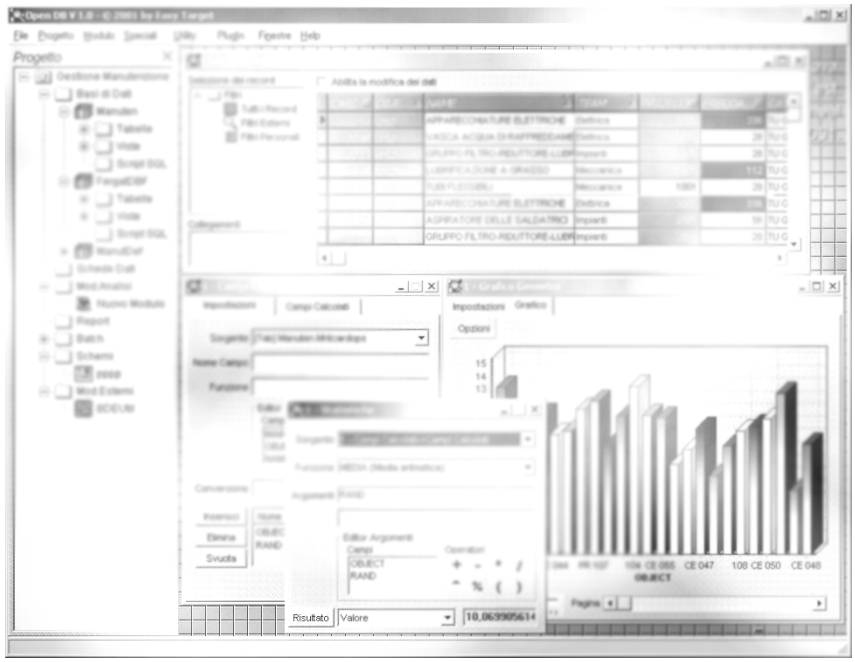
# Open DB 1.3

## Manuale per l'Utente



# Open DB 

***Accesso a database multipli ·***

***Definizione di Tabelle e Query ·***

***Creazione di Moduli d'Analisi ·***

***Stampa di Pagine Dati ·***

***Creazione di Report Master Detail ·***

***Collegamneti Dinamici ·***

***Analisi Multidimensionale ·***

***Calcoli Statistici ·***

***Generazione di Grafici ·***

***Alberi di Raggruppamento ·***

***Reti Neurali ·***

***Procedure Batch ·***

***Esplorazione Automatica ·***

***Esportazione cross database ·***

***e molto altro ...***

# Sommario

# *Introduzione a Open DB*

I DataBase sono il cuore di ogni sistema informatico e moltissimi utenti hanno la necessità di analizzare in vari modi i propri dati.
Purtroppo reperire sul mercato degli strumenti per la gestione dei dati che si adattino alle esigenze di ogni singolo utente è piuttosto difficile.

Attualmente un utente che non abbia le competenze necessarie per sviluppare i programmi di cui ha bisogno, può contare essenzialmente sulle funzionalità di applicazioni come Excel che, però, non sono in grado di agire direttamente sui dati, ma necessitano di una fase di importazione.
In alternativa si possono utilizzare strumenti come Access che consentono di creare applicazioni basate su database in modo abbastanza semplice, ma obbligano l'utente ad utilizzare il loro formato di gestione dei dati e comunque per analisi di una certa complessità o la generazione di grafici richiedono la conoscenza di un linguaggio di programmazione.

In pratica gli utenti sono costretti ad affidarsi ad una software house che sviluppi per loro delle applicazioni personalizzate. Questa soluzione ha i seguenti lati negativi:

- costo elevato
- tempi di sviluppo e messa a punto piuttosto lunghi
- necessità di interventi esterni all'insorgere di nuove esigenze

Vi sono poi molte realtà in cui si utilizzano più database (magari diversi) contemporaneamente e gli utenti sono costretti ad utilizzare molti programmi diversi, spesso con interfacce non omogene.

Esistono dei pacchetti software che consentono di mettere ordine in queste situazioni, ma sono caratterizzate da costi molto elevati e solitamente tendono a *"legare indissolubilmente"* gli utenti ai fornitori di tali applicazioni.

L'obiettivo di Open DB è di consentire ad un utente comune di definire l' ambiente personalizzato in cui intende operare, fornendogli tutti gli strumenti necessari per gestire i dati provenienti da qualunque sorgente.

In pratica si tratta di un ambiente di sviluppo di applicazioni per l'accesso ai dati di tipo *Ultra RAD*, in quanto la modalità operativa è simile a quella utilizzata con prodotti come Visual Basic o Delphi.
Questi ambienti semplificano lo sviluppo di applicazioni di qualunque tipo, mettendo a disposizione del programmatore una serie di componenti preconfezionati, ma richiedono comunque una buona conoscenza di un linguaggio di programmazione.

Open DB semplifica al massimo lo sviluppo di applicazioni di accesso ai dati, mettendo a disposizione dell'utente una serie di strumenti di alto livello che possono interagire all'interno dell'ambiente di lavoro e non necessitano di programmazione.

# Cenni Preliminari

Open DB non necessita di un motore di database particolare, ed è in grado di accedere a qualunque database per cui esista un driver ODBC e supporti il linguaggio SQL standard.

Il programma consente di creare dei progetti in cui vengono definiti tutti gli elementi per la gestione dei dati.

Durante l' utilizzo del programma, un utente può posizionare i vari elementi come preferisce e tali posizioni vengono mantenute automaticamente in modo da rendere più agevole l'esecuzione di compiti ripetitivi.

Ogni progetto viene salvato fisicamente in due file:

- il file primario con estensione **ODB** contiene le definizioni di tutti gli elementi
- il file secondario con estensione **ODP** contiene le posizioni dei vari moduli

La suddivisione consente di ottimizzare i tempi di salvataggio delle impostazioni.
Nel caso di utilizzo multi utente di un unico progetto, il file primario può essere salvato in una cartella di rete condivisa, mentre il file secondario potrà essere salvato sul disco locale del PC dei vari utenti, in modo che ognuno sia in grado di personalizzare il proprio ambiente di lavoro.

I progetti possono essere protetti da Password e supportano la gestione di profili utente multipli.
All' interno di un singolo progetto possono essere definite un numero qualsiasi di sorgenti dati indipendentemente dal motore che le gestisce; ad esempio si può accedere contemporaneamente:

- ad una directory contenente tabelle dBase
- ad una libreria su AS/400
- ad un file creato con Access
- ad un database Oracle
- ecc.

Gli elementi definibili all'interno di un progetto sono:

- Tabelle
- Query
- Istruzioni SQL
- Schede Dati
- Moduli complessi di Analisi
- Operazioni di trasferimento dati
- Report di tipo Master/Detail
- Schemi di Analisi
- Moduli Esterni

Inoltre sono sempre disponibili i moduli: Mixer Dati, Statistica, Grafico.
La definizione degli elementi è dinamica nel senso che, ad esempio un grafico cambierà al variare dei dati a cui fa riferimento.

All'interno del programma ogni elemento coinvolto in un'analisi può interagire con gli altri in

modo dinamico; ad esempio:

- un modulo tabella può essere utilizzato come sorgente per un modulo satistico
- è possibile filtrare i record di un modulo tabella sul record attivo di un modulo query
- un modulo grafico può essere agganciato ad un mixer dati

Il programma inoltre comprende le seguenti utility:

- Gestione Alias
- Analisi ed Esplorazione Database
- Ricerca Testo
- Informazioni Tabella
- Import / Export File di Testo formattato
- Creazione tabelle
- Esportazione Tabelle
- Console SQL
- Operazioni Batch

# Ottenere Aiuto

## *Hint*

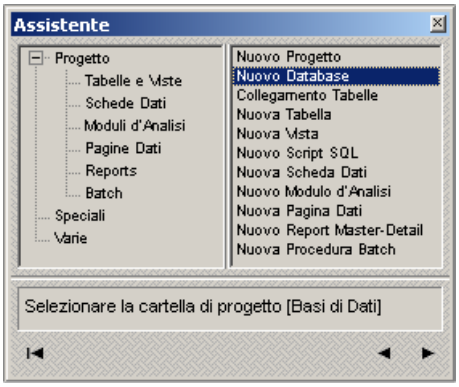
Ogni elemento dell'interfaccia è dotato di un aiuto sintetico che ne spiega la funzione.
Per visualizzare tale aiuto è sufficiente muovere il cursore del mouse sull'elemento desiderato: il testo descrittivo viene visualizzato sulla barra di stato nella parte inferiore della finestra principale.

## *Help contestuale*

Questo Help contiene la descrizione di ogni modulo dell'applicazione.
Premendo il tasto [F1] viene visualizzata la pagina di Help relativa alla finestra attiva.

## *Assistente*

L'Assistente è una finestra "Stay On Top" che contiene la descrizione passo-passo delle principali procedure operative.
Si basa su normali file di testo che possono essere modificati o ampliati anche dagli utenti.



# *Licenza d'Uso*

Il presente documento rappresenta un contratto tra il detentore della licenza (utente) e Easy Target. Easy Target fornisce il presente software e ne concede la licenza d'uso.
La responsabilità della scelta del software idoneo ad ottenere i risultati desiderati, nonché dell'installazione, dell'uso e dei risultati ottenuti dal software ricade sull'utente.
Easy Target non vende alcun diritto sul software, se non espressamente stipulato nel presente contratto, e si riserva qualsiasi diritto che non sia stato espressamente concesso al detentore della licenza.

## LICENZA

In caso di software monoutente, Easy Target concede al detentore della licenza un diritto non esclusivo e non cedibile di installare, utilizzare e visualizzare il software su un unico computer.
Il detentore della licenza potrà eseguire una copia del software in una qualsiasi forma leggibile su computer o stampata, allo scopo di avere una copia di riserva.
L'avviso di diritto d'autore (copyright) dovrà essere riprodotto e inserito in qualsiasi copia del software.
Qualsiasi decompilazione o disassemblaggio, integrale o parziale, è formalmente vietato.

## SCADENZA

La licenza rimarrà in vigore fino a quando non verrà annullata. Il detentore della licenza potrà annullarla in qualsiasi momento distruggendo il software e le relative copie. Essa verrà inoltre annullata in caso di mancato rispetto da parte del detentore della licenza di uno dei termini o delle condizioni del presente contratto. Il detentore della licenza accetta di distruggere il programma e le relative copie in caso di annullamento.

## GARANZIA LIMITATA

Il software viene fornito in buone condizioni senza alcun tipo di garanzia, esplicita od implicita, ivi comprese, senza che tuttavia tale limitazione sia esaustiva, le garanzie implicite per la commercializzazione o l'idoneità ad un uso particolare.
Tutti i rischi relativi alle qualità e alle prestazioni del programma sono sostenuti dal detentore della licenza. In caso di difetto del programma, il detentore della licenza (e non Easy Target) sosterrà tutte le spese necessarie per la localizzazione del difetto, la riparazione o la correzione.
Easy Target non garantisce che le funzioni contenute nei programmi corrisponderanno alle esigenze del detentore della licenza o che il funzionamento del programma non sarà soggetto ad interruzioni od errori.
Tuttavia, Easy Target garantisce che i supporti sui quali viene fornito il programma sono esenti da difetti di materiale e di assemblaggio per un periodo di trenta (30) giorni a decorrere dalla data di consegna al detentore della licenza, definita dalla copia del buono di ricevimento.
L'unica responsabilità di Easy Target e l'unico risarcimento dovuto al detentore della licenza sarà: 1. La sostituzione di qualsiasi supporto che non corrisponda alla garanzia limitata di Easy Target e che sarà stato restituito a Easy Target con una copia della ricevuta del detentore della licenza o 2. Nel caso in cui Easy Target si trovi nell'impossibilità di fornire un dischetto esente da difetti di materiale e di assemblaggio, il detentore della licenza potrà rescindere il presente contratto restituendo il programma e ricevere il relativo rimborso.
In nessun caso Easy Target sarà ritenuta responsabile nei confronti del detentore della licenza per qualsiasi tipo di danno, ivi comprese le perdite in termini di profitto o di risparmio o i danni accidentali o indiretti a terzi derivanti dall'utilizzo o dall'impossibilità di utilizzo del programma, anche nel caso in cui Easy Target sia stato informato della possibilità di tali danni o di qualsiasi reclamo.

## DISPOSIZIONI GENERALI

L'utente dichiara di avere letto il presente contratto, di averlo compreso e di accettarne i termini e le condizioni. Riconosce inoltre che tale contratto costituisce condizione integrale ed esclusiva dell'accordo e sostituisce qualsiasi proposta od accordo precedente, verbale o scritto, e qualsiasi altra comunicazione avvenuta relativamente all'oggetto del presente contratto.

# Capitolo 1

# *Utilizzo dell' Applicazione*

Il programma mette a disposizione dell' utente vari strumenti che consentono di sviluppare sistemi d' analisi anche piuttosto complessi senza richiedere competenze nel campo della programmazione.

## Strumenti del Programma

Gli strumenti del programma possono essere suddivisi in tre sezioni:

### *Moduli di Progetto*

**Tabelle:** Sono dei moduli che visualizzano i dati contenuti all'interno delle tabelle sotto forma di griglia. Consentono di definire dei filtri e dei collegamenti ad altri moduli per definire relazioni di tipo Master/Detail

**Query:** Sono moduli simili alle tabelle, ma visualizzano i dati relativi ad una interrogazione SQL.

**Istruzioni SQL:** Mantengono la definizione di istruzioni SQL per l'esecuzione di operazioni ripetitive.

**Schede Dati:** Sono moduli che visualizzano i dati contenuti in una tabella o Vista SQL in modalità scheda. Il layout della scheda viene definito tramite il relativo editor.

**Moduli di Analisi:** Sono moduli creati con l'apposito editor in cui si possono inserire griglie dati, grafici ed una serie di componenti per effettuare calcoli statistici.

**Pagine Dati:** Consentono di impaginare i risultati di un'analisi, includendo griglie, grafici, forme, testo ecc.

**Report:** Mantengono le definizioni per la stampa di report di tipo Master/Detail

**Operazioni Batch:** Mantengono le definizioni di operazioni di trasferimento dati tra due tabelle.

**Schemi di Analisi:** Mantengono le impostazioni di visualizzazione di un insieme di moduli

**Moduli Esterni:** Mantengono i collegamenti a DLL o eseguibili esterni

## *Moduli Speciali*

**Mixer Dati:** Consente di miscelare in un' unica tabella le colonne provenienti da moduli esterni

**Campi Calcolati:** Consente di creare dei dataset contenenti campi presi da sorgenti esterne e campi calcolati

**Statistica:** Consente di effettuare dei calcoli sui dati contenuti nelle tabelle di moduli esterni

**Grafico:** Consente di creare dei grafici sui dati contenuti nelle tabelle di moduli esterni

**Raggruppa Record:** Consente di creare un albero di raggruppamento di record utile per tabelle di grandi dimensioni

**Foglio di Calcolo:** Consente di manipolare i dati similmente a Microsoft Excel

**Rete Neurale:** Consente di definire, addestrare ed utilizzare una Rete EBPN

## *Utility*

**Gestione Alias:** Consente di modificare la definizione degli Alias di sistema

**Esplorazione Database:** Esegue un' analisi delle relazioni esistenti tra le tabelle di un database e ne consente l'esplorazione.

**Ricerca Testo:** Consente di effettuare la ricerca di un particolare testo all'interno di tutti i campi di una tabella o di un intero database.

**Informazioni Tabella:** Visualizza la struttura di una tabella, i suoi indici, la sua rappresentazione sotto forma di script SQL e i dati in essa contenuti.

**Creazione tabelle:** Consente di definire la struttura di una tabella e di crearla.

**Importa/Esporta Testo:** Importa o esporta un file di testo formattato

**Esportazione Tabelle:** Consente di esportare i dati di una tabella o di un intero database verso un altro database. Nel caso il DB destinazione non contenga le tabelle, queste vengono create.

**Console SQL:** Consente di eseguire script SQL e di interrogare un database qualsiasi.

**Operazioni Batch:** Consente di trasferire dati tra due tabelle effettuando eventuali conversioni. La sezione di progetto omonima consente di operare soltanto nell'ambito dei DB definiti nel progetto stesso, mentre questa utility può operare su qualunque database.

# Analisi DataBase Personali

Un utente che non abbia competenze nel campo della programmazione può trovare in Open DB uno strumento potente e versatile per definire il proprio sistema d'analisi.

Ad esempio potrebbe utilizzarlo per accedere ad un database di tipo Access che si trova sul suo disco locale, ma non esiste alcuna limitazione all'accesso a DB aziendali ammesso che ne abbia i necessari diritti.

Open DB può essere utilizzato come strumento di produttività personale da affiancare ai normali strumenti tipo Office. In particolare la possibilità di esportare qualunque Set di dati verso Excel può essere di grande utilità.

# Analisi in ambiente Multi Utente

In un'azienda di medie dimensioni, Open DB può rivestire un ruolo importante nel lavoro degli operatori incaricati della manutenzione del sistema informatico.

Un programmatore del C.E.D. spesso deve creare programmi, di per se semplici, ma estremamente variegati, per rendere agevole l'accesso ai dati ai vari utenti.
Ogni utente necessita di accedere ad una particolare area del sistema gestionale e la creazione e manutenzione di tutti programmi specifici richiede un'ingente quantità di risorse.

L' esempio tipico è rappresentato dalle aziende che utilizzano sistemi come AS/400.

Open DB può quindi essere utilizzato dai programmatori per creare progetti specifici in modo rapido e tali progetti possono essere utilizzati contemporaneamente da molti utenti che possono operare con diversi profili di autorizzazione.

Tutti i progetti avranno interfacce omogenee e offriranno funzionalità standard in un' ottica orientata agli oggetti e questo ridurrà i tempi di addestramento e apprendimento.

# Sviluppo Progetti d'Analisi

Una Software House che opera in campo aziendale, normalmente deve sviluppare un numero elevato di applicazioni verticali di dimensioni modeste, ma che devono essere personalizzate sulle esigenze specifiche di ogni cliente.
Ogni utente ha un proprio modo di lavorare ed è abituato a visualizzare i dati in modo particolare, quindi non è facile convincerlo ad adattarsi ad un programma prefatto.

Open DB, grazie alla sua versatilità può risolvere la maggior parte dei problemi di personalizzazione.

Può essere utilizzato in associazione ad un' applicazione preesistente per implementare parti accessorie di analisi dei dati; tali progetti possono poi essere modificati direttamente dall'utente all'insorgere di nuove esigenze.

Consente di sviluppare applicazioni relativamente semplici che non richiedono elaborazioni particolari, ma che comunque consentono la visualizzazione, la modifica e l'analisi dei dati.

I progetti sviluppati con Open DB, possono anche essere distribuiti con il modulo RunTime che non ne consente la modifica.

# Capitolo 2

# *Interfaccia Utente*

## Ambiente di Lavoro

Open DB è un' applicazione di tipo MDI cioè caratterizzata da una finestra principale che funge da contenitore per tutti gli altri moduli.
La finestra principale del programma presenta i seguenti elementi:

Menu principale nella parte superiore
Pannello del Progetto a sinistra che visualizza un albero con le varie sezioni
Barra di stato in basso che visualizza l' aiuto contestuale

Il Pannello del Progetto può essere nascosto per liberare spazio nell'area di lavoro.
Quando è nascosto, è possibile accedere a tutti gli oggetti del progetto utilizzando il Popup menù che viene attivato clickando in un punto qualunque dell'area di lavoro.
Le impostazioni relative alla posizione e dimensione della finestra principale vengono salvate in automatico nel registro di sistema alla chiusura dell'applicazione e ripristinate alla successiva apertura.

Alla chiusura del programma vengono salvate nel registro di sistema le seguenti informazioni:

Posizione e dimensione della finestra principale
Dimensione del pannello di progetto
Percorso dell'ultimo progetto aperto

Lo sfondo delle finestre del programma può essere impostato con la relativa voce nel menù File.

All'interno di ogni finestra, premendo il tasto F1 viene visualizzato l'aiuto relativo.

Tutti i moduli Child del programma sono in grado in interagire dinamicamente.
Ad esempio:
Due o più moduli Tabella possono lavorare in modalità Master / Detail
Due moduli Query possono essere usati come sorgente per un Mixer Dati
Un grafico può essere agganciato ad una serie calcolata con il modulo statistico
Un modulo d'analisi custom può operare su una sorgente esterna
Il dinamismo stà nel fatto che al variare di un Set di dati, tutti i moduli agganciati a quel dataset vengono aggiornati automaticamente.

# Pannello di Progetto

Il pannello visualizza la gerarchia di tutti gli oggetti definiti all'interno del progetto, può essere ridimensionato a piacere ed eventualmente chiuso per poter disporre di tutto lo spazio dell'area di lavoro.

Consente le seguenti operazioni:

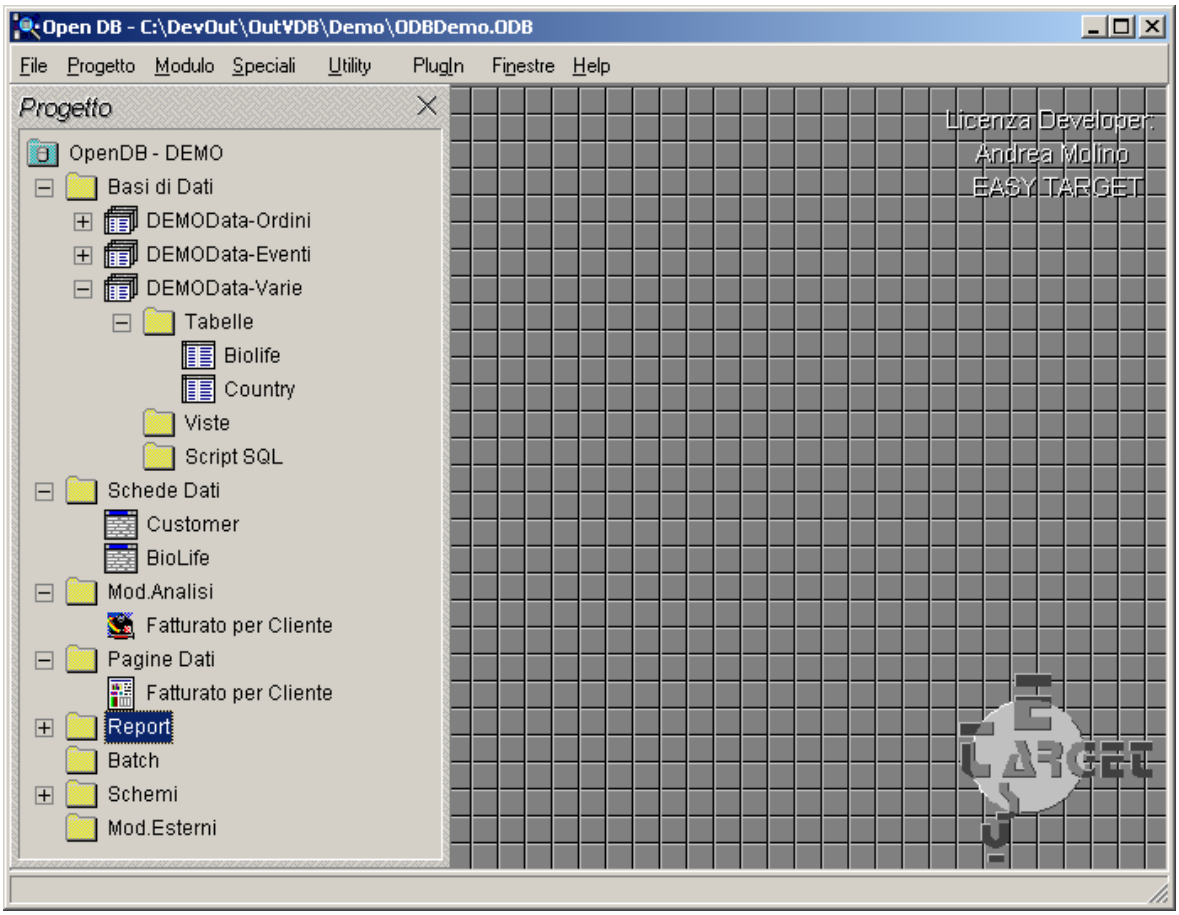
- muovendo il mouse sui vari oggetti, la barra di stato visualizza la relativa descrizione
- premendo il tasto destro del mouse, viene visualizzato il menù contestuale
- con un doppio click, l'oggetto selezionato viene aperto

Il menù contestuale da accesso alle operazioni di modifica del progetto nel modo seguente:

Attivandolo sulla radice dell'albero è possibile
- Cambiare nome e descizione al progetto
- Cambiare le impostazioni relative ai controlli di sicurezza

Attivandolo su una cartella è possibile:
- Creare un nuovo oggetto
- Eliminare tutti gli oggetti contenuti

Attivandolo su di un oggetto è possibile
- Ridenominarlo
- Modificarne la definizione
- Duplicarlo
- Eliminarlo



# Comandi del Menù

## *Menù File*

| | |
|---|---|
| Nuovo: | Crea un nuovo progetto |
| Apri...: | Apre un progetto |
| Riapri...: | Riapre uno dei progetti aperti in precedenza |
| Salva: | Salva il progetto |
| Salva con nome... | Salva il progetto con un nome diverso |
| Chiudi | Chiude il progetto |
| Imposta Sfondo... | Imposta il bitmap da usare come sfondo per le finestre |
| Sfondo di Sistema | Imposta lo sfondo delle finestre di sistema |
| Uscita | Esce dal programma |

## *Menù Progetto*

| | |
|---|---|
| Visualizza Progetto | Visualizza o nesconde il pannello di progetto |
| Nuovo | Crea un Nuovo Oggetto |
| Rinomina | Rinomina l'oggetto selezionato |
| Modifica | Modifica l'oggetto selezionato |
| Duplica | Crea un nuovo oggetto con le stesse proprietà di quello selezionato |
| Elimina | Elimina l'oggetto selezionato |
| Elimina Tutti | Elimina tutti gli oggetti contenuti |
| Aggiungi Tabelle | Crea in automatico i moduli per la visualizzazione delle tabelle |
| Gestione Sicurezza | Visualizza il modulo per l'impostazione dei controlli di sicurezza |

## *Menù Modulo*

| | |
|---|---|
| Salva Impostazioni | Rende persistenti le impostazioni attuali del modulo |
| Ripristina | Ripristina le ultime impostazioni salvate del modulo |
| Visualizza Record | Apre il pannello per la visualizzazione del record |
| Modifica Record | Apre il pannello per la modifica dei dati |
| Esporta Dati/Stampa | Apre il pannello per l'esportazione e la stampa dei dati visualizzati |
| Copia Testo SQL | Copia nella ClipBoard il testo della vista SQL con gli attuali filtri |
| Visualizza Struttura | Visualizza le informazioni relative alla struttara della tabella |
| Personalizza Griglia | Modifica le impostazioni della Griglia |
| Ripristina Griglia | Ripristina le impostazioni della griglia ai valori di default |
| Visualizza Allegati | Gestione degli allegati al record selezionato |
| Elimina tutti gli Alleg. | Elimina gli allegati agganciati a tutti i Record |
| Nuovo Filtro | Aggiunge un nuovo Filtro |
| Modifica Filtro | Modifica il filtro selezionato |
| Elimina filtro | Elimina il filtro selezionato |
| Svuota Filtri | Elimina tutti i filtri personali |
| Aggiungi Collegam. | Aggiunge un link ad un modulo esterno per il filtraggio Master/Detail |
| Elimina Collegamento | Elimina il collegamento selezionato |
| Svuota Elenco Coll. | Elimina tutti i collegamenti a moduli esterni |

## *Menù Speciali*

| | |
|---|---|
| Mixer Dati | Apre un modulo per il mix dei dati |
| Campi Calcolati | Apre un modulo per la creazione di campi calcolati |
| Statistica | Apre un modulo per l'analisi statistica |
| Grafico | Apre un modulo per la visualizzazione di grafici |
| Raggruppa Record | Apre un modulo per effettuare il raggrupp. dei record di una tabella |
| Foglio di calcolo | Apre un modulo utilizzabile come un normale Foglio di calcolo |
| Rete Neurale | Apre un modulo oer definire, Addestrare e Utilizzare una Rete Neurale |

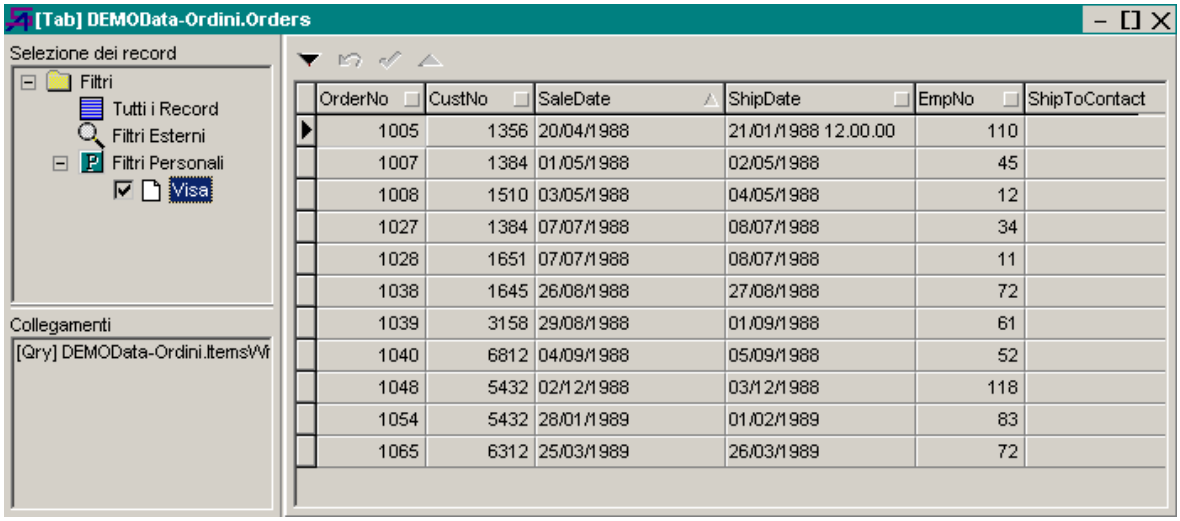
## *Menù Utility*

| | |
|---|---|
| Gestione Alias | Consente di modificare la definizione degli Alias di sistema |
| Esplora DataBase | Analizza ed Esplora un database |
| Ricerca Testo | Ricerca parole in tutti i campi di una tabella o di un intero Database |
| Info Tabelle | Visualizza le informazioni relative ad una tabella |
| Imp./Esp. Testo | Importazione un file di testo formattato in una tabella<br>Esportazione di una tabella in un file di testo fromattato |
| Crea Tabella | Crea una nuova tabella |
| Esportazione | Esportazione di tabelle da un DB ad un altro |
| Console SQL | Selezione dati o esecuzione comandi SQL |
| Procedura Batch | Trasferimento o aggiornamento di dati |

# Capitolo 3

# *Sezioni di Progetto*

## *Tabelle e Viste SQL*




I moduli Tabella e Vista SQL possono essere utilizzati per la visualizzazione e la modifica dei dati e offrono molte funzioni:

- Filtraggio
- Definizione relazioni tipo Master/Detail
- Esportazione dati
- Stampa
- Gestione Allegati

Ogni modulo è suddiviso in tre sezioni:

### Albero dei filtri per la selezione dei record

Esistono due tipi di filtro:

- Fitri esterni: viene utilizzato il record di una tabella *master* per il filtraggio
- Filtri personali: definiti tramite l'apposito · modulo

Utilizzando il popup menù associato è possibile:

- Creare, modificare, eliminare i filtri
- Selezionare la modalità di congiunzione di filtri multipli (AND / OR)
- Abilitare l'aggiornamento dinamico per i filtri esterni

La sezione dei filtri esterni di un modulo, contiene il riferimento a tutti i moduli di tipo tabella o vista attualmente aperti e compatibili cioè contenenti un collegamento al modulo stesso.

### *Elenco dei collegamenti Master/Detail*

I collegamenti vengono utilizzati per definire relazioni tipo Master/Detail tra due tabelle o viste.
Ad esempio supponiamo di avere due tabelle: **Clienti** (Master) e **Ordini (**Detail). Nel modulo dei Clienti si puo' definire un collegamento al modulo degli ordini in modo che selezionando un particolare cliente sul primo modulo, vengano visualizzati i suoi ordini sull'altro. Occorre utilizzare il relativo popup menù per gestire i collegamenti.

### Griglia per la visualizzazione dei dati

La griglia dati fornisce diverse funzionalità:

- Personalizzazione
- Vista Scheda Record
- Ordinamento dei record
- Ricerca Incrementale
- Filtro al Volo

Tramite il popup menù relativo si ha accesso alle principali funzioni.

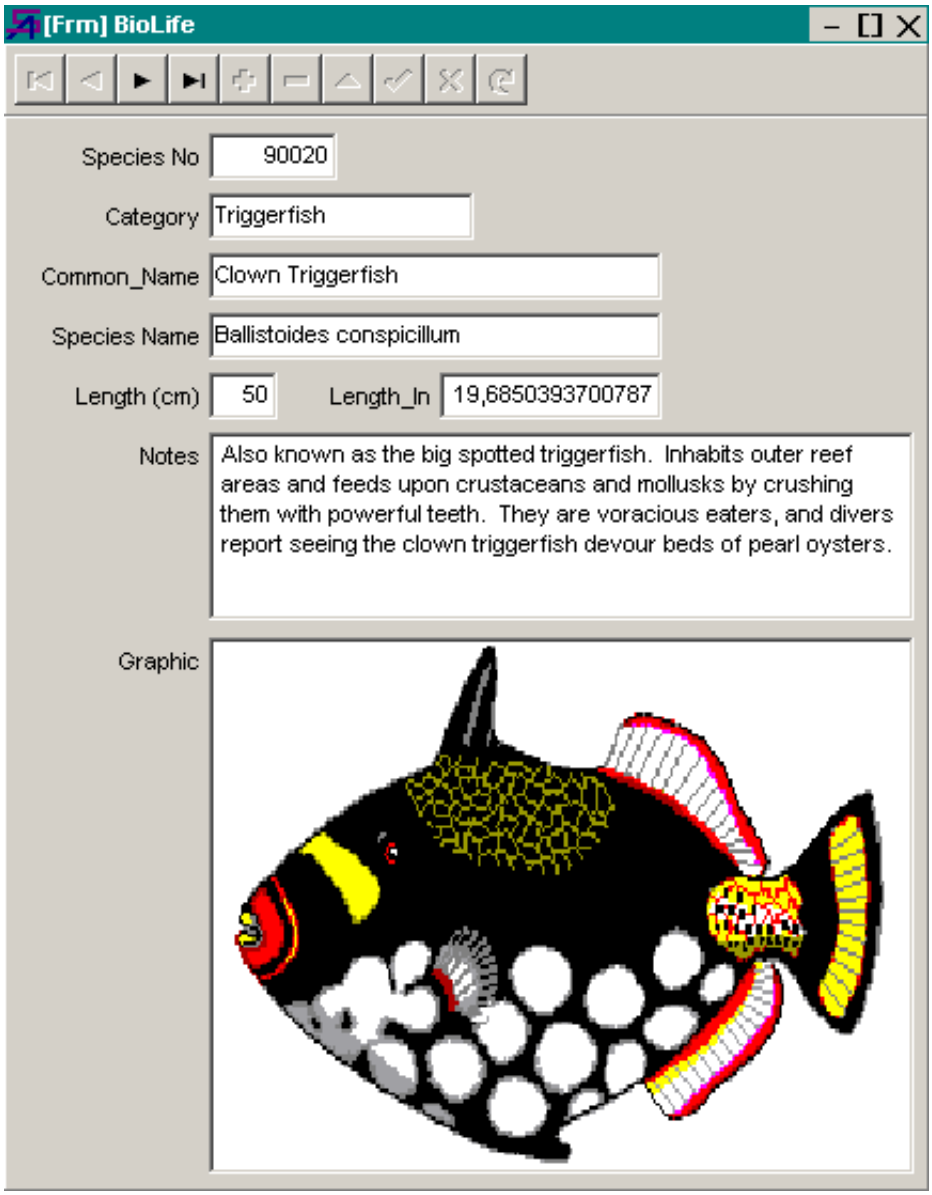
# Script SQL

All'interno di un progetto è possibile definire degli script SQL per eseguire operazioni sui dati. E' utile per oparzioni ripetitive come può essere ad esempio lo svuotamento di una tabella ausiliaria.

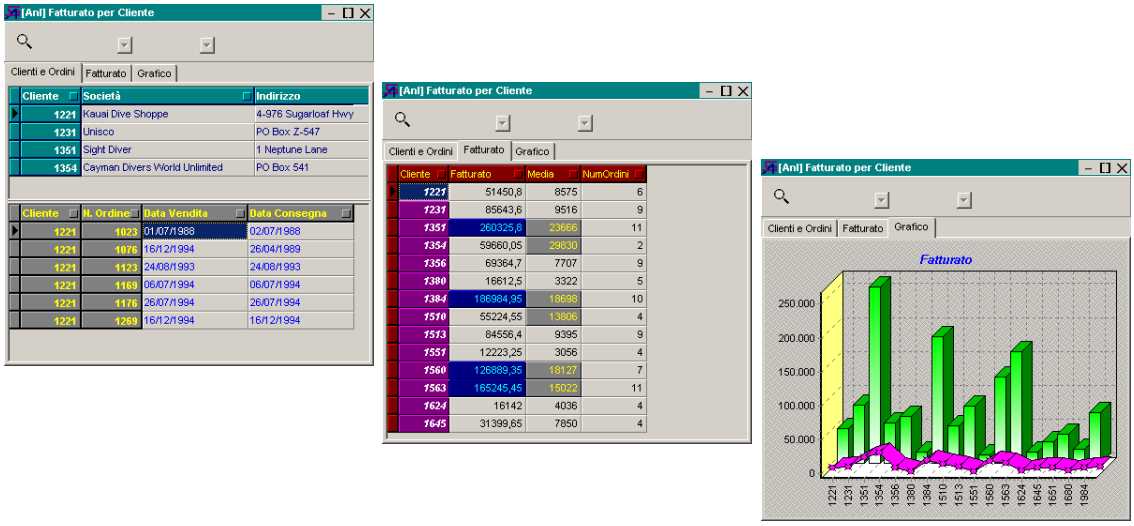
## *Scheda Dati*

I moduli di questo tipo vengono creati utilizzando l'apposito Editor e possono essere utilizzati per visualizzare o modificare i dati di una tabella sotto forma di scheda.
Rispetto alla visualizzazione standard, le schede consentono:

- di personalizzare il layout
- visualizzare campi Memo e Bitmap
- impostare lookup e liste di valori perdefiniti



# Moduli D'Analisi




Le analisi personalizzate consentono di definire dei moduli con varie sorgenti dati provenienti anche da database multipli. I dati possono essere visualizzati tramite griglie e grafici.

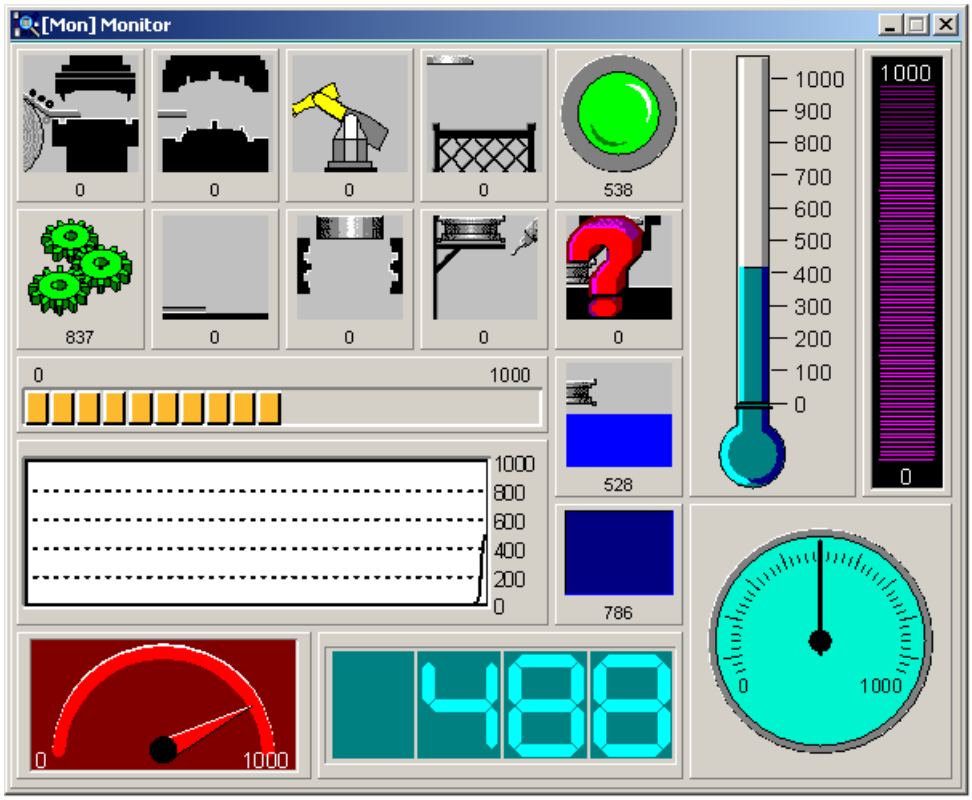
Sui dati è possibile effettuare calcoli creando dei dataset risultato; tali dataset possono essere utilizzati per calcoli ulteriori. Tramite un'analisi personalizzata è possibile generare campi calcolati, effettuare statistiche, unire due o più dataset in un'unica tabella e tali operazioni possono essere reiterate sulle tabelle risultanti.

# Monitor Real Time

I monitor in tempo reale sono in grado di visualizzare in modo grafico il valore di segnali vari provenienti da sistemi di monitoraggio.

Possono essere utilizzati per visualizzare dati contenuti all'interno di tabelle che v engono aggiornate con una frequenza alta come quelle relative, ad esempio, ad un sistema di controllo dell'avanzamento della produzione o al monitoraggio dello stato di indicatori di un qualunque processo.

E' anche possibile visualizzare dati provenienti da sorgenti diverse da un database standard; infatti i moduli di monitoraggio sono in grado di agganciarsi a DLL esterne che provvederanno a fornire i dati necessari.

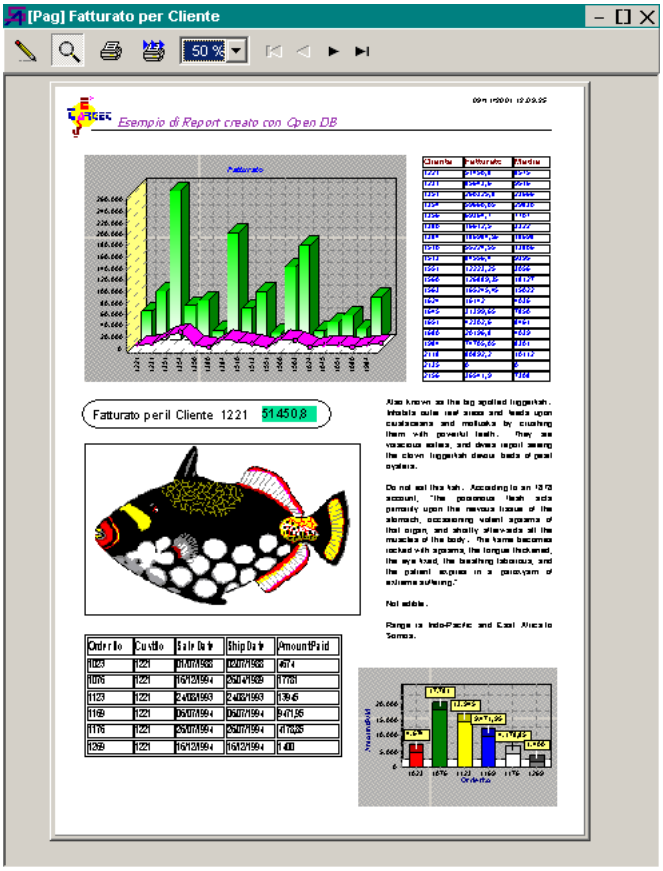



## *Pagine Dati*

Le Pagine Dati consentono di creare dei report contenenti i dati generati utilizzando qualunque modulo dell'Applicazione.
L'editor integrato è a tutti gli effetti un impaginatore che permette di posizionare i vari elementi secondo uno schema completamente personalizzabile.
Ogni report può contenere un numero qualunque di pagine in formato A4 o A3, in orizzontale o in verticale.
In una pagina è possibile inserire i seguenti elementi:

- Forma: Linee, Rettangoli, Cerchi...
- Testo
- Dato di Sistema: Titolo del Report, Num. Pagina, Data, Ora...
- Immagine
- Griglia su DataSet: visualizza i dati contenuti in un dataset in forma tabellare
- Griglia su Query: visualizza i dati prelevati tramite una Query SQL in forma tabellare
- Record su DataSet: visualizza i dati contenuti nel record attivo di un dataset
- Record su Query: visualizza i dati di un record prelevato tramite una Query SQL
- Campo su DataSet: visualizza un campo nel record attivo di un dataset
- Campo su Query: visualizza un campo in un record prelevato con una Query SQL
- Collegamento a Griglia: visualizza una griglia contenuta in un modulo esterno
- Collegamento a Grafico: visualizza un grafico contenuto in un modulo esterno

Se il pulsante di rinfresco viene attivato, gli elementi che risultano collegati a dataset, griglie o grafici esterni, vengono aggiornati dinamicamente al variare degli oggetti a cui puntano.




E' possibile effettuare stampe singole e stampe multiple: le prime stampano le pagine così come sono visualizzate, mentre con le stampe multiple è possibile stampare report di tipo master/detail collegando la stampa ad un dataset. In pratica per ogni record del dataset verrà rigenerato dinamicamente il report prima della stampa.

# *Procedure Batch*

Queste procedure sono particolarmente utili agli sviluppatori che necessitano di trasferire dati da una vecchia base dati ad una nuova, come succede ad esempio durante lo sviluppo di una nuova applicazione che va a sostituirne una preesistente. Consente di:

- Trasferire dati tra due tabelle qualsiasi anche in database diversi
- Aggiornare la tabella di destinazione con dati provenienti dalla tabella sorgente
- Modificare i dati contenuti in una tabella.

La peculiarità principale del modulo è la capacità di applicare delle funzioni di trasferimento ai dati da trasferire.

Ad esempio è possibile concatenare stringhe provenienti da due campi in uno solo o convertire un campo valuta da Lire a Euro.

La funzione di trasferimento può essere condizionata da situazioni particolari come il valore contenuto nel campo destinazione.

# Report

Oltre ad i report standard stampabili a partire da ogni tabella (funzione di esportazione/ stampa), è possibile creare dei report di tipo Master/Detail.
Per la creazione, viene utilizzato un editor che consente di ottenere dei buoni risultati visivi senza la necessità di posizionare i singoli campi, operazione questa che solitamente comporta un notevole impegno.

I report possono avere fino a tre livelli di dettaglio come ad esempio:

Aree geografiche
Clienti di ogni area
Ordini di ogni cliente
Articoli richiesti in ogni ordine

I record possono essere stampati sia sottoforma di tabella che di scheda.

# Schemi

Alcuni tipi di analisi che utilizzano i moduli standard di progetto vengono effettuate ripetutamente. Per semplificare l'esecuzione di queste analisi, si possono definire degli schemi di finestre che possono essere salvati e ripristinati in seguito.

In pratica oltre alla capacità standard di ricordare l'utima posizione di ogni finestra, il programma consente di salvare la posizione di gruppi di finestre ed i loro collegamenti in uno schema. Così un modulo che viene utilizzato per effettuare varie analisi può essere posizionato nel modo più conveniente a secoda dei casi e la sua posizione viene mantenuta in un successivo utilizzo.

# Moduli Esterni

Sebbene l'applicazione consenta di creare dei progetti con funzionalità molto estese e complete, adeguate nella maggior parte dei casi, è possibile "agganciare" ad un progetto dei moduli esterni, essenzialmente delle Dll o degli eseguibili per eseguire delle operazioni complesse.

Tali moduli potranno essere lanciati a partire dal progetto come i moduli interni all'applicazione. Un esempio tipico è rappresentato da moduli che eseguono delle elaborazioni complicate per la generazione di report di stampa.

# Capitolo 4

# *Funzionalità di base*

## *Gestione Allegati*

Tramite questo modulo è possiibile allegare un numero qualunque di documenti ad ogni record di qualunque tabella.
Ad esempio è possibile allegare un' immagine o un testo creato con Word ad un record di anagrafica; tale documento non viene incluso nel database, ma ne viene mantenuto un collegamento all'interno del modulo che visualizza la tabella.

Il modulo di gestione degli allegati visualizza l'elenco dei documenti collegati al record selezionato e consente di aggiungere, eliminare, visualizzare o aprire ogni documento.

I formati non direttamente visualizzabili tramite il visualizzatore interno vengono aperti con l'applicazione di defaul.

## *Visualizzatore*

Questo modulo viene utilizzato per visualizzare gli allegati ad un record.
Tramite il visualizzatore è possibile vedere i seguenti fromati:

- Immagini: Bmp, Jpg, Gif, Tiff, Pcx
- Vettoriali: Wmf, Emf
- Testo: Txt, Rtf, Html
- Media: Wav, Avi, Mid

## *Anteprima HTML*

Questo modulo visualizza la griglia di dati in formato Html generata dalla funzione di esportazione.
Consente di sciegliere se visualizzare la griglia e di impostare la larghezza della griglia stessa.
I dati possono essere salvati in un file o aperti con il browser di default.

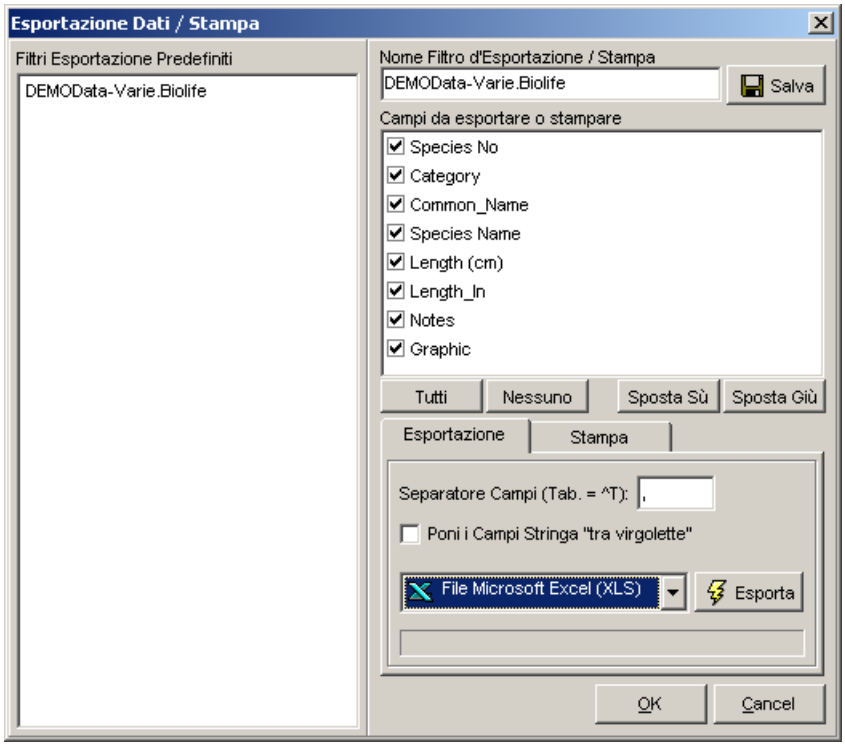
# *Esportazione Dati / Stampa*

All'interno dell'applicazione, qualunque griglia dati ha un popup menù associato da cui si può accedere alla funzione di esportazione e stampa.
Questa funzionalità viene utilizzata impostando una serie di parametri all'interno di una finestra di dialogo.
I parametri definiti possono essere salvati per un utilizzo futuro.
La finestra di dialogo consente di selezionare i campi da esportare o stampare ed il loro ordine.
Per quanto riguarda l'esportazione è possibile selezionare il formato tra i seguenti:

- File di testo formattato (è possibile impostare il carattere di separazione dei campi)
- File in forrmato Excel
- Invio diretto a Excel tramite OLE
- File in formato HTML
- Tabella Paradox
- Tabella dBase
- Tabella FoxPro
- Tabella Esistente (occorre poi selezionare una tabella con formato compatibile)
- Copia negli Appunti

Per la stampa è possibile impostare i seguenti parametri:

- Font per i titoli delle colonne
- Font per i dati
- Spaziatura Verticale (distanza tra le righe)
- Spaziatura orizzontale (distanza tra i campi)
- Bordo Celle (utile se si stampa la griglia)
- Stampa in Orizzontale (landscape)
- Stampa la Griglia
- Usa titoli colonne abbreviati (occorre il file con gli alias per i titoli \Res\ShortCap.txt)
- Modalità (Normale, Centrato, Adattato)



# *Personalizzazione delle griglie*

Tramite questo modulo è possibile personalizzare ogni aspetto delle griglie dati. Le modifiche effettuate sono immediatamente visibili sulla griglia stessa.

Il pulsante [Ripristina] consente di riportare la griglia all'aspetto che aveva prima dell'inizio della personalizzazione.

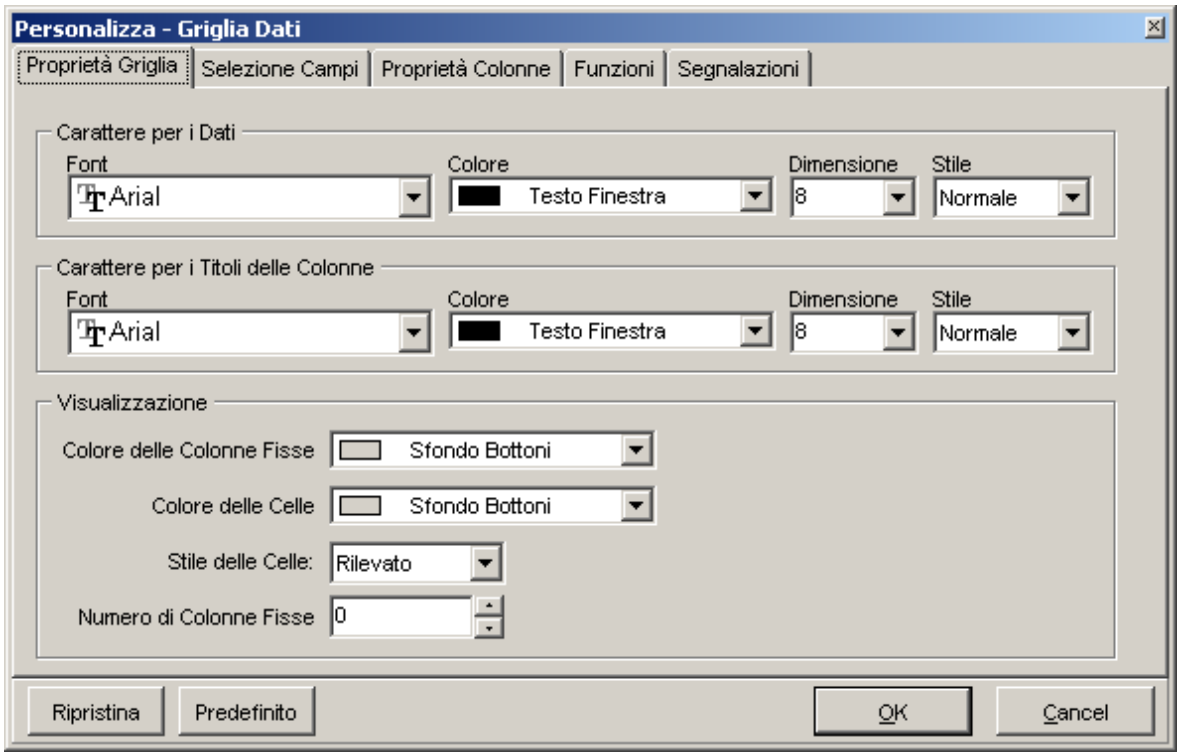
Il pulsante [Predefinito] consente di riportare la griglia all'aspetto standard della griglia.

Il modulo di personalizzazione è suddiviso in cinque pannelli:

## Proprietà della griglia

Consente di impostare le seguenti proprietà:

- Carattere per i dati di tutte le colonne
- Carattere per i titoli di tutte le colonne
- Colore delle colonne fisse
- Colore delle celle dei dati
- Stile delle celle
- Numero di colonne fisse



## Selezione dei campi

A sinistra si vede l'elenco dei campi disponibili, ma non visualizzati, mentre a destra si vede l'elenco dei campi visualizzati nella griglia.

Le funzioni dei vari pulsanti sono le seguenti::

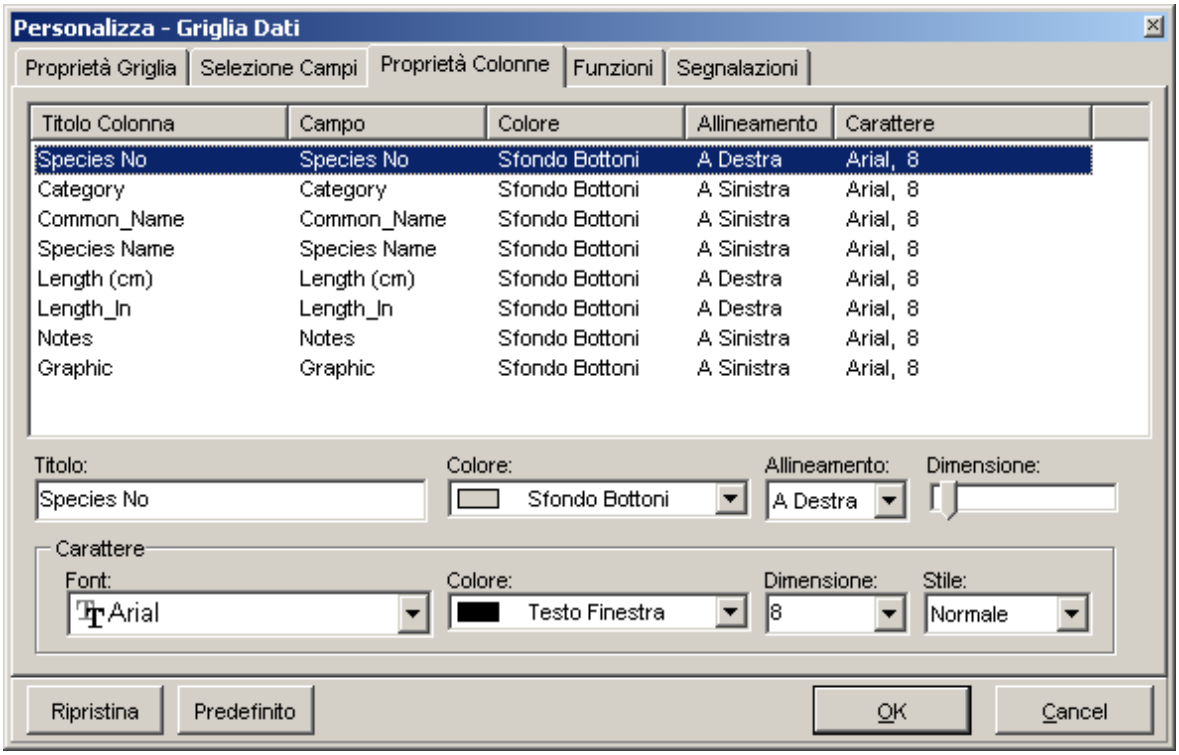
| | |
|---|---|
| [] | Crea le colonne standard per tutti i campi della tabella |
| > | Visualizza il campo selezionato |
| >> | Visualizza tutti i campi |
| < | Nasconde il campo selezionato |
| << | Nasconde tutti i campi |

Con un [Doppio Click] è possibile visualizzare o nascondere i vari campi
Utilizzando il Drag & Drop è possibile Cambiare l'ordine dei campi visualizzati
Con l'apposito pulsante è possibile creare un nuova colonna che potrà essere utilizzata per visualizzare i valori di un campo con l'applicazione di una funzione di conversione.
Ad esempio un campo che esprime una durata in secondi può essere visualizzato anche convertito in ore e centesimi.

## Proprietà della colonne

Per ogni colonna visualizzata è possibile impostare le seguenti proprietà:

- Titolo
- Colore
- Allineamento
- Dimensione
- Carattere



### Funzioni

Ad ogni colonna può essere applicata una funzione di conversione dei dati. Tale funzione può essere applicata a tutti i record o condizionata a particolari valori.

Le Funzioni disponibili sono:

[Conversione]
- Sostituisci con il valore...
- Aggiungi una Stringa in testa
- Aggiungi una Stringa in coda
- Moltiplica per il valore...
- Dividi per il valore...
- Aggiungi il valore...
- Sottrai il valore...
- Esprimi come % rispetto a
- Da secondi a hh.mm.ss
- Da secondi a HHH,cc
- Da secondi a GGG HH

[Calcoli]
- Parm1 * Parm2
- Parm1 / Parm2
- Parm1 + Parm2
- Parm1 - Parm2
- Parm1 % Parm2

[Conversione]
- Parm1 —> hh.mm.ss
- Parm1 —> HHH,cc
- Parm1 —> GGG HH

A seconda della funzione occorre impostare 1 o 2 parametri che possono essere costanti o fare riferimento ai valori di altri campi nello stesso record.

### Segnalazioni

I valori di ogni colonna possono essere evidenziati con colori differenti a senconda dei valori dei reletivi campi.
In pratica si possono colorare in modo diverso i campi di una stessa colonna impostando varie soglie.

## *Vista del record come scheda*

E' un pannello che visualizza il record selezionato sotto forma di scheda.
La scheda viene creata automaticamente in modo da visualizzare campi nel modo migliore possibile.
I campi avranno le stesse proprietà (font, colore, evidenziazione) delle relative colonne nella griglia, definite tramite la personalizzazione della stessa.
La scheda contiene anche un "navigatore" per spostarsi tra i vari record.
Se è stata abilitata la modalità di modifica, i valori dei vari campi saranno editabili.

## *Ordinamento*

Le Griglie consentono di cambiare l'ordinamento dei dati modificando automaticamente la clausola *ORDER BY* delle selezioni SQL.
Su titoli di ogni colonna è visibile un piccolo bottone quadrato che permette di selezionare i campi in base ai quali ordinare i dati.
Il primo click con il tasto sinistro del mouse sul bottone aggiunge il campo all'ordinamento, il secondo inverte l'ordine (*DESCEND*), mentre con un click del tasto destro, il campo viene eliminato dall'ordinamento.
E' possibile impostare l'ordinamento su più campi. Ad esempio una rubrica può essere ordinata prima per ***Città*** poi per ***Cognome*** ed infine per ***Nome***.

## *Ricerca Incrementale*

Le griglie dati consentono di effettuare una ricerca incrementale su ogni campo.
Per attivare la ricerca, è sufficiente clickare con il tasto sinistro del mouse sul titolo della colonna relativa al campo su cui si desidera eseguire la ricerca.
Inserendo dei caratteri nella casella di testo che appare dopo il click, il cursore della tabella si sposterà sul primo record il cui valore del campo inizia con i caratteri inseriti. Se non viene localizzato un record che corrisponde al testo inserito, il cursore non viene spostato.
I caratteri possono essere inseriti ignorando le maiuscole perchè la ricerca è "*Case Insensitive*".
Per nascondere la casella di testo, è sufficiente clickare in un punto qualunque della griglia.

## *Filtro "al volo"*

Nel caso fosse necessario filtrare i record di una tabella con un filtro semplice, (cioè utilizzando soltanto un operatore. *Es. val.campo > 100*) è possibile utilizzare il filtro fornito dalle griglie dati.
Per impostare il filtro, è sufficiente clickare con il tasto destro del mouse sul titolo della colonna relativa al campo in base al quale eseguire il filtraggio.
Nella casella di testo che appare dopo il click, occorre inserire l'operatore ed il valore desiderati.
Ad esempio si possono utilizzare filtri come i seguenti:

- > 100
- <= 10
- = 'SI'
- > 'mac'

Gli operatori utilizzabili sono: > , < , = , >=, <=, <>.
Per attivare il filtro basta premere [INVIO].
Per disattivare il filtro ci sono due possibilità:

- svuotare la casella di testo e premere [INVIO]
- clickare con il tasto sinistro sulla casella di testo

Per nascondere la casella di testo, è sufficiente clickare in un punto qualunque della griglia.
N.B. I moduli per la visualizzazione delle viste SQL (quelli con prefisso [Qry]) non consentono di utilizzare il filtro "al volo".

# Capitolo 5

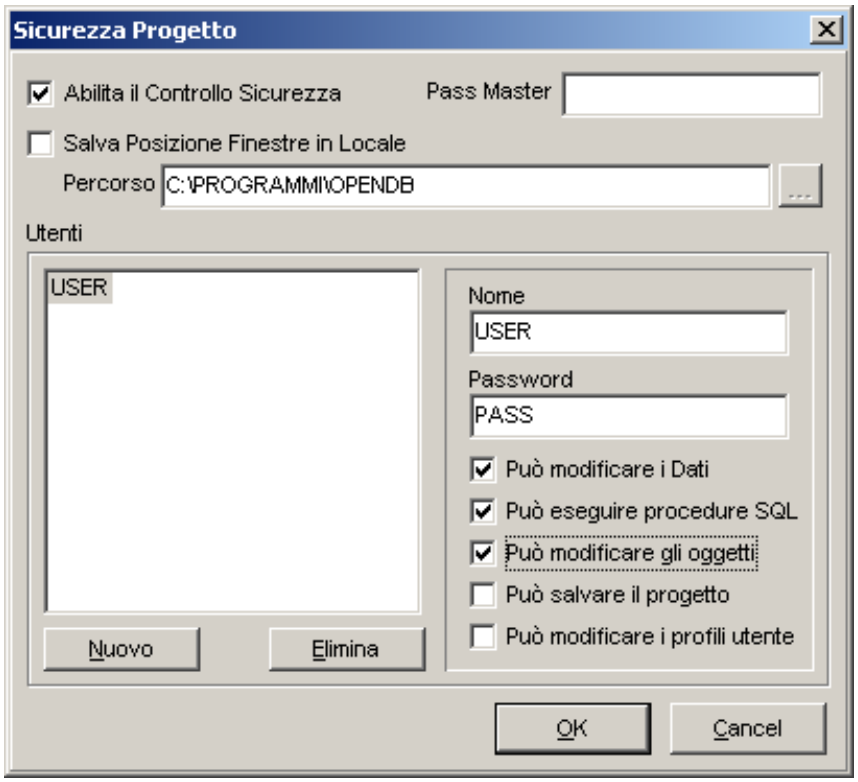
# *Definizione Progetto*

## *Sicurezza Progetto*

*Abilita Controllo di sicurezza*
Questa opzione consente di abilitare il controllo di sicurezza e quindi di richiedere una password all'apertura del progetto.
Nella casella *Pass Master* occorre inserire la password del creatore del progetto che assicura il massimo dei diritti di accesso.

*Salva posizione finestre in locale*
Nel caso di utilizzo multiutente di uno stesso progetto, è possibile istruire il programma affinchè le impostazioni relative, ad esempio, alla posizione delle varie finestre venga salvato sul disco locale di ogni utente in modo che ognuno possa impostarle a suo piacimento senza interferire con gli altri.

*Profili Utente*
E' possibile creare un numero illimitato di profili utente, assegnado ad ognuno una password e i diritti di eseguire le varie operazioni.



## *Nuovo Database*

Questo modulo consente di aggiungere al progetto un nuovo database. E' possibile utilizzare un Alias o selezionare il percorso di una directory. Se viene specificato un Alias occorre inserire anche nome utente e password.

## *Creazione Moduli Tabella*

Questo modulo consente di selezionare le tabelle di un database per le quali creare automaticamente i moduli di visualizzazione.

## *Nuova Tabella*

Tramite questo modulo è possibile specificare una tabella per cui si desidera creare un modulo di visualizzazione.

## *Creazione Guidata Query*

Chi non ha molta dimestichezza con il linguaggio SQL può utilizzare il Wizard che consente di creare delle viste SQL relativamente semplici in quanto è possibile selezionare dati da una singola tabella.
Query più complesse possono comunque essere create utilizzando le modalità avanzate.

Le modalità disponibili sono:

- Tramite Wizard
- Definizione Libera
- Query Libera

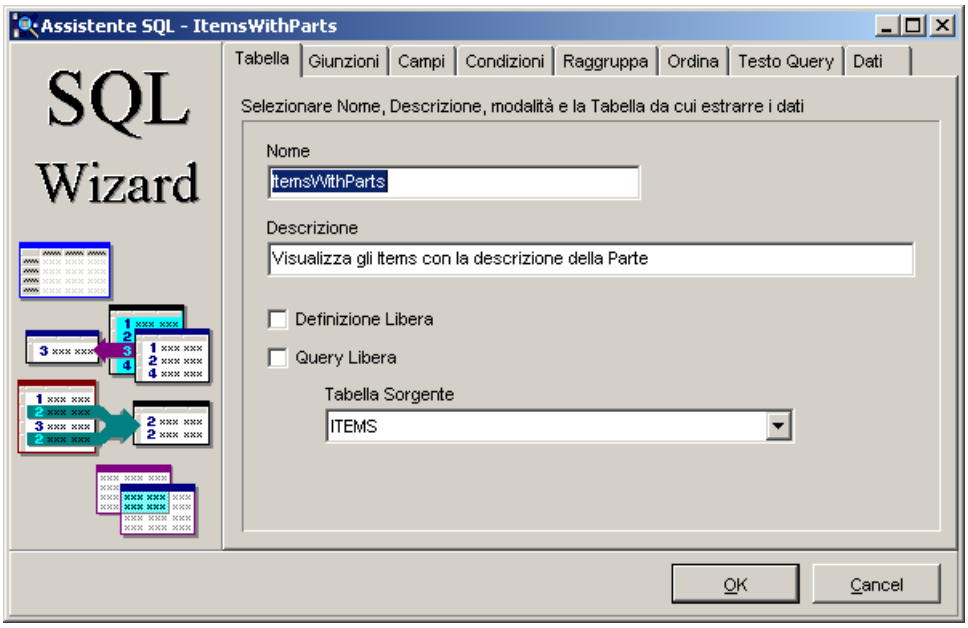
A seconda della modalità selezionata le pagine del wizard vengono visualizzate o nascoste. La correttezza formale della query viene controllata e il risultato della selezione dei dati viene visualizzato in tempo reale nella griglia da cui è stato lanciato il wizard in modo che si possa modificare la query per ottenere il risultato desiderato.

N.B. Con alcuni Database, le date nelle Condizioni devono essere espresse nella forma americana: MM/DD/YYYY

### *Tramite Wizard*

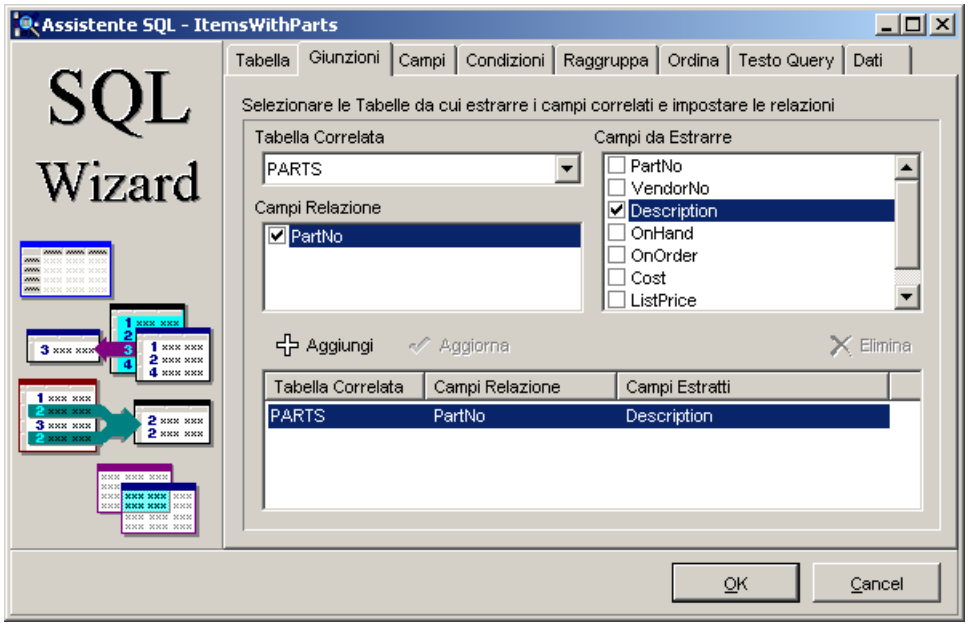
Il wizard è composto da 7 pagine in cui occorre selezionare i parametri della selezione e nell'ottava verranno visualizzati i dati estratti:

**Tabella**: selezione della tabella principale dall'elenco delle tabelle del database selezionato

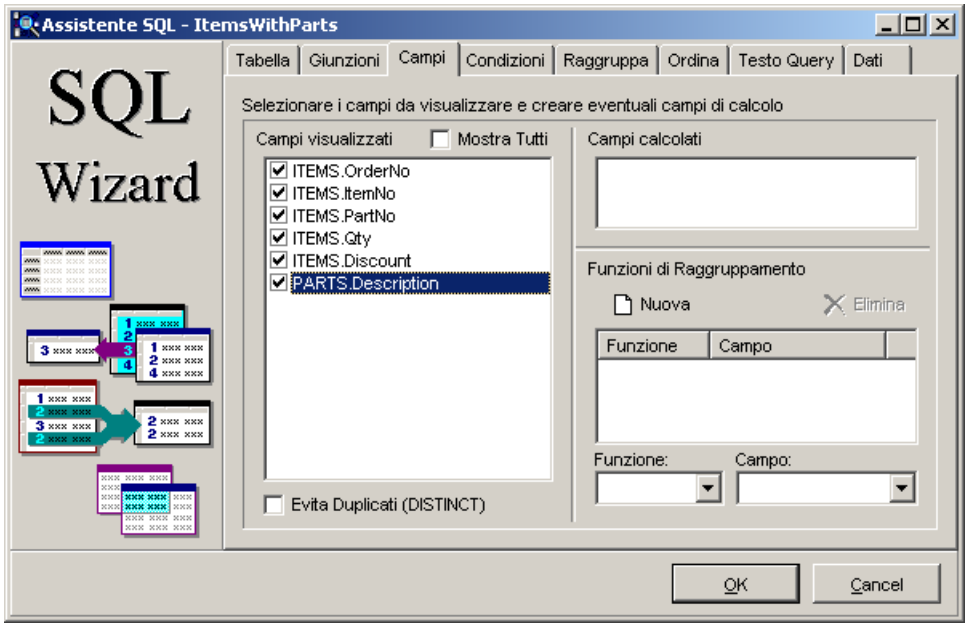



**Giunzioni**: Definizione dei JOIN con la tabella principale. Occorre specificare:

- Tabella Correlata*: tabella da cui estrarre i campi aggiuntivi*
- Campi Relazione*: i campi che definiscono la relazione*
- Campi da Estrarre*: i campi che dovranno essere estratti dalla tabella correlata*



**Campi**: selezione dei campi, creazione di campi calcolati e di campi di aggregazione




**Condizioni**: inserimento delle condizioni. Una condizione viene definita da:

- Campo*: viene selezionato dall'elenco dei campi della tabella selezionata*
- Operatore*: >, <, =, >=, <=, IS, LIKE*
- *Valore*: oltre ai valori di tipo numerico o stringa si possono usare dei valori speciali:

  NULL: da usare in associazione con l'operatore IS
  NOT NULL: da usare in associazione con l'operatore IS
  [RICHIEDI VALORE]: all'apertura della query verra richiesto il valore
  [RICHIEDI DATA]: all'apertura della query verra richiesto la data
  [RICHIEDI ORA]: all'apertura della query verra richiesto l'ora
  [GIORNO ATTUALE]: verrà utilizzata sempre la data del giorno in cui avviene l'apertura
  [ORA ATTUALE]: verrà utilizzata sempre l'ora al momento dell'apertura
  [GIORNO/ORA ATTUALE]: verrà utilizzata sempre la data/ora dell'apertura
  [DATA DI IERI]: all'apertura verrà utilizzata sempre la data del giorno precedente
  [INIZIO MESE IN CORSO]: verrà utilizzato sempre l'inizio del mese in corso
  [INIZIO ANNO IN CORSO]: verrà utilizzato sempre l'inizio del'anno in corso
  [DT INIZIO DEFAULT]: da utilizzare nei moduli d'analisi custom con il selettofre date
  [DT FINE DEFAULT]: da utilizzare nei moduli d'analisi custom con il selettofre date.

**Raggruppa**: selezione dei campi in base ai quali raggruppare

**Ordina**: selezione dei campi in base ai quali ordinare

**Testo Query**: visualizza il testo generato in automatico

### *Definizione Libera*

La definizione libera propone uno schema della query composto da cinque sezioni corrispondenti alla clausole principali di una query di selezione. In ogni sezione è sufficiente inserire la lista dei campi o tabelle o condizioni e la query verrà costruita in automatico.

Esempio:

**[SELECT]**
CustNo
Count(OrderNo) As Count_OrderNo
**[FROM]**
ORDERS
**[WHERE]**
PaymentMethod = 'Visa'
Terms = '30'
**[GROUP BY]**
CustNo
**[ORDER BY]**
CustNo

### *Query Libera*

In questo caso si ha a disposizione un semplice editor di testo in cui inserite la query e ovviamente richiede una conoscenza di base del linguaggio SQL.

Esempio: SELECT **CustNo**, COUNT(**OrderNo**) As **Count_OrderNo**
FROM **ORDERS**
WHERE (**PaymentMethod** = 'Visa' AND **Terms** = '30')
GROUP BY **CustNo**
ORDER BY **CustNo**

**Nota sulle condizioni nelle definizioni libere**

Nelle modalità Definizione libera e Query Libera è possibile condizionare la selezione facendo riferimento a valori che verranno richiesti al momento dell'apertura o ad un intervallo temporale che verrà impostato con il selettore relativo nei moduli d'analisi.
Per richiedere un valore all'apertura i parametri nelle Query devono iniziare con un carattere che ne indica il tipo:

'i' -> Integer
'f' -> Float
'd' -> Date
't' -> Time
'k' -> DateTime

Esempio: Select * from ORDLAV
where QUANTITAPREVISTA > **:iParmQtPrev**
And DATAINIZIOEFF > **:dParmDTInizio**

Per fare riferimento alle date d'inizio e di fine del selettore occorre utilizzare i valori speciali:
dDefaultDTFrom: Il parametro per la Data Inizio di default
dDefaultDTTo: Il parametro per la Data Fine di default

Esempio: Select * from ORDLAV
where DATAORDINE > **dDefaultDTFrom**
And DATAORDINE < **dDefaultDTTo**

# Impostazioni Tabelle e Viste SQL

## Editor dei filtri Personali

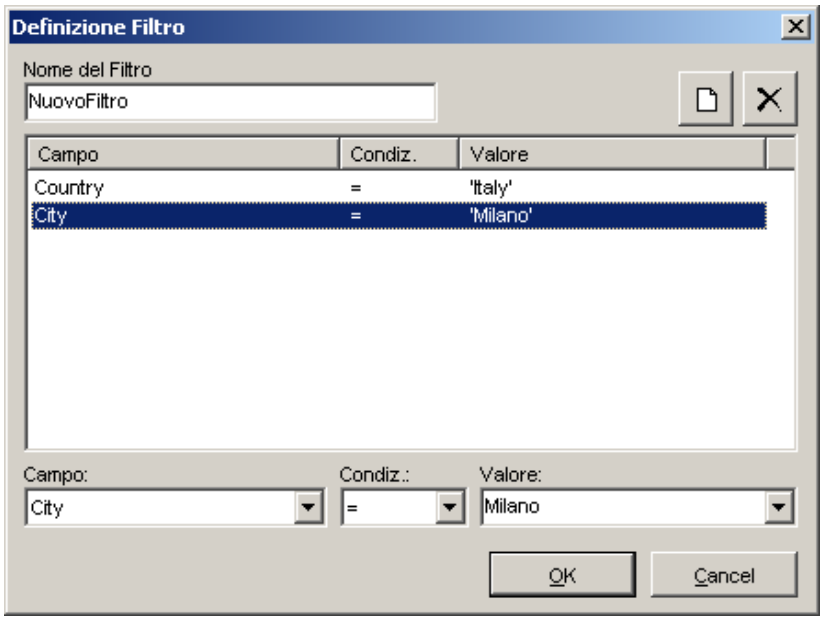
Questo editor consente di definire un filtro per la selezione dei record di una tabella o vista.
Per creare un nuovo filtro occorre:

- Specificare un nome (univoco all'interno del modulo)
- Aggiungere il numero di condizioni desiderato
- Definire le condizioni selezionando:
  - Nome del campo
  - Operatore di confronto
  - Valore di confronto

Tra i valori di confronto è possibile specificare i seguenti valori speciali:

| | |
|---|---|
| [RICHIEDI VALORE] | All'attivazione del filtro verrà richiesto il valore |
| [RICHIEDI DATA] | All'attivazione del filtro verrà richiesta la data |
| [RICHIEDI ORA] | All'attivazione del filtro verrà richiesto l'ora |
| [GIORNO ATTUALE] | Verrà utilizzata la data "*odierna*" |
| [ORA ATTUALE] | Verrà utilizzata l'ora attuale al momento dell'attivazione |
| [GIORNO/ORA ATTUALE] | Verrà utilizzata data e ora attuale |
| [DATA DI IERI] | Verrà utilizzata la data del giorno precedente |
| [INIZIO MESE IN CORSO] | Verrà utilizzata la data dell'inizio del mese in corso |
| [INIZIO ANNO IN CORSO] | Verrà utilizzata la data dell'inizio dell'anno in corso |

N.B. Con alcuni Database, le date devono essere espresse nella forma: MM/DD/YYYY




## Definizione Collegamento

Tramite questo modulo vengono definiti i collegamenti di tipo Master / Detail tra i moduli Tabella.
Per definire il collegamnento occorre selezionare il modulo di Dettaglio tra quelli aperti (occorre quindi aprire preventivamente i moduli desiderati) e i campi che devono essere utilizzati come chiave.

# *Editor Schede Dati*

Quando si decide di creare o modificare un modulo per visualizzare una scheda dati, viene aperto il modulo e l'editor per la definizione degli elementi necessari.
Le schede dati possono essere utilizzate per visualizzare o modificare i dati contenuti in una tabella. La disposizione dei campi d'immissione può essere completamente personalizzata.
Ogni scheda dati contiene una barra di navigazione per spostarsi sul record desiderato.
L'editor è suddiviso in tre parti:

- Barra Oggetti
- Albero Oggetti
- Pannello Proprietà

Tramite la Barra Oggetti è possibile creare sul modulo i seguenti elementi:

Accesso ai dati

- Query SQL: per la selezione dei dati da visualizzare o modificare
- DataSource: usata per collegare le query ai vari controlli visuali

Visualizzazione/Modifica

- Label:
  Etichetta usata ad esempio per i nomi dei campi

- DBText:
  Etichetta per visualizzare i valori di un campo

- DBEdit:
  casella di testo per la modifica di un campo

- DBMemo:
  per la modifica di un campo di timo Memo

- DBCheckBox:
  per la modifica dei campi di tipo Booleano

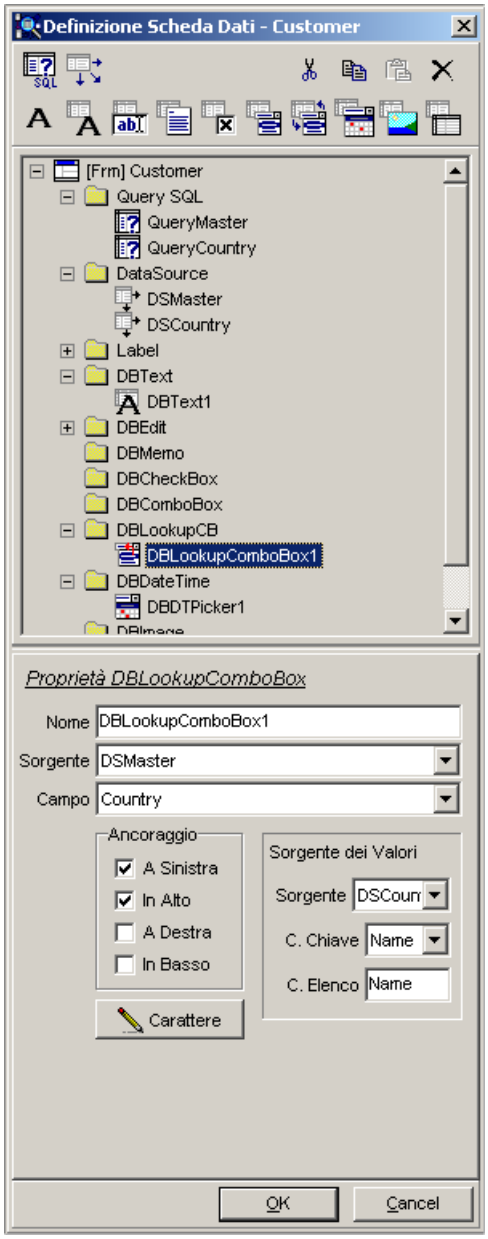
- DBComboBox:
  per l'impostazione di un valore preso da una lista

- DBLookupCB:
  per l'impostazione di un valore preso da un campo di un'altra tabella

- DBDateTime:
  per la modifica di un campo di tipo Data / Ora

- DBImage:
  per la visualizzazione di un'immagine

- DBGrid:
  griglia dati



# *Editor Analisi Personalizzate*

Quando si decide di creare o modificare un modulo per le analisi personalizzate, viene aperto il modulo e l'editor per la definizione degli elementi necessari.
Le analisi personalizzate consentono di definire dei moduli con varie sorgenti dati provenienti anche da database multipli. I dati possono essere visualizzati tramite griglie e grafici. Sui dati è possibile effettuare calcoli creando dei dataset risultato; tali dataset possono essere utilizzati per calcoli ulteriori. Tramite un'analisi personalizzata è possibile generare campi calcolati, effettuare statistiche, unire due o più dataset in un'unica tabella e tali operazioni possono essere reiterate sulle tabelle risultanti.

E' anche possibile definire delle analisi "standard" da utilizzare all'interno dell'applicazione facendo riferimento a sorgenti dati su altri moduli.
In questo caso verrà visualizzato un selettore che propone la lista delle sorgenti dati disponibili sugli altri moduli
E' disponibile anche un selettore per la definizione di un intervallo temporale: tale intervallo può essere utilizzato per condizionare le query di selezione dei dati.

L'editor è suddiviso in tre parti:

- Barra Oggetti
- Albero Oggetti
- Pannello Proprietà

Tramite la Barra Oggetti è possibile creare sul modulo i seguenti elementi:

### *Visuali*

- Pannello:
  contenitore per gli altri elementi

- Griglia:
  per la visualizzazione dei dati

- Grafico:
  per la creazione di grafici sui dati

### *Di accesso ai dati*

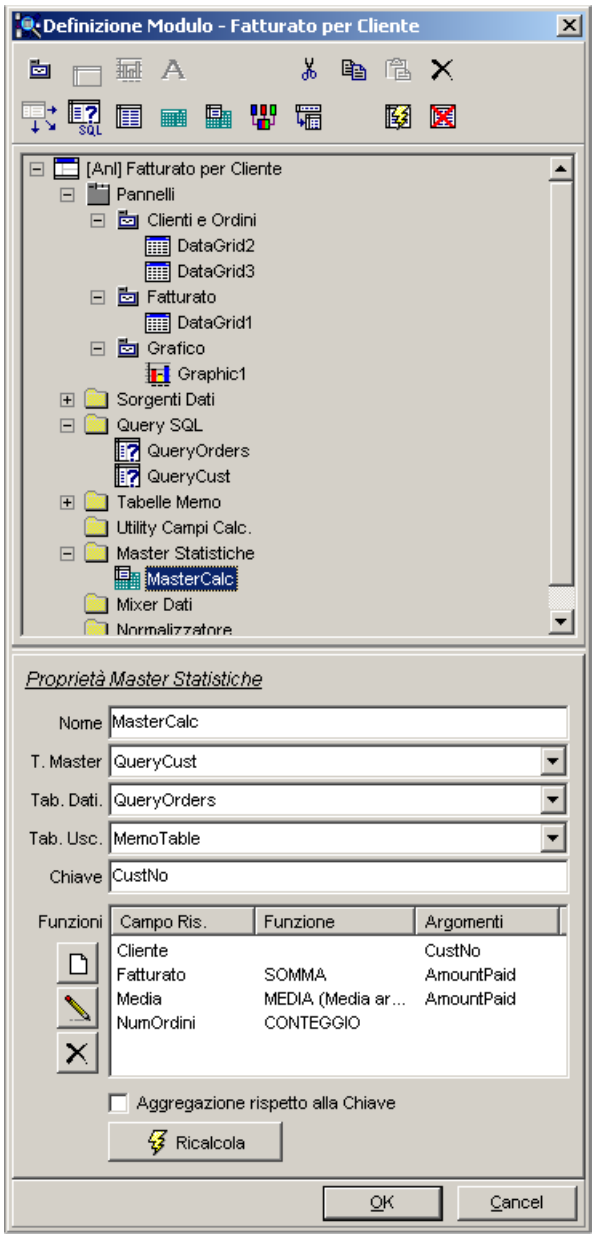
- Sorgente dati:
  che collega i dataset agli elementi visuali

- Query:
  per l'estrazione dei dati fisici

- Tabelle Memo:
  per contenere i dati calcolati

### *Di calcolo*

- Campi Calcolati
- Master Statistiche
- Mixer Dati
- Normalizzatore Record



L'albero degli Oggetti visualizza tutti gli elementi che sono stati creati sul modulo, consentendo di selezionarli e visualizzare le loro proprietà sull'apposito pannello.

*Note Operative*

- Le pagine "contenitore" possono essere di vari tipi
- Per posizionare griglie e grafici occorre clickare sull'area di desiderata una pagina.
- Le proprietà di una griglia possono essere impostate tramite l'apposito editor.

## Proprietà elementi di calcolo

### *Campi Calcolati*

Questo componente viene utilizzato per creare un dataset contenente campi provenienti da un altro dataset e campi calcolati. Supponiamo di avere la tabella degli Ordini con i campi **Cliente, Ordine, Importo, AliquotaIVA, %Sconto**, con questo componente possiamo creare un dataset con i campi Cliente, Ordine, Importo, ImportoIVA e ImportoSconto.

Di seguito sono riportate le proprietà che occorre impostare per utilizzare questo componente.

- Tabella Ingresso: Dataset dei dati d'igresso
- Tabella Uscita: Tabella memo in cui verranno inseriti i dati calcolati
- Campii: elenco dei campi da inserire nella tabella di uscita

### *Master Statistiche*

Questo componente consente di eseguire dei calcoli come la somma o la media dei valori di un campo di record di dettaglio di un dataset master.
Ad esempio supponiamo di avere la tabella dei Clienti e la tabella degli ordini: utilizzando questo componente possiamo creare una tabella contenente i campi Cliente, NumeroOrdini, TotaleOrdini.

In pratica per ogni cliente il componente seleziona i relativi ordini, ne calcola il numero e l'ammontare complessivo. La tabella risultante può, ad esempio, essere utilizzata per creare un grafico.Il dataset master deve essere collegato al dataset detail in modo che scrollando i suoi record, il dataset detail venga filtrato in base alla chiave.
In alternativa, se ad esempio si avesse soltanto la tabella degli ordini, è possibile ottenere lo stesso risultato aggregando i record in base alla Chiave Cliente.

Questa modalità equivale in pratica ad un'istruzione SQL del tipo:

Select **Cliente**, Count(**Ordine**), Sum(**ImportoOrdine**)
From **Ordini**
Group By **Cliente**

Di seguito sono riportate le proprietà che occorre impostare per utilizzare questo componente.

- Tabella Master: Dataset Master dei dati d'igresso
- Tabella Dati: Dataset Detail dei dati d'ingresso
- Tabella Uscita: Tabella memo in cui verranno inseriti i dati calcolati
- Chiave: Campo chiave del dataset master
- Funzioni: elenco delle funzioni da applicare ai record detail per ogni record Master.

### *Mixer Dati*

Supponiamo di avere due dataset contenenti rispettivamente i campi: Cliente, TotOrdini1999 e Cliente, TotOrdini2000 e di voler ottenere un dataset con tutti e tre i campi per poter creare un grafico di confronto.
Se i dataset sono tabelle dello stesso database, sarebbe sufficiente una selezione SQL (left join) ma se, ad esempio si tratta di tabelle memo provenienti da calcoli precedenti possiamo utilizzare questo componente.

E' possibile selezionare al massimo quattro dataset che devono avere almeno un campo in comune da usare come campo chiave. Per ogni dataset si possono selezionare un numero qualunque di campi tramite l'apposito Editor.
Il componente può essere utilizzato anche per effettuare dei LookUp invece che dei Join.

### *Normalizzatore Record*

Può capitare di eseguire analisi su database che non sono stati progettati utilizzando la forma normale prevista per i DB relazionali: questo componente è in grado di "normalzzare" i record di una tabella.
Vediamo un esempio per chiarire la sua utilità.

Abbiamo una tabella che contiene i dati relativi al fatturato mensile delle varie filiali di una grande azienda. Normalmente una tabella con dati simili dovrebbe avere una struttura del tipo:

FILIALE
MESE
FATTURATO

La nostra tabella invece ha la seguente struttura:

FILIALE
FATT_GENNAIO
FATT_FEBBRAIO
.....
FATT_DICEMBRE

Supponiamo di dover creare un grafico a barre relativo al fatturato mensile, con una barra per ogni mese.

Il componente è in grado di prendere in ingresso i record della tabella e creare in uscita una tabella con gli stessi dati ma organizzati secondo la forma normale.

E' possibile eseguire la conversione dell'intera tabella in ingresso, oppure del solo record attivo.

I record normalizzati possono essere utilizzati per la creazione del grafico.

## Selezione Pannello

Consente di selezionare il tipo di pannello da inserire nel modulo. Sono disponibili i seguenti tipi di pannello

- Pannello Semplice
- Pannello Doppio: con due sezioni in verticale
- Pannello Triplo: con tre sezioni in verticale
- Pannello a Y: con due sezioni in alto e una in basso
- Pannello a T: con una sezione in alto e due in basso
- Pannello a K: con una sezione a sinistra e due a destra
- Pannello a X: con quattro sezioni

## Parametri Campi Calcolati

Ogni campo calcolato viene definito utilizzando le seguenti proprietà:

- Nome: cioè il nome del nuovo campo calcolato.
- Funzione: può essere il singolo nome di un campo o una qualunque espressione matematica.
- Tipo dati: occorre specificarlo ad esempio per ottenere deii campi data o ora.
- Conversione: ad esempio per arrotondare il risultato

## Parametri Statistiche

Ogni funzione statistica viene definita utilizzando le seguenti proprietà:

- Nome: cioè il nome del nuovo campo calcolato.
- Funzione: una qualunque tra le funzioni statistiche disponibili nell'elenco.
- Argomenti: possono essere un singolo nome di un campo o una qualunque espressione matematica.
- Tipo dati: occorre specificarlo ad esempio per ottenere deii campi data o ora.
- Conversione: ad esempio per arrotondare il risultato

## Parametri Grafico

Per creare un grafico occorre impostare le seguenti proprietà:

- Tabella/Query: il dataset da cui prelevare i dati
- Tipo Grafico: è possibile creare grafici lineari, a barre, a torta ecc.
- Campo Entità: definisce il campo sull'asse delle X
- Campo Val. 1: definisce il campo sell'asse delle Y
- Campo Val. 2: da utilizzare con il tipo grafico Gantt per definire l'intervallo dei valori (Val1 -> Val2)
- Stile Etichette: Valore, Percentuale ecc.
- Colore: il colore del grafico

## Selezione Campi

Tramite questo modulo è possibile selezionare un numero qualsiasi di campi di una tabella. Per effettuare la selezione occorre agire sulle caselle di spunta alla sinistra del nome dei campi.

# *Editor Monitor RT*

Quando si decide di creare o modificare un Monitor RT, viene aperto il modulo e l'editor per la definizione degli elementi necessari.
Gli elementi fondamentali di un Monitor, sono gli indicatori che devono essere associati ad un segnale; tale segnale verrà utilizzato per prelevare i dati da visualizzare.
L'editor è suddiviso in tre parti:

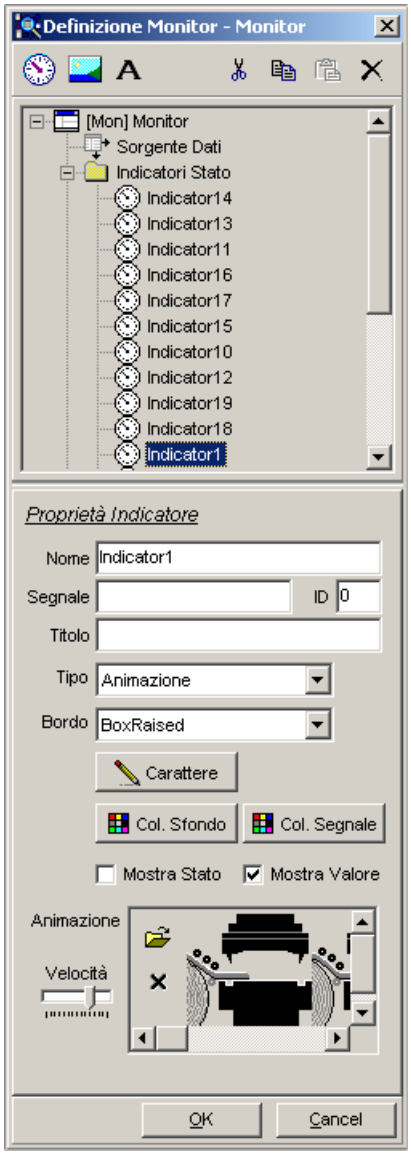
- Barra Oggetti
- Albero Oggetti
- Pannello Proprietà

Tramite la Barra Oggetti è possibile creare sul modulo i seguenti elementi:

- Indicatori
- Etichette
- Immagini

Sono disponibili i seguenti tipi di Indicatore:

- Animazione
- Livello
- Valore
- Tachimetro
- Immagine di Stato
- Led Colorato
- Grafico
- Termometro
- LCD Digitale
- Livello Verticale
- Semicircolare




Per associare ad un indicatore un determinato segnale, sono disponibili due proprietà:

- Segnale di tipo alfanumerico
- ID di tipo intero

Nel caso i dati si trovino in una tabella, le proprietà di cui sopra verranno usate come chiave per individuare un record.
La Tabella deve avere campi associabili a:

- Segnale
- ID (opzionale)
- Status (LV, AT, FE, PA....)
- Valore

Se i dati verranno letti tramite una DLL esterna, le proprietà verranno passate alla funzione di lettura e sarà compito della DLL interpretarle correttamente.
La DLL deve esporre le seguenti funzioni:

- InitMonitor()
- GetStatus(Var Data: TMonitorRecord)
- StopMonitor()

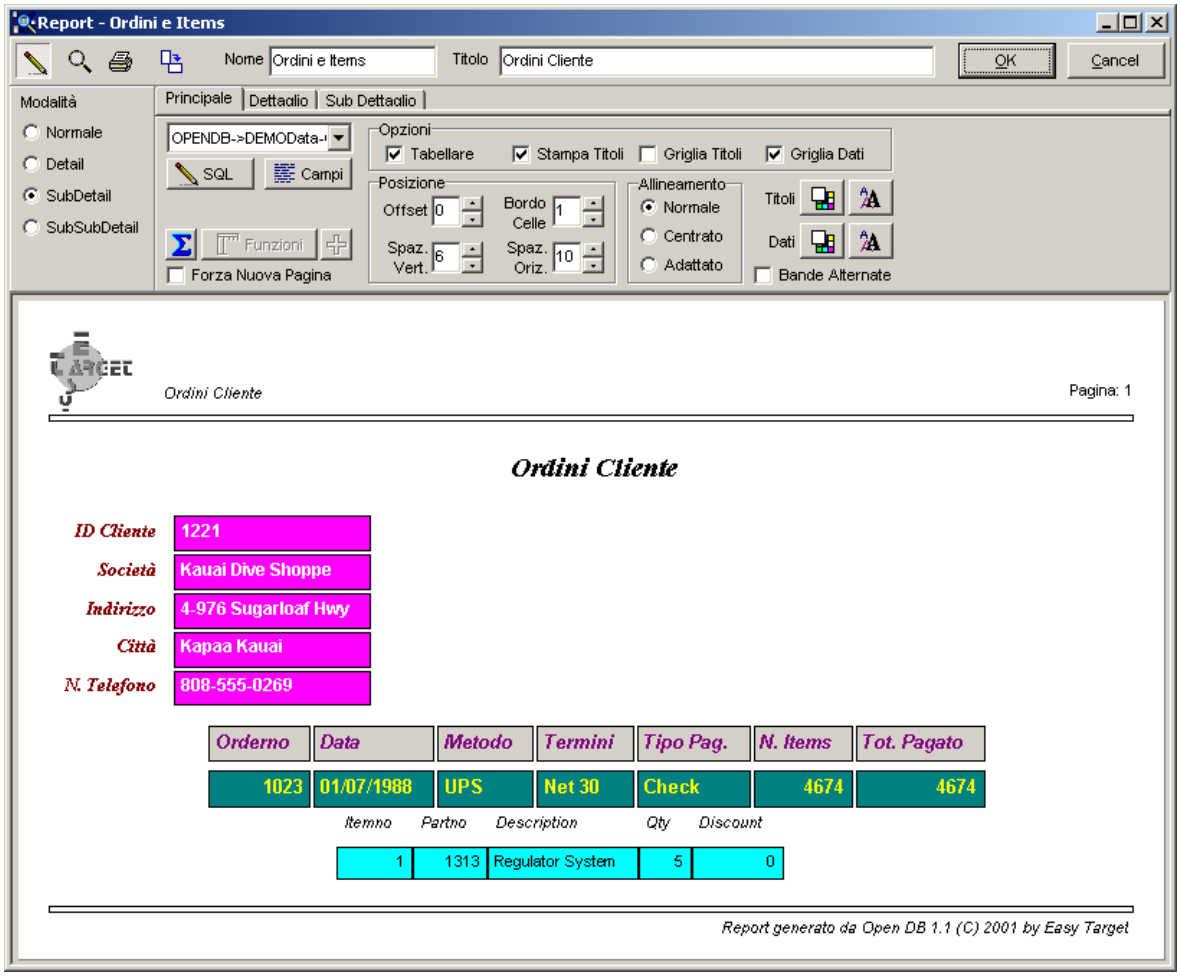
# *Creazione Report*

L'idea di base di un report Master/Detail è di selezionare una o più tabelle che abbiano un legame logico come ad esempio una tabella di Clienti e la relativa tabella degli Ordini, per poi stampare in sequenza ogni record Master seguito dai relativi record di Dettaglio.

Questo modello è applicabile ad una grande quantità di casi ed è possibile automatizzare molte delle operazioni necessarie per la definizione di ogni report.
Mentre con i normali tool di creazione di un report, di solito occorre creare le varie "sezioni" e poi disporre i vari campi manualmente, con questo editor è sufficiente definire le sorgenti dei dati ed i parametri visuali.

La finestra di dialogo per la creazione di un report consente di scegliere il tipo di report tra i seguenti:

- Semplice: Con una sorgente dati
- Master/Detail: Con due sorgenti dati
- Master/Detail Con Sub Detail: Con tre sorgenti dati
- Master/Detail Con Sub Detail e SubSub Detail: Con quattro sorgenti dati



Per ogni livello occorre specificare:

- DataBase
- Selezione SQL
- Campo Indice Dettaglio
- Campi da stampare con eventualmente i titoli per le colonne
- Font per i titoli delle colonne
- Font per i dati
- Offset (spostamento da sinistra)
- Spaziatura Verticale (distanza tra le righe)
- Spaziatura orizzontale (distanza tra i campi)
- Bordo Celle (utile se si stampa la griglia)
- Stampa in formato scheda
- Stampa i titoli
- Stampa la Griglia sui titoli
- Stampa la griglia sui dati
- Usa titoli colonne abbreviati (occorre creare un apposito file)
- Modalità (Normale, Centrato, Adattato)

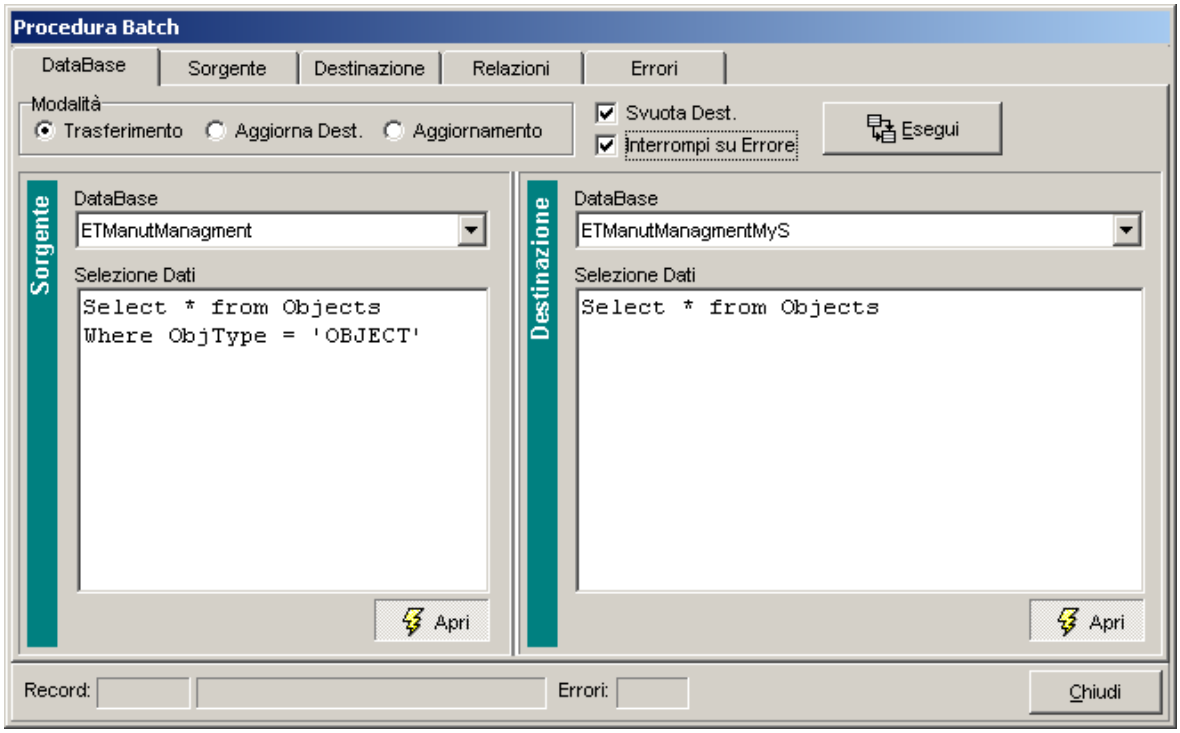
Le opzioni disponibili sono:

- cambio pagina al termine della stampa del record
- Bande colorate alternate per una maggiore leggibilità

# *Procedure Batch*

Questo modulo è un'utility particolarmente utile agli sviluppatori che necessitano di trasferire dati da una vecchia base dati ad una nuova, come succede ad esempio durante lo sviluppo di una nuova applicazione che va a sostituirne una preesistente. Consente di:

- Trasferire dati tra due tabelle qualsiasi anche in database diversi
- Aggiornare la tabella di destinazione con dati provenienti dalla tabella sorgente
- Modificare i dati contenuti in una tabella.




La peculiarità principale del modulo è la capacità di applicare delle funzioni di trasferimento ai dati da trasferire.
Ad esempio è possibile concatenare stringhe provenienti da due campi in uno solo o convertire un campo valuta da Lire a Euro.
La funzione di trasferimento può essere condizionata da situazioni particolari come il valore contenuto nel campo destinazione.

## *Sorgenti Dati*

Tramite il pannello [Database], vengono selezionate i dataset da utilizzare come sorgente e destinazione.

E' possibile selezionare la modalità operativa e alcune opzioni. I dataset vengono definiti con una query SQL.

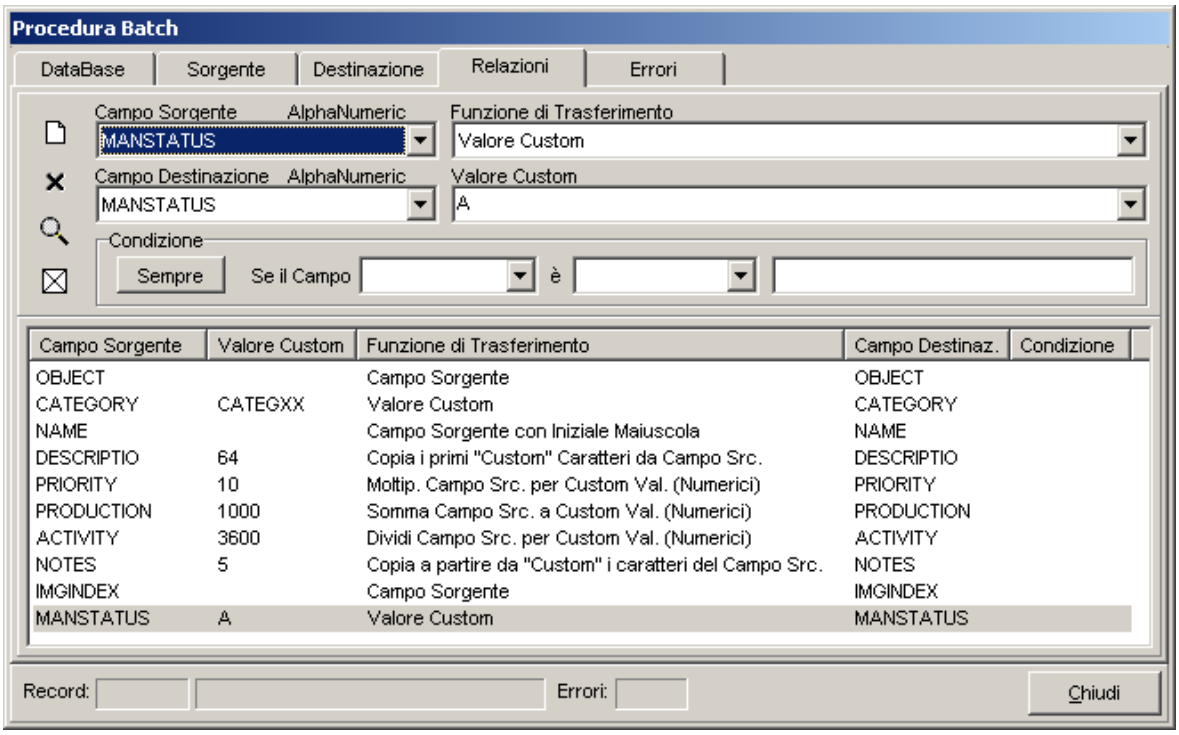
Dopo l'apertura delle query è possibile modificare direttamente i dati utilizzando le relative griglie sui pannelli [Sorgente] e [Destinazione].

### *Relazioni di trasferimento*

Tramite il pannello [Relazioni], vengono definite le funzioni di trasferimento dei vari campi. Usando il pulsante [Lente], vengono create le relazioni di default per un trasferimento semplice dei dati contenuti nei campi con nome uguale.

Il trasferimento può comunque essere eseguito tra campi con nome diverso. E' possibile anche eseguirlo tra campi di tipo diverso purchè siano compatibili. Ad esempio è possibile trasferire:

- campi numerici in campi stringa
- campi stringa in campi numerici se i campi sorgente rappresentano valori come "1234".




Ogni relazione è definita dai seguenti elementi:

- Campo Sorgente
- Campo Destinazione
- Funzione di Trasferimento
- Eventuale Valore Custom
- Condizione per eseguire il trasferimento

Le funzioni disponibili sono le seguenti:

- Campo Sorgente
- Valore Custom
- Somma Campo Sorgente a Valore Custom (Stringhe)
- Somma Custom Val. a Campo Sorgente (Stringhe)
- Somma Campo Sorgente a Valore Custom (Numerici)
- Sottrai Campo Sorgente da Valore Custom (Numerici)
- Sottrai Valore Custom da Campo Sorgente (Numerici)
- Moltip. Campo Sorgente per Valore Custom (Numerici)
- Dividi Campo Sorgente per Valore Custom (Numerici)
- Dividi Valore Custom per Campo Sorgente (Numerici)
- Somma Campo Sorgente a Campo con nome "Custom" (Stringhe)
- Somma Campo con nome "Custom" a Campo Sorgente (Stringhe)
- Somma Campo Sorgente a Campo con nome "Custom" (Numerici)
- Arrotonda Campo Sorgente alla "Custom" cifra Decimale
- Tronca Campo Sorgente alla "Custom" cifra Decimale
- Copia i primi "Custom" Caratteri da Campo Sorgente
- Copia a partire da "Custom" i caratteri del Campo Sorgente
- Campo Sorgente Tutto MAIUSCOLO
- Campo Sorgente Tutto minuscolo
- Campo Sorgente con Iniziale Maiuscola
- Auto Incrementante con Valore Iniziale = "Custom"
- Da tipo Data/Ora a tipo Stringa con formato "Custom"
- Da tipo Stringa con formato "Custom" a tipo Data/Ora
- Da tipo Numerico AAMG a tipo Data
- Da tipo Numerico HMS a tipo Ora
- Da tipo Numerico a tipo stringa di lunghezza "Custom"
- Giorno della settimana (1..7) della data Sorgente
- [DATA] di tipo Data/Ora
- [ORA] di tipo Data/Ora
- [DATA e ORA] di tipo Data/Ora
- [DATA/ORA] di tipo Stringa con formato "Custom"

E' possibile condizionare il trasferimento ai valori contenuti nel campo sorgente o destinazione. Ad esempio si può definire un trasferimento come il seguente:

Poni "VERO" nel campo destinazione [FLAG] se il campo sorgente [PRESENTE] contiene il valore <True>.
Poni "FALSO" nel campo destinazione [FLAG] se il campo sorgente [PRESENTE] contiene il valore <False>.
Poni il valore del campo sorg. [COD] nel campo destinazione [CODICE] se esso contiene il valore <0>.

Le condizioni possono essere definite utilizzando i seguenti operatori:

- uguale a
- diverso da
- maggiore di
- minore di
- Vuoto
- Contenente
- Contenuto in

## *Definizione Schemi*

Per creare un nuovo schema è sufficiente selezionare nel pannello di progetto la cartella degli schemi, attivare il menu popup e selezionare la voce [Nuovo].
Verrà richiesto di inserire nome e descrizione per il nuovo schema e dopo aver confermato verrà salvato lo schema delle finestre che saranno state aperte fino a quel momento. Un messaggio avviserà dell'avvenuto salvataggio.

## *Nuovo Modulo Externo*

Questo modulo consente di aggiungere al progetto un modulo esterno che può essere un eseguibile o una DLL.
E' possibile specificare la modalità con cui verrà mantenuto il collegamento:

- Percorso relativo a quello del progetto
- Percorso relativo a quello dell'applicazione
- Percorso assoluto

Per utilizzare i primi due, il modulo esterno deve trovarsi nella stessa directory in cui si trovano il progetto o l'applicazione o in una sttodirectory della stessa. E' una modalità utile se occorre distribuire il progetto con i moduli esterni senza conoscere a priori la directory in cui verrà installato.

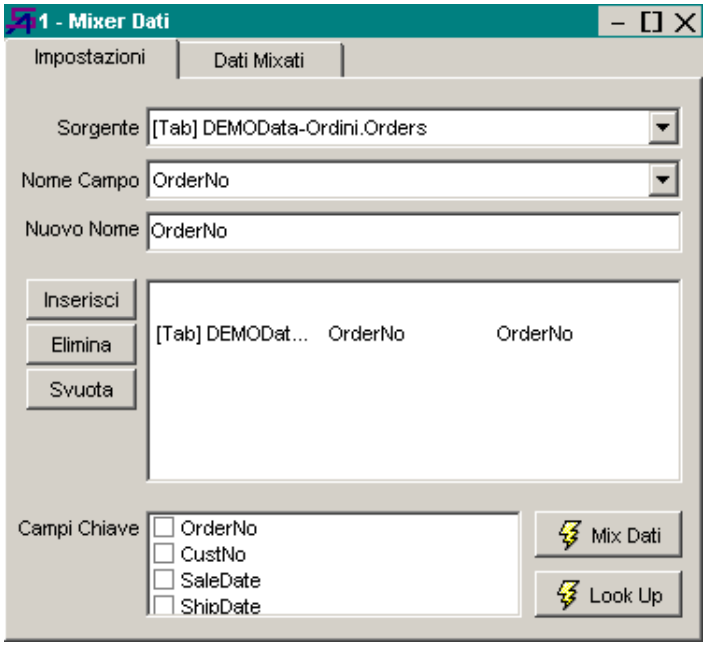
Nel caso si stia collegando una DLL, occorre specificare il nome della funzione da chiamare.

# Capitolo 6

# *Moduli Speciali*

## *Mixer Dati*

Questo modulo può essere utilizzato per generare dei dataset che contengono dati provenienti da vari dataset esterni.




Per definire i campi da inserire nel nuovo Dataset, occorre specificare i seguenti parametri:

- Sorgente dati (da cui leggere i dati di partenza)
- Nome del Campo (cioè il nome del campo sorgente)
- Nuovo Nome (cioè il nome da usare nel dataset risultante)
- Conversione (opzionale)

Dopo l'impostazione dei vari parametri, premendo il pulsante [Inserisci] il nuovo campo verrà aggiunto al dataset risultante.
Per eseguire il mixaggio occorre selezionare i campi chiave dall'apposito elenco.

Il mixaggio puo essere di due tipi:

- Unione 1 a 1
- LookUp

### *Unione 1 a 1*

Supponiamo di avere aperto due tabelle rispettivamente con i campi
[Cliente, Fatturato98] e [Cliente, Fatturato99]
e di voler ottenere un unico dataset con i campi
[Cliente, Fatturato98, Fatturato99]
per poter effettuare dei confronti o generare un grafico.

Esiste una relazione [1 a 1] tra i clienti ed il fatturato relativo.

E' necessario inserire i seguenti campi:

| ***Sorgente*** | ***Campo*** | ***Nuovo Nome*** |
|---|---|---|
| Tabella1 | Fatturato98 | Fatt1998 |
| Tabella2 | Fatturato99 | Fatt1999 |

Al termina dell'inserimento occorre selezionare il campo chiave [Cliente].

Dopo aver completato la definizione, premendo il pulsante [Mix. Dati] viene generato il dataset risultante.
Verranno inseriti soltanto i record la cui chiave esiste in entrambe le tabelle.

### *LookUp*

Supponiamo di avere aperto due tabelle rispettivamente con i campi
[Cliente, IDOrdine] e [IDOrdine, DescOrdine]
e di voler ottenere un unico dataset con i campi
[Cliente, IDOrdine, DescOrdine]
per poter stampare un Report.

Esiste una relazione [1 a molti] tra i Clienti e gli Ordini.

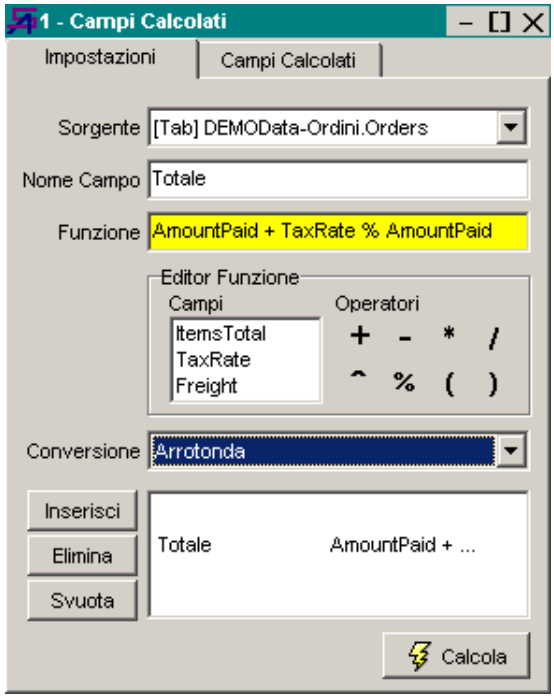
E' necessario inserire i seguenti campi:

| ***Sorgente*** | ***Campo*** | ***Nuovo Nome*** |
|---|---|---|
| Tabella1 | IDOrdine | Ordine |
| Tabella2 | DescOrdine | Descrizione |

Al termina dell'inserimento occorre selezionare il campo chiave [Cliente].
Dopo aver completato la definizione, premendo il pulsante [LookUp] viene generato il dataset risultante.
Verranno inseriti tutti i record della tabella1 con le descrizioni trovate in base alla chiave nella tabella2.

# *Campi Calcolati*

Questo modulo può essere utilizzato per generare dei dataset che includono i dati provenienti da altri moduli e dei campi calcolati.




Un esempio può chiarire meglio questa funzionalità.
Supponiamo di avere aperto una tabella con i campi
[NumeroFattura, DataFattura, Imponibile]
e di voler aggiungere i campi
[IVA, Totale].

Per definire i campi da inserire nel nuovo Dataset, occorre specificare i seguenti parametri:

- Sorgente dati (da cui leggere i dati di partenza)
- Nome del Campo (cioè il nome del campo risultante)
- Funzione (con cui calcolare il risultato)
- Conversione (opzionale)

La Sorgente deve essere la stessa per tutti i campi.

Per aggiungere un campo sorgente senza eseguire alcun calcolo ci sono due possibilità:

- Campo = [nome sorgente] / Funzione = “”
- Campo = [qualunque] / Funzione = [nome sorgente] (solo per campi numerici)

Per definire la Funzione da applicare si può utilizzare (ma è possibile scriverla direttamente) l’editor che fornisce il nome dei campi disponibili e gli operatori ammessi.
Nel caso fosse necessario utilizzare funzioni matematiche, si può utilizzare il Popup menù collegato con il campo di edit della funzione.
Per una panoramica sulle regole di frmattazione e un elenco delle Funzioni disponibili vedere Valutazione Espressioni.

Dopo l'impostazione dei vari parametri, premendo il pulsante [Inserisci] il nuovo campo verrà aggiunto al dataset risultante.

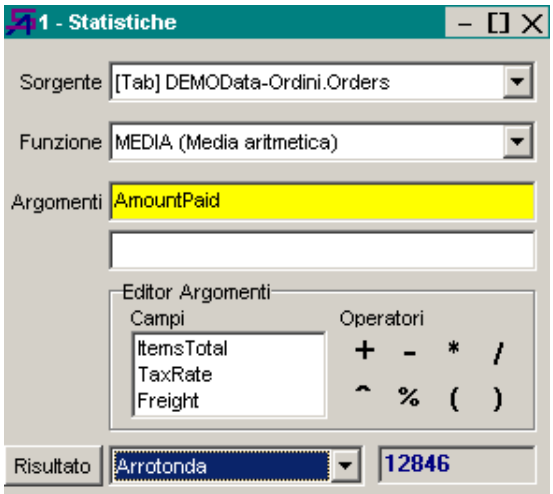
Tornando al nostro esempio, sarebbe necessario inserire i seguenti campi:

| ***Campo*** | ***Funzione*** |
|---|---|
| NumeroFattura | |
| DataFattura | |
| Imponibile | |
| IVA | 20 % Imponibile |
| Totale | Imponibile * 1.2 |

Dopo aver completato la definizione, premendo il pulsante [Calcola] viene generato il dataset risultante.

## *Statistiche*

Questo modulo può essere utilizzato per eseguire calcoli statistici sui dati contenuti in un dataset esterno.




Un esempio può chiarire meglio questa funzionalità.

Supponiamo di avere aperto una tabella con i campi
[NumeroFattura, DataFattura, Imponibile]
e di voler calcolare il totale Incassato.

Per eseguire il calcolo, occorre specificare i seguenti parametri:

- Sorgente dati (da cui leggere i dati di partenza)
- Funzione Statistica
- Argomento (Campo o funzione su cui applicare la statistica)
- Conversione (opzionale)

Le Funzioni Statistiche disponibili selezionabili dall'elenco sono:

- ASIMMETRIA
- CONTEGGIO
- CORRELAZIONE
- COVARIANZA
- CURTOSI
- DEV. QUAD. (Somma dei quadrati delle deviazioni)
- DEV. STAND. (Deviazione standard)
- MASSIMO
- MEDIA (Media aritmetica)
- MEDIA ARMON. (Media armonica)
- MEDIA DEV. (Media delle deviazioni assolute rispetto alla media)
- MEDIA DIST. (Media delle distanze)
- MEDIA GEOM. (Media geometrica)
- MINIMO
- PENDENZA
- PRODOTTO
- SOMMA
- VARIANZA

Per definire gli argomenti della Funzione, si può utilizzare (ma è possibile scrivere direttamente) l'editor che fornisce il nome dei campi disponibili e gli operatori ammessi.

Nel caso fosse necessario utilizzare funzioni matematiche, si può utilizzare il Popup menù collegato con il campo di edit della funzione. Per una panoramica sulle regole di frmattazione e un elenco delle Funzioni disponibili vedere Valutazione Espressioni.

Tornando al nostro esempio, sarebbe necessario specificare i seguenti parametri:

| ***Funzione*** | ***Argomento*** |
|---|---|
| Somma | 120 % Imponibile |

Dopo l'impostazione dei vari parametri, premendo il pulsante [Risultato] il calcolo viene eseguito.

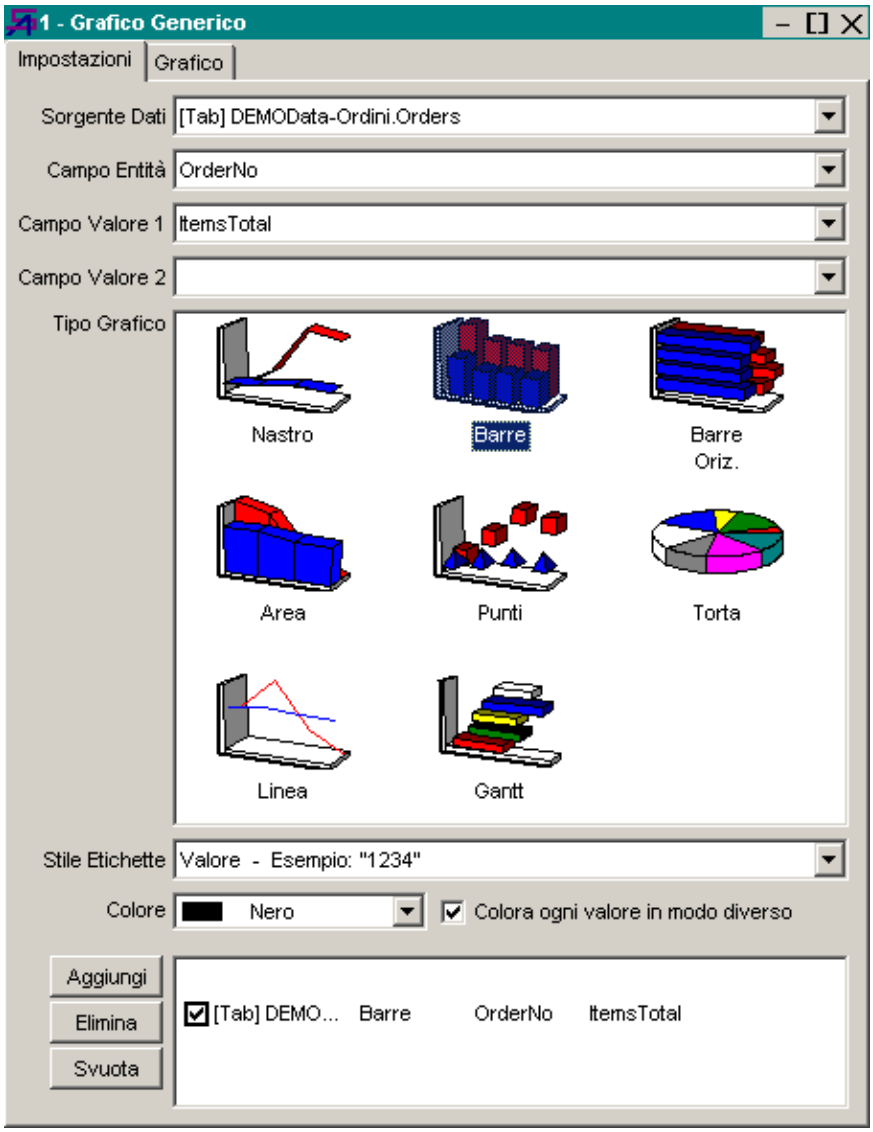
# *Grafici Generici*

Questo modulo può essere utilizzato per creare grafici con dati provenienti da vari dataset esterni.

Per definire i grafici da visualizzare, occorre specificare i seguenti parametri:

- Sorgente dati (da cui leggere i dati di partenza)
- Campo Entità (il valore sull'asse X)
- Campo Valore 1 (cioè il nome del campo sorgente)
- Campo Valore 2 (solo per il Gantt)
- Tipo del Grafico
- Stile Etichette (opzionale)
- Colore (Opzionale)

Dopo l'impostazione dei vari parametri, premendo il pulsante [Aggiungi] il nuovo grafico verrà visualizzato.

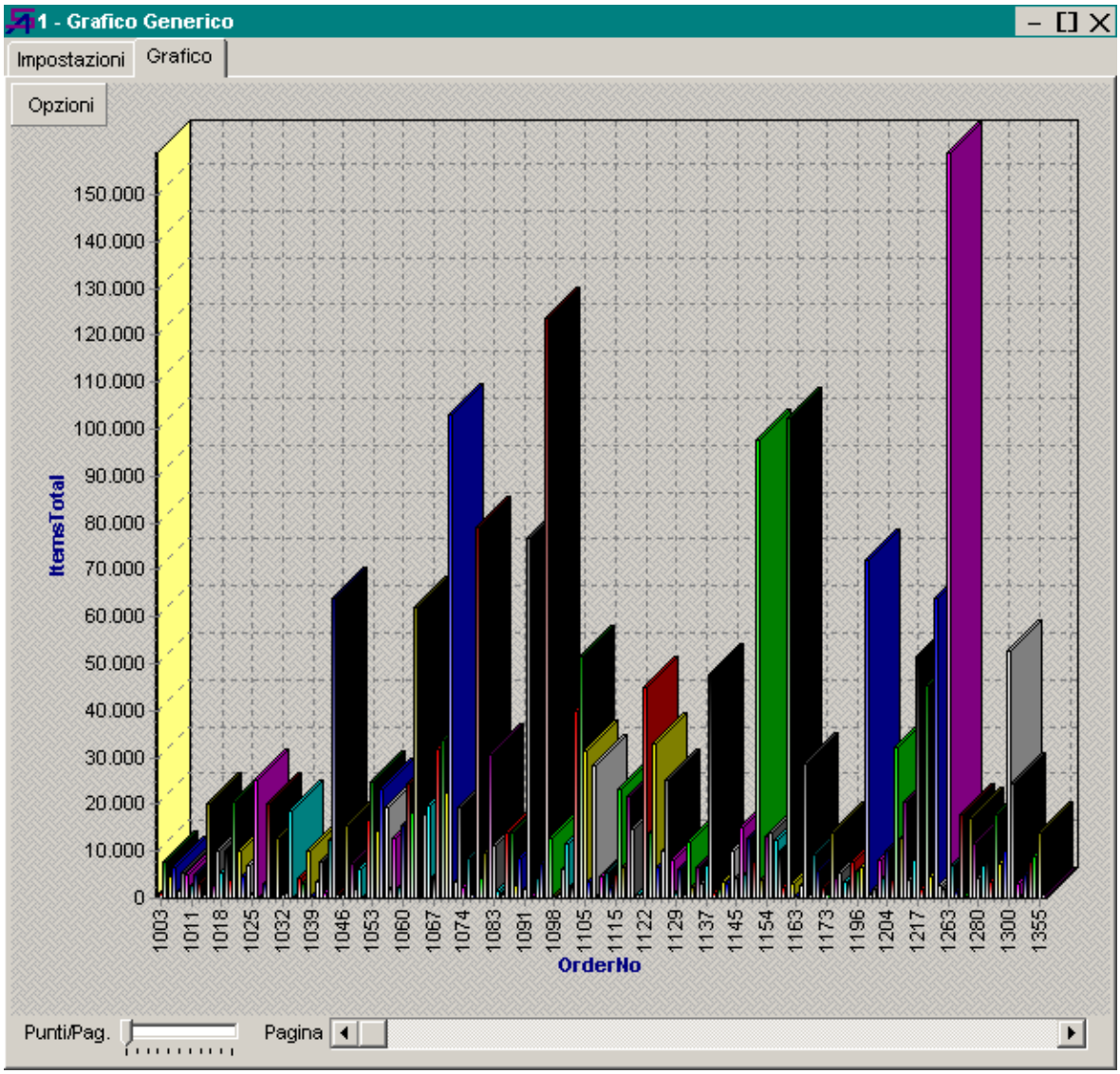


Supponiamo di avere aperto due tabelle rispettivamente con i campi
[Cliente, Fatturato98] e [Cliente, Fatturato99]
e di voler creare un grafico per confrontare il fatturato dei vari clienti nelle due annate ( 1998, 1999 ).

Occorre selezionare il campo Entità [Cliente].

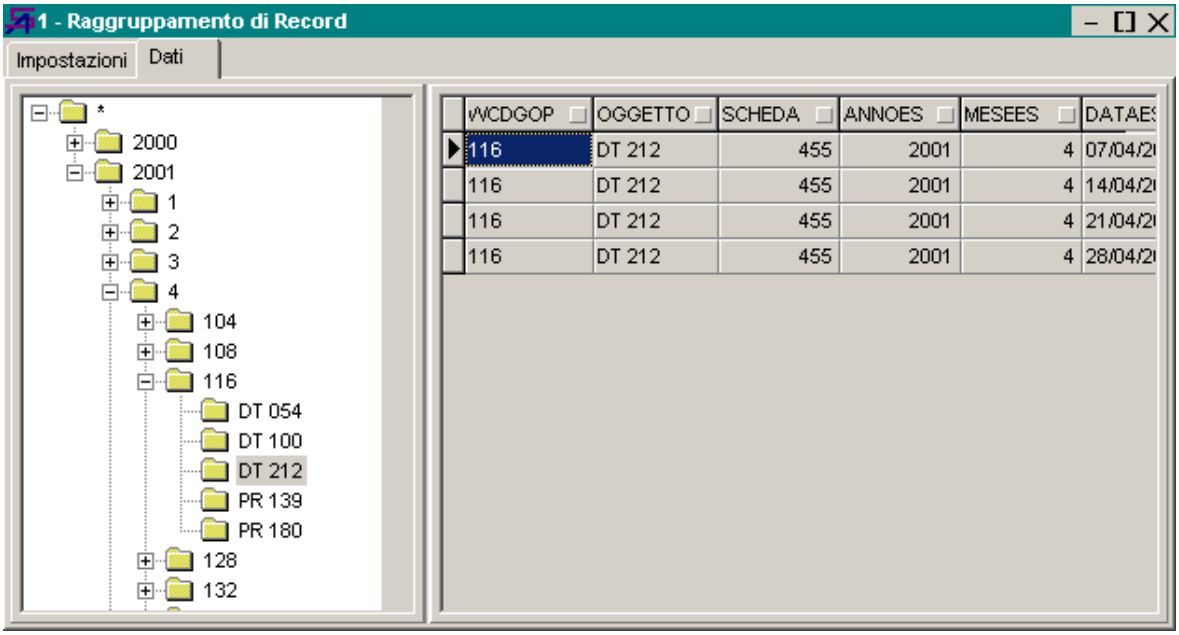
E' necessario, poi inserire i seguenti grafici:

| ***Sorgente*** | ***Campo Valore 1*** | ***Tipo Grafico*** | ***Colore*** |
|---|---|---|---|
| Tabella1 | Fatturato98 | a Barre | Rosso |
| Tabella2 | Fatturato99 | a Barre | Blu |



# *Raggruppamento di Record*

Questo modulo consente di creare un albero per il raggruppamento dei record di una tabella in base ai campi chiave definiti dall'utente.


E' utile nel caso in cui si debba consultare tabelle di grandi dimensioni come, ad esempio uno storico.

Supponiamo di avere una tabella con i seguenti campi
[Regione, Provincia, Cliente, Anno, Ordine, Data, Articolo, Quantità, Prezzo]
che contiene lo storico di tutti gli ordini effettuati dai vari clienti con i relativi articoli ordinati.

Dopo aver selezionato il database e la tabella è sufficiente selezionare i campi da utilizzare per il raggruppamento; nel nostro esempio inseriremo i campi:
[Regione, Provincia, Cliente, Anno, Ordine].

Visualizzando la pagina dei Dati vedremo l'albero generato e la griglia con i dati. Selezionando una foglia dell'albero verranno visualizzati i record che rispettano la chiave formata dal valore di quella foglia e di tutte le foglie gerarchicamente superiori.

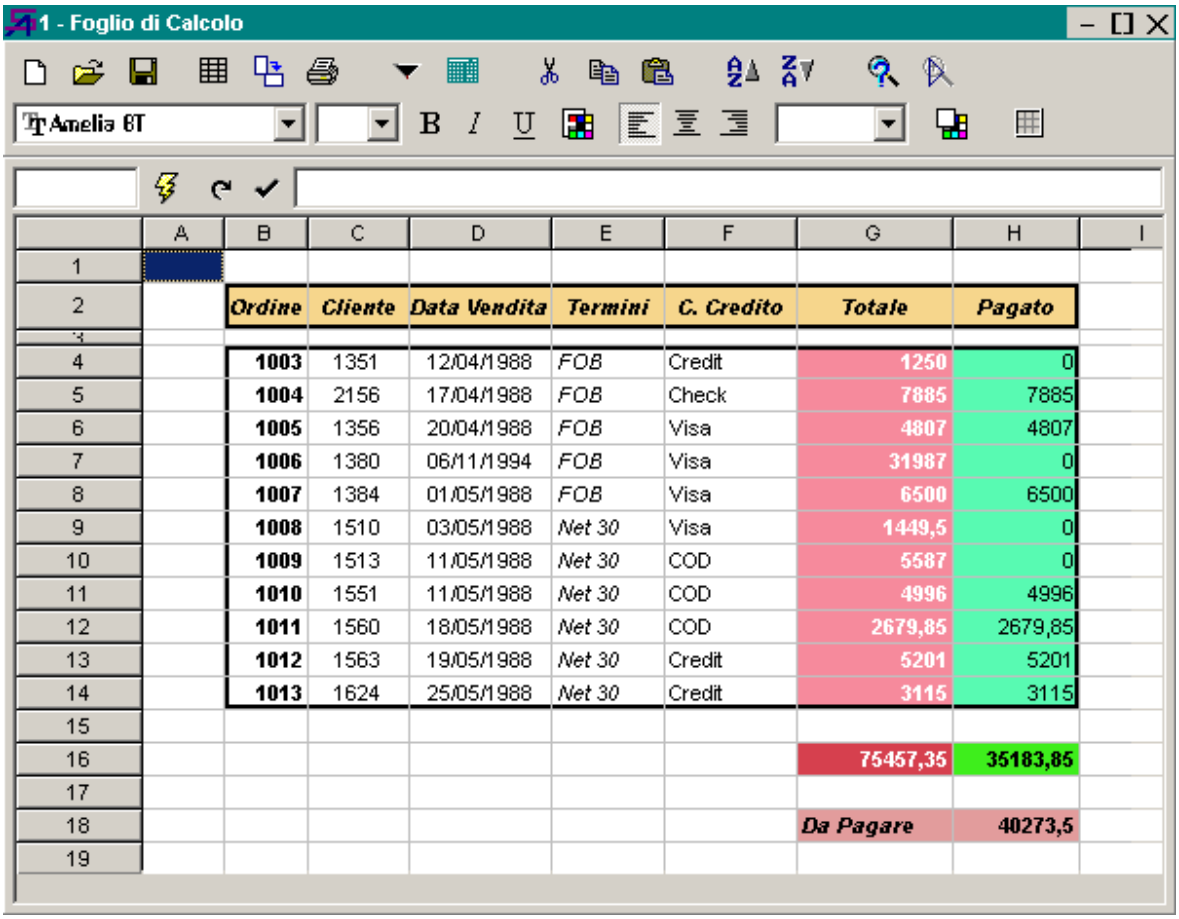
Nell' esempio seguente vengono specificati i record che verranno visualizzati a seconda della foglia selezionata:

| | | | | |
|---|---|---|---|---|
| * | | | | Tutti gli Articoli |
| | **Piemonte** | | | Tutti gli Articoli degli Ordini del Piemonte |
| | | **Torino** | | Tutti gli Articoli degli Ordini di Torino |
| | | | **Verdi** | Tutti gli Articoli degli Ordini del Sig. Verdi |
| | | | **1998** | Tutti gli Articoli degli Ordini fatti da Verdi ha nel 1998 |
| | | | | **1234** Tutti gli Articoli dell'Ordine 1234 |
| | | | | **1235** Tutti gli Articoli dell'Ordine 1235 |
| | | | | **...** |
| | | | **1999** | Tutti gli Articoli degli Ordini fatti da Verdi ha nel 1999 |
| | | | | **1656** Tutti gli Articoli dell'Ordine 1656 |
| | | | | **1657** Tutti gli Articoli dell'Ordine 1657 |
| | | | | **...** |
| | | | **...** | |
| | | | **Bianchi** | Tutti gli Articoli degli Ordini del Sig. Bianchi |
| | | | **...** | |
| | | **Asti** | | Tutti gli Articoli degli Ordini di Asti |
| | | | **Rossi** | Tutti gli Articoli degli Ordini del Sig. Rossi |
| | | | **....** | |
| | | **....** | | |
| | **....** | | | |

# *Foglio di Calcolo*

Questo modulo può essere utilizzato come un normale foglio elettronico alla stregua di Excel o simili.
Consente di lavorare su dati provenienti da dataset contenuti nei moduli dell'applicazione tramite un'importazione diretta, però non supporta l'aggiornamento dinamico al variare del dataset.
Il contenuto del Foglio elettronico può essere Salvato in un file, Copiato negli Appunti e stampato.

E' anche possibile selezionare un'area da rendere disponibile come dataset per il collegamento ad altri moduli.




| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | *Ordine* | *Cliente* | *Data Vendita* | *Termini* | *C. Credito* | *Totale* | *Pagato* | |
| 4 | | **1003** | 1351 | 12/04/1988 | *FOB* | Credit | 1250 | 0 | |
| 5 | | **1004** | 2156 | 17/04/1988 | *FOB* | Check | 7885 | 7885 | |
| 6 | | **1005** | 1356 | 20/04/1988 | *FOB* | Visa | 4807 | 4807 | |
| 7 | | **1006** | 1380 | 06/11/1994 | *FOB* | Visa | 31987 | 0 | |
| 8 | | **1007** | 1384 | 01/05/1988 | *FOB* | Visa | 6500 | 6500 | |
| 9 | | **1008** | 1510 | 03/05/1988 | *Net 30* | Visa | 1449,5 | 0 | |
| 10 | | **1009** | 1513 | 11/05/1988 | *Net 30* | COD | 5587 | 0 | |
| 11 | | **1010** | 1551 | 11/05/1988 | *Net 30* | COD | 4996 | 4996 | |
| 12 | | **1011** | 1560 | 18/05/1988 | *Net 30* | COD | 2679,85 | 2679,85 | |
| 13 | | **1012** | 1563 | 19/05/1988 | *Net 30* | Credit | 5201 | 5201 | |
| 14 | | **1013** | 1624 | 25/05/1988 | *Net 30* | Credit | 3115 | 3115 | |
| 15 | | | | | | | | | |
| 16 | | | | | | | 75457,35 | 35183,85 | |
| 17 | | | | | | | | | |
| 18 | | | | | | | *Da Pagare* | 40273,5 | |
| 19 | | | | | | | | | |

# *Rete Neurale*

Questo modulo può essere utilizzato per addestrare ed eseguire una Rete Neurale.
Il modulo implementa reti multistrato di tipo EBPN (Error Back Propagation Net). Si possono cioè definire reti che utilizzano il metodo della retropropagazione dell'errore per l'addestramento

## *Definizione*

Nella prima pagina vengono impostate le proprietà della rete:

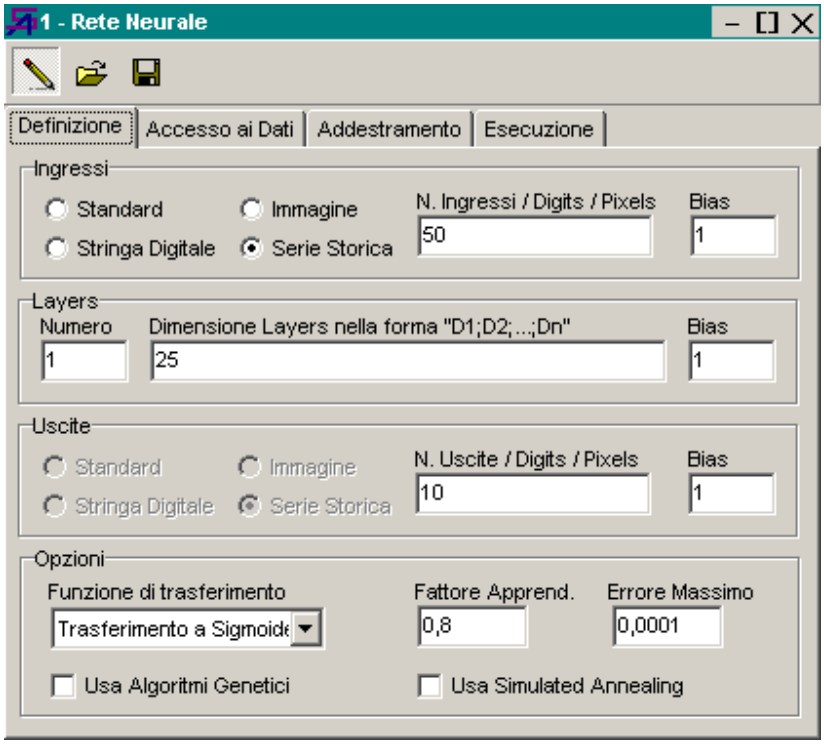
- Ingresso: Tipo dati, Numero d'ingressi, Bias
- Strati (Layers): : Numero di strati intermedi, numero di neuroni nei vari strati
- Uscita: Tipo dati, Numero di uscite, Bias
- Opzioni: Fattore di apprendimento, Massimo errore per campione, Alg. Gen, Sim. Annealing

A seconda del tipo di rete e di dati da analizzare, l'utilizzo dell'algoritmo genetico e del Simulated Annealing possono portare ad un miglioramento dell'addestramento, ma conviene comunque iniziare con il metodo standard e poi eventualmente attivare queste opzioni per migliorare il risultato.
Il tipo di dati consente di specificare come deve essere interpretato il contenuto dei campi usati come input e output:

- Standard: Ogni campo contiene un valore che rappresenta un segnale (campi numerici)
- Stringa Digitale: Il campo contiene una stringa del tipo "0010110110" in cui ogni digit rappresenta un segnale
- Immagine: Il campo contiene un'immagine quadrata di tipo Bitmap
- Serie Storica: Il campo contiene valori numerici e verranno usati N record per N segnali

Nota: Con il tipo standard è anche possibile utilizzare campi stringa che contengano valori tipo Flag e in questo caso occorrerà specificare l'elenco dei possibili valori onde poter eseguire la normalizzazione.

## *Accesso ai dati*



Nella seconda pagina vengono impostate le sorgenti dei dati:

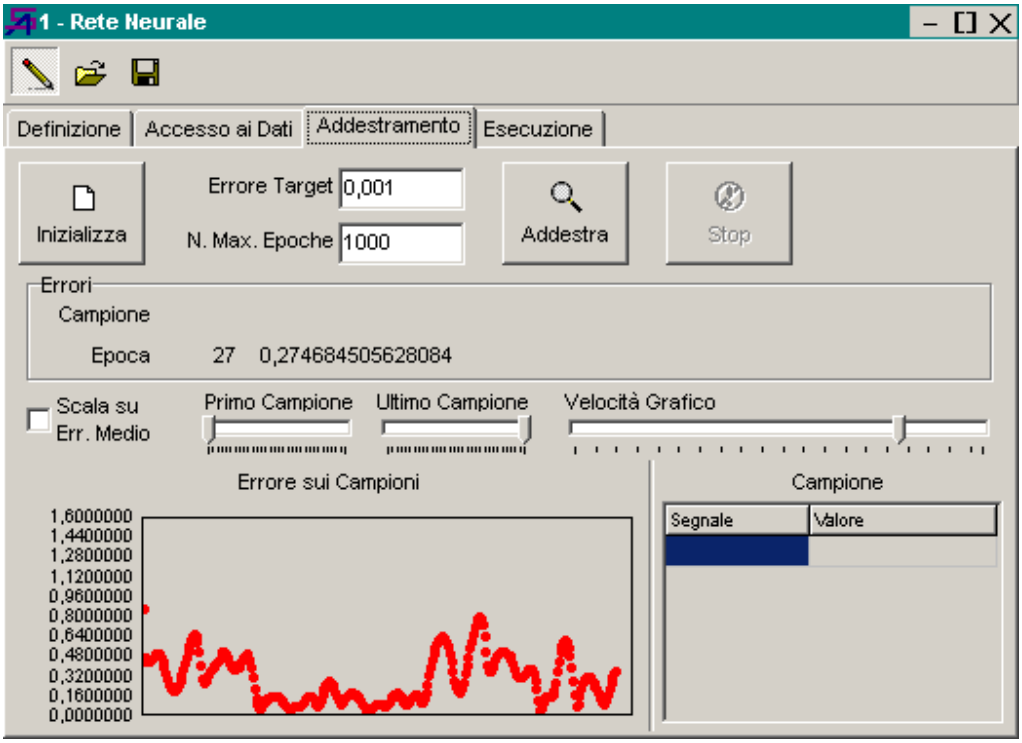
- Database: Alias del database da cui prelevare i dati
- Query: Definizione della sorgente dati
- Ingressi: elenco dei campi da cui prelevare i dati di ingresso
- Uscite: elenco dei campi da cui prelevare i dati di uscita

Per ogni campo occorre specificare i seguenti parametri che verranno utilizzati per normalizzare i valori in modo che rientrino nell'intervallo [0..1] in ingresso e "Denormalizzati" in uscita:

- Minimo: Limite inferiore del range dei possibili valori
- Massimo: Limite Superiore del range dei possibili valori
- Valori: Elenco dei possibili valori se il campo è di tipo stringa

## *Addestramento*

Nella terza pagina si trovano i controlli per effettuare l'addestramento e visualizzare il grafico degli errori.
I parametri fondamentali da impostare sono i seguenti:
- Errore Target: L'errore massimo che si vuole raggiungere
- Massimo Num. Epoche: Il massimo numero di Iterazioni da effettuare indipendentemente dall'errore




## *Esecuzione*

Nella quarta pagina si trovano i controlli per utilizzare una rete addestrata.
Occorre selezionare la sorgente dei dati su cui eseguire l'analisi. Nel caso in cui si stia operando un'analisi su una serie storica i risultati verranno inseriti ina tabella temporanea, altrimenti verrà aggiornato direttamente il dataset selezionato.

# *Valutazione Espressioni*

I moduli per i campi calcolati e le statistiche incorporano un valutatore di Espressioni matematiche in grado di eseguire il calcolo di qualunque funzione.

Un espressione può contenere:

- Operatori
- Valori numerici (come sep. decimale viene usato il punto)
- Funzioni (anche multi parametro; come sep. parametri viene usata la virgola)
- Variabili (i nomi dei campi di una tabella)
- Parentesi

Non ci sono limitazioni per l'annidamento di operazioni e funzioni.
Gli spazi possono essere utilizzati a piacere

## *Esempi di espressioni valide*

```
( X + Y - Z ) * 2
SIN(X+Y)
POWER( SIN(X+Y), LOG(Z) )
10 % MEAN(X, Y, Z, K, J)
X ^ Y / 2
```

## *Elenco Operatori in ordine inverso di precedenza*

| | |
|---|---|
| % | Percentuale |
| + | Somma |
| - | Differenza |
| * | Moltiplicazione |
| / | Divisione |
| ^ | Elevamento a potenza |

Es.
15 %100 + 50 - 5 * 80 / 2^3 = 15 %(100 + 50 - 5 * 80 / 8) =15% ( 100 + 50 - 50) = 15

## *Elenco Funzioni Supportate*

**Trigonometriche**

| | |
|---|---|
| COS | Coseno |
| SIN | Seno |
| TAN | Tangente |
| ACOS | Arco Coseno |
| ASIN | Arco Seno |
| ATAN | Arco Tangente |
| COSH | Coseno Iperbolico |
| SINH | Seno Iperbolico |
| TANH | Tangente Iperbolica |
| ACOSH | Arco Coseno Iperbolica |
| ASINH | Arco Seno Iperbolica |
| ATANH | Arco Tangente Iperbolica |
| SEC | Secante |
| COSEC | Cosecante |
| COTAN | Cotangente |

**Matematiche**

| | |
|---|---|
| ABS | Valore Assoluto |
| SQRT | Radice Quadrata |
| EXP | Esponenziale |
| HYPOT | Ipotenusa [es. HYPOT(3, 4) = Sqrt(3^2 + 4^2) = 5] |
| LN | Logaritmo Naturale |
| LOG | Logaritmo in base 10 |
| LG2 | Logaritmo in base 2 |
| LOGN | Logaritmo in base N [es. LOGN(N, N^3) = 3] |
| POWER | Elev. a Potenza [es. POWER(3, 2) = 9] |
| SIGN | Segno [es. SIGN(-5) = -1; SIGN(3) = 1] |
| TRUNC | Restituisce la parte intera di un numero in virgola mobile |
| FRAC | Restituisce la parte frazionaria di un numero in virgola mobile |
| ROUND | Arrotonda un numero in vrgola mobile |
| ROUNDX | Arrotonda al decimale X [es. ROUNDX(2, 5.43751) = 5.44] |

**Statistiche**

| | |
|---|---|
| MEAN | Media aritmetica [es. MEAN(3, 1, 5, 2, 6, 7) = 4] |
| MIN | Minimo |
| MAX | Massimo |
| NORM | Norma |
| RANDG | Gen. Casuale [uso RANDG(Media, Dev.Standard)] |
| SUM | Somma |
| SUMSQR | Somma dei Quadrati |
| STDDEV | Deviazione Standard su un campione |
| STDDEVPOP | Deviazione Standard sull'intera popolazione |
| VARIANCE | Varianza su un campione |
| VARIANPOP | Varianza sull'intera popolazione |
| VARIANTOT | Varianza totale ovvero Sommatoria(i=1,N)[(X(i) - Media)^2] |

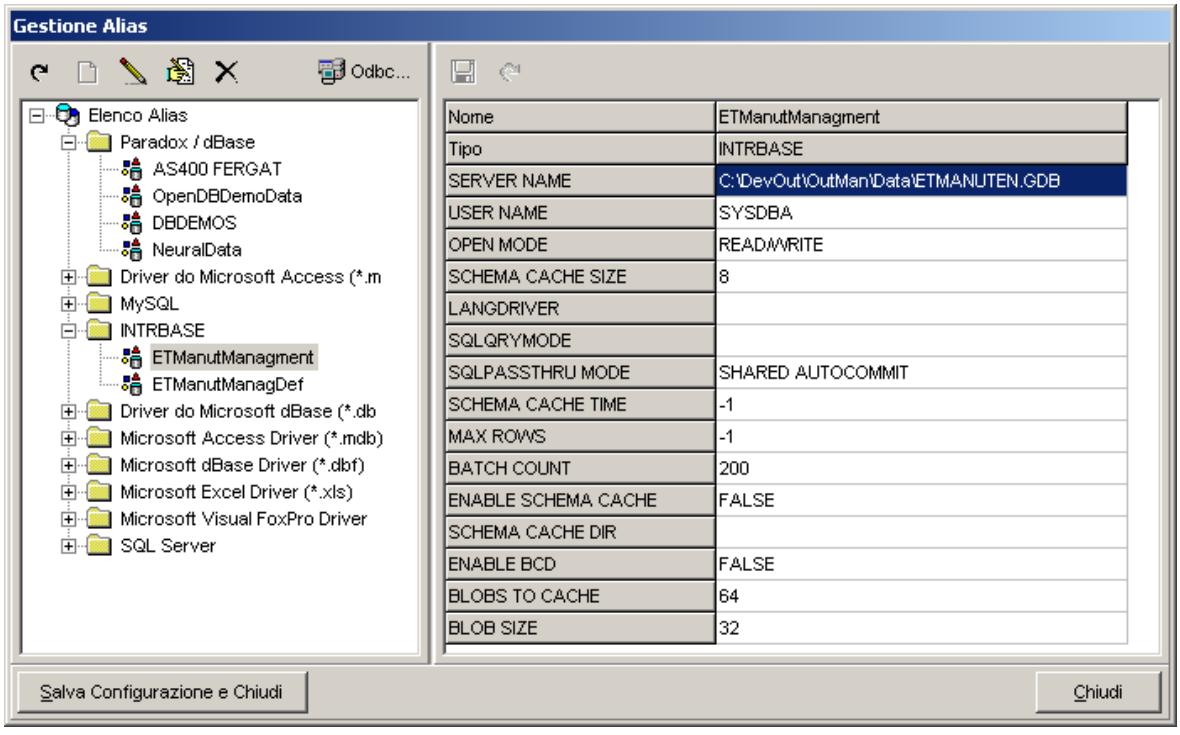
# Capitolo 7

# *Utility*

## *Gestione Alias*

L'applicazione utilizza il Borland Database Engine come interfaccia per l'accesso ai dati.
Il BDE consente di accedere a tabelle Paradox o dBase specificando il percorso della directory che le contiene.
Per accededere ad altri tipi di database come Interbase o SQL Server è necessario definire degli Alias.
Un Alias è un'associazione tra un nome univoco nel sistema e i parametri di collegamento al database.
Ad esempio per accedere ad un DB Interbase occorre specificare:

- Percorso del DB
- Modalità di accesso (LOCAL / SERVER)
- Abilitazione alla scrittura
- Nome Utente
- Password




Questo modulo consente di gestire gli Alias di sistema.
E' possibile inserire, modificare o eliminare gli Alias.

### Inserimento
Se si seleziona un driver e si preme il pulsante [Nuovo] viene creato un Alias per il driver selezionato, altrimenti viene richiesto di specificare il driver.

### Ridenominazione
Per modificare il nome di un Alias basta attivare l'editing sull'albero (click sul nome e attesa) o premere il pulsante [Rinomina].

### Modifica
Per modificare un Alias occorre selezionarlo e premere il pulsante [Modifica]. Al termine della modifica è possibile premere il pulsante [Applica] o il pulsante [Annulla] per uscire dalla modalità di modifica.

Per l'impostazione di alcune proprietà dell'Alias sono disponibili delle scorciatoie a cui si accede con un [Doppio Click] sulla riga della proprietà desiderata.

Le proprietà con scorciatoie sono:

- SERVER NAME
- SYSTEM DATABASE
- DATABASE NAME
- PATH
- OPEN MODE
- SQLQRYMODE
- SQLPASSTHRU MODE
- SCHEMA CACHE DIR
- ENABLE ....

### *Salvataggio Configurazione*
Le modifiche alla configurazione degli Alias, è temporanea e nel caso di un riavvio del sistema viene ripristinata la configurazione precedente.

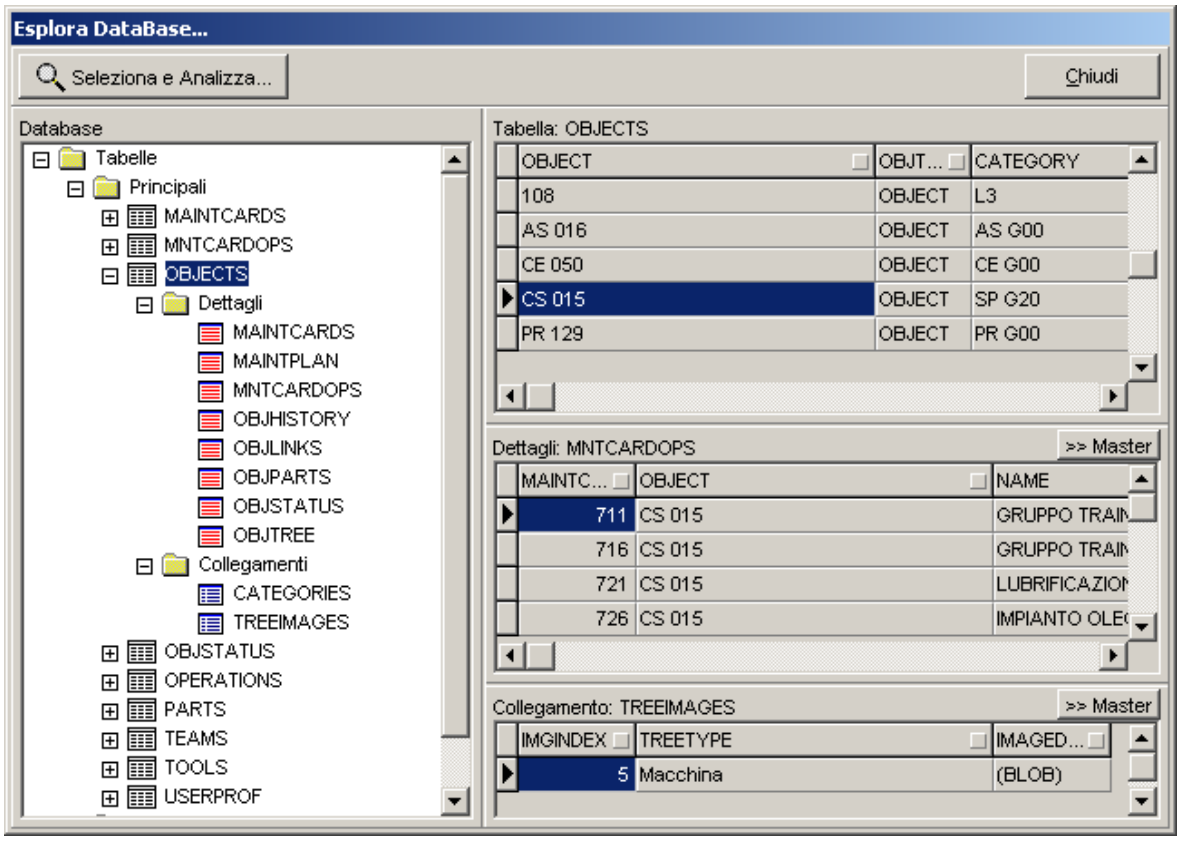
Per rendere la configurazione persistente occorre salvarla esplicitamente con il relativo pulsante.

# *Esplorazione DataBase*

Questo modulo consente di analizzare la struttura di un database e di creare un albero con:

- Tabelle principali: quelle per cui esistono tabelle di dettaglio
- Tabelle secondarie: quelle che contengono almeno riferimenti ad altre tabelle
- Altre Tabelle: le tabelle ausiliare senza riferimenti ad altre tabelle

E' utile soprattutto quando sia necessario comprendere il funzionamento di un database sconosciuto per cui non esiste una documentazione precisa, come spesso succede nei progetti di conversione per l'avvio di nuove applicazioni.




Il sistema utilizzato per l'analisi è "euristico" nel senso che non utilizza i riferimenti espliciti come le Foreign Key eventualmente definiti sul database. In pratica controlla la struttura delle tabelle, le chiavi primarie e gli altri indici e individua le tabelle che contengono campi con lo stesso nome di altre.

Le regole per la determinazione della struttura sono le seguenti:

Se **la Chiave Primaria di TAB1** è uguale ad
**un Indice di TAB2** allora
**TAB2** è una tab. di dettaglio **di TAB1**

Se **i campi della Chiave Primaria di TAB1** sono compresi tra
**i campi della Chiave Primaria di TAB2** allora
**TAB2** è una tab. di dettaglio **di TAB1**

Se **i campi della Chiave Primaria di TAB2** sono compresi tra
**i campi di TAB1** allora
**TAB2** è una tab. di Lookup **di TAB1**

Questo sistema non è perfetto ma permette di ottenere dei buoni risultati, almeno nel caso di database progettati con una logica relazionale anche se non sono stati definiti esplicitamente i collegamenti tra le tabelle come ad esempio nei database che utilizzano tabelle dBase.

Per eseguire l'analisi basta premere il pulsante [Seleziona e Analizza]; verrà visualizzata una finestra che consente di:

- selezionare il database
- selezionare le tabelle da analizzare
- eseguire l'analisi
- salvare un file di testo contenete la struttura

Alla conferma, nel modulo di esplorazione verrà visualizzato l'albero delle tabelle. Ogni tabella può avere le sottocartelle [Dettagli] e [Collegamenti] che contengono rispettivamente le tabelle di dettaglio e di lookup.

Espandendo il ramo di una tabella di primo livello, i suoi dati verranno visualizzati nella griglia primaria.

Clickando su una tabella di dettaglio, i suoi dati verranno visualizzati nella griglia dettagli.

Clickando su una tabella di lookup, i suoi dati verranno visualizzati nella griglia collegamento.

Le tabelle di dettaglio e di lookup restano collegate dinamicamente alla tabella primaria in modo che selezionando un record su quest'ultima, esse vengono aggiornate per visualizzare i dettagli ed i collegamenti relativi a quel record.

Agendo sui pulsanti [>> Master] si ottiene un cambio di contesto, ovvero la tabella, ad esempio, di dettaglio viene aperta come master (se si tratta di una tabella principale) ed è possibile continuare l'esplorazione ricorsivamente.

# *Ricerca Testo*

Questo modulo consente di effettuare delle ricerche di testo all'interno di una tabella o di un intero database. In pratica si agisce come se si stesse cercando una parola all'interno di un file di testo.

Il programma cerca il testo desiderato all'interno di ogni campo di tipo stringa o memo in ogni record.

Per effettuare la ricerca occorre seguire la procedura:

- Selezionare il Database
- Selezionare la tabella o attivare la ricerca sull'intero database
- Immettere il testo da cercare.
- Sciegliere se occorre rispettare le Maiuscole/minuscole
- Premere il pulsante [Trova Primo]

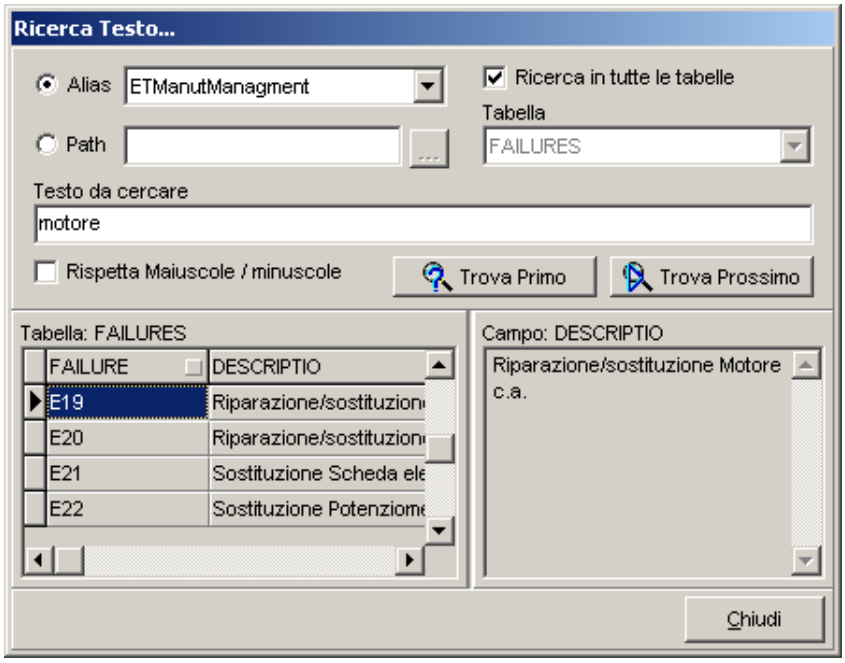
Se viene trovato il testo desiderato, è possibile continuare la ricerca sugli altri record premendo il pulsante [Trova Prossimo].

Nel testo da cercare è possibile specificare più parole separate dal segno [+].

In questo modo verrà trovati solo i campi che contengono tutte le parole immesse.

Ad esempio se si immette "Ruota + Ferro" verranno individuati tutti i campi che contengono sia la parola "Ruota" che la parola "Ferro", mentre verranno ignorati i campi che contengono solo una delle due parole.

Non ci sono limiti al numero di parole specificabili.

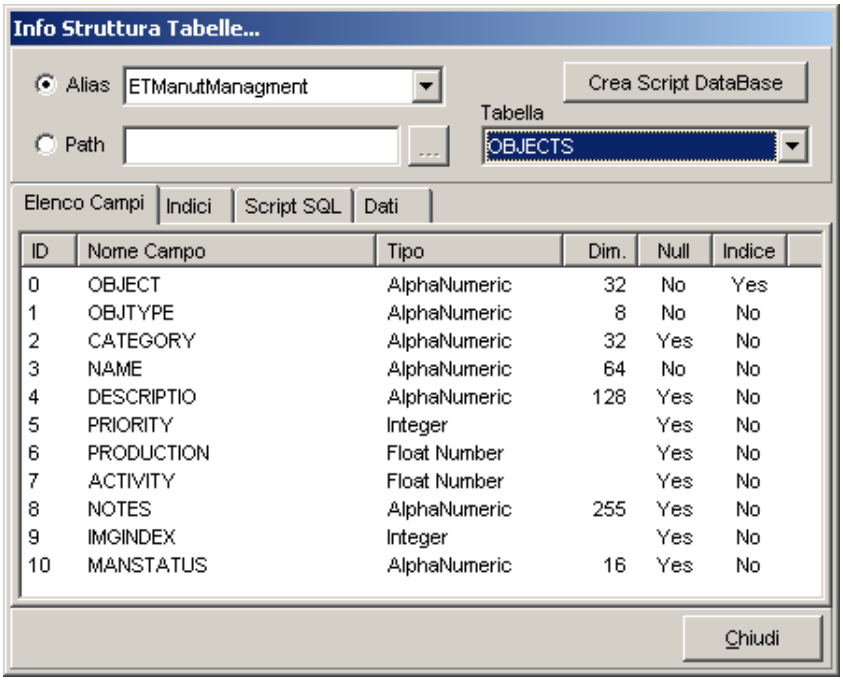


# *Informazioni Tabella*

Selezionando il database tramite alias o percorso, verrà creato l'elenco delle tabelle disponibili. Selezionando una tabella verranno visualizzate le seguenti informazioni:

- Struttura dei campi
- Indici definiti sulla tabella
- Statistiche (Num. record, Dim. record ecc.)
- Script SQL per la creazione di una tabella con la stessa struttura
- Griglia dei dati contenuti

Premendo il pulsante [Crea Script Database] verrà creato lo script completo per la creazione di tutte le tabelle contenute nel database.



# *Creazione Tabelle*

Per creare una nuova tabella è necessario seguire la seguente procedura:

- Selezionare il database di destinazione
- Definire la struttura inserendo per ogni campo: Nome, Tipo, Dim., Richiesto
- Definire gli Indici
- Premere il pulsante [Crea Tabella]

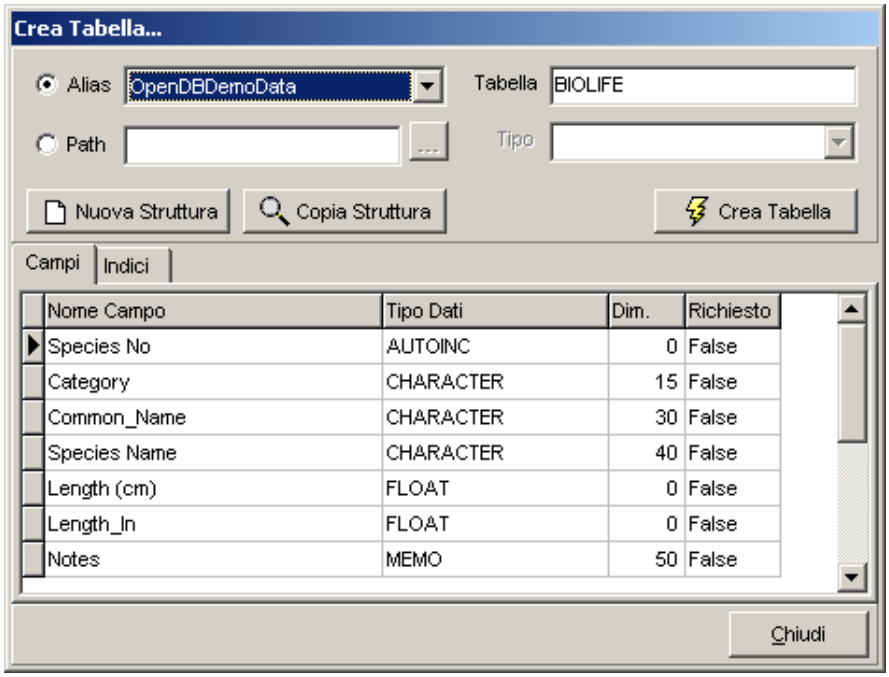
Per definire la struttura è possibile partire da zero utilizzando il pulsante [Nuova Struttura] o copiare e modificare la struttura di una tabella preesistente.

Premendo il pulsante [Copia Struttura] verrà aperto il modulo di selezione della tabella e a selezione effettuata, la struttura verrà visualizzata per un eventuale modifica.

Per la definizione degli indici occorre seguire la procedura:

- Premere il pulsante [+]
- Inserire il nome dell'indice
- Selezionare le opzioni: primario, unico, discendente
- Selezionare i campi che compongono l'indice

Per aggiungere un campo occorre un Doppio Click sulla lista dei campi della tabella
Per togliere un campo occorre un Doppio Click sulla lista dei campi dell'Indice



# *Importa / Esporta Testo*

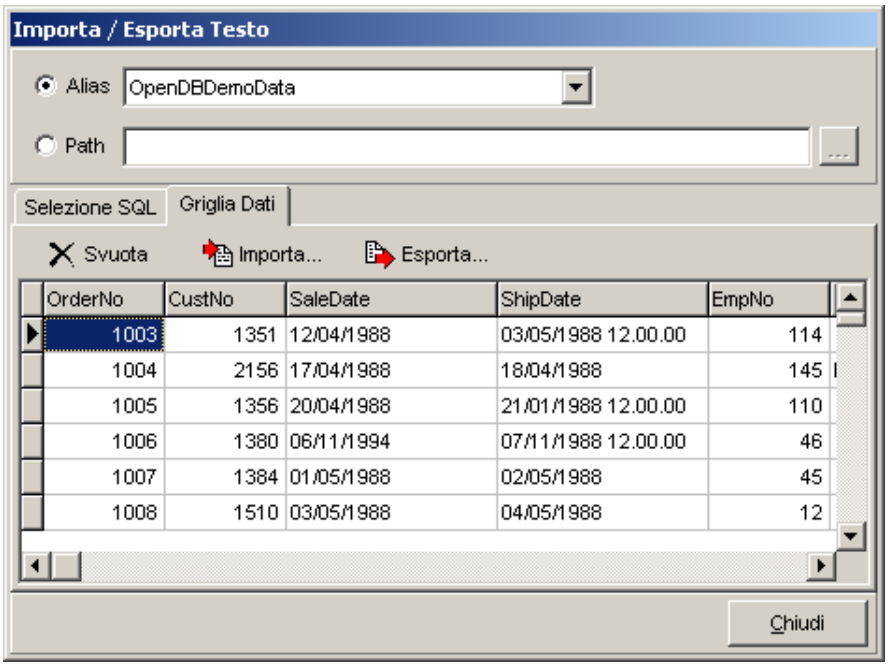
Questa utility consente di importare ubn file di testo formattato in una tabella preesistente o di esportare un dataset creando un nuovo file di testo.

Per selezionare la tabella per l'importazione o il dataset per l'esportazione occorre scrivere una query SQL.

Se la query inserita è corretta il dataset o la tabella vengono aperti ed i dati vengono visualizzati nella griglia.

Nella pagina della griglia ci sono tre pulsanti:

- Svuota: elimina tutti i record dalla tabella (utile prima di effettuare un'importazione)
- Importa: apre il modulo per selezionare i dati da importare.
- Esporta: apre il modulo per creare il file in cui esportare i dati.



## *Importazione Testo*

In questo modulo, attivato dall'utility di Importazione/Esportazione del testo, dopo aver selezionato il file da importare, occorre eseguire le seguenti impostazioni:

- immettere il carattere di separazione dei campi utilizzato nel file
- scomporre il primo record in modo da visualizzare i valori dei vari campi
- scegliere di non importare il primo record (se rappresenta i titoli delle colonne)
- selezionare i campi da importare ed i relativi campi di destinazione
- premere il pulsante [OK] per eseguire l'importazione.

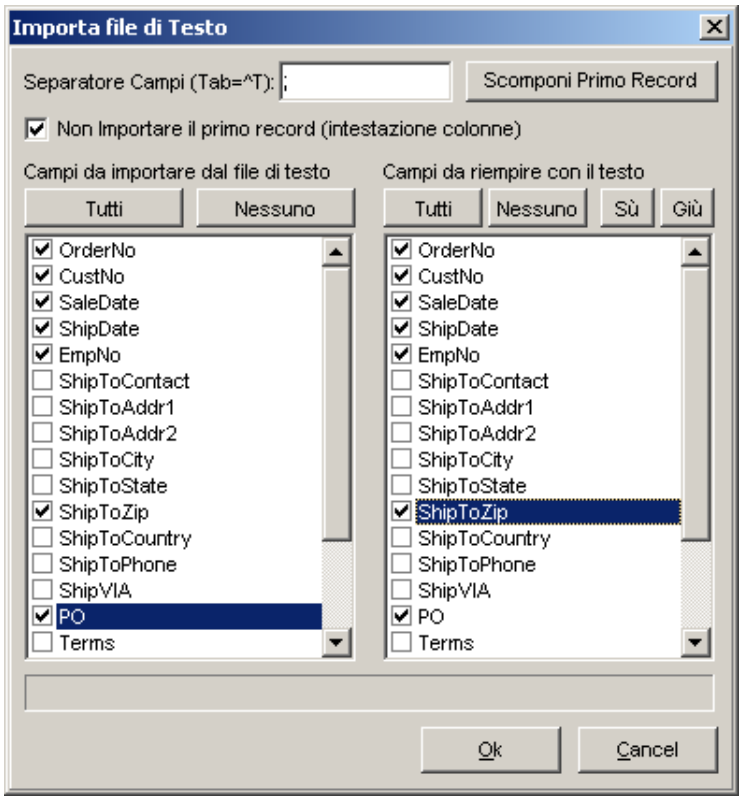
I valori da importare verranno inseriti nei campi adiacenti, quindi occorre eventualmente cambiare l'ordine dei campi di destinazione.

## *Esportazione Testo*

In questo modulo, attivato dall'utility di Importazione/Esportazione del testo, occorre eseguire le seguenti impostazioni:

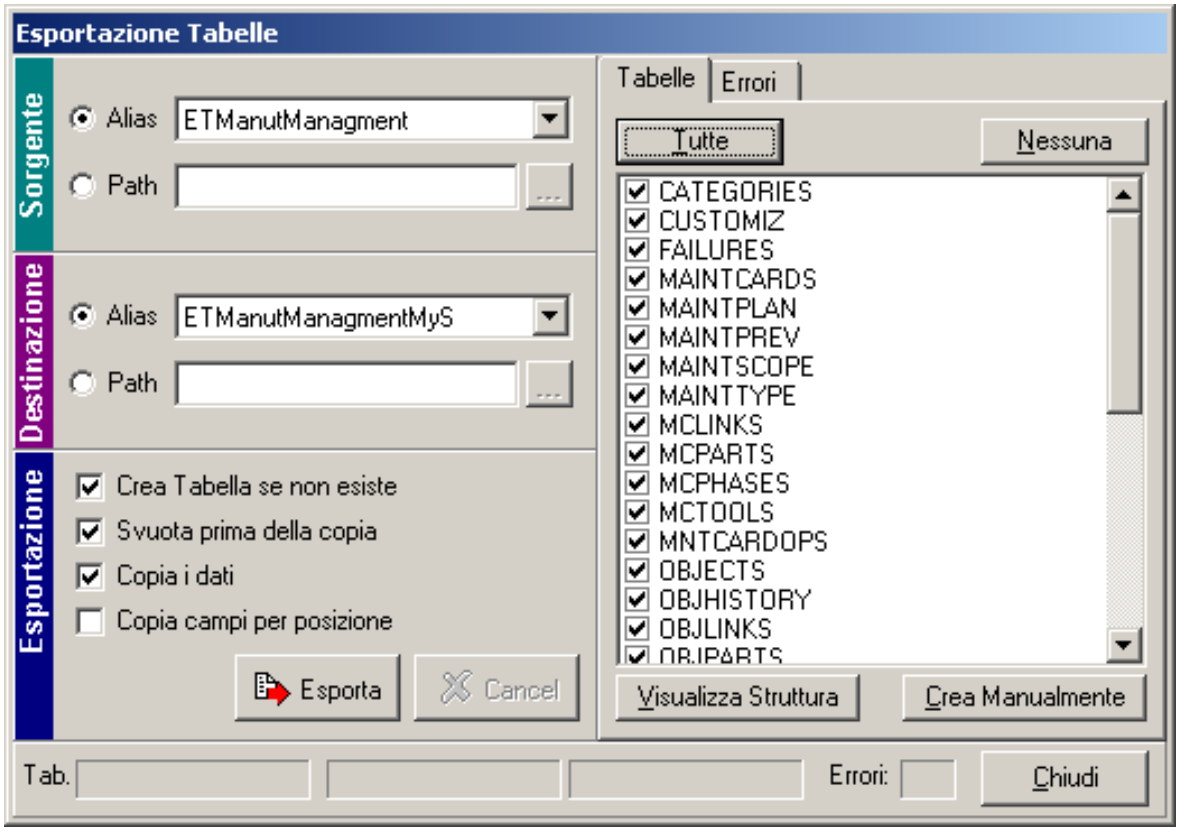
- selezionare il carattere da utilizzare come separatore dei campi
- scegliere se porre i campi di tipo stringa tra "virgolette"
- selezionare i campi da esportare ed il loro ordine

Al termine delle impostazioni premendo il pulsante [esporta] verrà richiesto di specificare il nome ed il percorso del file di testo da creare.




# *Esportazione Tabelle*

Questa utility è in grado di trasferire dati tra due database qualunque. L'unico requisito necessario è l'esistenza del database di sestinazione.




Questo modulo infatti è in grado di creare le tabelle nel database di destinazione se non esistono ancora. In effetti l'utilità maggiore, è rappresentata dalla possibilità di trasferire un database completo (o una parte delle tabelle) cambiando il formato.

Ad esempio può capitare di dover creare una applicazione nuova che però deve utilizzare i dati contenuti in un vecchio database, poniamo formato da tabelle dBase.

Supponiamo che la nuova applicazione si basi su Interbase: con questa utility è possibile creare le tabelle con lo stesso formato di quelle sorgenti e trasferire i dati in un'unica operazione. E' anche possibile creare soltanto la struttura senza trasferire i dati.

Gli unici limiti sono dati da eventuali incompatibilità tra tipi differenti di database. Ad esempio una tabella Paradox può avere campi di tipo booleano, mentre Interbase non supporta questo tipo; in questo caso verrà creato un campo di tivo VARCHAR(1) e poi verrà riempito con valori [T] e [F].

Altri problemi possono derivare dalle diverse definizioni per gli indici e le chiavi primarie, ma il sistema di esportazione riesce a risolvere la maggior parte delle incongruenze.

Per effettuare l'esportazione occorre seguire la procedura:

- · Selezionare il database Sorgente
- · Selezionare il database Destinazione
- · Selezionare le tabelle da esportare
- · Impostare l'opzione di creazione delle tabelle inesistenti
- · Impostare l'opzione di svuotamente delle tabelle già esistenti che contengono dati
- · Impostare l'opzione di copia dei dati
- · Impostare l'opzione di copia per posizione (utile se i nomi con corrispondono)
- · Premere il pulsante [Esporta]

Sotto l'elenco delle tabelle ci sono due pulsanti utili per risolvere incompatibilità tra database diversi:

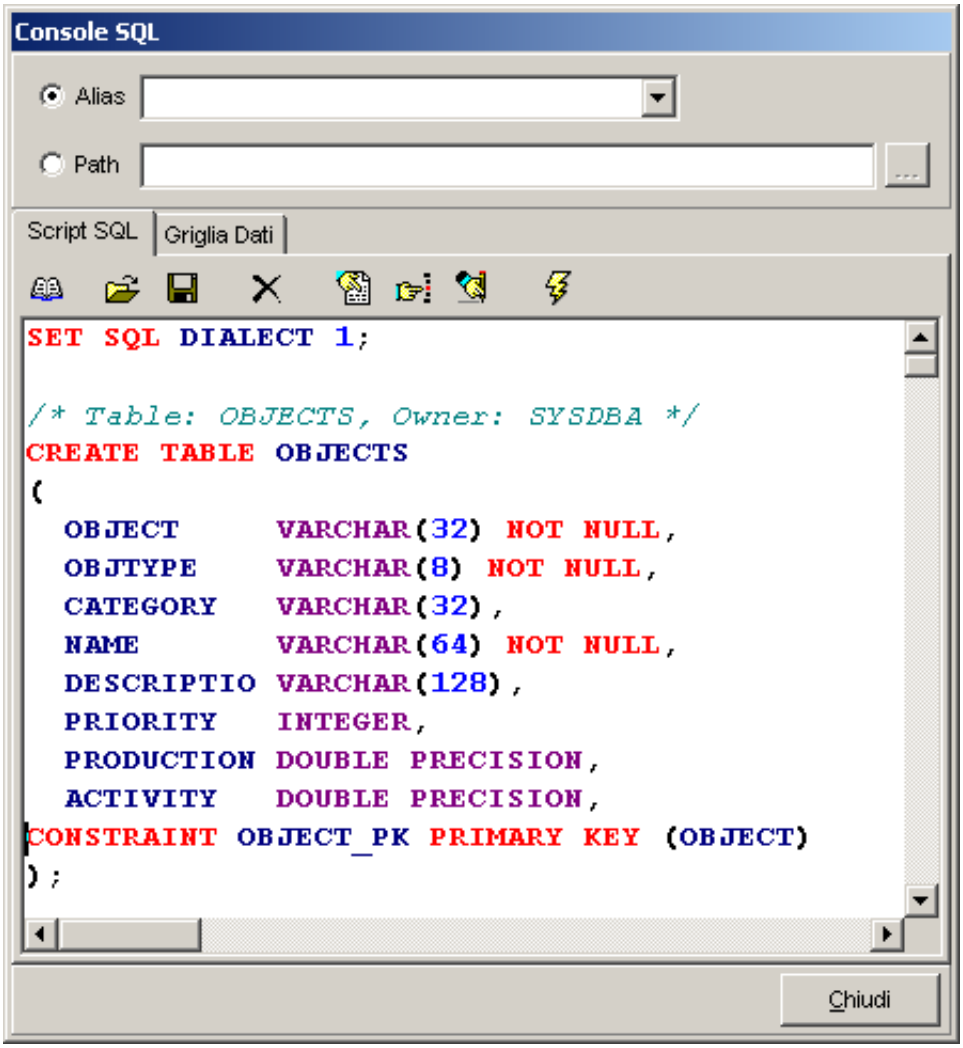
Visualizza struttura: visualizza la struttura della tabella selezionata

Crea manualmente: apre il modulo di creazione delle tabelle e imposta i vari parametri

Eventuali errori che si dovessero verificare durante l'esportazione saranno visualizzati nella relativa pagina.

## Console SQL

Con questo modulo è possibile eseguire qualunque Script SQL. Possono essere attivate delle selezioni di dati che vengono visualizzati nell'apposita griglia. Nel caso di query diverse da quelle di selezione, è anche possibile definire degli script con query multiple; tali query devono essere separate da [;].



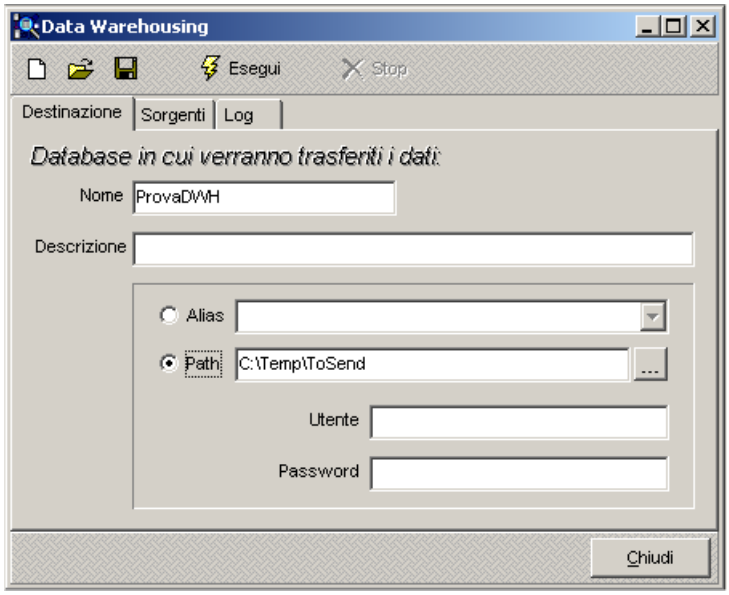

## Procedure Batch

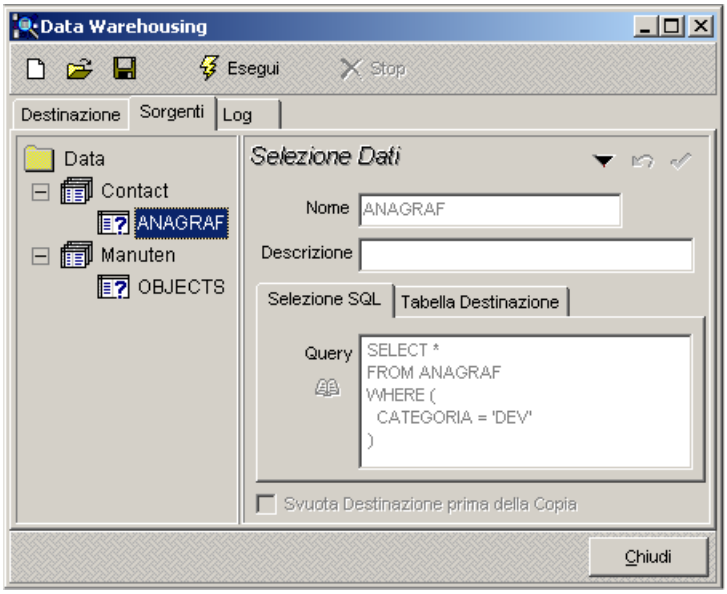
Questa utility è equivalente all'omonimo modulo speciale.
L'unica differenza è che in questo caso è possibile operare su qualunque database mentre il modulo speciale consente di operare solo sui DB definiti nel progetto.

# *Data Warehousing*

Con questo modulo è possibile definire qualunque procedura per la creazione di un data warehouse.
E' possibile definire un numero qualunque di sorgenti dati all'interno di DB diversi.
Sarà necessario definire la destinazione dei dati e per ogni dataset sorgente sarà possibile specificare il nome e la struttura della tabella di destinazione.

Nel caso la tabella di destinazione sia stata creata in precedenza, è possibile specificare la necessità di svuotarla prima di eseguire il trasferimento. Se la tabella destinazione non viene trovata, viene creata automaticamente.



# Tutorials

***Le Basi di Dati ·***

***Il Linguaggio SQL ·***

***Le Reti Neurali ·***

# *Le Basi di Dati*

Uno dei principali compiti dei sistemi informatici è l'attività di raccolta, organizzazione e conservazione dei dati. Tali sistemi garantiscono che questi dati siano conservati in modo permanente su dispositivi per la loro memorizzazione, permettendone l'aggiornamento e rendendone possibile l'accesso da parte degli utenti.

Questo tutorial ha come argomento la gestione dei dati tramite sistemi informatici; ha come obiettivo trattare in maniera semplice ma esaustiva i concetti che stanno dietro ad una tale gestione.

Di seguito verranno introdotti i concetti di sistema informativo e di base di dati, definendo poi quali sono i requisiti che deve avere un sistema informatico per gestire una base di dati.

# Introduzione

## Sistemi informativi, informazioni e dati

Per sistema informativo si intende quel sistema che permette la disponibilità e la gestione delle informazione. L'esistenza di un sistema informativo è indipendente dalla sua automazione; lo dimostra il fatto che archivi e servizi anagrafici esistono da vari secoli. Per indicarne la porzione automatizzata viene utilizzato il termine sistema informatico.
La diffusione dell'informatica ha fatto si che la quasi totalità dei sistemi informativi siano anche sistemi informatici.

Le informazioni vengono rappresentate e scambiate in varie forme, quali la lingua, disegni, figure, numeri. In alcuni casi può anche non esistere una rappresentazione esplicita delle informazioni, come nel caso di informazioni trasmesse oralmente e ricordate a memoria.
Col progredire delle attività umane, tuttavia, è nata l'esigenza di individuare opportune codifiche per la memorizzazione dei dati.

Nei sistemi informatici il concetto di rappresentazione e codifica viene portato all'estremo: le informazioni vengono rappresentate per mezzo di dati, che hanno bisogno di essere interpretati per fornire informazioni.

## Basi di dati, la definizione

La più generale definizione di una base di dati è collezione di dati utilizzati per rappresentare le informazioni di interesse per un sistema informativo.
Tale definizione è molto semplicistica e troppo generale. Nel paragrafo seguente si cercare di definire il termine in maniera più precisa.

Occorre, tuttavia, trarre una prima considerazione sulle basi di dati. Se prendiamo come esempio i dati relativi alle applicazioni bancarie noteremo che essi hanno una struttura sostanzialmente invariata da decenni, mentre le procedure che agiscono su di essi variano con una certa frequenza. Inoltre, quando viene introdotta una nuova procedura occorre, prima di tutto, "ereditare" (=importare) i dati dalla vecchia, se pur con le necessarie trasformazioni.

Questa caratteristica di stabilità porta ad affermare che i dati costituiscono una "risorsa" per l'organizzazione che li gestisce, un patrimonio significativo da sfruttare e proteggere. Le normative attuali in fatto di privacy e tutela delle basi di dati lo dimostra.

# Sistemi di gestione di basi di dati

Sebbene la gestione dei dati abbia catalizzato, fin dalle origini dell'informatica, l'attenzione delle applicazioni, solo negli anni settanta nascono linguaggi specificatamente dedicati alla gestione dei dati. Un esempio di tali linguaggi è il COBOL, nato in quegli anni e ormai superato, che è presente ancor oggi in un numero incredibile di applicazioni.

L'approccio "convenzionale" alla gestione dei dati sfrutta la presenza di archivi (o file) per memorizzare i dati in modo persistente sulla memoria di massa.

Tale approccio presenta delle macroscopiche deficienze per quanto riguarda la ricerca e la condivisione dei dati, in pratica annullata; infatti con una simile metodologia di lavoro ogni utente lavora con la propria copia "locale", con i relativi problemi *ridondanza* e possibilità di *incoerenze*. Le basi di dati sono state concepite in buona parte per ovviare ad inconvenienti di questo tipo.

Un sistema di gestione di basi di dati, detto DBMS (Data Base Management System) è un sistema software in grado di gestire collezioni di dati che siano *grandi*, *condivise* e *persistenti* assicurando la loro *affidabilità* e *privatezza*. Inoltre, in quanto prodotto informatico, deve essere *efficiente* e *efficace*. Una base di dati è una collezione di dati gestita da un DBMS.

Riassumendo un DBMS si occupa di basi di dati con le seguenti caratteristiche:

***Grandi dimensioni**:*
nel senso che possono avere anche dimensioni enormi ( terabyte e oltre ) e quindi oltre le capacita della memoria centrale di un elaboratore. Di conseguenza un DBMS deve essere in grado di gestire memorie secondarie.

***Risorse Condivise**:*
perché un DBMS deve permettere a più utenti di accedere contemporaneamente ai dati comuni. In tal modo viene anche ridotta la *ridondanza* e *inconsistenza* dei dati, dato che esiste una sola copia dei dati. Per controllare l'accesso condiviso di più utenti il DBMS dispone di un meccanismo apposito, *detto controllo di concorrenza*.

***Affidabilità**:*
dato che un DBMS deve garantire l'integrità dei dati anche in caso di malfunzionamento hardware e software, prevedendo per lo meno procedure di recupero dei dati. I DBMS forniscono, per tali scopi, procedure di salvataggio e ripristino della base di dati (*backup* e *recovery*).

***Privatezza**:*
i DBMS gestiscono un sistema di *autorizzazioni* che definisce i *diritti* di ciascun utente ( lettura, scrittura ecc.).

# Modelli di dati

Un modello di dati è un insieme di concetti utilizzati per organizzati i dati di interesse e descrivere la struttura in modo che essa risulti comprensibile ad un elaboratore.

Ogni modello di dati fornisce meccanismi di strutturazione, analoghi ai costruttori di tipo dei linguaggi di programmazione, che permettono di definire nuovi tipi sulla base di tipi elementari predefiniti.

Il modello relazionale dei dati (modello su cui si concentra l'attenzione di questo tutorial) permette di definire tipi per mezzo del costruttore di *relazione*, che consente di organizzare i dati in insiemi di record a struttura fissa. Una relazione viene spesso rappresentata mediante una tabella in cui le righe rappresentano i specifici record e le colonne corrispondono ai campi dei record.

## *Schemi ed istanze*

Esistono, oltre al modello relazionale, altri modelli di database quali il modello gerarchico, il modello reticolare, il modello ad oggetti. Tutti i modelli di basi di dati sono, però, accomunati da dalla presenza di una parte che rimane invariata nel tempo, detta *schema*, e da una parte, detta istanza o stato della base di dati, costituita dai valori effettivi.

Si dice anche che lo schema è la parte intensionale della base di dati mentre l'istanza è la parte estensionale.

## *Livelli di astrazione nei DBMS*

Esiste una proposta di struttura standardizzata per i DBMS articolata su tre livelli, detti *esterno, logico e interno*; per ciascun livello esiste uno schema:

Lo ***schema logico*** (o ***concettuale***), che costituisce la descrizione dell'intera base di dati per mezzo del modello logico adottato dal DBMS (cioè tramite uno dei modelli citati in precedenza, relazionale, gerarchico, reticolare o a oggetti).

Lo ***schema interno*** costituisce la rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione.

Uno ***schema esterno*** costituisce la descrizione di una porzione della base di dati di interesse, per mezzo del modello logico. Uno schema esterno può prevedere organizzazioni dei dati diverse rispetto a quelle utilizzate nello schema logico, che riflettono il punto di vista di un particolare utente o insieme di utenti. Pertanto, è possibile associare ad uno schema logico vari schemi esterni.

Nei sistemi moderni il livello esterno non è esplicitamente presente, ma è possibile definire relazioni derivate (o viste, dall'inglese *views*).

### *Indipendenza dei dati*

L'architettura così definita garantisce l'indipendenza dei dati, ovvero la principale proprietà dei DBMS.
Questa proprietà permette agli utenti ed ai programmi applicativi di utilizzare una base di dati ad un elevato livello di astrazione, che prescinde dai dettagli realizzativi utilizzati per la base di dati stessa.
In particolare, l'indipendenza dei dati può essere caratterizzata ulteriormente come indipendenza fisica e logica:

***L'indipendenza fisica*** consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati. In base a questa proprietà è possibile modificare le strutture fisiche senza influire sulle descrizioni dei dati ad alto livello e quindi sui programmi che utilizzano i dati stessi.

***L'indipendenza logica*** consente di interagire con il livello esterno della base di base in modo indipendente dal livello logico.

# I Database Relazionali

Rappresenta il modello su cui si basa la maggior parte dei sistemi di basi di dati oggi sul mercato. Tale modello fu proposto in una pubblicazione scientifica nel 1970 al fine di superare le limitazioni logiche dei modelli allora utilizzati, che non permettevano di realizzare efficacemente la proprietà di indipendenza dei dati, già riconosciuta come fondamentale.
Sebbene i primi prototipi di db basati sul modello relazionale risalgano ai primi anni settanta bisognerà aspettare la metà degli anni ottanta perché tale modello acquisisca una frazione significativa di mercato.

La lentezza di affermazione del modello relazionale deriva principalmente dal suo alto livello di astrazione: non è stato immediato per gli operatori del settore imparare ad individuare relazioni efficienti.

## Il modello relazionale

Vengono qui illustrate le modalità secondo cui esso questo modello permette di organizzare i dati, come il concetto di relazione possa essere mutuato dalla teoria degli insiemi ed utilizzato, con le debite varianti, per rappresentare le informazioni di interesse in una base di dati. Vengono approfonditi i concetti di corrispondenza fra dati in strutture diverse, informazione completa e vincoli di integrità.

## Modelli logici nei sistemi di basi di dati

Il modello relazionale si basa su due concetti fondamentali: relazione e tabella.
Mentre il concetto di tabella è facilmente intuibile, quello di relazione proviene dalla matematica, ed in particolare dalla teoria degli insiemi.

E' opinione diffusa che parte del successo del modello relazionale derivi dalla presenza contemporanea di questi due concetti, uno intuitivo ed uno formale.
Infatti, mentre le tabelle risultano naturali e facilmente comprensibili le relazioni garantiscono una formalizzazione semplice e chiara che ha permesso uno sviluppo teorico del modello finalizzato al raggiungimento di risultati di interesse concreto.

Il modello relazionale risponde al requisito dell'indipendenza dei dati e, pertanto, prevede un livello fisico ed un livello logico. Utenti e programmatori interagiscono solo col livello logico e quindi non è necessario che essi conoscano le strutture fisiche della base di dati.
Anche questo aspetto è responsabile del suo successo dato che i suoi principali concorrenti (reticolare e gerarchico) obbligavano gli utilizzatori a conoscerne, almeno a grandi linee, la struttura realizzativa.

# Relazioni

Il concetto di relazione è legato al concetto puramente matematico di prodotto cartesiano tra due insiemi.
Avendo due insiemi $D_1$ e $D_2$ il prodotto cartesiano ($D_1$ x $D_2$) è l'insieme delle coppie ordinate ($v_1$ e $v_2$) tale che $v_1$ è un elemento di $D_1$ e $v_2$ è un elemento di $D_2$.
Quindi il prodotto cartesiano è l'insieme di tutti le combinazioni tra gli insiemi dati.

La relazione è un sottoinsieme della relazione matematica tra due insiemi, detti domini della relazione, rappresentato da un insieme di tuple omogenee, dove per tuple (traslitterazione dell'inglese "tuple") si intende un elemento definito tramite i suoi attributi.
Le tuple sotto questo aspetto sono diverse dal concetto matematico di n-uple, elemento individuato tramite posizione, e tupla, in cui l'elemento è individuato tramite i suoi attributi.

Ad esempio dati due insiemi A e B dove A= {1,2,3} e B={h,k} il prodotto cartesiano è uguale all'insieme A x B={(1,h),(2,h),(3,h),(1,k),(2,k),(3,k)} mentre una relazione possibile è {(1,h),(1,k),(3,h)}.

Generalizzando la relazione ad un numero di insiemi n>0 avremmo $D_1$,$D_2$,…,$D_n$ il prodotto $D_1$ x $D_2$ x … x $D_n$ ed un sottoinsieme che descriverà la relazione.
Il numero n delle componenti dell'insieme è detto grado del prodotto cartesiano e della relazione.
Il numero degli elementi che della relazione è detta cardinalità della relazione.

Le tabelle nascono dall'esigenza di rappresentare graficamente le relazioni presentandole in una forma più facilmente comprensibile. In questo caso le righe della tabella saranno rappresentate dalle tuple mentre le colonne ne rappresentano i campi.

E' importante chiarire che in una relazione, in quanto insieme, non vi è alcun ordinamento fra le tuple che lo compongono; nelle tabelle che la rappresentano l'ordine c'è per necessità, ma è occasionale in quanto due tabelle con le stesse righe, ma in ordine diverso, rappresentano la stessa relazione.
Inoltre le tuple di una relazione sono distinte l'una dall'altra, in quanto tra gli elementi di un insieme non possono essere presenti due elementi uguali; da cui si deduce che una tabella può rappresentare una relazione solo se le righe che la formano sono diverse l'una dall'altra.

# Informazioni incompleta e valori nulli

Un'altra caratteristica importante dei sistemi relazionali è la presenza di un particolare valore che può assumere un'istanza di una tabella.
Tale valore è detto null e viene utilizzato per indicare una serie di situazioni che è possibile trovare in un campo di una tabella.
Ad esempio ci si può trovare di fronte ad una tabella del tipo:

| **Città** | **Indirizzo prefettura** |
|---|---|
| Roma | Via Quattro Novembre |
| Firenze | Null |
| Tivoli | Null |
| Prato | Null |

In questo esempio il valore null indica tre diverse situazioni:

· **Valore sconosciuto**: nel primo caso dato che Firenze è un capoluogo di provincia ed avrà sicuramente una prefettura; significa che il DBMS non dispone del suo indirizzo.
· **Valore inesistente**: nel caso di Tivoli, dato che questa città non ha di Prefettura.
· **Senza informazione**: siccome la provincia di Prato è di recente istituzione significa che non sappiamo se la mancanza di un indirizzo dipenda dal fatto che essa ancora non esista, oppure da un deficit del DBMS.

# Vincoli di integrità

I vincoli di integrità sono quei vincoli, caratteristica fondamentale delle basi di dati relazionali, che indicano la "bontà" delle informazioni. Ci possono essere, infatti, casi in cui i dati non rispettino, per una serie di motivi che andremo ad analizzare, l'integrità logica del database che, in quanto insieme formalizzato di informazioni, ha delle regole molto precise (e spesso rigide) che devono essere obbligatoriamente rispettate.

I vincoli di un DBMS si dividono in intrarelazionali, se riguardano l'interno nella relazioni, ed extrarelazionali, più pesanti perché riguardano i legami fra le relazioni e quindi la natura stessa di un DBMS relazionale.

Il più caratterizzante esempio di violazione di un vincolo extrarelazionale si ha qualora non vi sia corrispondenza tra le istanze di due tabelle che, per ragioni intrinseche allo schema della base di dati, sono legate tra loro da un vincolo detto di integrità referenziale, il quale impone che per ogni valore di una tabella vi sia un corrispondente nell'altra.

Un vincolo intrarelazionale, al contrario, trova il suo soddisfacimento rispetto alle singole relazioni del DBMS; esso può essere:

· **vincolo di tupla**; spiegato di seguito
· **vincolo su valori**, o **vincolo di dominio**: quando si impone a certi valori del database di rientrare in determinate caratteristiche (ad esempio il voto di un esame universitario deve obbligatoriamente rientrare tra il 18 ed il 30, ed un DBMS incaricato raccogliere tali dati deve prevederlo).

# Vincoli di tupla

Il vincolo di tupla è un vincolo che esprime delle condizioni sui valori di ciascuna tupla, indipendentemente dalle altre tuple.

Ad esempio consideriamo l'esempio dei voti degli esami universitari e ipotizziamo di avere un DBMS che debba memorizzarli; poniamo, inoltre, di destinare un campo all'eventuale lode (campo di tipo booleano, vero/falso).

In questo caso avremmo che l'attributo "lode" potrà essere vero solo se il voto di quella tupla è uguale a trenta. Cercando di dare una forma a tale vicolo potremmo scrivere che:

se ( voto <> 30 ) allora ( lode = falso )

mentre in questo caso il vincolo di dominio (citato nel paragrafo precedente) potrebbe essere espresso con la formula:

```
( voto >= 18 ) and ( voto <= 30 )
```

# Chiavi

Una chiave è un insieme di attributi (anche uno solo) utilizzato per identificare in maniera univoca le tuple di una relazione. Questo per la natura stessa del modello relazionale che pone questo vincolo come suo assioma.

Ne consegue che la violazione del vincolo di chiave, inammissibile in un DBMS relazionale, è la presenza di due o più tuple aventi la stessa chiave.

# Cenni di algebra e calcolo relazionale

L'algebra relazionale è un linguaggio procedurale, basato su concetti di tipo algebrico che deriva da quella parte della matematica che si occupa degli insiemi.
Siccome i concetti trattati sono molto astratti, poco legati al pratico utilizzo dei DBMS quanto a quello realizzativo, in questo capitolo ci limiteremo a descrivere i concetti generali senza entrare nel dettaglio delle formule e delle dimostrazioni.
Di seguito descriveremo gli operatori fondamentali dell'algebra relazionale, lo stretto necessario per comprendere da cosa deriva l' SQL.

## Algebra relazionale

L'algebra relazionale definisce una serie di operatori e regole per la creazione e la modifica delle relazioni. Di seguito vengono trattati i più comuni.

### *Unione*

Appurato che le relazioni sono insiemi, ha senso intervenire sulle relazioni con gli operatori dell'algebra insiemistica, quali l'unione la differenza e l'intersezione.
Tuttavia, dato che una relazione è un insieme di tuple omogenee (ovvero definite dagli stessi attributi) questi operatori potranno essere usati solo con relazioni aventi gli stessi attributi, o almeno degli attributi comuni.

Detto questo è possibile dire che l'unione tra due relazioni r1 ed r2, definite sullo stesso insieme di attributi X ( si scrive r1(X) ed r2(X) ), è indicata con r1Ur2 ed è una relazione ancora su X contenente le tuple che appartengono ad r1 oppure ad r2, oppure ad entrambe.

Nella fattispecie delle basi di dati l'unione di due tabelle, che come abbiamo visto sono le corrispondenti delle relazione, è l'insieme delle righe che appartengono alle tabelle ( alla prima, alla seconda o ad entrambe).

### *Intersezione*

L'intersezione di r1(X) ed r2(X) è indicata con r1  r2 ed è una relazione su X contenente le tuple che appartengono sia ad r1 che ad r2.

Riferito alle tabelle l'intersezione di due di queste è l'insieme delle righe che appartengono ad entrambe.

### *Differenza*

La differenza di r1(X) ed r2(x) è indicata con r1 - r2 ed è una relazione su X contenente le tuple che appartengono ad r1 e non appartengono ad r2.

Quindi la differenza tra due tabelle è l'insieme delle righe che appartengono alla prima tabella e non alla seconda.

### *Ridenominazione*
Per risolvere il problema derivante dal fatto che gli operatori sopra citati possono essere usati solo con attributi uguali, l'algebra relazionale ci mette a disposizione l'operatore di ridenominazione.
Questo operatore ci permette di ridefinire il nome dell'attributo in modo da operare operazioni di unione, differenza ed intersezioni anche su relazioni con diversi attributi (sempre ammesso che ciò possa avere un senso).

### *Selezione*
La selezione è quell'operatore dell'algebra relazionale che produce un sottoinsieme di tuple, producendo una relazione che ha gli stessi attributi dell'operando ma un numero minore (o uguale in casi particolari) di istanze.

La selezione introduce un concetto fondamentale dell'algebra relazionale e, di conseguenza, delle basi di dati : il concetto di condizione (trattato in maniera più esauriente nel capitolo successivo). Infatti la discriminante che permette di stabilire quali tuple andranno a formare il risultato della selezione è il fatto che essa rispondano meno ad una determinata condizione.

### *Proiezione*
La proiezione è un concetto simile a quello della selezione. Essa produce, alla pari della selezione, un sottoinsieme della relazione genitrice; a differenza della selezione, però, il sottoinsieme prodotto contiene tutte le tuple originali ma con un numero inferiore di attributi.

### *Join*
L'operatore di join è il più caratteristico dell'algebra relazionale.
Il join naturale è un operatore che correla i dati di due relazioni sulla base di valori uguali in attributi con lo stesso nome.

Esistono diversi tipi di join, a seconda delle diverse situazione di relazioni, che sono tuttavia riconducibili al modello del join naturale. Un'analisi più approfondita esula dai compiti di questo lavoro, volto all'aspetto pratico delle basi di dati. Ulteriori informazioni circa il join si possono trovare nel capitolo successivo, che mostra l'implementazione di tale operatore nel linguaggio SQL.

### *Viste*
La vista non è un operatore dell'algebra relazionale; più semplicemente è una tecnica dell'algebra che permette di mettere a disposizione dell'utente rappresentazioni diverse degli stessi dati.

La tecnica che ci permette di raggiungere questo scopo è quella delle relazioni derivate, relazioni il cui contenuto è funzione del contenuto di altre relazioni. Queste relazioni si contrappongono alle relazioni di base (le relazioni vere e proprie), il cui contenuto è autonomo. A livello di implementazioni si possono dividere le viste in due tipi:

· **viste materializzate**: relazioni derivate effettivamente memorizzate sulla base di dati;
· **viste virtuali**: relazioni definite per mezzo di funzioni, non memorizzate sulla base di dati, ma utilizzabili nelle interrogazioni come se effettivamente lo fossero.

# Creazione di Database

I database di un qualunque Motore DBMS sono rappresentati da un insieme di tabelle contenenti dati e da altri oggetti quali viste, indici, stored procedure e trigger, definiti per supportare le attività eseguite con i dati.
I dati archiviati in un database sono in genere correlati a un oggetto o processo particolare, quali le informazioni dell'inventario del magazzino di una fabbrica.

Ogni database consente di archiviare dati correlati o dati non correlati a quelli presenti in altri database. Ad esempio, è possibile che in un server sia disponibile un database in cui sono archiviati i dati sul personale e un altro in cui sono archiviati i dati correlati ai prodotti.
In alternativa, è possibile che in un database siano archiviati i dati aggiornati sugli ordini dei clienti e in un altro database correlato siano archiviati gli ordini precedenti dei clienti da utilizzare per la creazione di report annuali.

Prima di creare un database, è importante conoscere le parti che lo compongono e come progettarle per ottenere prestazioni ottimali dal database dopo l'implementazione.

I database includono un insieme di tabelle in cui viene archiviato uno specifico set di dati strutturati.
Le tabelle contengono un insieme di righe e colonne (a volte denominate attributi).
Le singole colonne delle tabelle vengono impostate in modo che contengano un tipo di informazioni specifico, ad esempio date, nomi, importi in valuta o numeri.
Alle tabelle sono associati numerosi tipi di controlli (vincoli, regole, trigger, valori predefiniti e tipi di dati definiti dall'utente) che garantiscono la validità dei dati.

Nelle tabelle è possibile includere indici, con caratteristiche molto simili a quelli dei libri, che consentono di trovare rapidamente le righe.

È possibile aggiungere i vincoli di integrità referenziale dichiarativa (DRI, Declarative Referential Integrity) alle tabelle per garantire che i dati interrelati delle diverse tabelle rimangano consistenti.

Nei database è inoltre possibile archiviare viste che consentono l'accesso personalizzato ai dati della tabella.

## Considerazioni sulla progettazione di database

Per progettare un database è necessario conoscere le funzioni business che si desidera modellare e le caratteristiche e i concetti relativi al database utilizzati per rappresentare le funzioni business.
È importante progettare un database in modo accurato per adattarlo alle proprie esigenze perché una significativa modifica alla struttura di un database già implementato può richiedere tempi lunghi.
Inoltre, le prestazioni di un database progettato in modo appropriato saranno migliori.
Durante la progettazione di un database, è necessario considerare:

· Le finalità del database e l'effetto di queste sulla struttura. Creare un piano di database adeguato alle finalità.
· Le regole di normalizzazione del database che impediscono di commettere errori durante la progettazione del database.
· La protezione dell'integrità dei dati.
· I requisiti di protezione del database e le autorizzazioni per gli utenti.
· I requisiti dell'applicazione in termini di prestazioni.

# Creazione di piani di database

Il primo passaggio della procedura di creazione di un database è la creazione di un piano da utilizzare come guida quando il database viene implementato e come specifica funzionale per il database dopo l'implementazione. La complessità e i dettagli della struttura di un database dipendono dalla complessità e dalle dimensioni dell'applicazione di database e dal tipo di utenti.

La natura e la complessità di un'applicazione di database e il processo di pianificazione possono variare notevolmente. Un database può essere relativamente semplice e progettato per essere utilizzato da un singolo utente oppure può essere di grandi dimensioni, complesso e progettato, ad esempio, per consentire la gestione di migliaia di client. Nel primo caso, per progettare il database potrebbe essere sufficiente creare una semplice bozza su un foglio di carta.
Nel secondo caso, è probabile che il progetto sia un documento formale con centinaia di pagine che includono tutti i dettagli sul database.
Durante la pianificazione di un database è consigliabile attenersi ai passaggi seguenti, indipendentemente dalle dimensioni e dalla complessità:

· Raccolta di informazioni.
· Identificazione degli oggetti.
· Modellazione degli oggetti.
· Identificazione dei tipi di informazioni per ogni oggetto.
· Identificazione delle relazioni tra oggetti.

### *Raccolta di informazioni*

Prima di creare un database, è necessario determinare in modo chiaro quale funzione dovrà essere svolta dal database. Se il database sostituirà un sistema di informazioni basato su materiale cartaceo o gestito manualmente, il sistema esistente fornirà la maggior parte delle informazioni necessarie.
È importante consultare tutti gli utenti che utilizzano il sistema per conoscere le loro mansioni e determinare in quale modo il database può soddisfare le loro esigenze. È inoltre importante identificare le funzioni che gli utenti desiderano vengano svolte dal nuovo sistema nonché individuare problemi, limiti e colli di bottiglia dei sistemi esistenti.
Infine, raccogliere copie di note informative sui clienti, di elenchi di inventario, di report amministrativi e altri documenti che fanno parte del sistema esistente perché saranno utili per progettare il database e le interfacce.
Identificazione degli oggetti
Durante il processo di raccolta delle informazioni, è necessario identificare gli oggetti o le entità chiave che verranno gestiti dal database.

L'oggetto può essere un elemento tangibile, ad esempio una persona o un prodotto, oppure può essere un elemento meno tangibile, quale una transazione business, il reparto di un'azienda o un periodo di retribuzione.
Esiste in genere un numero limitato di oggetti primari. Dopo che sono stati identificati, gli elementi correlati diventano evidenti. A ogni elemento distinto del database dovrebbe corrispondere una tabella.

Supponiamo che l'oggetto primario di un database di un'azienda sia un libro.
Gli oggetti correlati ai libri dell'attività di questa azienda sono gli autori che scrivono i libri, gli editori che li producono, i negozi che li vendono e le transazioni di vendita eseguite con i negozi. Ognuno di questi oggetti rappresenta una tabella nel database.

### *Modellazione di oggetti*

Quando gli oggetti del sistema vengono identificati, è importante registrarli in modo da rappresentare il sistema visivamente. È possibile utilizzare il modello di database come riferimento durante l'implementazione del database.

A tale scopo, gli sviluppatori di database utilizzano strumenti che per complessità variano da semplici appunti su carta a programmi di elaborazione di testi, fogli di calcolo e programmi dedicati alla modellazione di dati per la progettazione di database. Indipendentemente dallo strumento utilizzato, è importante mantenerlo aggiornato.

### *Identificazione dei tipi di informazioni per ogni oggetto*

Dopo aver identificato gli oggetti di database primari che si desidera inserire nelle tabelle, è necessario identificare i tipi di informazioni che si desidera archiviare per ogni oggetto, che diventeranno le colonne nella tabella dell'oggetto.
Le colonne di una tabella di database contengono alcuni tipi di informazioni comuni:

· Colonne di dati non elaborati
Tali colonne contengono informazioni tangibili, quali i nomi, determinate da un'origine esterna al database.

· Colonne di categoria
Tali colonne classificano o raggruppano i dati e contengono una selezione limitata di dati quali true/false, married/single, VP/Director/Group Manager e così via.

· Colonne identificatore
Tali colonne consentono di identificare ogni elemento archiviato nella tabella. I nomi di queste colonne indicano in genere che le colonne includono un ID o un numero (ad esempio, **ID_Impiegato**, **Num_Telefono**).
La colonna identificatore è il componente primario utilizzato dagli utenti e dagli strumenti di elaborazione interni al database per accedere a una riga di dati nella tabella. A volte l'oggetto si presenta nella forma tangibile di ID che è possibile utilizzare nella tabella (ad esempio, un numero di previdenza sociale), ma nella maggior parte delle situazioni è possibile definire la tabella in modo da creare un ID artificiale affidabile per la riga.

· Colonne relazionali o referenziali
Queste colonne stabiliscono un collegamento tra le informazioni di una tabella e le informazioni correlate di un'altra tabella. Ad esempio, una tabella che tiene traccia delle transazioni di

vendita viene in genere collegata alla tabella dei clienti per consentire di associare le informazioni complete sui clienti alla transazione di vendita.

### *Identificazione delle relazioni tra oggetti*

Un punto di forza dei database relazionali è rappresentato dalla possibilità di correlare o associare informazioni su elementi diversi del database.
È possibile archiviare in modo distinto tipi isolati di informazioni, ma il motore di database consente di combinare i dati se necessario.
Per identificare le relazioni tra oggetti durante il processo di progettazione, è necessario esaminare le tabelle, determinare in che modo sono correlate logicamente e aggiungere colonne relazionali che stabiliscono i collegamenti tra le singole tabelle.

Ad esempio, durante la progettazione di un database di libri verranno create le tabelle per i titoli e gli editori.
La tabella **Libri** include informazioni per ogni libro: una colonna identificatore denominata **ID_Libro**, colonne di dati non elaborati per il titolo, il prezzo del libro e la relativa data di pubblicazione nonché alcune colonne con le informazioni di vendita del libro.
La tabella include una colonna di categoria denominata **tipo** che consente di raggruppare i libri in base al tipo di contenuto.
A ogni libro è associato un editore, ma le informazioni sull'editore sono incluse in un'altra tabella. Pertanto, la colonna **ID_Editore** della tabella **Libri** includerà soltanto l'ID dell'editore. Quando viene aggiunta una riga di dati per un libro, l'ID dell'editore viene archiviato con le altre informazioni relative al libro.

## Tipi di Applicazione

Molti programmi rientrano in due categorie principali di applicazioni di database:

- Elaborazione delle transazioni in linea (OLTP, Online Transaction Processing)
- Supporto decisionale

Le caratteristiche di questi tipi di applicazioni hanno un effetto determinante sulle considerazioni relative alla progettazione di un database.

### *Elaborazione delle transazioni in linea*

Le applicazioni di database per l'elaborazione delle transazioni in linea sono appropriate per la gestione dei dati che vengono modificati e in genere supportano un numero elevato di utenti che eseguono contemporaneamente più transazioni per la modifica dei dati in tempo reale.

Sebbene le singole richieste di dati tendano a fare riferimento a pochi record, molte delle richieste vengono eseguite contemporaneamente. Esempi comuni di questi tipi di database sono rappresentati dai sistemi di biglietteria aerea e dai sistemi di transazione bancaria.
Gli aspetti più importanti da considerare per questo tipo di applicazioni sono la concorrenza e l'atomicità.

I controlli della concorrenza in un sistema di database assicurano che gli stessi dati non

vengano modificati da due utenti contemporaneamente oppure che un utente non modifichi determinati dati prima che un altro utente abbia terminato di modificarli.
Ad esempio, se un cliente chiede all'impiegato di una biglietteria aerea di prenotare l'ultimo posto disponibile su un volo e l'impiegato inizia il processo di prenotazione del posto a nome del cliente, un altro impiegato non dovrebbe essere in grado di trovare tale posto disponibile per assegnarlo a un altro passeggero.

L'atomicità assicura che tutti i passaggi di una transazione vengono ritenuti correttamente completati solo se considerati tutti come un unico gruppo.
Se un passaggio ha esito negativo, non sarà possibile completare gli altri passaggi. Ad esempio, una transazione bancaria può comportare due passaggi: prelievo di fondi da un conto corrente e deposito su un conto di risparmio.

Se il passaggio durante il quale vengono prelevati i fondi dal conto corrente ha esito positivo, è necessario verificare che l'importo venga depositato sul conto di risparmio o ridepositato sul conto corrente.

***Considerazioni sulla progettazione per i sistemi di elab. delle trans. in linea***
I database dei sistemi di elaborazione delle transazioni dovrebbero essere progettati in modo da garantire:

· Il posizionamento corretto dei dati.
I colli di bottiglia delle operazioni di I/O rappresentano un problema rilevante per i sistemi OLTP a causa del numero di utenti che modificano i dati in tutto il database. È opportuno determinare quali saranno le più probabili modalità di accesso ai dati e raggruppare i dati a cui si accede più di frequente. A tale scopo, utilizzare i filegroup e i sistemi RAID (matrice ridondante di dischi indipendenti, Redundant Array of Independent Disks).

· Transazioni brevi per ridurre al minimo i blocchi a lungo termine e ottimizzare la concorrenza.
Evitare l'interazione dell'utente durante le transazioni. Quando possibile, eseguire una singola stored procedure per elaborare l'intera transazione. L'ordine con cui si fa riferimento alle tabelle nelle transazioni può avere effetto sulla concorrenza. Inserire nella parte finale della transazione i riferimenti alle tabelle a cui si accede di frequente per ridurre al minimo la durata di mantenimento dei blocchi.

· Backup in linea.
I sistemi OLTP sono spesso caratterizzati da operazioni continue (24 ore al giorno, 7 giorni alla settimana) con periodi di inattività minimi. Sebbene in molti motori DBMS sia consentita l'esecuzione del backup di un database mentre viene utilizzato, è opportuno pianificare il processo di backup in modo che venga eseguito durante i periodi di minore attività per ridurre al minimo gli effetti sugli utenti.

· Elevata normalizzazione del database.
Ridurre il più possibile le informazioni ridondanti per aumentare la velocità degli aggiornamenti e quindi ottimizzare la concorrenza. La riduzione dei dati aumenta inoltre la velocità di esecuzione dei backup perché è necessario eseguire il backup di una minore quantità di dati.

· Assenza o riduzione al minimo di dati storici o aggregati.
I dati a cui si fa riferimento solo di rado possono essere archiviati in altri database oppure spostati dalle tabelle aggiornate più frequentemente alle tabelle che includono soltanto dati storici. In questo modo, le dimensioni delle tabelle vengono limitate al massimo e vengono

pertanto ottimizzati i tempi di esecuzione dei backup e le prestazioni delle query.

· Utilizzo accurato degli indici.
È necessario aggiornare gli indici ogni volta che una riga viene aggiunta o modificata. Per evitare una indicizzazione eccessiva delle tabelle aggiornate con maggiore frequenza, mantenere basso il numero di colonne degli indici. Per progettare gli indici, utilizzare l'Ottimizzazione guidata indici.

· Configurazione hardware ottimale per gestire il numero elevato di utenti concorrenti e i tempi di risposta veloci necessari in un sistema OLTP.

### *Supporto decisionale*
Le applicazioni di database per il supporto decisionale sono indicate per le query che non modificano i dati.

Ad esempio, un'azienda può riepilogare periodicamente i dati di vendita per data, area di vendita o per prodotto e archiviare queste informazioni in un database distinto da utilizzare per le analisi da parte dei responsabili.

Per prendere decisioni strategiche, gli utenti devono essere in grado di determinare rapidamente le tendenze delle vendite eseguendo query sui dati in base a vari criteri, ma non è necessario che modifichino questi dati.
Le tabelle in un database per il supporto decisionale sono intensamente indicizzate e i dati non elaborati vengono in genere pre-elaborati e organizzati per supportare i vari tipi di query da utilizzare. Poiché gli utenti non modificano i dati, la concorrenza e l'atomicità non rappresentano un problema.
I dati vengono modificati soltanto in occasione di aggiornamenti di massa periodici eseguiti durante gli orari di minore attività del database.

#### *Considerazioni sulla progettazione per i sistemi di supporto decisionale*
I database dei sistemi di supporto decisionale dovrebbero essere progettati in modo da garantire:

· Un'intensa indicizzazione.
I sistemi per il supporto decisionale gestiscono una quantità di aggiornamenti limitata ma volumi di dati estesi. Utilizzare numerosi indici per ottimizzare le prestazioni delle query.

· Denormalizzazione del database.
Introdurre dati preaggregati o riepilogati per soddisfare i requisiti comuni delle query e ottimizzare i tempi di risposta delle query stesse.

· Utilizzare uno schema a stella o a fiocco di neve per organizzare i dati all'interno del database.

# Normalizzazione

L'attività di progettazione logica del database, incluse le tabelle e le relazioni tra tabelle, è di fondamentale importanza per ottenere un database relazionale ottimizzato.
Se si esegue una progettazione logica corretta, si hanno buone probabilità di ottenere prestazioni ottimali dal database e dall'applicazione. Se si esegue una progettazione logica inadeguata, potrebbero verificarsi effetti negativi sulle prestazioni dell'intero sistema.

La normalizzazione della progettazione logica del database comporta l'utilizzo di metodi formali per la suddivisione dei dati in più tabelle correlate. Un numero maggiore di tabelle compatte (con un numero minore di colonne) è tipico di un database normalizzato.
Un numero minore di tabelle di grandi dimensioni (con un numero maggiore di colonne) è tipico di un database non normalizzato.

Una normalizzazione ragionevole spesso consente di ottimizzare le prestazioni. Quando sono disponibili indici utili, il motore DBMS seleziona in modo efficace join tra tabelle rapidi ed efficienti.

Di seguito vengono indicati alcuni vantaggi garantiti dalla normalizzazione:

· Ordinamento e creazione dell'indice più veloci.
· Indici cluster più numerosi.
· Indici più compatti e con un minor numero di colonne.
· Un numero minore di indici per tabella con conseguente ottimizzazione delle prestazioni delle istruzioni INSERT, UPDATE e DELETE.
· Un numero minore di valori Null e una minore probabilità di inconsistenze, con conseguente aumento della compattezza del database.

Aumentando il grado di normalizzazione, aumentano il numero e la complessità dei join necessari per recuperare i dati.

Un numero troppo elevato di join relazionali complessi tra un numero troppo elevato di tabelle può rallentare le prestazioni. Una normalizzazione ragionevole è spesso caratterizzata da un numero basso di query eseguite regolarmente le quali utilizzano join che coinvolgono più di quattro tabelle.

A volte la progettazione logica del database è già stata eseguita e una riprogettazione totale non è realizzabile. Anche in questo caso, tuttavia, sarà possibile normalizzare selettivamente una tabella di grandi dimensioni in diverse tabelle più piccole.

Se per accedere al database si utilizzano stored procedure, è possibile implementare questa modifica dello schema senza interessare le applicazioni. In caso contrario, è possibile creare una vista che nasconde la modifica dello schema alle applicazioni.

## *Progettazione ottimale dei database*

Nella teoria della progettazione di database relazionali, le regole di normalizzazione identificano alcuni attributi che devono essere presenti o assenti in un database ben progettato.
Una completa discussione sulle regole di normalizzazione esula dall'ambito di questo argomento.

Tuttavia, esistono alcune regole che consentono di ottenere una struttura di database affidabile:

· Includere un identificatore in ogni tabella.
La regola fondamentale della teoria della progettazione di database prevede di includere in ogni tabella un identificatore di riga univoco, una colonna o un set di colonne che è possibile utilizzare per distinguere ogni record dagli altri record della tabella. A ogni tabella è necessario associare una colonna di ID e non sarà possibile assegnare lo stesso ID a record diversi. Una o più colonne utilizzate come identificatore di riga univoco per una tabella rappresentano la chiave primaria della tabella.

· In una tabella è opportuno archiviare soltanto i dati di un solo tipo di entità.
Se si tenta di archiviare troppe informazioni in una tabella, la gestione efficiente e affidabile dei dati della tabella può essere ostacolata. Nel database dei Libri le informazioni su titoli ed editori sono archiviate in due tabelle distinte.
Nonostante nella tabella **Libri** sia possibile includere colonne con informazioni sia per il libro sia per l'editore del libro, questa struttura comporta diversi problemi.
Le informazioni sull'editore devono essere aggiunte e archiviate in modo ridondante per ogni libro pubblicato da un editore e questo comporta l'utilizzo di spazio di archiviazione aggiuntivo nel database. Se l'indirizzo dell'editore viene modificato, la modifica deve essere apportata per ogni libro. Se l'ultimo libro di un editore viene rimosso dalla tabella, le informazioni su tale editore vengono perdute.
Poiché nel database dei Libri le informazioni relative ai libri e agli editori sono archiviate nelle tabelle **Libri** e **Editori**, è possibile immettere le informazioni sull'editore una sola volta e quindi collegarle a ogni libro. Pertanto, se le informazioni sull'editore vengono modificate, la modifica deve essere apportata una sola volta e le informazioni sull'editore saranno presenti anche se nel database non sono inclusi libri associati a tale editore.

· In una tabella non includere colonne che supportano valori Null.
Nelle tabelle è possibile definire colonne che supportano valori Null. Un valore Null indica che non è presente alcun valore.
In alcuni casi può essere utile consentire l'inserimento di valori Null, ma in genere è preferibile utilizzarli solo se necessario perché richiedono una gestione speciale che aumenta la complessità delle operazioni sui dati.
Se è disponibile una tabella con più colonne che supportano valori Null e in diverse righe delle colonne sono inclusi valori Null, è consigliabile inserire tali colonne in un'altra tabella collegata alla tabella primaria. L'archiviazione dei dati in due tabelle distinte consente di ottenere una tabella primaria con una struttura semplice ma che consente di archiviare queste informazioni se si presenta la necessità.

· In una tabella non includere colonne o valori ripetuti.
La tabella di un elemento del database non deve includere l'elenco di valori relativi a un'informazione specifica. Ad esempio, è possibile che un libro nel database dei Libri sia stato scritto da più autori. Se nella tabella **Libri** è presente una colonna per il nome dell'autore, si verifica un problema.
Una soluzione consiste nell'archiviare i nomi dei coautori nella colonna, ma in questo modo è più difficile visualizzare un elenco dei singoli autori. Un'altra soluzione consiste nel modificare la struttura della tabella aggiungendo un'altra colonna per il nome del secondo autore, ma in questo modo è possibile specificare soltanto due autori. Se il libro è stato scritto da tre autori, sarà infatti necessario aggiungere un'altra colonna.
Se è necessario archiviare un elenco di valori in un'unica colonna o se sono disponibili più colonne per un unico dato (**Autore1**, **Autore2** e così via), è consigliabile inserire i dati duplicati in un'altra tabella con un collegamento alla tabella primaria.
Nel database dei Libri verrà creata una tabella per inserire le informazioni sui libri e un'altra

tabella in cui sono archiviati soltanto gli ID dei libri e gli ID degli autori dei libri.
Questa struttura consente di specificare un numero qualsiasi di autori per un libro senza modificare la definizione della tabella e non assegna lo spazio di archiviazione inutilizzato ai libri con un unico autore.

# Integrità dei dati

Se si ottiene l'integrità dei dati si garantisce la qualità dei dati nel database.
Ad esempio, se a un dipendente viene assegnato il numero di matricola (**ID_Impiegato**) **123** , il database non dovrebbe consentire che a un altro dipendente venga assegnato un ID con lo stesso valore.
Se è disponibile una colonna **Valutazione_Impiegato** nella quale si prevede di includere valori da **1** a **5**, il database non dovrebbe accettare il valore **6**. Se nella tabella è disponibile una colonna **ID_Reparto** nella quale viene archiviato il numero di reparto del dipendente, il database dovrebbe consentire soltanto l'inserimento dei valori validi per i numeri di reparto dell'azienda.
Due operazioni importanti per la pianificazione di tabelle sono identificare i valori validi per una colonna e determinare come ottenere l'integrità dei dati della colonna.
L'integrità dei dati rientra nelle seguenti categorie:

- Integrità di entità
- Integrità di dominio
- Integrità referenziale

## *Integrità di entità*

L'integrità di entità definisce una riga come entità univoca per una tabella specifica. L'integrità di entità determina l'integrità di una o più colonne identificatore o della chiave primaria di una tabella (tramite indici, vincoli UNIQUE, vincoli PRIMARY KEY o proprietà IDENTITY).

## *Integrità di dominio*

L'integrità di dominio è la validità delle voci di una colonna specifica. È possibile ottenere l'integrità di dominio limitando il tipo (tramite i tipi di dati), il formato (tramite i vincoli CHECK e le regole) o l'intervallo di valori possibili (tramite i vincoli FOREIGN KEY, i vincoli CHECK, le definizioni DEFAULT, le definizioni NOT NULL e le regole).

## *Integrità referenziale*

L'integrità referenziale mantiene le relazioni definite tra le tabelle quando i record vengono inseriti o eliminati. Solitamente in un motore DBMS, l'integrità referenziale si basa sulle relazioni tra le chiavi esterne e le chiavi primarie o tra le chiavi esterne e le chiavi univoche.

L'integrità referenziale garantisce che i valori di chiave siano consistenti tra le varie tabelle. Per ottenere tale consistenza, è necessario non fare riferimento a valori inesistenti e se viene modificato un valore di chiave, tutti i riferimenti a tale valore devono essere modificati in modo consistente in tutto il database.

# *Il Linguaggio SQL*

SQL è l'acronimo di *Standard Query Language* e identifica un linguaggio di interrogazione (gestione) per basi di dati relazionali. Le sue origini risalgono alla fine degli anni '70 e questo giustifica la sua sintassi prolissa e verbale tipica dei linguaggi dell'epoca, come il COBOL.

Allo stato attuale, data la sua evoluzione e standardizzazione, l'SQL rappresenta un riferimento fondamentale per la gestione di una base di dati relazionale.

A parte il significato originale dell'acronimo, SQL è un linguaggio completo per la gestione di una base di dati relazionale, includendo le funzionalità di un DDL (*Data Description Language*), di un DML (*Data Manipulation Language*) e di un DCL (*Data Control Language*).

Data l'età, e la conseguente evoluzione di questo linguaggio, si sono definiti nel tempo diversi livelli di standard. I più importanti sono: SQL89; SQL92 detto anche SQL2; SQL3.
Il livello SQL3 è ancora in corso di definizione.

L'aderenza dei vari sistemi DBMS allo standard SQL2 non è mai completa e perfetta, per questo sono stati definiti dei sottolivelli di questo standard per definire il grado di compatibilità di un DBMS. Si tratta di: *entry SQL*, *intermediate SQL* e *full SQL*. Si può intendere che il primo sia il livello di compatibilità minima e l'ultimo rappresenti la compatibilità totale.

Lo standard di fatto è rappresentato prevalentemente dal primo livello, che coincide fondamentalmente con lo standard precedente, SQL89.

## *Concetti Fondamentali*

Convenzionalmente, le istruzioni di questo linguaggio sono scritte con tutte le lettere maiuscole. Si tratta solo di una tradizione di quell'epoca. SQL non distingue tra lettere minuscole e maiuscole nelle parole chiave delle istruzioni e nemmeno nei nomi di tabelle, colonne e altri oggetti. Solo quando si tratta di definire il contenuto di una variabile, allora le differenze contano.

La tradizione richiederebbe che, quando si fa riferimento a istruzioni SQL, queste vengano indicate utilizzando solo lettere maiuscole; occorre però dire che alcuni motori di database accettano istruzioni scritte anche con lettere minuscole.

I nomi degli oggetti (tabelle e altro) possono essere composti utilizzando lettere, numeri e il simbolo di sottolineatura; il primo carattere deve essere una lettera oppure il simbolo di sottolineato.

Le istruzioni SQL possono essere distribuite su più righe, senza una regola precisa. Si distingue la fine di un'istruzione dall'inizio di un'altra attraverso la presenza di almeno una riga vuota. Alcuni sistemi SQL richiedono l'uso di un simbolo di terminazione delle righe, che potrebbe essere un punto e virgola.

L'SQL standard prevede la possibilità di inserire commenti; per questo si può usare un trattino doppio (**—**) seguito dal commento desiderato, fino alla fine della riga.

# Tipi di dati

I tipi di dati gestibili con il linguaggio SQL sono molti. Fondamentalmente si possono distinguere tipi contenenti: valori numerici, stringhe e informazioni data-orario. Nelle sezioni seguenti vengono descritti solo alcuni dei tipi definiti dallo standard.

## Stringhe di caratteri

Si distinguono due tipi di stringhe di caratteri in SQL: quelle a dimensione fissa, completate a destra dal carattere spazio, e quelle a dimensione variabile.

CHARACTER | CHARACTER(*<dimensione>*)

CHAR | CHAR(*<dimensione>*)

Quelle appena mostrate sono le varie sintassi alternative che possono essere utilizzate per definire una stringa di dimensione fissa. Se non viene indicata la dimensione tra parentesi, si intende una stringa di un solo carattere.

CHARACTER VARYING(*<dimensione>*)

CHAR VARYING(*<dimensione>*)

VARCHAR(*<dimensione>*)

Una stringa di dimensione variabile può essere definita attraverso uno dei tre modi appena elencati. È necessario specificare la dimensione massima che questa stringa potrà avere. Il minimo è rappresentato dalla stringa nulla.

## Costanti stringa

Le costanti stringa si esprimono delimitandole attraverso apici singoli, oppure apici doppi, come nell'esempio seguente:

'Questa è una stringa letterale per SQL'
"Anche questa è una stringa letterale per SQL"

Non tutti i sistemi SQL accettano entrambi i tipi di delimitatori di stringa. In caso di dubbio è bene limitarsi all'uso degli apici singoli.

## Valori numerici

I tipi numerici si distinguono in ***esatti*** e ***approssimati***, intendendo con la prima definizione quelli di cui si conosce il numero massimo di cifre numeriche intere e decimali, mentre con la seconda si fa riferimento ai tipi a virgola mobile.
In ogni caso, le dimensioni massime o la precisione massima che possono avere tali valori dipende dal sistema in cui vengono utilizzati.

NUMERIC | NUMERIC(*<precisione>*[,*<scala>*])

Il tipo **NUMERIC** permette di definire un valore numerico composto da un massimo di tante cifre numeriche quante indicate dalla precisione, cioè il primo argomento tra parentesi.

Se viene specificata anche la scala, si intende riservare quella parte di cifre per quanto appare dopo la virgola.
Per esempio, con **NUMERIC(5,2)** si possono rappresentare valori da +999.99 a -999.99.

Se non viene specificata la scala, si intende che si tratti solo di valori interi; se non viene specificata nemmeno la precisione, viene usata la definizione predefinita per questo tipo di dati, che dipende dalle caratteristiche del DBMS.

DECIMAL | DECIMAL( <*precisione*>[, <*scala*>])

DEC | DEC( <*precisione*>[, <*scala*>])

Il tipo **DECIMAL** è simile al tipo **NUMERIC**, con la differenza che le caratteristiche della precisione e della scala rappresentano le esigenze minime, mentre il sistema potrà fornire una rappresentazione con precisione o scala maggiore.

INTEGER | INT

SMALLINT

I tipi **INTEGER** e **SMALLINT** rappresentano tipi interi la cui dimensione dipende generalmente dalle caratteristiche del sistema operativo e dall'hardware utilizzato.
L'unico riferimento sicuro è che il tipo **SMALLINT** permette di rappresentare interi con una precisione inferiore o uguale al tipo **INTEGER**.

FLOAT | FLOAT( <*precisione*>)

REAL

DOUBLE PRECISION

Il tipo **FLOAT** definisce un tipo numerico approssimato (a virgola mobile) con una precisione binaria pari o superiore di quella indicata tra parentesi (se non viene indicata, dipende dal sistema).

Il tipo **REAL** e il tipo **DOUBLE PRECISION** sono due tipi a virgola mobile con una precisione prestabilita. Questa precisione dipende dal sistema, ma in generale, il secondo dei due tipi deve essere più preciso dell'altro.

### *Costanti numeriche*

I valori numerici costanti vengono espressi attraverso la semplice indicazione del numero senza delimitatori. La virgola di separazione della parte intera da quella decimale si esprime attraverso il punto (**.**).

### *Valori Data-orario e intervalli di tempo*

I valori data-orario sono di tre tipi e servono rispettivamente a memorizzare un giorno particolare, un orario normale e un'informazione data-ora completa.

DATE

TIME | TIME(*<precisione>*)

TIME WITH TIME ZONE | TIME(*<precisione>*) WITH TIME ZONE

TIMESTAMP | TIMESTAMP(*<precisione>*)

TIMESTAMP WITH TIME ZONE | TIMESTAMP(*<precisione>*) WITH TIME ZONE

Il tipo **DATE** permette di rappresentare delle date composte dall'informazione anno-mese-giorno.

Il tipo **TIME** permette di rappresentare un orario particolare, composto da ore-minuti-secondi ed eventualmente frazioni di secondo.
Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo.

Il tipo **TIMESTAMP** è un'informazione oraria più completa del tipo **TIME** in quanto prevede tutte le informazioni, dall'anno ai secondi, oltre alle eventuali frazioni di secondo.
Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo.

L'aggiunta dell'opzione **WITH TIME ZONE** serve a specificare un tipo orario differente, che assieme all'informazione oraria aggiunge lo scostamento, espresso in ore e minuti, dell'ora locale dal tempo universale (UTC).

Per esempio, 22:05:10+1:00 rappresenta le 22.05 e 10 secondi dell'ora locale italiana (durante l'inverno), e il tempo universale corrispondente sarebbe invece 21:05:10+0:00.

Quanto mostrato fino a questo punto, rappresenta un valore che indica un momento preciso nel tempo: una data o un'orario, o entrambe le cose. Per rappresentare una durata, si parla di intervalli.

Per l'SQL si possono gestire gli intervalli a due livelli di precisione: anni e mesi; oppure giorni, ore, minuti, secondi, ed eventualmente anche le frazioni di secondo.
L'intervallo si indica con la parola chiave **INTERVAL**, seguita eventualmente dalla precisione con qui questo deve essere rappresentato:

INTERVAL [ *<unità-di-misura-data-orario>* [ TO *<unità-di-misura-data-orario>* ] ]

In pratica, si può indicare che si tratta di un intervallo, senza specificare altro, oppure si possono definire una o due unità di misura che limitano la precisione di questo (pur restando nei limiti a cui si è già accennato).

Tanto per fare un esempio concreto, volendo definire un'intervallo che possa esprimere solo ore e minuti, si potrebbe dichiarare con: **INTERVAL HOUR TO MINUTE**.

La tabella elenca le parole chiave che rappresentano queste unità di misura.

| Parola chiave | Significato |
|---|---|
| YEAR | Anni |
| MONTH | Mesi |
| DAY | Giorni |
| HOUR | Ore |
| MINUTE | Minuti |
| SECOND | Secondi |

### *Costanti data-orario*

Le costanti che rappresentano informazioni data-orario sono espresse come le stringhe, delimitate tra apici.
Il sistema DBMS potrebbe ammettere più forme differenti per l'inserimento di queste, ma i modi più comuni dovrebbero essere quelli espressi dagli esempi seguenti.

```
'1999-12-31'
'12/31/1999'
'31.12.1999'
```

Questi tre esempi rappresentano la stessa data: il 31 dicembre 1999. Per una questione di uniformità, dovrebbe essere preferibile il primo di questi formati, corrispondente allo stile ISO 8601.
Anche gli orari che si vedono sotto, sono aderenti allo stile ISO 8601; in particolare per il fatto che il fuso orario viene indicato attraverso lo scostamento dal tempo universale, invece che attraverso una parola chiave che definisca il fuso dell'ora locale.

```
'12:30:50+1.00'
'12:30:50.10'
'12:30:50'
'12:30'
```

Il primo di questa serie di esempi rappresenta un orario composto da ore, minuti e secondi, oltre all'indicazione dello scostamento dal tempo universale (per ottenere il tempo universale deve essere sottratta un'ora).
Il secondo esempio mostra un orario composto da ore, minuti, secondi e centesimi di secondo. Il terzo e il quarto sono rappresentazioni normali, in particolare nell'ultimo è stata omessa l'indicazione dei secondi.

```
'1999-12-31 12:30:50+1.00'
'1999-12-31 12:30:50.10'
'1999-12-31 12:30:50'
'1999-12-31 12:30'
```

Gli esempi mostrano la rappresentazione di informazioni data-orario complete per il tipo **TIMESTAMP**.
La data è separata dall'ora da uno spazio.

### *Costanti che esprimono intervalli*

Un'informazione che rappresenta un intervallo di tempo inizia sempre con la parola chiave **INTERVAL**, ed è seguita da una stringa che contiene l'indicazione di uno o più valori, seguiti ognuno dall'unità di misura relativi (ammesso che ciò sia necessario).
Si osservino i due esempi seguenti:

```
INTERVAL '12 HOUR 30 MINUTE 50 SECOND'

INTERVAL '12:30:50'
```

Queste due forme rappresentano entrambe la stessa cosa: una durata di 12 ore, 30 minuti e 50 secondi. In generale, dovrebbe essere preferibile la seconda delle due forme di rappresentazione, ma nel caso di unità più grandi, diventa impossibile.

```
INTERVAL '10 DAY 12 HOUR 30 MINUTE 50 SECOND'

INTERVAL '10 DAY 12:30:50'
```

Come prima, i due esempi che si vedono sopra sono equivalenti. Intuitivamente, si può osservare che non ci può essere un altro modo di esprimere una durata in giorni, senza specificarlo attraverso la parola chiave **DAY**.

Per completare la serie di esempi, si aggiungono anche i casi in cui si rappresentano esplicitamente quantità molto grandi, e per questo approssimate al mese (come richiede lo standard SQL92):

```
INTERVAL '10 YEAR 11 MONTH'

INTERVAL '10 YEAR'
```

Gli intervalli di tempo possono servire per indicare un tempo trascorso rispetto al momento attuale. Per specificare espressamente questo fatto, si indica l'intervallo come un valore negativo, aggiungendo all'inizio un trattino (il segno meno).

```
INTERVAL '- 10 YEAR 11 MONTH'
```

L'esempio che si vede sopra, esprime precisamente 10 anni e 11 mesi fa.

# *Operatori, funzioni ed espressioni*

SQL, pur non essendo un linguaggio di programmazione completo, mette a disposizione una serie di operatori e di funzioni utili per la realizzazione di espressioni di vario tipo.

## *Operatori aritmetici*

Gli operatori che intervengono su valori numerici sono elencati nella tabella.

| Operazione | Descrizione |
|---|---|
| -*<op>* | Inverte il segno dell'operando. |
| *<op1>* + *<op2>* | Somma i due operandi. |
| *<op1>* - *<op2>* | Sottrae dal primo il secondo operando. |
| *<op1>* * *<op2>* | Moltiplica i due operandi. |
| *<op1>* / *<op2>* | Divide il primo operando per il secondo. |
| *<op1>* % *<op2>* | Modulo: il resto della divisione tra il primo e il secondo operando. |

Nelle espressioni, tutti i tipi numerici esatti e approssimati possono essere usati senza limitazioni. Dove necessario, il sistema provvede a eseguire le conversioni di tipo.

## *Operazioni con i valori data-orario e intervallo*

Le operazioni che si possono compiere utilizzando valori data-orario e intervallo, hanno significato solo in alcune circostanze. La tabella elenca le operazioni possibili e il tipo di risultato che si ottiene in base al tipo di operatori utilizzato.

| Operazione | Risultato |
|---|---|
| *<data-orario>* - *<data-orario>* | Intervallo |
| *<data-orario>* +\|- *<intervallo>* | Data-orario |
| *<intervallo>* + *<data-orario>* | Data-orario |
| *<intervallo>* +\|- *<intervallo>* | Intervallo |
| *<intervallo>* *\|/ *<numerico>* | Intervallo |
| *<numerico>* * *<intervallo>* | Intervallo |

## *Operatori di confronto e operatori logici*

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano: *Vero* o *Falso*. Gli operatori di confronto sono elencati nella tabella.

| Operazione | Descrizione |
|---|---|
| *<op1>* = *<op2>* | *Vero* se gli operandi si equivalgono. |
| *<op1>* <> *<op2>* | *Vero* se gli operandi sono differenti. |
| *<op1>* < *<op2>* | *Vero* se il primo operando è minore del secondo. |
| *<op1>* > *<op2>* | *Vero* se il primo operando è maggiore del secondo. |
| *<op1>* <= *<op2>* | *Vero* se il primo operando è minore o uguale al secondo. |
| *<op1>* >= *<op2>* | *Vero* se il primo operando è maggiore o uguale al secondo. |

Quando si vogliono combinare assieme diverse espressioni logiche si utilizzano gli operatori logici. Come in tutti i linguaggi di programmazione, si possono usare le parentesi tonde per raggruppare le espressioni logiche in modo da chiarire l'ordine di risoluzione. Gli operatori logici sono elencati nella tabella.

| Operazione | Descrizione |
|---|---|
| NOT *<op>* | Inverte il risultato logico dell'operando. |
| *<op1>* AND *<op2>* | *Vero* se entrambi gli operandi restituiscono il valore *Vero*. |
| *<op1>* OR *<op2>* | *Vero* se almeno uno degli operandi restituisce il valore *Vero*. |

Il meccanismo di confronto tra due operandi numerici è evidente, mentre può essere meno evidente con le stringhe di caratteri.
Per la precisione, il confronto tra due stringhe avviene senza tenere conto degli spazi finali, per cui, le stringhe **'ciao'** e **'ciao '** dovrebbero risultare uguali attraverso il confronto di uguaglianza con l'operatore **=**.

Con le stringhe, tuttavia, si possono eseguire dei confronti basati su modelli, attraverso gli operatori **IS LIKE** e **IS NOT LIKE**. Il modello può contenere dei metacaratteri rappresentati dal simbolo di sottolineato (_), che rappresenta un carattere qualsiasi, e dal simbolo di percentuale (**%**), che rappresenta una sequenza qualsiasi di caratteri.
La tabella riassume quanto detto.

| Espressioni e modelli | Descrizione |
|---|---|
| *<stringa>* IS LIKE *<modello>* | *Vero* se il modello corrisponde alla stringa. |
| *<stringa>* IS NOT LIKE *<modello>* | *Vero* se il modello non corrisponde alla stringa. |

| | |
|---|---|
| [ _ ] | Rappresenta un carattere qualsiasi. |
| [ %] | Rappresenta una sequenza indeterminata di caratteri. |

La presenza di valori indeterminati impone la presenza di operatori di confronto in grado di determinarne l'esistenza. La tabella riassume gli operatori ammissibili in questi casi.

| Operatori | Descrizione |
|---|---|
| *<espress.>* IS NULL | *Vero* se l'espressione genera un risultato indeterminato. |
| *<espress.>* IS NOT NULL | *Vero* se l'espressione non genera un risultato indeterm. |

Infine, occorre considerare una categoria particolare di espressioni che permettono di verificare l'appartenenza di un valore a un intervallo o a un elenco di valori.
La tabella riassume gli operatori utilizzabili.

| Operatori e operandi | Descrizione |
|---|---|
| *<op1>* IN (*<elenco>*) | *Vero* se il primo operando è contenuto nell'elenco. |
| *<op1>* NOT IN (*<elenco>*) | *Vero* se il primo operando non è contenuto nell'elenco. |
| *<op1>* BETWEEN *<op2>* AND *<op3>* | *Vero* se il primo operando è compreso tra il secondo e il terzo. |
| *<op1>* NOT BETWEEN *<op2>* AND *<op3>* | *Vero* se il primo operando non è compreso nell'intervallo. |

### *Funzioni Numeriche*

| | |
|---|---|
| ABS(n) | valore assoluto di n. |
| ROUND(n[,m]) | n arrotond. a m cifre decimali; m=0 di default; m puo' essere negativo. |
| TRUNC(n[,m]) | n troncato a m cifre decimali; m=0 di default; m puo' essere negativo. |
| SIGN(n) | 1 se n e' positivo; 0 se n e' 0; -1 se n e' negativo. |
| CEIL(n) | il piu' piccolo intero maggiore o uguale a n |
| FLOOR(n) | il piu' grande intero minore o uguale a n |
| MOD(n,m) | il resto della divisione di n per m |
| POWER(n,m) | n elevato alla m |
| SQRT(n) | radice quadrata di n |

### *Funzioni su Stringhe*

| | |
|---|---|
| SUBSTR(char,m[,n]) | una sottostringa di char, che inizia al carattere m, lunga n byte (se n manca, lunga fino alla fine della stringa char) |
| LENGTH(char) | lunghezza della stringa char in byte |
| CHR(n) | carattere con valore ASCII n |
| ASCII(char) | valore ASCII del primo carattere della stringa char |
| UPPER(char) | stringa char con tutte le lettere maiuscole |
| LOWER(char) | stringa char con tutte le lettere minuscole |
| INITCAP(char) | stringa char con l' iniziale di ogni parola maiuscola |
| REPLACE(char,str1[,str2]) | char con ogni occorrenza di string1 sostituita da string2 (se manca string2, string1 viene cancellata) |
| TRANSLATE(char,from,to) | char con ogni carattere presente in from sostituito col corrispondente carattere di to |
| RPAD(char1,n[,char2]) | char1, riempito a destra di char2 fino alla lunghezza n |
| LPAD(char1,n[,char2]) | char1, riempito a sinistra di char2 fino alla lunghezza n |
| RTRIM(char[,set]) | char, con i caratteri finali cancellati dopo l' ultimo car. non in set |
| LTRIM(char[,set]) | char, con i car. iniziali cancellati prima del primo car. non in set |

# Tabelle

SQL tratta le «relazioni» attraverso il modello tabellare, e di conseguenza si adegua tutta la sua filosofia e il modo di esprimere i concetti nella sua documentazione. Le tabelle di SQL vengono definite nel modo seguente dalla documentazione standard.

La tabella è un insieme di più righe. Una riga è una sequenza non vuota di valori. Ogni riga della stessa tabella ha la stessa cardinalità e contiene un valore per ogni colonna di quella tabella. L'*i*-esimo valore di ogni riga di una tabella è un valore dell'*i*-esima colonna di quella tabella. La riga è l'elemento che costituisce la più piccola unità di dati che può essere inserita in una tabella e cancellata da una tabella.

Il grado di una tabella è il numero di colonne della stessa. In ogni momento, il grado della tabella è lo stesso della cardinalità di ognuna delle sue righe, e la cardinalità della tabella (cioè il numero delle righe contenute) è la stessa della cardinalità di ognuna delle sue colonne. Una tabella la cui cardinalità sia zero viene definita come vuota.

In pratica, la tabella è un contenitore di informazioni organizzato in righe e colonne. La tabella viene identificata per nome, così anche le colonne, mentre le righe vengono identificate attraverso il loro contenuto.

Nel modello di SQL, le colonne sono ordinate, anche se ciò non è sempre un elemento indispensabile, dal momento che si possono identificare per nome. Inoltre sono ammissibili tabelle contenenti righe duplicate.

## *Creazione di una tabella*

La creazione di una tabella avviene attraverso un'istruzione che può assumere un'articolazione molto complessa, a seconda delle caratteristiche particolari che da questa tabella si vogliono ottenere. La sintassi più semplice è quella seguente:

CREATE TABLE *<nome-tabella>* ( *<specifiche>* )

Tuttavia, sono proprio le specifiche indicate tra le parentesi tonde che possono tradursi in un sistema molto confuso.
La creazione di una tabella elementare può essere espressa con la sintassi seguente:

CREATE TABLE *<nome-tabella>* (*<nome-colonna>* *<tipo>*[,...])

In questo modo, all'interno delle parentesi vengono semplicemente elencati i nomi delle colonne seguiti dal tipo di dati che in esse possono essere contenuti.

L'esempio seguente rappresenta l'istruzione necessaria a creare una tabella composta da cinque colonne, contenenti rispettivamente informazioni su: codice, cognome, nome, indirizzo e numero di telefono.

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40),
        Nome            char(40),
        Indirizzo       varchar(60),
        Telefono        varchar(40)
)
```

### *Valori predefiniti*

Quando si inseriscono delle righe all'interno della tabella, in linea di principio è possibile che i valori corrispondenti a colonne particolari non siano inseriti esplicitamente.

Se si verifica questa situazione (purché ciò sia consentito dai vincoli), viene attribuito a questi elementi mancanti un valore predefinito. Questo può essere stabilito all'interno delle specifiche di creazione della tabella, e se questo non è stato fatto, viene attribuito **NULL**, corrispondente al valore indefinito.

La sintassi necessaria a creare una tabella contenente le indicazioni sui valori predefiniti da utilizzare è la seguente:

```
CREATE TABLE <nome-tabella> (
        <nome-colonna> <tipo> [DEFAULT <espressione>] [,...]
)
```

L'esempio seguente crea la stessa tabella già vista nell'esempio precedente, specificando come valore predefinito per l'indirizzo, la stringa di caratteri: «sconosciuto».

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40),
        Nome            char(40),
        Indirizzo       varchar(60)  DEFAULT 'sconosciuto',
        Telefono        varchar(40)
)
```

### *Vincoli interni alla tabella*

Può darsi che in certe situazioni, determinati valori all'interno di una riga non siano ammissibili, a seconda del contesto a cui si riferisce la tabella.
I vincoli interni alla tabella sono quelli che possono essere risolti senza conoscere informazioni esterne alla tabella stessa.

Il vincolo più semplice da esprimere è quello di non ammissibilità dei valori indefiniti. La sintassi seguente ne mostra il modo.

```
CREATE TABLE <nome-tabella> (
        <nome-colonna> <tipo> [NOT NULL] [,...]
)
```

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti, specificando che il codice, il cognome, il nome e il telefono non possono essere indeterminati.

```
CREATE TABLE Indirizzi (
        Codice          integer NOT NULL,
        Cognome         char(40) NOT NULL,
        Nome            char(40) NOT NULL,
        Indirizzo       varchar(60) DEFAULT 'sconosciuto',
        Telefono        varchar(40) NOT NULL
)
```

Un altro vincolo importante è quello che permette di definire che un gruppo di colonne deve rappresentare dati unici in ogni riga, cioè che non siano ammissibili righe che per quel gruppo di colonne abbiano dati uguali. Segue lo schema sintattico relativo.

```
CREATE TABLE <nome-tabella> (
        <nome-colonna> <tipo> [,...], UNIQUE ( <nome-colonna> [,...] ) [,...]
)
```

L'indicazione dell'unicità può riguardare più gruppi di colonne in modo indipendente.
Per ottenere questo si possono indicare più opzioni **UNIQUE**.

È il caso di osservare che il vincolo **UNIQUE** non implica che i dati non possano essere indeterminati. Infatti, il valore indeterminato, **NULL**, è diverso da ogni altro **NULL**.

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti, specificando che i dati della colonna del codice devono essere unici per ogni riga.

```
CREATE TABLE Indirizzi (
        Codice          integer NOT NULL,
        Cognome         char(40) NOT NULL,
        Nome            char(40) NOT NULL,
        Indirizzo       varchar(60) DEFAULT 'sconosciuto',
        Telefono        varchar(40) NOT NULL,
        UNIQUE (Codice)
)
```

Quando una colonna, o un gruppo di colonne, costituisce un riferimento importante per identificare le varie righe che compongono la tabella, si può utilizzare il vincolo **PRIMARY KEY**, che può essere utilizzato una sola volta.

Questo vincolo stabilisce anche che i dati contenuti, oltre a non poter essere doppi, non possono essere indefiniti.

```
CREATE TABLE <nome-tabella> (
        <nome-colonna> <tipo> [,...], PRIMARY KEY ( <nome-colonna>[,...] )
)
```

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti specificando che la colonna del codice deve essere considerata la chiave primaria.

```
CREATE TABLE Indirizzi (
        Codice          integer NOT NULL,
        Cognome         char(40) NOT NULL,
        Nome            char(40) NOT NULL,
        Indirizzo       varchar(60) DEFAULT 'sconosciuto',
        Telefono        varchar(40) NOT NULL,
        PRIMARY KEY (Codice)
)
```

### *Vincoli esterni alla tabella*

I vincoli esterni alla tabella riguardano principalmente la connessione con altre tabelle, e la necessità che i riferimenti a queste siano validi. La definizione formale di questa connessione è molto complessa e qui non viene descritta.
Si tratta, in ogni caso, dell'opzione **FOREIGN KEY** seguita da **REFERENCES**.

Vale la pena però di considerare i meccanismi che sono coinvolti. Infatti, nel momento in cui si inserisce un valore, il sistema può impedire l'operazione perché non valida in base all'assenza di quel valore in un'altra tabella esterna specificata.

Il problema nasce però nel momento in cui nella tabella esterna viene eliminata o modificata una riga che era oggetto di un riferimento da parte della prima. Si pongono le alternative seguenti.

**CASCADE**
Se nella tabella esterna il dato a cui si fa riferimento è stato cambiato, viene cambiato anche il riferimento nella tabella di partenza; se nella tabella esterna la riga corrispondente viene rimossa, viene rimossa anche la riga della tabella di partenza.

**SET NULL**
Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore indefinito.

**SET DEFAULT**
Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore predefinito.

**NO ACTION**
Se viene a mancare l'oggetto a cui si fa riferimento, non viene modificato il dato contenuto nella tabella di partenza.

Le azioni da compiere si possono distinguere in base all'evento che ha causato la rottura del riferimento: cancellazione della riga della tabella esterna o modifica del suo contenuto.

### *Modifica della struttura della tabella*

La modifica della struttura di una tabella riguarda principalmente la sua organizzazione in colonne.
Le cose più semplici che si possono desiderare di fare sono l'aggiunta di nuove colonne e l'eliminazione di colonne esistenti. Vedendo il problema in questa ottica, la sintassi si riduce ai due casi seguenti.

```
ALTER TABLE <nome-tabella> (
        ADD [COLUMN] <nome-colonna> <tipo> [<altre caratteristiche>]
)

ALTER TABLE <nome-tabella> (
        DROP [COLUMN] <nome-colonna>
)
```

Nel primo caso si aggiunge una colonna, della quale si deve specificare il nome e il tipo, ed eventualmente si possono specificare i vincoli; nel secondo si tratta solo di indicare la colonna da eliminare.
A livello di singola colonna può essere eliminato o attribuito un valore predefinito.

```
ALTER TABLE <nome-tabella> (
        ALTER [COLUMN] <nome-colonna> DROP DEFAULT
)

ALTER TABLE <nome-tabella> (
        ALTER [COLUMN] <nome-colonna> SET DEFAULT <valore-predefinito>
)
```

### *Eliminazione di una tabella*

L'eliminazione di una tabella, con tutto il suo contenuto, è un'operazione semplice che dovrebbe essere autorizzata solo all'utente che l'ha creata.

```
DROP TABLE <nome-tabella>
```

# *Inserimento, eliminazione e modifica dei dati*

L'inserimento, l'eliminazione e la modifica dei dati di una tabella è un'operazione che interviene sempre a livello delle righe. Infatti, come già definito, la riga è l'elemento che costituisce l'unità di dati più piccola che può essere inserita o cancellata da una tabella.

## *Inserimento di righe*

L'inserimento di una nuova riga all'interno di una tabella viene eseguito attraverso l'istruzione **INSERT**.
Dal momento che nel modello di SQL le colonne sono ordinate, è sufficiente indicare ordinatamente l'elenco dei valori della riga da inserire, come mostra la sintassi seguente:

```
INSERT INTO <nome-tabella>
        VALUES (<espressione-1>[,...<espressione-N>])
```

Per esempio, l'inserimento di una riga nella tabella **Indirizzi** già mostrata in precedenza, potrebbe avvenire nel modo seguente:

```
INSERT INTO Indirizzi
        VALUES (01, 'Pallino', 'Pinco', 'Via Biglie 1', '0222,222222')
```

Se i valori inseriti sono meno del numero delle colonne della tabella, i valori mancanti, in coda, ottengono quanto stabilito come valore predefinito, o **NULL** in sua mancanza (sempre che ciò sia concesso dai vincoli della tabella).

L'inserimento dei dati può avvenire in modo più chiaro e sicuro elencando prima i nomi delle colonne, in modo da evitare di dipendere dalla sequenza delle colonne memorizzata nella tabella. La sintassi seguente mostra il modo di ottenere questo.

```
INSERT INTO <nome-tabella> (<colonna-1>[,...<colonna-N>])]
        VALUES (<espressione-1>[,...<espressione-N>])
```

L'esempio già visto potrebbe essere tradotto nel modo seguente, più prolisso, ma anche più chiaro.

```
INSERT INTO Indirizzi (Codice, Cognome, Nome, Indirizzo, Telefono)
        VALUES (01, 'Pallino', 'Pinco', 'Via Biglie 1', '0222,222222')
```

Questo modo esplicito di fare riferimento alle colonne garantisce anche che eventuali modifiche di lieve entità nella struttura della tabella non debbano necessariamente riflettersi nei programmi.
L'esempio seguente mostra l'inserimento di alcuni degli elementi della riga, lasciando che gli altri ottengano l'assegnamento di un valore predefinito.

```
INSERT INTO Indirizzi (Codice, Cognome, Nome, Telefono)
        VALUES (01, 'Pallino', 'Pinco', '0222,222222')
```

### *Aggiornamento delle righe*

La modifica delle righe può avvenire attraverso una scansione della tabella, dalla prima all'ultima riga, eventualmente controllando la modifica in base all'avverarsi di determinate condizioni.
La sintassi per ottenere questo risultato, leggermente semplificata, è la seguente:

```
UPDATE <tabella>
        SET <colonna-1>=<espressione-1>[,...<colonna-N>=<espressione-N>]
 [WHERE <condizione>]
```

L'istruzione **UPDATE** esegue tutte le sostituzioni indicate dalle coppie *<colonna>=<espressione>*, per tutte le righe in cui la condizione posta dopo la parola chiave **WHERE** si avvera.
Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le righe della tabella.

L'esempio seguente aggiunge una colonna alla tabella degli indirizzi, per contenere il nome del comune di residenza; successivamente viene inserito il nome del comune «Sferopoli» in base al prefisso telefonico.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30)


UPDATE Indirizzi
        SET Comune='Sferopoli'
WHERE Telefono >= '022' AND Telefono < '023'
```

Eventualmente, al posto dell'espressione si può indicare la parola chiave **DEFAULT** che fa in modo di assegnare il valore predefinito per quella colonna.

### *Eliminazione di righe*

La cancellazione di righe da una tabella è un'operazione molto semplice. Richiede solo l'indicazione del nome della tabella e la condizione in base alla quale le righe devono essere cancellate.

```
DELETE FROM <tabella> [WHERE <condizione>]
```

**N.B. Se la condizione non viene indicata, si cancellano tutte le righe!**

# *Interrogazioni di tabelle*

L'interrogazione di una tabella è l'operazione con cui si ottengono i dati contenuti al suo interno, in base a dei criteri di filtro determinati. L'interrogazione consente anche di combinare assieme dati provenienti da tabelle differenti, in base a delle relazioni che possono intercorrere tra queste.

## *Interrogazioni elementari*

La forma più semplice di esprimere la sintassi necessaria a interrogare **una** sola tabella è quella espressa dallo schema seguente:

SELECT *<espress-col-1>*[,...*<espress-col-N>*]
FROM *<tabella>*
[WHERE *<condizione>*]

In questo modo è possibile definire le colonne che si intendono utilizzare per il risultato, e le righe si specificano, eventualmente, con la condizione posta dopo la parola chiave **WHERE**.

L'esempio seguente mostra la proiezione delle colonne del cognome e nome della tabella di indirizzi già vista negli esempi delle altre sezioni, senza porre limiti alle righe.

```
SELECT Cognome, Nome FROM Indirizzi
```

Quando si vuole ottenere una selezione composta dalle stesse colonne della tabella originale, nel suo stesso ordine, si può utilizzare un carattere jolly particolare, l'asterisco (*****).
Questo rappresenta l'elenco di tutte le colonne della tabella indicata.

```
SELECT * FROM Indirizzi
```

È bene osservare che le colonne si esprimono attraverso un'espressione, questo significa che le colonne a cui si fa riferimento sono quelle del risultato finale, cioè della tabella che viene restituita come selezione o proiezione della tabella originale.

L'esempio seguente emette una sola colonna contenente un ipotetico prezzo scontato del 10%, in pratica viene moltiplicato il valore di una colonna contenente il prezzo per 0,90, in modo da ottenerne il 90% (100% meno lo sconto).

```
SELECT Prezzo * 0.90 FROM Listino
```

In questo senso si può comprendere l'utilità di attribuire esplicitamente un nome alle colonne del risultato finale, come indicato dalla sintassi seguente:

SELECT *<espress-col-1>* AS *<nome-col-1>*
[,...*<espress-col-N>* AS *<nome-col-N>*]
FROM *<tabella>*
[WHERE*<condizione>*]

In questo modo, l'esempio precedente può essere trasformato come segue, dando un nome alla colonna generata e chiarendone così il contenuto.

```
SELECT Prezzo * 0.90 AS Prezzo_Scontato FROM Listino
```

Finora è stata volutamente ignorata la condizione che controlla le righe da selezionare. Anche se potrebbe essere evidente, è bene chiarire che la condizione posta dopo la parola chiave **WHERE** può fare riferimento solo ai dati originali della tabella da cui si attingono. Quindi, non è valida una condizione che utilizza un riferimento a un nome utilizzato dopo la parola chiave **AS** abbinata alle espressioni delle colonne.

Per qualche motivo che verrà chiarito in seguito, può essere conveniente attribuire un alias alla tabella da cui estrarre i dati.
Anche in questo caso si utilizza la parola chiave **AS**, come indicato dalla sintassi seguente:

```
SELECT <specificazione-della-colonna-1>
        [,...<specificazione-della-colonna-N>]
FROM <tabella> AS <alias>
[WHERE <condizione>]
```

Quando si vuole fare riferimento al nome di una colonna, se per qualche motivo questo nome dovesse risultare ambiguo, si può aggiungere anteriormente il nome della tabella a cui appartiene, separandolo attraverso l'operatore punto (**.**).
L'esempio seguente è la proiezione dei cognomi e dei nomi della solita tabella degli indirizzi. In questo caso, le espressioni delle colonne rappresentano solo le colonne corrispondenti della tabella originaria, con l'aggiunta dell'indicazione esplicita del nome della tabella stessa.

```
SELECT Indirizzi.Cognome, Indirizzi.Nome FROM Indirizzi
```

A questo punto, se al nome della tabella viene abbinato un alias, si può esprimere la stessa cosa indicando il nome dell'alias al posto di quello della tabella, come nell'esempio seguente:

```
SELECT Ind.Cognome, Ind.Nome FROM Indirizzi AS Ind
```

## *Interrogazioni ordinate*

Per ottenere un elenco ordinato in base a qualche criterio, si utilizza l'istruzione **SELECT** con l'indicazione di un'espressione in base alla quale effettuare l'ordinamento.
Questa espressione è preceduta dalle parole chiave **ORDER BY**:

```
SELECT <espress-col-1>[,...<espress-col-N>]
FROM <tabella>
[WHERE <condizione>]
ORDER BY <espressione> [ASC|DESC] [,...]
```

L'espressione può essere il nome di una colonna, oppure un'espressione che genera un risultato da una o più colonne; l'aggiunta eventuale della parola chiave **ASC**, o **DESC**, permette di specificare un ordinamento crescente, o discendente.
Come si vede, le espressioni di ordinamento possono essere più di una, separate con una virgola.

```
SELECT Cognome, Nome
FROM Indirizzi
ORDER BY Cognome
```

L'esempio mostra un'applicazione molto semplice del problema, in cui si ottiene un elenco delle sole colonne **Cognome** e **Nome**, della tabella **Indirizzi**, ordinato per **Cognome**.

```
SELECT Cognome, Nome
FROM Indirizzi
ORDER BY Cognome, Nome
```

Quest'altro esempio, aggiunge l'indicazione del nome nella chiave di ordinamento, in modo che in presenza di cognomi uguali, la scelta venga fatta in base al nome.

```
SELECT Cognome, Nome
FROM Indirizzi
ORDER BY TRIM( Cognome ), TRIM( Nome )
```

Quest'ultimo esempio mostra l'utilizzo di due espressioni come chiave di ordinamento. Per la precisione, la funzione **TRIM()**, usata in questo modo, serve a eliminare gli spazi iniziali e finali superflui. In questo modo, se i nomi e i cognomi sono stati inseriti con degli spazi iniziali, questi non vanno a influire sull'ordinamento.

## *Interrogazioni simultanee di più tabelle*

Se dopo la parola chiave **FROM** si indicano più tabelle (ciò vale anche se si indica più volte la stessa tabella), si intende fare riferimento a una tabella generata dal prodotto di queste.

Se per esempio si vogliono abbinare due tabelle, una di tre righe per due colonne e un'altra di due righe per due colonne, quello che si ottiene sarà una tabella di quattro colonne composta da sei righe. Infatti, ogni riga della prima tabella risulta abbinata con ogni riga della seconda.

SELECT *<specificazione-della-colonna-1>*[,...*<specificazione-della-colonna-N>*]

Vediamo un esempio molto semplice di gestione del magazzino.

***Articoli***

| *Codice* | *Descrizione* |
|---|---|
| vite30 | Vite 3 mm |
| dado30 | Dado 3 mm |
| rond50 | Rondella 5 mm |

***Movimenti***

| *Codice* | *Data* | *Carico* | *Scarico* |
|---|---|---|---|
| dado30 | 01/01/1999 | 1200 | 0 |
| vite30 | 01/01/1999 | 0 | 800 |
| vite30 | 03/01/1999 | 2000 | 0 |
| rond50 | 03/01/1999 | 0 | 500 |

Da questa situazione si vuole ottenere il join della tabella **Movimenti** con tutte le informazioni corrispondenti della tabella **Articoli**, basando il riferimento sulla colonna **Codice**.

In pratica si vuole ottenere la seguente tabella.

| *Codice* | *Data* | *Carico* | *Scarico* | *Descrizione* |
|---|---|---|---|---|
| dado30 | 01/01/1999 | 1200 | 0 | Dado 3 mm |
| vite30 | 01/01/1999 | 0 | 800 | Vite 3 mm |
| vite30 | 03/01/1999 | 2000 | 0 | Vite 3 mm |
| rond50 | 03/01/1999 | 0 | 500 | Rondella 5 mm |

Considerato che da un'istruzione **SELECT** contenente il riferimento a più tabelle si genera il prodotto tra queste, si pone poi il problema di eseguire una proiezione delle colonne desiderate, e soprattutto di selezionare le righe.
In questo caso, la selezione deve essere basata sulla corrispondenza tra la colonna **Codice** della prima tabella, con la stessa colonna della seconda. Dovendo fare riferimento a due colonne di tabelle differenti, aventi però lo stesso nome, diviene indispensabile indicare i nomi delle colonne prefissandoli con i nomi delle tabelle rispettive.

```
SELECT Movimenti.Codice,
       Movimenti.Data,
       Movimenti.Carico,
       Movimenti.Scarico,
       Articoli.Descrizione
FROM Movimenti, Articoli
WHERE Movimenti.Codice = Articoli.Codice;
```

L'interrogazione simultanea di più tabelle si presta anche per elaborazioni della stessa tabella più volte. In tal caso, diventa obbligatorio l'uso degli alias.
Si osservi il caso seguente:

```
SELECT Ind1.Cognome, Ind1.Nome
FROM Indirizzi AS Ind1, Indirizzi AS Ind2
WHERE Ind1.Cognome = Ind2.Cognome AND Ind1.Nome <> Ind2.Nome
```

Il senso di questa interrogazione, che utilizza la stessa tabella degli indirizzi per due volte con due alias differenti, è quello di ottenere l'elenco delle persone che hanno lo stesso cognome, avendo però un nome differente.

Esiste anche un altra situazione in cui si ottiene l'interrogazione simultanea di più tabelle: l'***unione***.
Si tratta semplicemente di attaccare il risultato di un'interrogazione su una tabella con quello di un'altra tabella, quando le colonne finali appartengono allo stesso tipo di dati.

```
SELECT <specificazione-della-colonna-1>[,...<specificazione-della-colonna-N>]
FROM <specificazione-della-tabella-1>[,...<specificazione-della-tabella-N>]
[WHERE <condizione>]
        UNION
SELECT <specificazione-della-colonna-1>[,...<specificazione-della-colonna-N>]
FROM <specificazione-della-tabella-1>[,...<specificazione-della-tabella-N>]
[WHERE <condizione>]
```

Lo schema sintattico dovrebbe essere abbastanza esplicito: si uniscono due istruzioni **SELECT** in un risultato unico, attraverso la parola chiave **UNION**.

### *Condizioni*

La condizione che esprime la selezione delle righe può essere composta come si vuole, purché il risultato sia di tipo logico e i dati a cui si fa riferimento provengano dalle tabelle di partenza.
Quindi si possono usare anche altri operatori di confronto, funzioni, e operatori booleani.

È bene ricordare che il valore indefinito, rappresentato da **NULL**, è diverso da qualunque altro valore, compreso un altro valore indefinito. Per verificare che un valore sia o non sia indefinito, si deve usare l'operatore **IS NULL** oppure **IS NOT NULL**.

### *Aggregazioni*

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una tabella, in forma di tabella contenente una sola riga.
Per questo si utilizzano delle funzioni speciali al posto dell'espressione che esprime le colonne del risultato.

Queste funzioni restituiscono un solo valore, e come tali concorrono a creare un'unica riga.
Le funzioni di aggregazione sono: **COUNT()**, **SUM()**, **MAX()**, **MIN()**, **AVG()**.
Per intendere il problema, si osservi l'esempio seguente:

```
SELECT COUNT(*) FROM Movimenti WHERE ...
```

In questo caso, quello che si ottiene è solo il numero di righe della tabella **Movimenti** che soddisfano la condizione posta dopo la parola chiave **WHERE** (qui non è stata indicata).

L'asterisco posto come parametro della funzione **COUNT()** rappresenta effettivamente l'elenco di tutti i nomi delle colonne della tabella **Movimenti**.

Quando si utilizzano funzioni di questo tipo, occorre considerare che l'elaborazione si riferisce alla tabella virtuale generata dopo la selezione posta da **WHERE**.

La funzione **COUNT()** può essere descritta attraverso la sintassi seguente:

```
COUNT( * )

COUNT( [DISTINCT|ALL] <lista-colonne>)
```

Utilizzando la forma già vista, quella dell'asterisco, si ottiene solo il numero delle righe della tabella.

L'opzione **DISTINCT**, seguita da una lista di nomi di colonne, fa in modo che vengano contate le righe contenenti valori differenti per quel gruppo di colonne.

L'opzione **ALL** è implicita quando non si usa **DISTINCT**, e indica semplicemente di contare tutte le righe.

Il conteggio delle righe esclude in ogni caso quelle in cui il contenuto di tutte le colonne selezionate è indefinito (**NULL**).

Le altre funzioni aggreganti non prevedono l'asterisco, perché fanno riferimento a un'espressione che genera un risultato per ogni riga ottenuta dalla selezione.

SUM( [DISTINCT|ALL] *<espressione>*)

MAX( [DISTINCT|ALL] *<espressione>*)

MIN( [DISTINCT|ALL] *<espressione>*)

AVG( [DISTINCT|ALL] *<espressione>*)

In linea di massima, per tutti questi tipi di funzioni aggreganti, l'espressione deve generare un risultato numerico, sul quale calcolare la sommatoria, **SUM()**, il valore massimo, **MAX()**, il valore minimo, **MIN()**, e la media **AVG()**.

L'esempio seguente calcola lo stipendio medio degli impiegati, ottenendo i dati da un'ipotetica tabella **Emolumenti**, limitandosi ad analizzare le righe riferite a un certo settore.

```
SELECT AVG( Stipendio )
FROM Emolumenti
WHERE Settore = 'Amministrazione'
```

L'esempio seguente è una variante in cui si estraggono rispettivamente lo stipendio massimo, medio e minimo.

```
SELECT MAX( Stipendio ), AVG( Stipendio ), MIN( Stipendio )
FROM Emolumenti
WHERE Settore = 'Amministrazione'
```

L'esempio seguente è invece volutamente **errato**, perché si mescolano funzioni aggreganti assieme a espressioni di colonna normali.

— Esempio errato:

```
SELECT MAX( Stipendio ), Settore
FROM Emolumenti
WHERE Settore = 'Amministrazione'
```

### *Raggruppamenti*

Le aggregazioni possono essere effettuate in riferimento a gruppi di righe, distinguibili in base al contenuto di una o più colonne.
In questo tipo di interrogazione si può generare solo una tabella composta da tante colonne quante sono quelle prese in considerazione dalla clausola di raggruppamento, e da altre contenenti solo espressioni di aggregazione.
Alla sintassi normale già vista nelle sezioni precedenti, si aggiunge la clausola **GROUP BY**.

SELECT *<specificazione-della-colonna-1>*[,...*<specificazione-della-colonna-N>*]
FROM *<specificazione-della-tabella-1>*[,...*<specificazione-della-tabella-N>*]
[WHERE *<condizione>*]
GROUP BY *<colonna-1>*[,...]

Per comprendere l'effetto di questa sintassi, si deve scomporre idealmente l'operazione di selezione da quella di raggruppamento:

la tabella ottenuta dall'istruzione **SELECT...FROM** viene filtrata dalla condizione **WHERE**;

la tabella risultante viene riordinata in modo da raggruppare le righe in cui i contenuti delle colonne elencate dopo la clausola **GROUP BY** sono uguali;

su questi gruppi di righe vengono valutate le funzioni di aggregazione.

Si osservi la tabella riportata in figura 3, mostra la solita sequenza di carichi e scarichi di magazzino.

***Movimenti***

| *Codice* | *Data* | *Carico* | *Scarico* |
|---|---|---|---|
| vite40 | 01/01/1999 | 1200 | 0 |
| vite30 | 01/01/1999 | 0 | 800 |
| vite40 | 01/01/1999 | 1500 | 0 |
| vite30 | 02/01/1999 | 0 | 1000 |
| vite30 | 03/01/1999 | 2000 | 0 |
| rond50 | 03/01/1999 | 0 | 500 |
| vite40 | 04/01/1999 | 2200 | 0 |

Si potrebbe porre il problema di conoscere il totale dei carichi e degli scarichi per ogni articolo di magazzino. La richiesta può essere espressa con l'istruzione seguente:

```
SELECT Codice, SUM( Carico ), SUM( Scarico )
FROM Movimenti
GROUP BY Codice
```

Quello che si ottiene appare nella figura seguente.

| *Codice* | *SUM(Carico)* | *Sum(Scarico)* |
|---|---|---|
| vite40 | 4900 | 0 |
| vite30 | 2000 | 1800 |
| rond50 | 0 | 500 |

Volendo si possono fare i raggruppamenti in modo da avere i totali distinti anche in base al giorno, come nell'istruzione seguente:

```
SELECT Codice, Data, SUM( Carico ), SUM( Scarico )
FROM Movimenti
GROUP BY Codice, Data
```

Si è detto che la condizione posta dopo la parola chiave **WHERE** serve a filtrare inizialmente le righe da considerare nel raggruppamento.

Se quello che si vuole è filtrare ulteriormente il risultato di un raggruppamento, occorre usare la clausola **HAVING**.

SELECT *<specificazione-della-colonna-1>*[,...*<specificazione-della-colonna-N>*]
FROM *<specificazione-della-tabella-1>*[,...*<specificazione-della-tabella-N>*]
[WHERE *<condizione>*]
GROUP BY *<colonna-1>*[,...]
HAVING *<condizione>*

L'esempio seguente serve a ottenere il raggruppamento dei carichi e scarichi degli articoli, limitando però il risultato a quelli per i quali sia stata fatta una quantità di scarichi consistente (superiore a 1000 unità).

```
SELECT Codice, SUM( Carico ), SUM( Scarico )
FROM Movimenti
GROUP BY Codice
HAVING SUM( Scarico ) > 1000
```

Dall'esempio precedente risulterebbe escluso l'articolo **rond50**.

# *Trasferimento di dati in un'altra tabella*

Alcune forme particolari di richieste SQL possono essere utilizzate per inserire dati in tabelle esistenti o per crearne di nuove.

## *Creazione di una nuova tabella a partire da altre*

L'istruzione **SELECT** può servire per creare una nuova tabella a partire dai dati ottenuti dalla sua interrogazione.

```
SELECT <specificazione-della-colonna-1>[,...<specificazione-della-colonna-N>]
INTO TABLE <tabella-da-generare>
FROM <specificazione-della-tabella-1>[,...<specificazione-della-tabella-N>]
[WHERE <condizione>]
```

L'esempio seguente crea la tabella **Mia_prova** come risultato della fusione delle tabelle **Indirizzi** e **Presenze**.

```
SELECT Presenze.Giorno, Presenze.Ingresso, Presenze.Uscita,
        Indirizzi.Cognome, Indirizzi.Nome
INTO TABLE Mia_prova
FROM Presenze, Indirizzi
WHERE Presenze.Codice = Indirizzi.Codice;
```

## *Inserimento in una tabella esistente*

L'inserimento di dati in una tabella esistente prelevando da dati contenuti in altre, può essere fatta attraverso l'istruzione **INSERT** sostituendo la clausola **VALUES** con un'interrogazione (**SELECT**).

```
INSERT INTO <nome-tabella> [(<colonna-1>...<colonna-N>)]
SELECT <espressione-1>, ... <espressione-N>
FROM <tabelle-di-origine>
[WHERE <condizione>]
```

L'esempio seguente aggiunge alla tabella dello storico delle presenze le registrazioni vecchie che poi vengono cancellate.

```
INSERT INTO PresenzeStorico ( PresenzeStorico.Codice, PresenzeStorico.Giorno,
                              PresenzeStorico.Ingresso, PresenzeStorico.Uscita )
SELECT Presenze.Codice, Presenze.Giorno,
        Presenze.Ingresso, Presenze.Uscita
FROM Presenze
WHERE Presenze.Giorno <= '01/01/1999';


DELETE FROM Presenze WHERE Giorno <= '01/01/1999';
```

## *Viste*

Le viste sono delle tabelle virtuali ottenute a partire da tabelle vere e proprie o da altre viste, purché non si formino ricorsioni. Il concetto non dovrebbe risultare strano.

In effetti, il risultato delle interrogazioni è sempre in forma di tabella. La vista crea una sorta di interrogazione permanente che acquista la personalità di una tabella normale.

CREATE VIEW *<nome-vista>* [(*<colonna-1>*[,...*<colonna-N>*)] AS *<richiesta>*

Dopo la parola chiave **AS** deve essere indicato ciò che compone un'istruzione **SELECT**.

L'esempio seguente, genera la vista dei movimenti di magazzino del solo articolo **vite30**.

```
CREATE VIEW Movimenti_Vite30
        AS
 SELECT Codice, Data, Carico, Scarico
 FROM Movimenti
 WHERE Codice = 'vite30'
```

L'eliminazione di una vista si ottiene con l'istruzione **DROP VIEW**, come illustrato dallo schema sintattico seguente:

DROP VIEW *<nome-vista>*

Volendo eliminare la vista **Movimenti_Vite30**, si può intervenire semplicemente come nell'esempio seguente:

```
DROP VIEW Movimenti_Vite30
```

# *Controllare gli accessi*

La gestione degli accessi in una base di dati è molto importante e potenzialmente indipendente dall'eventuale gestione degli utenti del sistema operativo sottostante.

Per quanto riguarda i sistema Unix, il DBMS può riutilizzare la definizione degli utenti del sistema operativo, farvi riferimento, oppure astrarsi completamente.

Un DBMS SQL richiede la presenza di un DBA (*Data Base Administrator*) che in qualità di amministratore ha sempre tutti i privilegi necessari a intervenire come vuole nel DBMS.
Il nome simbolico predefinito per questo utente dal linguaggio SQL è **_SYSTEM**.

Il sistema di definizione degli utenti è esterno al linguaggio SQL, perché SQL si occupa solo di stabilire i privilegi legati alle tabelle.

## *Creatore*

L'utente che crea una tabella, o un'altra risorsa, è il suo creatore. Su tale risorsa è l'unico utente che possa modificarne la struttura e che possa eliminarla.
In pratica è l'unico che possa usare le istruzioni **DROP** e **ALTER**. Chi crea una tabella, o un'altra risorsa, può concedere o revocare i privilegi degli altri utenti su di essa.

## *Tipi di privilegi*

I privilegi che si possono concedere o revocare su una risorsa sono di vario tipo, ed espressi attraverso una particolare parola chiave. È bene considerare i casi seguenti:

**SELECT** — rappresenta l'operazione di lettura del valore di un oggetto della risorsa, per esempio dei valori di una riga da una tabella (in pratica si riferisce all'uso dell'istruzione **SELECT**);

**INSERT** — rappresenta l'azione di inserire un nuovo oggetto nella risorsa, come l'inserimento di una riga in una tabella;

**UPDATE** — rappresenta l'operazione di aggiornamento del valore di un oggetto della risorsa, per esempio la modifica del contenuto di una riga di una tabella;

**DELETE** — rappresenta l'eliminazione di un oggetto dalla risorsa, come la cancellazione di una riga da una tabella;

**ALL PRIVILEGES** — rappresenta simultaneamente tutti i privilegi possibili riferiti a un oggetto.

### *Concedere i privilegi*

I privilegi su una tabella, o su un'altra risorsa, vengono concessi attraverso l'istruzione **GRANT**.

GRANT *<privilegi>* ON *<risorsa>*[,...] TO *<utenti>* [WITH GRANT OPTION]

Nella maggior parte dei casi, le risorse da controllare coincidono con una tabella. L'esempio seguente permette all'utente **Pippo** di leggere il contenuto della tabella **Movimenti**.

```
GRANT SELECT ON Movimenti TO Pippo
```

L'esempio seguente, concede tutti i privilegi sulla tabella **Movimenti** agli utenti **Pippo** e **Arturo**.

```
GRANT ALL PRIVILEGES ON Movimenti TO Pippo, Arturo
```

L'opzione **WITH GRANT OPTION** permette agli utenti presi in considerazione di concedere a loro volta tali privilegi ad altri utenti.

L'esempio seguente concede all'utente **Pippo** di accedere in lettura al contenuto della tabella **Movimenti** e gli permette di concedere lo stesso privilegio ad altri.

```
GRANT SELECT ON Movimenti TO Pippo WITH GRANT OPTION
```

### *Revocare i privilegi*

I privilegi su una tabella, o un'altra risorsa, vengono revocati attraverso l'istruzione **REVOKE**.

REVOKE *<privilegi>* ON *<risorsa>*[,...] FROM *<utenti>*

L'esempio seguente toglie all'utente **Pippo** il permesso di accedere in lettura al contenuto della tabella **Movimenti**.

```
REVOKE SELECT ON Movimenti FROM Pippo
```

L'esempio seguente toglie tutti i privilegi sulla tabella **Movimenti** agli utenti **Pippo** e **Arturo**.

```
REVOKE ALL PRIVILEGES ON Movimenti FROM Pippo, Arturo
```

# *Controllo delle transazioni*

Una transazione SQL, è una sequenza di istruzioni che rappresenta un corpo unico dal punto di vista della memorizzazione effettiva dei dati.
In altre parole, secondo l'SQL, la registrazione delle modifiche apportate alla base di dati avviene in modo asincrono, raggruppando assieme l'effetto di gruppi di istruzioni determinati.

Una transazione inizia nel momento in cui l'interprete SQL incontra delle istruzioni determinate, e termina con l'istruzione **COMMIT**, oppure **ROLLBACK**: nel primo caso si conferma la transazione che viene memorizzata regolarmente, mentre nel secondo si richiede di annullare le modifiche apportate dalla transazione:

```
COMMIT [WORK]

ROLLBACK [WORK]
```

Stando così le cose, si intende la necessità di utilizzare regolarmente l'istruzione **COMMIT** per memorizzare i dati quando non esiste più la necessità di annullare le modifiche.

```
COMMIT
        INSERT INTO Indirizzi
        VALUES (01, 'Pallino', 'Pinco', 'Via Biglie 1', '0222,222222' )
COMMIT
```

L'esempio mostra un uso intensivo dell'istruzione **COMMIT**, dove dopo l'inserimento di una riga nella tabella **Indirizzi**, viene confermata immediatamente la transazione.

```
COMMIT
        INSERT INTO Indirizzi
        VALUES (01, 'Pallino', 'Pinco', 'Via Biglie 1', '0222,222222' )
ROLLBACK
```

Quest'altro esempio mostra un ripensamento (per qualche motivo).
Dopo l'inserimento di una riga nella tabella **Indirizzi**, viene annullata la transazione, riportando la tabella allo stato precedente.

# Cursori

Quando il risultato di un'interrogazione SQL deve essere gestito all'interno di un programma, si pone un problema nel momento in cui ciò che si ottiene è più di una sola riga.

Per poter scorrere un elenco ottenuto attraverso un'istruzione **SELECT**, riga per riga, si deve usare un ***cursore***.

La dichiarazione e l'utilizzo di un cursore avviene all'interno di una transazione. Quando la transazione si chiude attraverso un **COMMIT** o un **ROLLBACK**, si chiude anche il cursore.

## Dichiarazione e apertura

L'SQL prevede due fasi prima dell'utilizzo di un cursore: la dichiarazione e la sua apertura:

```
DECLARE <cursore> [INSENSITIVE] [SCROLL] CURSOR FOR SELECT ...

OPEN <cursore>
```

Nella dichiarazione, la parola chiave **INSENSITIVE** serve a stabilire che il risultato dell'interrogazione che si scandisce attraverso il cursore, non deve essere sensibile alle variazioni dei dati originali; la parola chiave **SCROLL** indica che è possibile estrarre più righe simultaneamente attraverso il cursore.

```
DECLARE Mio_cursore CURSOR FOR
        SELECT Presenze.Giorno, Presenze.Ingresso, Presenze.Uscita,
                Indirizzi.Cognome, Indirizzi.Nome
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

L'esempio mostra la dichiarazione del cursore **Mio_cursore**, abbinato alla selezione delle colonne composte dal collegamento di due tabelle, **Presenze** e **Indirizzi**, dove le righe devono avere lo stesso numero di codice.
Per attivare questo cursore, lo si deve aprire come nell'esempio seguente:

```
OPEN Mio_cursore
```

## Scansione

La scansione di un'interrogazione inserita in un cursore, avviene attraverso l'istruzione **FETCH**. Il suo scopo è quello di estrarre una riga alla volta, in base a una posizione, relativa o assoluta.

```
FETCH [ [ NEXT | PRIOR | FIRST | LAST | { ABSOLUTE | RELATIVE } n ]
FROM <cursore> ] INTO :<variabile> [,...]
```

Le parole chiave **NEXT**, **PRIOR**, **FIRST**, **LAST**, permettono rispettivamente di ottenere la riga successiva, quella precedente, la prima, e l'ultima.

Le parole chiave **ABSOLUTE** e **RELATIVE** sono seguite da un numero, corrispondente alla scelta della riga *n*-esima, rispetto all'inizio del gruppo per il quale è stato definito il cursore (**ABSOLUTE**), oppure della riga *n*-esima rispetto all'ultima riga estratta da un'istruzione **FETCH** precedente.

Le variabili indicate dopo la parola chiave **INTO**, che in particolare sono precedute da due punti (**:**), ricevono ordinatamente il contenuto delle varie colonne della riga estratta.

Naturalmente, le variabili in questione devono appartenere a un linguaggio di programmazione che incorpora l'SQL, dal momento che l'SQL stesso non fornisce questa possibilità.

```
FETCH NEXT FROM Mio_cursore
```

L'esempio mostra l'uso tipico di questa istruzione, dove si legge la riga successiva (se non ne sono state lette fino a questo punto, si tratta della prima), dal cursore dichiarato e aperto precedentemente.
L'esempio seguente è identico dal punto di vista funzionale.

```
FETCH RELATIVE 1 FROM Mio_cursore
```

I due esempi successivi sono equivalenti, e servono a ottenere la riga precedente.

```
FETCH PRIOR FROM Mio_cursore
```

```
FETCH RELATIVE -1 FROM Mio_cursore
```

### *Chiusura*

Il cursore, al termine dell'utilizzo, deve essere chiuso:

```
CLOSE <cursore>
```

Seguendo gli esempi visti in precedenza, per chiudere il cursore **Mio_cursore** basta l'istruzione seguente:

```
CLOSE Mio_cursore
```

# *Reti Neurali*

## *Introduzione alle Reti Neurali*

### Il Cervello Umano

Il cervello umano è sicuramente la struttura più complessa dell universo e può essere considerato come una enorme rete neurale.

Circa 100 miliardi di neuroni costituiscono i nodi di tale rete. Ciascun neurone è collegato a decine di migliaia di altri neuroni ed esistono pertanto milioni di miliardi di connessioni.

Un neurone biologico è composto da un corpo cellulare o "soma" dal quale partono molti collegamenti(dendriti)che ricevono segnali da altri neuroni, e un collegamento di uscita (assone) con il quale il neurone trasmette informazioni ad altri neuroni (attraverso i loro dendriti).

Ogni neurone ha una soglia di attivazione caratteristica: se i segnali provenienti da altri neuroni la superano, il neurone si attiva e trasmette un segnale elettrico sull assone che arriva ad altri neuroni.

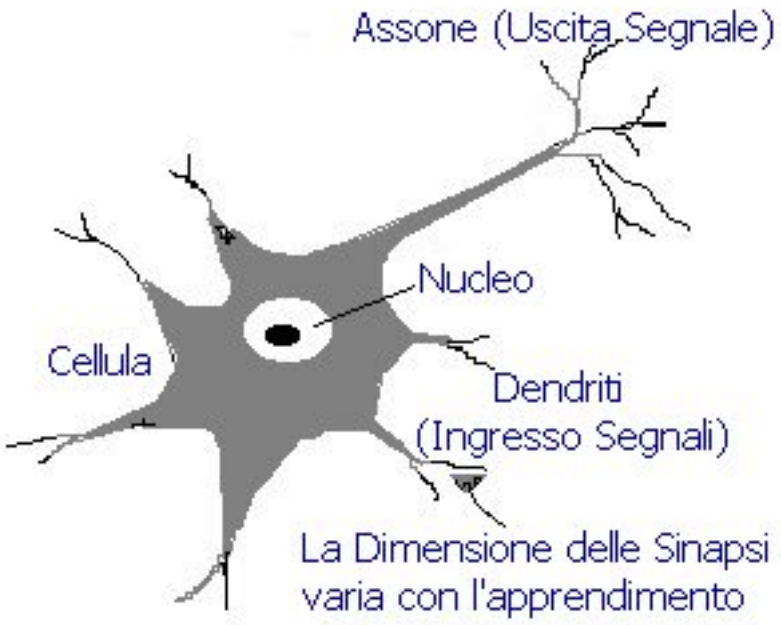
Fra assone e dendrite esiste una sottile intercapedine detta "sinapsi" che permette la trasmissione del segnale attraverso un processo elettrochimico. Lo spessore della sinapsi può variare nel tempo rafforzando o indebolendo il collegamento tra due neuroni.

Il contenuto informativo momentaneo del cervello è rappresentato dall insieme dei valori di attivazione di tutti i neuroni, mentre la memoria è rappresentata dai valori di collegamento(più o meno forte) di tutte le sinapsi.

Due sono le caratteristiche fondamentali del cervello: la plasmabilità e la scomposizione dell informazione in informazioni elementari contenute in ogni singolo neurone. La plasmabilità deriva dal fatto che le sinapsi possono modificarsi nel tempo interagendo con segnali dal mondo esterno.

Non è assolutamente ancora chiaro il meccanismo di apprendimento del cervello, ma è chiaro che il rafforzamento e l'indebolimento dei collegamenti sinaptici costitusce la memorizzazione di una informazione.

Figura 1 - Neurone Biologico




## Reti Neurali Artificiali

Le reti neurali sono lo stato dell arte nel trattamento dell informazione.
Sono basate su principi completamente differenti da quelli normalmente utilizzati nell AI classica per il trattamento dell informazione e il supporto alla decisione.
In effetti, in una rete neurale le informazioni sono scomposte in informazioni "elementari" contenute all interno di ogni singolo neurone.

Una rete neurale può essere vista come un sistema in grado di dare una risposta ad una domanda o fornire un output in risposta ad un input.

La combinazione in/out ovvero la funzione di trasferimento della rete non viene programmata ,ma viene ottenuta attraverso un processo di "addestramento" con dati empirici. In pratica la rete apprende la funzione che lega l output con l input attraverso la presentazione di esempi corretti di coppie input/output.

Effettivamente, per ogni input presentato alla rete, nel processo di apprendi mento, la rete fornisce un output che si discosta di una certa quantità DELTA dall output desiderato: l' algoritmo di addestramento modifica alcuni parametri della rete nella direzione desiderata.

Ogni volta che viene presentato un esempio, quindi, l'algoritmo avvicina un poco i parametri della rete ai valori ottimali per la soluzione dell esempio: in questo modo l'algoritmo cerca di "accontentare " tutti gli esempi un pò per volta.

I parametri di cui si parla sono essenzialmente i pesi o fattori di collagamento tra i neuroni che compongono la rete. Una rete neurale è infatti composta da un certo numero di

neuroni collegati tra loro da collegamenti "pesati" , proprio come lo sono i neuroni del cervello umano.

Ciò che ha portato alla realizzazione delle reti neurali è stato il tentativo di realizzare delle simulazioni delle strutture nervose del tessuto cerebrale.
Tale obiettivo è, però, sfociato nella identificazione di modelli matematici che non hanno molte affinità con i modelli biologici.

Un neurone del tessuto cerebrale può essere visto come una cella (corpo cellulare) che ha molti ingressi (dendriti) e una sola uscita(assone): una rete neurale biologica è composta da molti neuroni dove gli assoni di ogni neurone vanno a collegarsi ai dendriti di altri neuroni tramite un collegamento (la cui forza varia chimicamente in fase di apprendimento e costituisce una "microinformazione") che viene chiamato sinapsi.

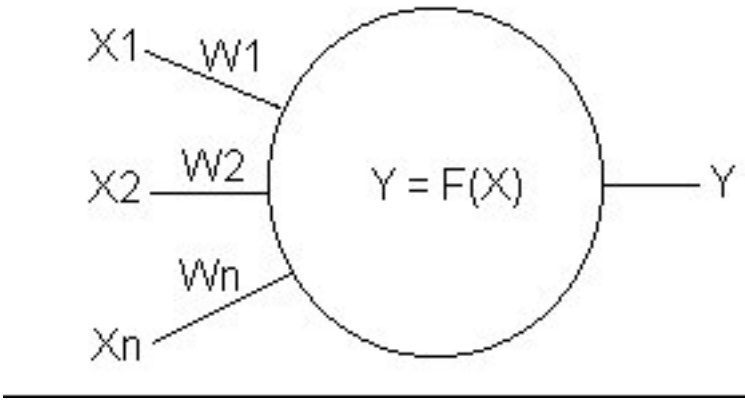
La fig.2 è la rappresentazione formale di un neurone biologico: come si può notare il neurone ha una sua interna funzione di trasferimento.

Non sono ancora chiari i meccanismi di apprendimento del cervello degli esseri viventi e le reti neurali artificiali sono attualmente solo un sistema di trattamento dell informazione in modo distribuito con algoritmi di apprendimento dedicati.

Bisogna sottolineare però che le reti neurali hanno caratteristiche sorprendentemente simili a quelle del cervello umano, come capacità di apprendere, scarsa precisione associata ad alta elasticità di interpretazione dell input e quindi capacità di estarpolazione.

Quella che abbiamo chiamato elasticità di interpretazione dell input viene comunemente chiamata "resistenza al rumore" o "capacità di comprendere dati rumorosi": un sistema programmato ha bisogno di un input ben preciso per dare una risposta corretta, mentre una rete neurale è in grado di dare una risposta abbastanza corretta ad un input parziale o impreciso rispetto a quelli utilizzati negli esempi di addestramento.

Figura 2 - Rapp. formale di un Neurone



## Tipi di Reti Neurale

Esistono molti tipi di reti neurali che sono differenziati sulla base di alcune caratteristiche fondamentali:

- tipo di utilizzo
- tipo di apprendimento (supervisionato/non supervisionato)
- algoritmo di apprendimento
- architettura dei collegamenti

La divisione fondamentale è quella relativa al tipo di apprendimento che può essere supervisionato o non supervisionato: nel primo caso si addestra la rete con degli esempi che contengono un input e l 'output associato desiderato, mentre nel secondo caso la rete deve essere in grado di estrarre delle informazioni di similitudine tra i dati forniti in input (senza associazioni con output desiderati) al fine di classificarli in categorie.

Dal punto di vista del tipo di utilizzo possiamo distinguere tre categorie basilari:

- memorie associative
- simulatori di funzioni matematiche complesse(e non conosciute)
- classificatori

*MEMORIE ASSOCIATIVE*:
possono apprendere associazioni tra patterns(insieme complesso di dati come l insieme dei pixels di una immagine) in modo che la presentazione di un pattern A dia come output il pattern B anche se il pattern A è impreciso o parziale(resistenza al rumore).
Esiste anche la possibilità di utilizzare la memoria associativa per fornire in uscita il pattern completo in risposta ad un pattern parziale in input.

*SIMULATORI DI FUNZIONI MATEMATICHE*:
sono in grado di comprendere la funzione che lega output con input in base a degli esempi forniti in fase di apprendimento.
Dopo la fase di apprendimento, la rete è in grado di fornire un output in risposta ad un input anche diverso da quelli usati negli esempi di addestramento.
Ne consegue una capacità della rete di interpolazione ed estrapolazione sui dati del training set.
Tale capacità è facilmente verificabile addestrando una rete con una sequenza di dati input/ output proveniente da una funzione nota e risulta, invece, utile proprio per il trattamento e la previsione di fenomeni di cui non sia chiaro matematicamente il legame tra input e output.

In ogni caso la rete si comporta come una "black box", poichè non svela in termini leggibili la funzione di trasferimento che è contenuta al suo interno.
Di questo tipo fa parte la rete a retropropagazione dell errore o error back propagation che è quella attualmente più utilizzata per efficacia e flessibilità.

*CLASSIFICATORI*:
con essi è possibile classificare dei dati in specifiche categorie in base a caratteristiche di similitudine.
In questo ultimo tipo di rete esiste il concetto di apprendimento non supervisionato o "autoorganizzante", nel quale i dati di input vengono distribuiti su categorie non predefinite.

L' algoritmo di apprendimento di una rete neurale dipende essenzialmente dal tipo di utilizzo della stessa , così come l architettura dei collegamenti.

Le reti multistrato prevedono ad esempio l algoritmo a retropropagazione dell errore o sono addestrate tramite algoritmi genetici. I classificatori normalmente derivano dall architettura delle mappe autorganizzanti di Kohnen.

Esistono diverse regole di base per l'apprendimento ma sono sempre in fase di studio nuovi paradigmi: quello delle reti neurali è un campo in cui c è ancora molto da inventare e da capire.

## *Caratteristiche Generiche di una Rete Neurale*

1) DEVE ESSERE COMPOSTA DA UN CERTO NUMERO DI NEURONI

2) OGNI NEURONE DEVE AVERE INGRESSI E USCITE E UNA PRECISA FUNZIONE DI TRASFERIMENTO

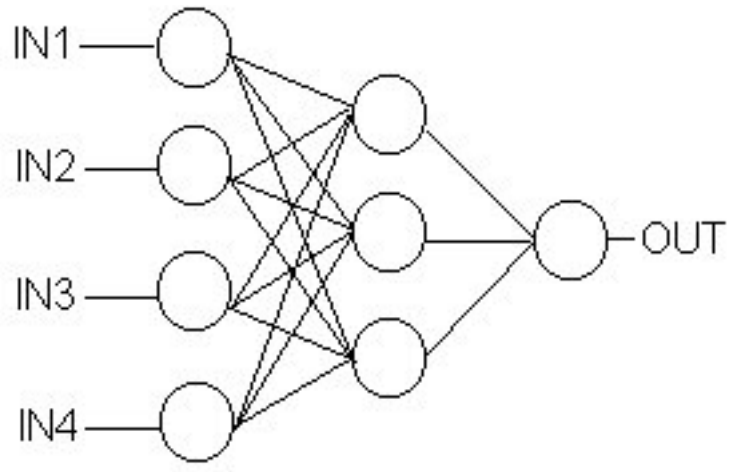
3) GLI INGRESSI E LE USCITE DEI NEURONI DEVONO ESSERE COLLEGATI TRAMITE "COLLEGAMENTI SINAPTICI" MODIFICABILI IN FASE DI ADDESTRAMENTO

4) DEVE ESISTERE UNA PRECISA LEGGE DI APPRENDIMENTO PER LA MODIFICA DEI PESI

Esistono anche reti neurali che non sono basate su specifici collegamenti tra i neuroni ma si evolvono modificando un parametro della funzione di trasferimento di ogni neurone sulla base delle attivazioni dei neuroni di un opportuno vicinato (come il "vicinato di Von Newmann": neuroni sopra, sotto, a destra e a sinistra in una griglia).

Esistono inoltre reti che possono essere studiate appositamente per risolvere problemi di ottimizzazione combinatoria con opportuni vincoli e una funzione di costo(energia) da minimizzare.

Figura 3 - Esempio di rete Neurale



# *Reti Neurali Error Back Propagation*

## Introduzione

Le memorie associative hanno una serie di limitazioni abbastanza gravi:

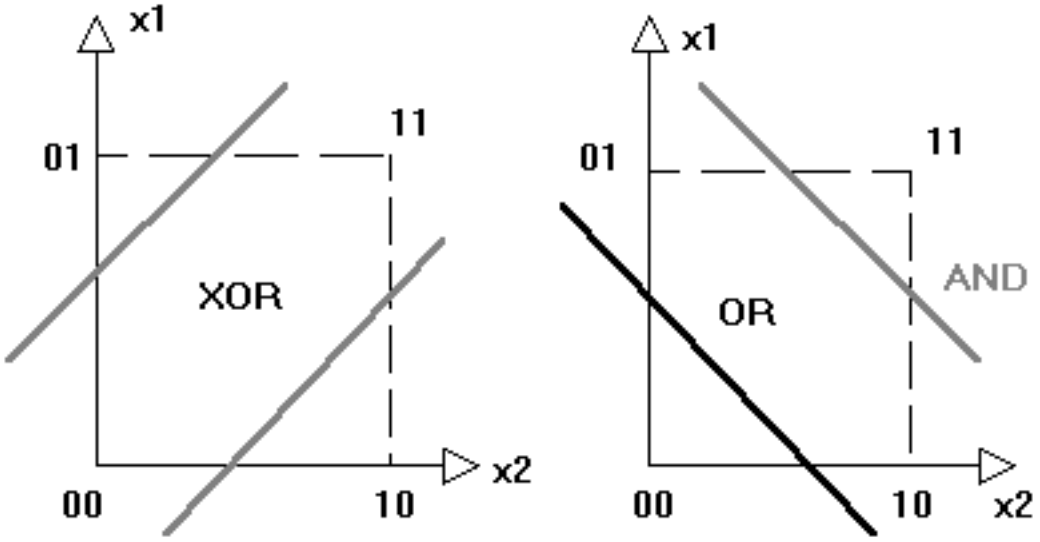
- 1) capacità di memoria bassa
- 2) spesso è necessaria ortogonalità dei vettori di input
- 3) le forme riconosciute devono essere linearmente separabli
- 4) incapacità di adattamento alla traslazione,all'effetto scala e alla rotazione.
- 5) possibilità di operare solo con dati booleani.

Spieghiamo meglio i punti 3 e 4 che forse possono risultare i più oscuri:

-la separabilità lineare di una forma da riconoscere si ha quando una retta(in due dimensioni) o un piano (in tre dimensioni) o un iperpiano(in n dimensioni) puo' separare i punti X(k)={x1(k),x2(k),...xn(k)} (corrispondenti alla forma k da riconoscere) da tutti gli altri punti che rappresentano forme differenti e quindi da discriminare.
Ad esempio le due forme logiche AND e OR sono linearmente separabili,mentre non lo e' la forma logica XOR (or esclusivo), come rappresentato in fig.1.

-l'incapacità di adattamento alla traslazione e alla rotazione o all'effetto scala è importante nel riconoscimento di immagini di oggetti che dovrebbero potere essere riconosciuti indipendentemente da questi tre fattori.

Figura 1 - Separazione AND e OR con uno stadio e separazione XOR con due stadi decisionali




## *Reti Neurali Multistrato*

Prossimamente ci occuperemo di reti neuronali che non sono impiegate come memorie associative ma sono in grado di svolgere funzioni più complesse, quali il riconoscimento di forme qualsiasi e la estrapolazione di correlazioni tra insiemi di dati apparentemente casuali.

Nel fare questo passaggio dalle memorie associative alle reti multistrato tralasciamo il

Perceptron, un tipo di rete neurale a due strati che ha senz'altro una importanza storica come capostipite delle attuali reti backprop e che vale la pena di menzionare.

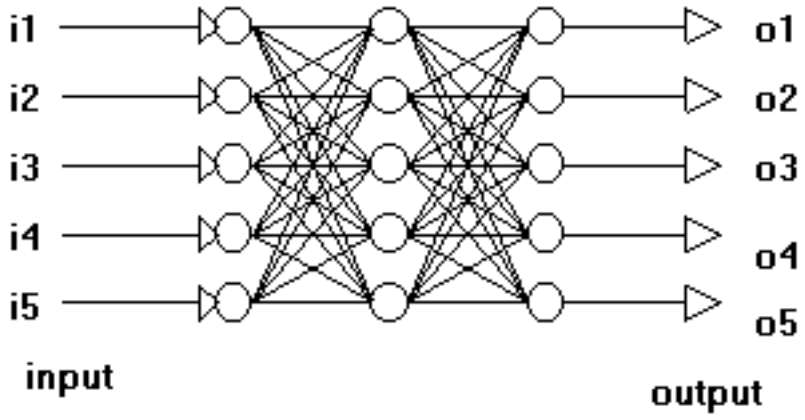
Il tipo di reti neurali che analizzeremo è unidirezionale nel senso che i segnali si propagano solamente dall'input verso l'output senza retroazioni che sono invece presenti nella memoria associativa BAM.

Nel tipo di reti che vedremo i neuroni possono assumere valori reali (compresi tra 0.0 e 1.0) e non più valori booleani 0 e1 per cui la flessibilità della rete risulta nettamente migliorata nella applicabilità a problemi reali.

Ogni neurone di ogni strato sara collegato ad ogni neurone dello strato successivo, mentre non vi saranno collegamenti tra i neuroni dello stesso strato (fig.2) .
Quella di figura 2 è la configurazione più semplice di rete multistrato, dotata di un solo strato intermedio normalmente chiamato "hidden layer": gli strati decisionali di tale rete sono lo strato intermedio e lo strato di output.
Questa è un affermazione che puo sembrare scontata ma qualcuno di voi si sara chiesto:"perche lo strato di neuroni di input non ha potere decisionale?" Ottima domanda!

Una rete neurale impara gli esempi dell' addestramento attraverso un particolare algoritmo che modificava i pesi dei collegamenti tra i neuroni.
Cio che una rete neurale impara sta proprio nel valore dei pesi che collegano i neuroni opportunamente modificato in base ad una legge di apprendimento su un set di esempi. Allora comprendiamo che lo strato di input non è uno strato decisionale perche non ha pesi modificabili in fase di apprendimento sui suoi ingressi.

Figura 2 - Rete Neurale Multistrato




*Note sulla forma delle espressioni matematiche*
E{K=1,N}(K) X(K) INDICA SOMMATORIA DI INDICE K DI X DA K=1 A K=N
(i limiti possono non essere presenti.Es: E(k) X(k)*Y(j))

## Funzione di trasferimento del Neurone

Ogni neurone ha una precisa funzione di trasferimento come avevamo gia accennato parlando delle memorie associative nelle quali i neuroni hanno una funzione di trasferimento a gradino.
In una rete error_back_propagation dove vogliamo avere la possibilita' di lavorare con valori reali e non booleani,dobbiamo utilizzare una funzione di trasferimento a sigmoide (fig.3) definita dalla formula:

O = 1/(1+exp(-I))

dove O=output del neurone,
I=somma degli input del neurone

e in particolare
I= E{k=1,n}(k) w(j)(k) * x(k)

La sigmoide di fig.3 e' centrata sullo zero ma in molte applicazioni e' decisamente opportuno che il centro delle sigmoidi di ogni neurone sia "personalizzato" dal neurone stesso per garantire una maggiore flessibilità di calcolo.

Si puo' ottenere cio' modificando l'ultima formula nel seguente modo:

I= E{k=1,n}(k) w(j)(k) * x(k) - S(k)

dove S(k) è il punto in cui è centrata la sigmoide del k_esimo neurone.

Affinche tale soglia sia personalizzata è necessario che essa venga appresa(cioè si modifichi durante la fase di apprendimento) esattamente come i pesi delle connessioni tra i neuroni dei diversi strati.
Con un piccolo trucco possiamo considerare tale soglia come un input aggiuntivo costante al valore 1 che è collegato al neurone k con un peso da "apprendere": la nostra rete si trasforma pertanto in quella di fig.4.

La formula per cacolare l'attivazione del neurone diventa:

I=E{k=1,n+1}(k) w(j)(k) * x(k)

Figura 3 - Funzione di trasferimento a sigmoide

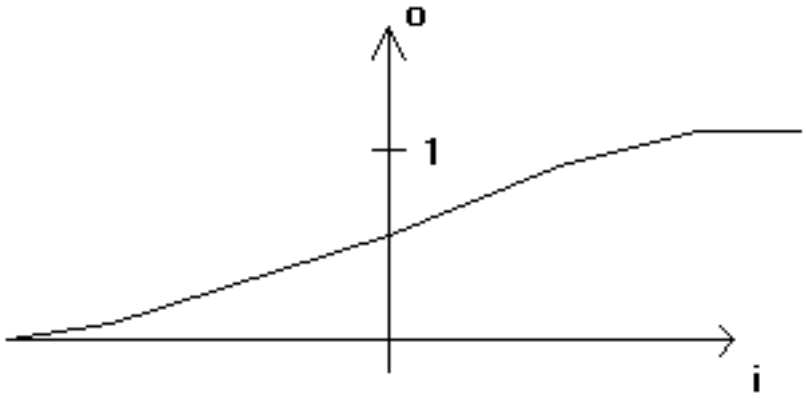
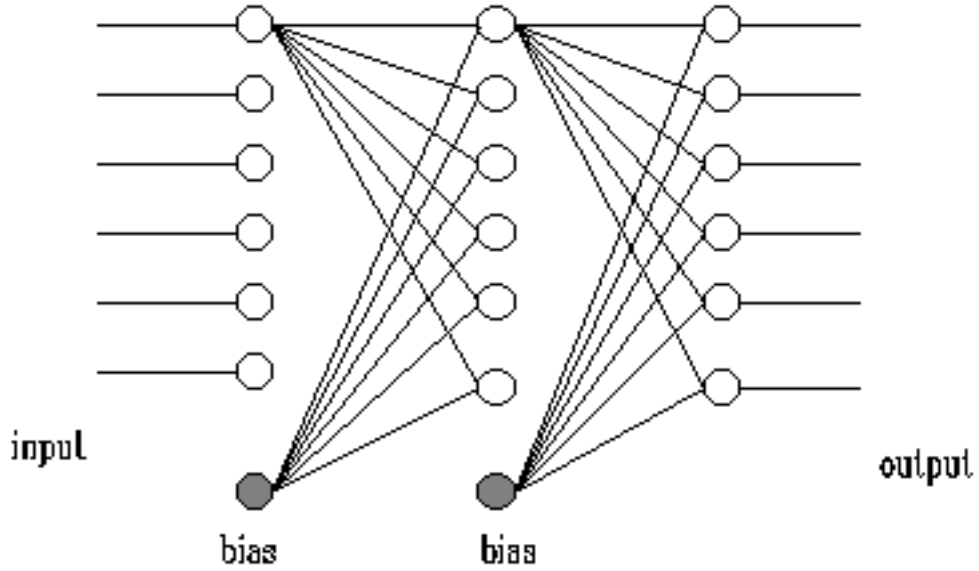


***Figura 4 - Inserimento del Bias***




## Algoritmo di Apprendimento

Il tipo di addestramento che si esegue su questa rete è chiamato "supervised" (supervionato) perchè associa ogni input ad un output desiderato:

Ciclo Epoche:
{
per ogni esempio:
{
- si fornisce l'input alla rete
- si preleva l'output da questa fornito
- si calcola la differenza con l'output desiderato
- si modificano i pesi di tutti i collegamenti tra i neuroni in base a tale errore con una regola (chiamata regola delta) in modo che tale che l'errore diminuisca.
}
- calcolo dell'errore globale su tutti gli esempi
- si ripete il ciclo epoche finche l'errore non raggiunge il valore accettato
}

Naturalmente per input e output si intende un insieme di n esempi ciascuno composto da k input (k sono gli input fisici della rete) e j output (j sono gli output fisici della rete).

Ogni ciclo come quello sopraesposto viene chiamato "epoca" di apprendimento. Chiameremo "potere di generalizzazione" la capacità della rete dopo l'addestramento di dare una risposta significativa anche ad un input non previsto negli esempi.

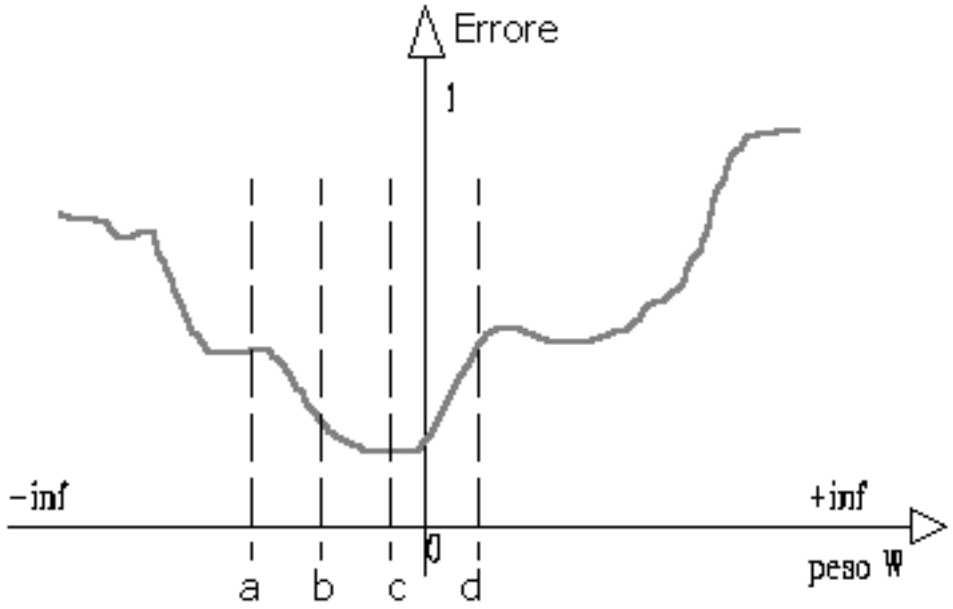
In pratica durante l'addestramento, la rete non impara ad associare ogni input ad un output ma impara a riconoscere la relazione che esiste tra input e output per quanto complessa possa essere: diventa pertanto una "scatola nera" che non ci svelerà la formula matematica che correla i dati ma ci permetterà di ottenere risposte significative a input di cui non

abbiamo ancora verifiche "sul campo" sia all'interno del range di valori di addestramento (interpolazione), che all'esterno di esso(estrapolazione).

Naturalmente l'estrapolazione è più difficoltosa e imprecisa che l'interpolazione e comunque entrambe danno risultati migliori quanto più è completo e e uniformemente distribuito il set di esempi.
Potremmo insegnare ad una rete neurale di questo tipo a fare delle somme tra due valori di input fornerndole in fase di apprendimento una serie di somme "preconfezionate" con risultato corretto in un range più vasto possibile: la rete riuscirà a fare somme anche di valori diversi da quelli degli esempi.

Addestrare una rete neurale a fare una somma è ovviamente inutile se non per motivi di studio o sperimentali inquanto essa è utile per studiare fenomeni di cui non sia conosciuta la correlazione matematica tra input e output (o sia estremamente complessa).

Figura 5 - Andamento dell'errore in funzione di un peso




## La regola Delta

Ci troviamo esattamente al punto c del ciclo visto sopra e calcoliamo la differenza dell'output avuto all'unità j con quello desiderato:

err(j)=D(j)-y(j)

Se per ogni esempio facessimo la somma algebrica degli errori di ogni output, poiche questi sono compresi tra +1 e -1, rischieremmo di annullare l'errore globale e, comunque, commetteremmo un errore concettuale dato che per noi l'errore è tale sia nel senso negativo che positivo.

Possiamo ottenere una misura dell'errore globale facendo una somma dei quadrati degli errori (o in un programma in c usando una funzione che ci restituisca il valore assoluto di un float).

err(j)=absolute(D(j)-y(j)) o err(j)=((D(j)-y(j))**2)/2

Supponiamo che l'errore globale della rete cosi calcolato sia e=0.3, mentre noi desideriamo che l'errore sia 0.01, cio significa che dobbiamo cambiare i pesi delle connessioni ...ma come?
Consideriamo il peso della connessione specifica che collega y(k) dello strato di output al neurone h(j) dello strato intermedio: naturalmente variando questo peso l'errore su y(k) varia,supponiamo con una legge come quella di fig.5.

Il valore al quale dovremmo settare il peso della connessione e` ovviamente il punto c che corrisponde al minimo dell'errore.
Pertanto se noi riuscissimo a cercare tale punto minimo per ogni connessione avremmo ottenuto il risultato, cioe` la rete sarebbe addestrata.

La difficolta` di tale procedimento sta nel fatto che non possiamo analizzare singolarmente ogni connessione in modo sequenziale perche la forma della funzione che lega ciascun peso con l'errore varia al variare degli altri pesi.

Il problema della ottimizzazione dei pesi di una rete di tale tipo è in realtà la ricerca di un minimo di una funzione in uno spazio n-dimensionale che puo essere ricondotto alla ricerca di un insieme di minimi coerenti di n funzioni in uno spazio bidimensionale.

Esiste solamente una tecnica empirica per ottenere la soluzione che viene denominata "discesa del gradiente".

Ricordando che la derivata di una funzione è una misura della pendenza della funzione in quel punto, possiamo spiegare tale tecnica come segue: partiamo da un punto casuale(i pesi all'inizio hanno valori casuali) e facciamo piccoli spostamenti di segno opposto e proporzionali alla derivata della funzione in quel punto(significa che se ci troviamo in A faremo un piccolo spostamento verso B e e se ci troviamo in D faremo un piccolo spostamento verso C).

In questo modo ci avviciniamo sempre più al minimo della funzione e quando sara raggiunto, essendo la derivata in quel punto 0, non effettueremo più spostamenti.

Con questa tecnica noi possiamo anche immaginare la forma della funzione modificarsi lentamente(a causa delle modifiche degli altri pesi) mentre noi ci muoviamo su di essa con spostamenti abbastanza piccoli da consentirci di tenere conto di tale movimento.

Naturalmente il minimo raggiunto potrebbe non essere il minimo assouto ma un minimo locale, pero nell'insieme di tutti i minimi raggiunti e` molto probabile che la grande maggioranza siano minimi assoluti.

Esprimiamo in formula matematica il pricipio esposto nel seguente modo:

delta_w2=-epsilon*(d err/d w2)

dove d err/d w = derivata dell errore rispetto al peso
epsilon = costante di apprendimento che incide sulla misura dello spostamento
a parita di "pendenza" nel punto specifico
(e` consiglaibile un valore compreso tra 0.1 e 0.9)

Proviamo a sviluppare questa derivata nel seguente modo:

delta_w2(k)(j)=-epsilon*(d err/d I(j))*(d I(j)/d w2(k)(j))

definendo delta(j)=d err/d I(j) si semplifica in

delta_w2(k)(j)=-epsilon * delta(j) * (d I(j)/d w2(k)(j))

e ricordiamo che la formula di attivazione di un neurone dello strato di output è

I(j)=E(k) w2(k)(j) * h(k)

dove h(k)=output del neurone k_esimo dello strato hidden

la sua derivata risulta d I(j)/d w2(k)(j)=h(k)

ritorniamo adesso a delta impostando

delta(j)=(d err/d y(j)) * (d y(j)/d I(j))

e risolviamo separatamente le due derivate in essa contenute:

prima derivata: d err/d y(j) = d (((D(j)-y(j))**2)/2)/d y(j) =-(D(j)-y(j))

seconda derivata: d y(j)/d I(j) = d (1/(1+e**(-I(j)))/d I(j) = y(j) * (1-y(j))

per cui dato che

delta_w2(j)=-epsilon * delta(j) * (d I(j)/d (w2(k)(j))

si ha

delta_w2=-epsilon * delta(j) * h(k)

e

delta(j)=(D(j)-y(j))*y(j)*(1-y(j))

abbiamo la formula che ci consente di calcolare i pesi delle connessioni tra i neuroni dello strato di output e quelli dello strato intermedio:

delta_w2[j][k]=epsilon*(D[j]-y[j]) * y[j] * (1-y[j]) * h[k]

dove conosciamo

D[j] come valore di output desiderato

y[j] come valore di output ottenuto

h[k] come stato di attivazione del neurone k del livello intermedio

## La retropropagazione dell'errore

Con la formula sopraesposta potremmo gia costrure un programma che effettui l'addestramento di una rete priva di strati nascosti (cioè con il solo strato di output decisionale), sostituendo allo strato nascosto lo strato di input.

Se consideriamo una rete con uno strato hidden dobbiamo invece ripetere il calcolo precedente (tra output e hidden), tra lo strato hidden e lo strato di input.

A questo punto sorge una difficoltà in più: nel primo calcolo noi conoscevamo l'errore sullo strato di output dato da(D[j]-y[j]), mentre adesso non conosciamo l'errore sullo strato hidden.
Qui entra in gioco il concetto di retropropagazione dell'errore proprio per calcolare quale errore è presente nello strato nascosto(uso indifferentemente i termini hidden e nascosto) in corrispondenza dell'errore nello strato di output.

La tecnica è quella di fare percorrere all'errore sull'output un cammino inverso attraverso le connessioni pesate tra output e hidden (da qui il nome "retropropagazione").

Tradurre in termini matematici ciò che sembra un concetto fisicamente banale risulta invece un lavoro un po più complesso ma ci proviamo lo stesso iniziando con il definire la formula di modifica per i pesi tra hidden e input analoga alla precedente (discesa del gradiente):

```
delta_w1[j][k]=-epsilon*(d err / d w1[j][k])
```

sviluppiamo la derivata nel seguente modo:

```
delta_w1[j][k]=-epsilon*(d err/d I[k])*(d I[k]/d w1[j][k])
```

chiamiamo

```
delta[k]=-(d err/d I[k])
```

e applicando la regola di composizione delle derivate possiamo scrivere

```
delta[k]=-(d err/d I[j])*(d I[j]/d h[k])*(d h[k]/d I[k])
```

dove h[k]=attivazione del k_esimo neurone hidden

quindi anche

```
delta[k]=-(E(j)(d err/d I[j])*(d I[j]/d h[k]))*(d h[k]/d I[k])
```

dato che l'operatore sommatoria agisce solo sui fattori aggiunti che si annullano.

Notiamo ora che il primo termine (d err/d I[j]) = delta[j] che abbiamo incontrato nei calcoli precedenti e che il secondo (d I[j]/d h[k]) puo essere calcolato come

```
d E(k) w2[k][j]*h[k]/d h[k] = w2[k][j]
```

mentre il terzo d h[k]/d I[k] = d h[k]/d (1/1+e**(-h[k]) = h[k]*(1-h[k]

otteniamo finalmente:

delta_w1[j][k]= epsilon * delta[k] * x[j]

in cui

x[i]= i_esimo input

e

delta[k]= (E[j] delta[j] * w2[k][j]) * h[k] * (1-h[k])

in cui h[k]= attivazione neurone k dello strato hidden (conosciuto ovviamente eseguendo la rete, cioè moltiplicando gli input per i pesi delle connessioni e sommando tutto)

delta[j] = (D[j]-y[j]) * (y[j]) * (1-y[j])

già calcolato nella parte precedente (quella relativa alla modifica dei pesi tra output e hidden).
Notate che delta[j] contiene effettivamente la retropropagazione dell'errore e infatti è l'unico termine che contiene dati relativi allo strato di output contenuto nel calcolo del delta_w1.

## Il Numero degli strati intermedi

Esiste una netta differenza tra le reti a due strati decisionali (output+1strato hidden) e quelle con tre strati decisionali (output+due strati hidden): come scritto in un rapporto della D.A.R.P.A. (Defensive Advanced Research Project Agency) sulle reti neurali, con uno strato decisionale (output) è possibile realizzare una separazione lineare, mentre con due strati (output+hidden1) possono essere separati spazi convessi e, con tre strati decisionali (output+ +hidden1 +hidden2) possono essere riconosciute forme qualsiasi (fig.6).

Possiamo pensare che il primo strato decisionale sia in grado di separare i punti appartenenti a forme diverse con un semipiano o un iperpiano(separazione lineare) per ogni nodo e che il secondo strato decisionale faccia altrettanto intersecando i suoi semipiani con quelli del primo strato a formare regioni convesse (per ogni nodo).

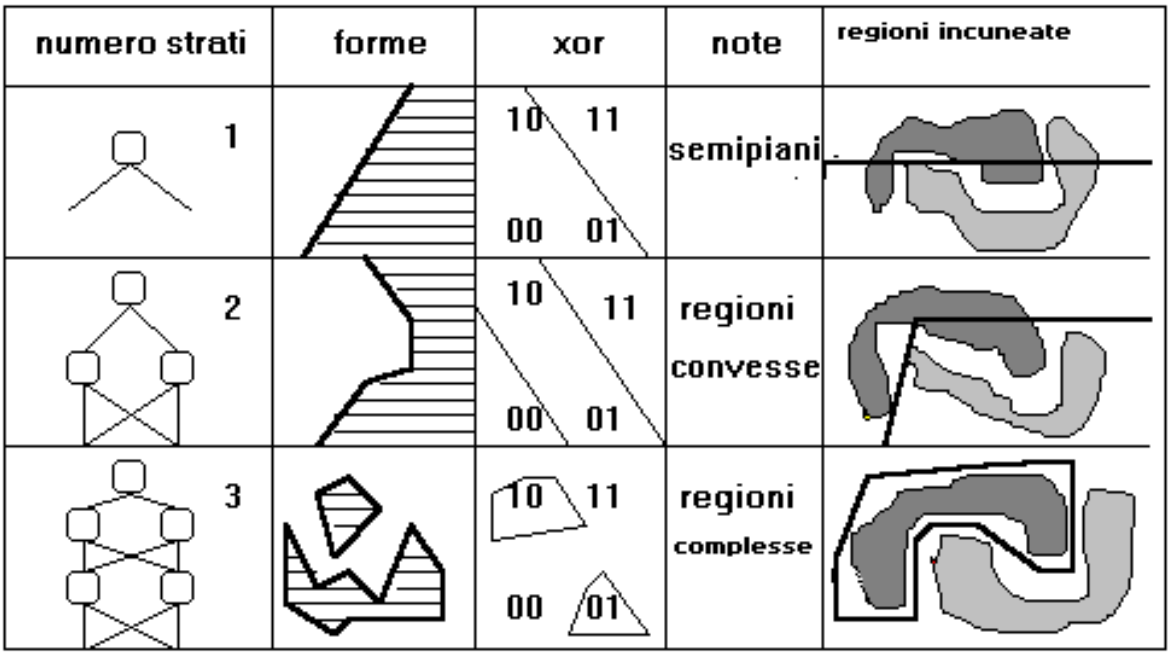
Il terzo strato decisionale associa regioni convesse di nodi differenti dello strato precedente creando forme qualsiasi la cui complessità dipende dal numero di nodi dei primi due strati e il cui numero dipende dal numero di nodi del terzo strato.

I vantaggi relativi all'uso di più di due strati nascosti sono decisamente inferiori.

Esistono delle formule empiriche per calcolare il numero dei neuroni degli strati nascosti in base alla complessità del problema ma, generalmente, è più la pratica che suggerisce tale numero (spesso è inferiore al numero degli input e output).

Con più neuroni negli strati hidden l'apprendimento diventa esponenzialmente più lungo, nel senso che ogni epoca occupa un tempo maggiore (n_pesi=n_neuroni1*n_neuroni2), ma non è escluso che il risultato (raggiungimento del target) venga raggiunto in un tempo simile a causa di una maggiore efficienza della rete.

Figura 6 - Forme riconoscibili con diversi numeri di strati




## Esperimento: Somma di due Numeri

Utilizziamo il training set di fig.7 che contiene 40 esempi normalizzati di somma di due numeri: il programma creato per l'esperimento, su una SPARK station, e' arrivato al raggiungimento del target 0.05 dopo pochi minuti, in circa 4300 epoche con un epsilon(tasso di apprendimento)=0.5 e con quattro neuroni per ogni strato intermedio(ovviamente n_input=2 e n_output=1).

Il raggiungimento del target_error 0.02 è stato ottenuto nelle stesse condizioni all'epoca ~13800. Se non volete attendere molto tempo e verificare ugualmente la convergenza della rete potete ridurre il training set a 10 o 20 esempi curando che siano adeguatamente distribuiti (8 è dato dalla somma 1+7 ma anche 4+4 e 7+1).

Con un training set ridotto la rete converge molto più velocemente ma è chiaro che il potere di generalizzazione risulta inferiore, per cui presentando poi alla rete degli esempi fuori dal training set l'errore ottenuto potrebbe essere molto piu' elevato del target_error raggiunto.

E' buona norma lavorare con delle pause su un numero di epoche non elevato al fine di poter eventualmente abbassare il tasso di apprendimento epsilon quando ci si accorge della presenza di oscillazioni(errore che aumenta e diminuisce alternativamente) dovute a movimenti troppo lunghi(epsilon alto) intorno ad un minimo (che speriamo sia il target e non un minimo locale).

Il raggiungimento del target 0.02 è relativamente rapido mentre da questo punto in poi la discesa verso il minimo è meno ripida e i tempi diventano più lunghi(è il problema della discesa del gradiente con movimenti inversamente proporzionali alla derivata nel punto). Inoltre avvicinandosi al minimo è prudente utilizzare un epsilon basso per il motivo esaminato prima.

E' sicuramente possibile che un errore target non sia raggiungibile in tempi accettabili se è troppo basso in relazione alla complessita' del problema.
In ogni caso bisogna ricordare che le reti neurali non sono sistemi di calcolo precisi ma sistemi che forniscono risposte approssimate a inputs approssimati.

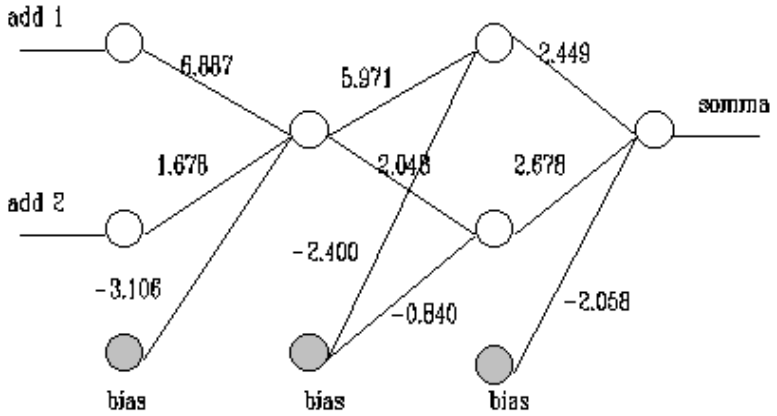
Se ripetete due o più volte lo stesso training con gli stessi dati e gli stessi parametri potreste sicuramente notare differenze di tempi di convergenza verso il target dovuti al fatto che inizialmente i pesi della rete sono settati a valori random(procedura weight_starter) che possono essere più o meno favorevoli alla soluzione del problema.

Nella fig.8 è visulizzato il know_how di una rete con un neurone nello strato hidden1 e due neuroni nello strato hidden2, dopo l'addestramento alla somma fino al target_error=0.05: sono evidenziati i numeri che rappresentano i pesi dei collegamenti.

Figura 7 - Somma di valori normalizzati (tra 0 e 1)

| **add1** | **add2** | **sum** | **add1** | **add2** | **sum** |
|---|---|---|---|---|---|
| 0.40 | 0.30 | 0.70 | 0.11 | 0.44 | 0.55 |
| 0.22 | 0.33 | 0.55 | 0.22 | 0.33 | 0.55 |
| 0.37 | 0.37 | 0.74 | 0.33 | 0.22 | 0.55 |
| 0.49 | 0.37 | 0.86 | 0.44 | 0.11 | 0.55 |
| 0.12 | 0.12 | 0.24 | 0.33 | 0.11 | 0.44 |
| 0.11 | 0.11 | 0.22 | 0.22 | 0.22 | 0.44 |
| 0.13 | 0.13 | 0.26 | 0.11 | 0.33 | 0.44 |
| 0.11 | 0.88 | 0.99 | 0.22 | 0.11 | 0.33 |
| 0.22 | 0.77 | 0.99 | 0.11 | 0.11 | 0.22 |
| 0.33 | 0.67 | 1.00 | 0.16 | 0.16 | 0.32 |
| 0.44 | 0.56 | 1.00 | 0.05 | 0.05 | 0.10 |
| 0.55 | 0.45 | 1.00 | 0.77 | 0.11 | 0.88 |
| 0.66 | 0.34 | 1.00 | 0.66 | 0.22 | 0.88 |
| 0.77 | 0.23 | 1.00 | 0.55 | 0.33 | 0.88 |
| 0.88 | 0.12 | 1.00 | 0.44 | 0.44 | 0.88 |
| 0.11 | 0.55 | 0.66 | 0.33 | 0.55 | 0.88 |
| 0.22 | 0.44 | 0.66 | 0.22 | 0.66 | 0.88 |
| 0.33 | 0.33 | 0.66 | 0.11 | 0.77 | 0.88 |
| 0.44 | 0.22 | 0.66 | 0.11 | 0.66 | 0.77 |
| 0.55 | 0.11 | 0.66 | 0.22 | 0.55 | 0.77 |

Figura 8 - Rete Nuurale addestrata



## Un esempio applicativo

Una rete neurale puo essere addestrata al riconoscimento di profili altimetrici: utilizziamo una rete con 5 input che costituiscono 5 valori di altezza consecutivi e 5 output corrispondenti alle seguenti scelte: (fig.9)

- 1) monte
- 2) valle
- 3) pendenza negativa
- 4) pendenza positiva
- 5) piano(pendenza nulla)

Utilizziamo il training set di fig.10 per addestrare la rete e il validation set di fig.11 per verificare la capacita` di generalizzazione della rete.

Nelle figure 11 e 12 sono presentati i risultati ottenuti su una rete con cinque neuroni su ogni strato hidden. Si arriva al raggiungimento del target_error=0.02 in ~5600 epoche e in ~23000 epoche si raggiunge l'errore 0.01 con un epsilon=0.5.

Come si può constatare il potere di generalizzazione, almeno per il validation set utilizzato, è ottimo nonostante che il training set sia costituito di soli 20 esempi(i valori ottenuti sono riferiti all esecuzione della rete con i pesi relativi al target_error=0.01).

Come noterete negli esempi i valori sono equamente distribuiti tra le cinque categorie per ottenere un buon risultato, poiche la rete neurale (un po come il nostro cervello) tende a diventare più sensibile agli eventi che si verificano più spesso o anche a quelli che si sono verificati più spesso nell ultimo periodo.

In questo semplice esempio applicativo abbiamo utilizzato una rete ebp in modo un po anomalo, cioè come classificatore, nel senso che non abbiamo una risposta "analogica" al nostro input ma una risposta booleana che assegna ad una classe il nostro input pattern(configurazione degli input).

Figura 9 - Profilo altimetrico

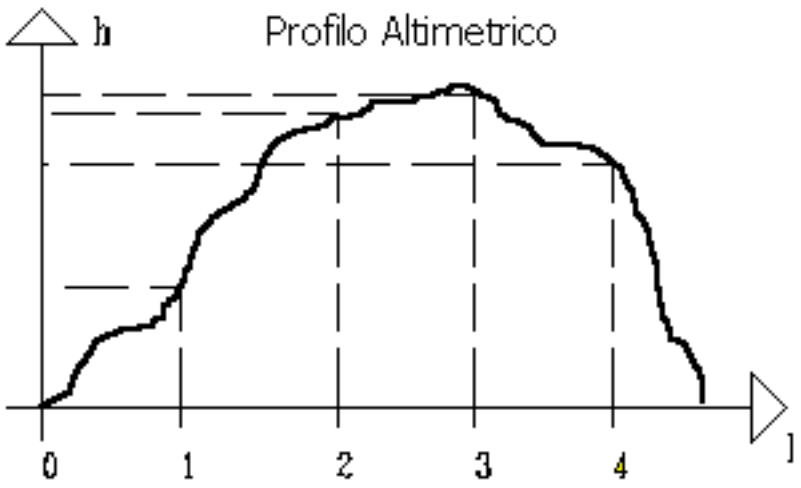


Figura 10 - Training Set

| **Inputs** | | | | | **Output atteso** | | | | | **Output ottenuto** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **h1** | **h2** | **h3** | **h4** | **h5** | **M** | **V** | **P+** | **P-** | **P0** | **m** | **v** | **p+** | **p-** | **p0** |
| 0.5 | 0.6 | 0.7 | 0.1 | 0.0 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.3 | 0.2 | 0.1 | 0.5 | 0.6 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.8 | 0.6 | 0.5 | 0.3 | 0.1 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |
| 0.4 | 0.5 | 0.6 | 0.4 | 0.3 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.6 | 0.5 | 0.3 | 0.4 | 0.7 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.4 | 0.5 | 0.7 | 0.8 | 0.9 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.7 | 0.5 | 0.4 | 0.3 | 0.1 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |
| 0.1 | 0.4 | 0.6 | 0.2 | 0.1 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.8 | 0.5 | 0.5 | 0.7 | 0.9 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.0 | 0.1 | 0.3 | 0.6 | 0.9 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.9 | 0.5 | 0.4 | 0.1 | 0.0 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |
| 0.3 | 0.6 | 0.9 | 0.6 | 0.5 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.8 | 0.6 | 0.5 | 0.7 | 0.9 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.5 | 0.6 | 0.7 | 0.9 | 1.0 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.7 | 0.5 | 0.4 | 0.3 | 0.1 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |

Figura 11 - Validation Set (test potere generaliz.)

| **inputs** | | | | | **output atteso** | | | | | **output ottenuto** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **h1** | **h2** | **h3** | **h4** | **h5** | **M** | **V** | **P+** | **P-** | **P0** | **m** | **v** | **p+** | **p-** | **p0** |
| 0.0 | 0.1 | 0.5 | 0.4 | 0.2 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.5 | 0.3 | 0.1 | 0.3 | 0.4 | 0 | 1 | 0 | 0 | 0 | .00 | .98 | .00 | .00 | .03 |
| 0.1 | 0.2 | 0.5 | 0.7 | 0.9 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.5 | 0.4 | 0.2 | 0.1 | 0.0 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |
| 0.3 | 0.4 | 0.5 | 0.3 | 0.2 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .01 |
| 0.6 | 0.4 | 0.1 | 0.3 | 0.8 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.4 | 0.5 | 0.6 | 0.7 | 0.9 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.4 | 0.3 | 0.2 | 0.1 | 0.0 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .98 | .00 |
| 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |
| 0.4 | 0.5 | 0.6 | 0.4 | 0.3 | 1 | 0 | 0 | 0 | 0 | .99 | .00 | .00 | .00 | .00 |
| 0.5 | 0.4 | 0.1 | 0.6 | 0.8 | 0 | 1 | 0 | 0 | 0 | .00 | .99 | .00 | .00 | .00 |
| 0.1 | 0.2 | 0.3 | 0.4 | 0.6 | 0 | 0 | 1 | 0 | 0 | .00 | .00 | .99 | .00 | .00 |
| 0.9 | 0.5 | 0.3 | 0.2 | 0.1 | 0 | 0 | 0 | 1 | 0 | .00 | .00 | .00 | .99 | .00 |
| 0.7 | 0.7 | 0.7 | 0.7 | 0.7 | 0 | 0 | 0 | 0 | 1 | .00 | .00 | .00 | .00 | .99 |

# *Applicazioni Pratiche di una Rete Neurale*

## *Introduzione*

I campi di applicazione delle reti neurali sono tipicamente quelli dove gli algoritmi classici, per la loro intrinseca rigidità (necessità di avere inputs precisi), falliscono.

In genere i problemi che hanno inputs imprecisi sono quelli per cui il numero delle possibili variazioni di input è così elevato da non poter essere classificato.
Ad esempio nel riconoscimento di un immagine di soli 25 pixels (5*5) in bianco e nero senza toni di grigio abbiamo 2**25 possibili immagini da analizzare. Se consideriamo una immagine di 1000 pixels con 4 toni di grigio dovremo analizzare 6**1000 immagini possibili.

Per risolvere questi problemi si utilizzano normalmente algoritmi di tipo probabilistico che, dal punto di vista della efficienza risultano, comunque, inferiori alle reti neurali e sono caratterizzati da scarsa flessibilità ed elevata complessità di sviluppo.

Con una rete neurale dovremo prendere in considerazione non tutte le immagini possibili ma soltanto quelle significative: le possibili variazioni in un range intorno all'immagine sono gia riconosciute dalla proprieta' intrinseca della rete di resistenza al rumore.

Un altro campo in cui gli algoritmi classici sono in difficoltà è quello dell' analisi di fenomeni di cui non conosciamo regole matematiche.
In realtà esistono algoritmi molto complessi che possono analizzare tali fenomeni ma, dalle comparazioni fatte sui risultati, pare che le reti neurali risultino nettamente più efficienti: questi algoritmi sfruttano la trasformata di Fourier per scomporre il fenomeno in componenti frequenziali e pertanto risultano molto complessi e riescono ad estrarre un numero limitato di armoniche generando notevoli approssimazioni.

Come vedremo, una rete neurale addestrata con i dati di un fenomeno complesso sarà in grado di fare previsioni anche sulle sue componenti frequenziali e ciò significa che realizza al suo interno una trasformata di Fourier anche se nessuno le ha mai insegnato come funziona!

## *Riconoscimento Immagini*

Con una rete neurale tipo error_back_propagation, al contrario di una memoria associativa (che ammette solo valori 1/0 di input), possiamo pensare di analizzare immagini con vari livelli di grigio abbinando ad ogni input un pixel dell'immagine e il livello di grigio definito dal valore (compreso tra 0.0 e 1.0) dell'input.

L'utilizzo potrebbe essere quello di riconoscere un particolare tipo di immagine o di classificare le immagini ricevute in input: il numero degli output pertanto è uguale al numero di classi previste ed ogni output assume un valore prossimo a 1 quando l'immagine appartiene alla classe ad esso corrispondente, altrimenti un valore prossimo a 0. Utilizzando una rete con 100 input si possono classificare immagini di 10*10 pixel (fig.1).

Dovendo analizzare immagini più complesse (esempio 900 pixel 30*30) risulta piu pratico utilizzare uno schema di apprendimento modulare anzichè una singola rete con 900 input. Ad esempio possiamo utilizzare 9 reti da 100 input ciascuna che riconoscono una parte definita della immagine divisa in zone, ed infine una rete neurale con 9 input viene addestrata con le combinazioni di uscita delle 9 reti (fig.2).

In tab.1 si vedono i dati di training della rete finale che ha il compito di riconoscere se il numero di reti precedenti che ha riconosciuto la stessa classe di appartenenza è sufficiente a stabilire che l'immagine appartiene a tale classe: in pratica la tabella rivela una funzione di AND che però risulta "elasticizzato" dalla rete(per semplicità consideriamo solo 3 zone).

In un tale sistema la resistenza al rumore della rete finale viene posta "in serie" con quella delle reti precedenti e pertanto è conveniente effettuare addestarmenti con raggiungimento di errori abbastanza piccoli al fine di evitare una eccessiva flessibilità del sistema.

Figura 1 - Esempio di carattere per OCR

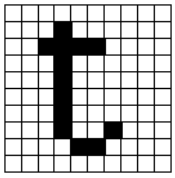


Figura 2 - Pool di Reti

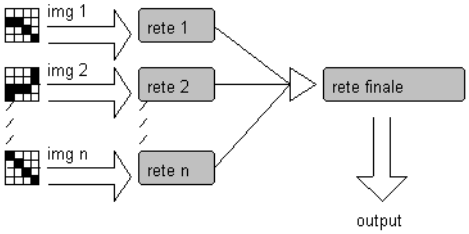



Tabella 1

| net1 | net2 | net3 | out |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

## *Riconoscimento Scrittura*

E abbastanza evidente come sia facilmente trasportabile il problema del riconoscimento scrittura manuale al problema del riconoscimento immagini.
Se pensiamo alle lettere manoscritte incasellate in un modulo quadrettato e letto da uno scanner, il problema si riduce al riconoscimento dell'immagine contenuta in ogni quadrato in modo sequenziale (fig.3).

Se la scrittura è invece completamente libera il problema diventa più complesso e si rende necessario un preprocessor che sia in grado di ricomporre la scrittura scomponendo le singole lettere su matrici definite: i risultati sono ovviamente meno affidabili.

Figura 3 - Esempio di scrittura manuale

### *Previsione di fenomeni complessi*

Una delle applicazioni più importanti delle reti neurali è sicuramente quella delle previsioni di fenomeni complessi come i fenomeni meteorologici o quelli finanziari o socio-economici.

Esistono metodi per la previsione di tali fenomeni che si basano su tre diverse linee di principio:

1) classico
2) frequenziale
3) moderno

Non vogliamo analizzare in questa sede queste tre teorie, ma accennare solo alla seconda che si basa sulla scomposizione in componenti armoniche secondo la legge di Fourier.
Il principale difetto di questo metodo è che i calcoli relativi alle componenti frequenziali piu alte appesantiscono eccessivamente l'algoritmo al punto che si rende necessario accontentarsi di selezionare le armoniche che si ritengono maggiormente influenti, ottenendo un evidente errore di approssimazione.

Con una rete neurale è possibile fare previsioni analizzando le serie storiche dei dati esattamente come con questi sistemi ma non è necessario fare supposizione alcuna per restringere il problema ne, tantomeno, applicare la trasformata di Fourier.

Un difetto comune ai metodi di analisi sopra elencati è quello di essere applicabili solamente se si attuano delle restrizioni nel problema utilizzando delle ipotesi che talvolta potrebbero rivelarsi errate. In pratica si addestra la rete neurale con successioni di serie storiche di dati del fenomeno che si vuole prevedere.

Supponiamo di voler prevedere, sulla base di n valori consecutivi che la variabile x(t) ha assunto, i successivi m valori che assumerà: addestriamo la rete di fig.4 con la prima serie storica di n dati della variabile in input e la seconda serie di successivi m dati in output e cosi via per altre coppie di serie storiche ciascuna delle quali rappresenta un esempio.

In questo modo la rete apprende l'associazione che esiste tra i valori della variabile in n punti e quelli in m punti successivi ma anche l' associazione tra le derivate inquanto l'informazione di due soli valori vicini della variabile è un indice della derivata in quel punto.

Esistono due tipi di previsione: univariata e multivariata.
La prima riguarda la previsione dei valori di una sola variabile di un fenomeno in base ai dati storici della variabile stessa.
La seconda riguarda la previsione dei valori di piu variabili in base ai dati storici delle stesse ed eventualmente anche di altre variabili: in questo secondo caso è l'insieme dei valori delle sequenze storiche di tutte le variabili di input che concorre alla determinazione dell'output di ognuna delle variabili su cui si vuole fare la previsione.

Esistono casi in cui le sequenze storiche elementari (quelle che rappresentano un esempio) devono essere composte da molti dati consecutivi per avere dei buoni risultati e ciò comporta la necessità di utilizzare reti con un numero di inputs molto elevato: è possibile fare questo ma, talvolta, si preferisce utilizzare "reti ricorrenti" che sono reti neurali dotate di "memorie sugli input" tali che ogni variabile possa avere un solo input fisico ma la rete ad'ogni istante t sia condizionata non solo dagli input dell istante t ma anche da quelli degli istanti precedenti (t-1 ... t-n , dove n rappresenta l'ampiezza delle sequenze storiche).

Figura 4

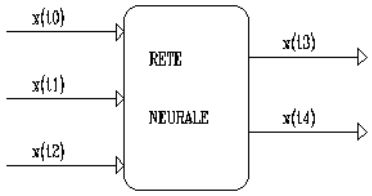
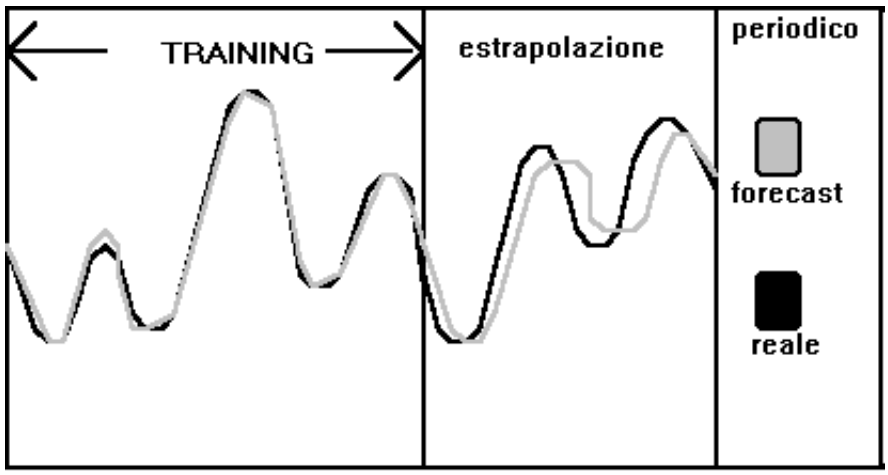



Figura 5




## Previsione Univariata

Si tratta in pratica di ciò di cui si è gia parlato riferendoci alla fig.4.
Vediamo due esempi di previsione di due processi dei quali il primo è di tipo periodico(naturalmente la previsione di un processo periodico ha senso solo a livello didattico), e il secondo aperiodico.
Possiamo effettuare previsioni sulla funzione:

x(t)=sin(2*pigreco*t/40)+sin(2*pigreco*t/100)
[due sinusoidi con periodi 40 e 100]

Usiamo una rete neurale con 2 inputs che corrispondono ad una serie storica di 2 valori e con 1 solo output che corrisponde al valore successivo previsto: con un training set di 40 esempi (120 valori del file period.lrn composto di 200 valori) e periodo 40 (la seconda sinusoide assume automaticamente periodo 100), otteniamo il risultato di fig.5 (il target_error=0.01.

E stato ottenuto con un numero considerevole di epoche su una spark station).
Il risultato viene ottenuto ripetendo piu volte la funzione exec del programma di simulazione di rete neurale con valori di input scelti nel range dei valori possibili della funzione.
Utilizzando una rete con due input e un output abbiamo una informazione su due valori precedenti ma anche sulla derivata in quel punto.

Per effettuare l'addestramento si può utilizzare lo stesso training set impostando diversamente il numero degli input e degli output (tenendo presente che deve essere (n_in + n_out)*n_es < n_gen).
Avendo il programma utilizzato una funzione di trasferimento a sigmoide che fornisce valori compresi tra 0 e 1, verranno generati degli esempi normalizzati con valori che variano tra 0.0 e 1.0.

NOTA IMPORTANTE: bisogna distinguere le previsioni fatte nel range del training set (interpolazioni), che non sono previsioni vere e proprie, da quelle effettuate al di fuori di esso che rappresentano la reale capacità predittiva della rete neurale.

IN PRATICA: nella previsione di fenomeni complessi si utilizzano serie storiche composte da un numero molto maggiore di dati e i valori da predire sono sempre piu di uno a seconda del campo predittivo che l'applicazione richiede.

Passiamo adesso ad analizzare una funzione aperiodica che risulta essere sicuramente piu interessante ai fini delle previsioni.
Utilizziamo una funzione composta dalla sovrapposizione di due sinusoidi di cui una con periodo incommensurabile:

x(t)=sin(2*pigreco*t/40)+sin(sqrt(2*pigreco*t/40))

Addestriamo una rete con tre inputs e un output (usiamo 5 neuroni in ogni strato hidden). Il risultato di previsione che otteniamo dopo un addestramento che porta ad un errore=0.01 è quello di fig.7.
L'addestramento è stato effettuato con 16 esempi che costituiscono 64 generazioni (16*(3input+1output)) delle 100 generate da sin_gen utilizzando l'opzione aperiodica e periodo=40.

Adesso proviamo a verificare se la rete addestrata con il processo aperiodico è in grado di fare predizioni anche sulle componenti frequenziali del processo stesso.

Possiamo generare, usando le componenti frequenziali, due serie storiche con i valori reali di tre dati consecutivi da usare come input e costruire la funzione in base al valore dato come output.

Sia nel caso della prima componente che nel caso della seconda abbiamo risultati di previsione analoghi a quelli della funzione composta e ne deduciamo che la rete è in grado di riconoscere le componenti frequenziali del processo, proprio come se applicasse una trasformata di Fourier "virtuale".

Per avere una controprova di ciò possiamo tentare di fare una previsione su una sinusoide di frequenza differente da quelle componenti: la rete fa delle previsioni con errori tangibilmente maggiori dimostrando di essere sensibile solamente alle componenti frequenziali del processo relativo ai dati di addestramento (fig.9 fig.10 fig.11).

Effettivamente questa controprova risulta molto più evidente se l'addestramento avviene con serie storiche più ampie di quella utilizzata, attraverso le quali la rete assume informazioni piu precise in merito alla frequenza delle varie componenti.

Figura 7

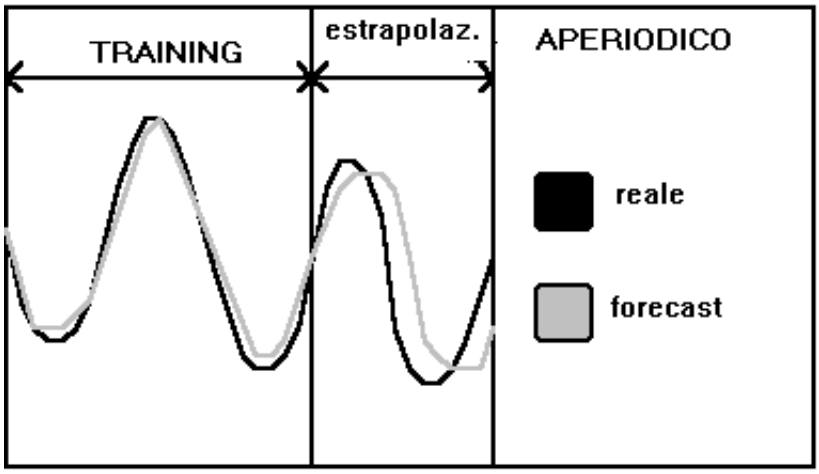
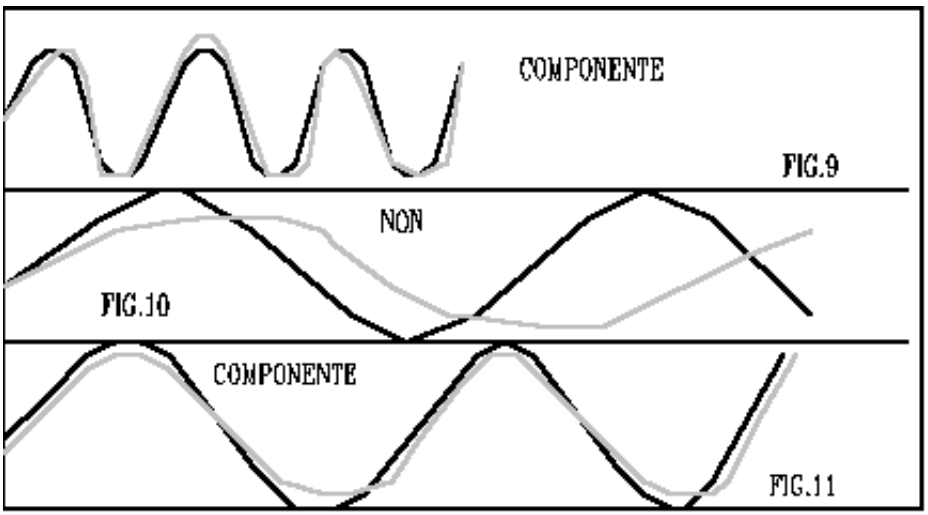



Figura 9-10-11




## *Processi non Stazionari*

Negli esempi precedenti abbiamo esaminato processi stazionari, cioè processi in cui la variabile considerata può variare entro un range ben definito.
Se consideriamo processi non stazionari la varabile da predire puo variare al di fuori di qualunque range di valori e ciò comporta un problema nella normalizzazione dei dati.

La normalizzazione infatti consiste nel riportare valori reali del processo a valori compresi tra 0 e 1 o tra -1 e 1 a seconda del tipo di rete e di funzione di trasferimento dei neuroni: si ottiene questo nel modo più semplice dividendo tutti i valori del processo per il valore più alto di essi, ma in un processo non stazionario quest'ultimo non è conosciuto.

Per risolvere questo problema si rende necessario effettuare una normalizzazione non lineare del processo in modo che la variabile analizzata, che può teoricamente variare tra -inf e +inf , vari tra -k e +k.

Si possono utilizzare funzioni logaritmiche, radici cubiche e funzioni nonlineari come la sigmoide utilizzata nella funzione di trasferimento dei neuroni di una rete e_b_p.
Naturalmente l'imprecisione aumenta se ci si avvicina agli estremi della funzione dato che il rapporto tra la variazione della variabile normalizzata e quella non normalizzata decresce.

esempio 1 di normalizzazione per proc. non stazionario:

x_normaliz= 1/(1+exp(-x))
dove -inf < x < +inf e 0 < x_normaliz < 1

esempio 2 di normalizzazione per proc. non stazionario:

x_normaliz=logn(x)/k
dove 0 < x < +inf e
k=costante valutata in base alla probabilità che la variabile oltrepassi un certo valore

Naturalmente con una normalizzazione di tipo 1 abbiamo la certezza che tutti i valori normalizzati cadano entro il range 0.0/1.0, mentre con il secondo metodo dobbiamo fare una ipotesi restrittiva sui valori massimi che la variabile potra assumere. In ogni caso, l'utilizzo di funzioni di questo tipo è piu utile per effettuare delle "ridistribuzioni dei dati" (non tratteremo qui questo argomento).

Per aggirare il problema dei processi non stazionari si possono considerare serie storiche di derivate anzichè di valori della variabile: in questo modo siamo sicuri che i dati in ingresso sono compresi tra due valori definiti.
Analogamente al caso di serie storiche di valori della variabile, gli intervalli di campionamento dei dati per l'addestramento dovranno essere tanto piu piccoli quanto piu alta è la frequenza di variazioni significative della variabile.

## *Previsione Multivariata*

Abbiamo precedentemente analizzato le problematiche relative alla previsione univariata, cioè quella in cui si cerca di stimare valori futuri di una variabile in base ai valori assunti precedentemente.

Il caso della previsione multivariata non è concettualmente molto più complesso inquanto in esso esistono soltanto delle variabili che sono tra loro correlate per cui la previsione di valori assunti da una o piu variabili dipende dall'insieme delle serie storiche di ognuna delle variabili di ingresso.
Il contesto di analisi della rete neurale diventa molto piu ampio, dato che l'output dipende da

n_informazioni=n_variabili_input * n_dati_serie_storica.

Ciò che rende differente e più interssante il problema è che i fenomeni complessi che si possono studiare analizzando l'evoluzione delle variabili appartenenti sono in genere sistemi nei quali le ucite non sono direttamente influenzate solo dalle variazioni degli ingressi ma sono funzione dello stato del sistema associato agli eventi negli ingressi.

In realtà dobbiamo considerare tre tipi di variabili:

1) di ingresso
2) di stato del sistema
3) di uscita.

Le variabili di stato sono funzione di se stesse e degli input e le variabili di uscita sono funzione delle variabili di stato e degli inputs:

dS(t)/dt= f(S,i) dove S=stato del sistema i=input
u(t) = f''(S,i) dove u=output

f e f'' sono rispettivamente la funzione di evoluzione dello stato rispetto agli ingressi e la funzione di evoluzione dell' uscita del sistema

Nella realtà è abbastanza lecito semplificare nel seguente modo:

u(t) = f''(S)

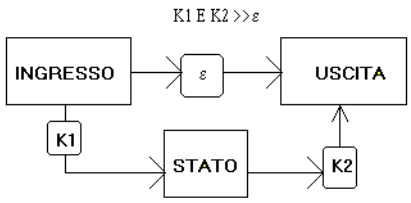
perchè le uscite sono, nella maggior parte dei casi, scarsamente influenzate direttamente dalle variazioni degli ingressi ma sono significativamente influenzate dalle variazioni di stato del sistema (fig.13).

Considerando un sistema in cui gli ingressi variano lentamente, si possono calcolare le uscite all'istante t sulla base dello stato e degli ingressi all'istante t-1:

S(t)=f(S(t-1),i(t-1))

u(t)=f''(S(t-1))=f''(f(S(t-1),i(t-1)))

Figura 13



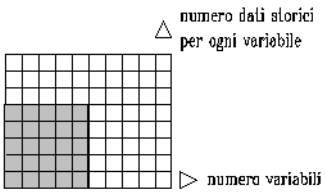

Considerando un sistema in cui gli ingressi variano molto velocemente bisogna tenere conto dell'errore che deriva dal trascurare l'evoluzione dello stato dovuta alla variazione degli ingressi dall'istante precedente a quello relativo alla previsione.

Per fare questo è bene inserire come variabili di input della rete, non soltanto variabili di stato del sistema ma, anche variabili di input del sistema in modo che la rete possa fare previsioni sulle evoluzioni degli stessi.

In ogni caso la previsione multivariata risulta molto più precisa e affidabile della previsione univariata, dato che si basa su un numero maggiore di informazioni.

Per questo motivo è possibile utilizzare serie storiche ridotte rispetto alla previsione univariata: possiamo parlare di quantità di informazione verticale (numero variabili analizzate) e orizzontale(estensione delle serie storiche) e considerare come misura della completezza dell'informazione l'area del rettangolo di fig.14.

Figura 14




## *Conclusioni*

Per uscire un pò dalla teoria vi fornisco un elenco di applicazioni pratiche realizzate con reti neurali in maggioranza di tipo error_back_propagation:

-sistema di guida autonoma di automobili che puo guidare alla velocità di circa 5 Km/h nei viali della Carnagie Mellon University, avendo come input l'immagine della strada.E' una rete neurale error_back_propagation addestrata con 1200 immagini in 40 epoche su un supercomputer Warp. Il tempo di addestramento è stato di 30 minuti mentre la sua esecuzione richiede circa 200 msec su Sun-3.

- Classificatore di segnali radar che raggiunge prestazioni che con tecniche Bayesiane non sono mai state raggiunte.

- Lettore di testi che può pronunciare parole mai viste prima con una accuratezza del 90%: si tratta di una rete error_back_propagation con 309 unità e 18629 connessioni implementata in linguaggio c su VAX_780.

- Riconoscitore di oggetti sottomarini attraverso sonar realizzato da Bendix Aerospace

- Un sistema di scrittura automatico su dettatura in lingua finlandese o giapponese che ha una accuratezza compresa tra l'80% e il 97%.L'hardware prevede un filtro passa_basso e un convertitore analogico_digitale, mentre la rete neurale è di tipo autoorganizzante realizzata su pc AT con coprocessore TMS-32010.

- Sistema di supporto alla decisione per la concessione prestiti realizzato con rete error_back_propagation con 100 input e 1 output(prestito concesso o no). L'apprendimento è stato effettuato con 270.000 casi di prestiti in 6 mesi.

- La rete Neuro-07 sviluppata da NEC riconosce al 90% caratteri stampati e la rete sviluppata dalla Neuristique riconosce il 96% dei caratteri numerici scritti a mano.La sua architettura prevede 256 input(16*16 pixels), uno strato nascosto di 128 neuroni ed un secondo di 16, uno strato di output di 10 neuroni(cifre da 0 a 9).

- La Hughes Research Lab ha realizzato una rete che può riconoscere visi umani anche con disturbi(barba/occhiali/invecchiamento).

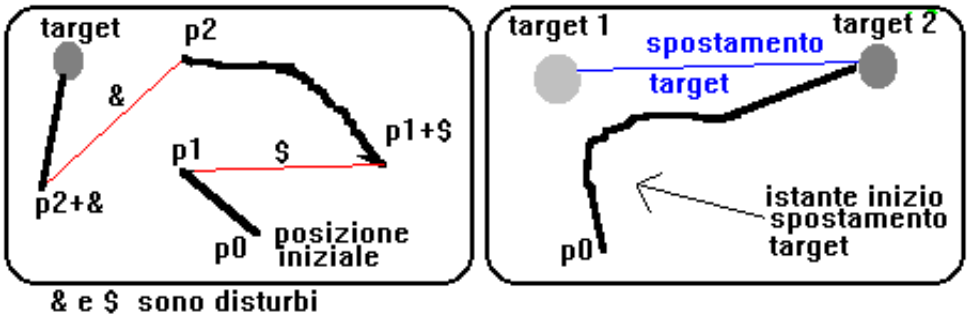
- E' stata realizzata una rete avente come input una immagine di 20*20 pixel con 16 livelli di grigio proveniente da proiezioni tomografiche di polmoni che restituisce in output l'immagine depurata del rumore(che è tanto più ampio quanto minore è il numero delle proiezioni).
In pratica la rete fa una interpolazione di immagini discontinue sulla base di un addestramento di 60 immagini di polmoni con validation set di 120 immagini.

- La PNN(Probabilistic Neural Network) sviluppata da Lockheed interpreta correttamente il 93% dei segnali sonar con un addestramento di 1700 esempi di segnali riflessi da sommergibili e di 4300 esempi di echi riflessi da navi o da movimenti dell'acqua in superfice.

- Una rete neurale è stata applicata in robotica per risolvere un problemma di "cinematica inversa" del tipo: un braccio meccanico con due snodi deve raggiungere un target di coordinate R_t e alpha_t(R=raggio alpha=angolo) partendo da una situazione degli snodi alpha1_b e alpha2_b.
Tale problema per essere risolto richiede il calcolo di funzioni trascendenti (trigonometriche inverse) ma con una rete neurale di 4 input(R_t,alpha_t,alpha1_b,alpha2_b), due strati intermedi rispettivamente di 6 e 4 neuroni e due neuroni di otput(alpha1_b,alpha2_b) si ottiene un risultato migliore con poco sforzo.
Tale rete che comanda il manipolatore INTELLEDEX 605T è simulata su un PC_AT ed è stata addestrata con 64 esempi contenenti posizioni relative diverse tra target e braccio.
Ciò che ha reso ancora più interessante l'applicazione è stato il fatto che la rete è risultata in grado di correggere disturbi al moto di avvicinamento del braccio e seguire un target in movimento (fig 15).

Per finire una considerazione: notate come con pochi neuroni negli strati intermedi si possano ottenere risultati applicativi di livello molto elevato.

Figura 15



# Metodi di Addestramento

## Addestramento con Algoritmo Genetico

Si puo addestrare la rete utilizzando il tipico algoritmo di retropropagazione dell errore, ma anche tramite un algoritmo che sfrutta i principi dell evoluzione di Darwin.
Gli algoritmi genetici sono ampiamente utilizzati nella moderna intelligenza artificiale per risolvere problemi di ottimizzazione.

La base fondamentale di un algoritmo genetico è una popolazione di individui (cromosomi) composti, ciascuno, da un certo numero di geni che rappresentano caratteri dell individuo. Un individuo può avere caratteri più adatti alla soluzione del problema di un altro: si dice che la "fitness function" per quell' individuo porta ad un valore più vicino alla soluzione.
Per fitness function si intende una funzione che lega i caratteri del cromosoma(le variabili indipendenti nel problema) alla variabile che rappresenta la soluzione del problema.

Un algoritmo genetico deve calcolare la fitness function per ogni individuo della popolazione e salvare gli individui migliori .
Partendo da questi individui, deve generare una nuova popolazione tramite gli operatori "crossover" e "mutation"che, rispettivamente, scambiano geni tra cromosomi e creano piccole mutazioni casuali su alcuni geni dei cromosomi.

Viene ricalcolata la fitness function per ogni individuo della nuova popolazione e salvati gli individui migliori.
Questo ciclo viene ripetuto un numero molto elevato di volte creando sempre nuove "generazioni di individui". Nel nostro caso, un individuo o cromosoma è rappresentato dal vettore contenente tutti i pesi dei collegamenti tra gli strati neuronali della rete e ogni singolo peso rappresenta un gene del cromosoma.

Ogni individuo viene testato e, in questo caso, la fitness function coincide con l'esecuzione della rete stessa Gli individui migliori sono quelli che portano ad un errore globale più vicino al target.

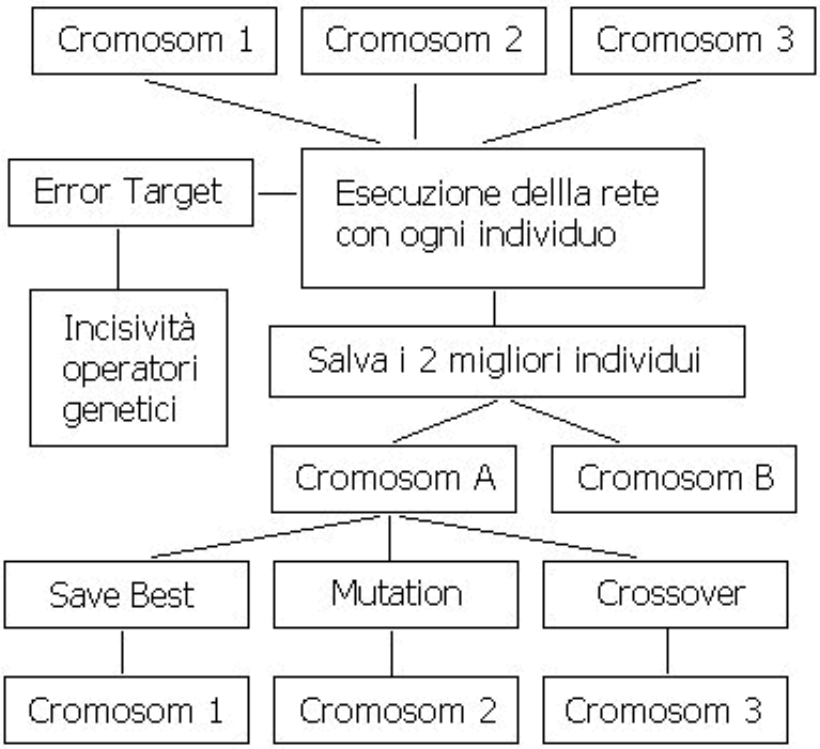
Si parte con tre individui e si salva il migliore e, come sopra esposto, si crea da esso una nuova popolazione di tre individui con gli operatori crossover e mutation, lasciando invariato un individuo per impedire possibili involuzioni.
Questo ciclo viene ripetuto fino al raggiungimento del target error.

Un addestramento con algoritmo genetico è utile su livelli molto bassi di errore della rete per due motivi: il primo è il fatto che un algoritmo genetico può superare i minimi locali e, il secondo è che su livelli alti di errore il classico algoritmo a retropropagazione dell'errore è molto efficente mentre su livelli di errore bassi, cioè quando ci si avvicina al minimo assoluto (o comunque ad un minimo), tale algoritmo produce avvicinamenti al target molto lenti(ricordate la tecnica della discesa del gradiente e il fatto che la derivata di una funzione su un punto di minimo è nulla?)

Per questo motivo l'apprendimento con algoritmo genetico può partire solamente da errori inferiori a 0.5, perchè per errori maggiori la retropropagazione degli errori risulta più efficiente.

Si può anche utilizzare un tool ibrido che inizia l'addestramento con algoritmo ebp e continua con algoritmo genetico, automaticamente, in prossimità del 10% di distanza dal target. Un diagramma di flusso dell algoritmo genetico è rappresentato in nella figura seguente.

Figura 1




## Simulated Annealing: Regime Termico Dinamico

Il termine "Simulated annealing" significa ricottura simulata e si ispira al processo di ricottura dei vetri di spin ("spin glasses").

Un vetro di spin è un metallo contenente delle impurità di un altro metallo in percentuali molto basse ed ha caratteristiche magnetiche particolari.

Un metallo puro può essere ferromagnetico (ferro) o non ferromagnetico (cromo).
I metalli ferromagnetici hanno tutti gli atomi con lo stesso momento magnetico o spin (vettori con stessa direzione e stesso verso).
Un metallo non ferromagnetico ha tutti gli atomi adicenti con spin in posizione di equilibrio (stessa direzione ma verso differente).

Portando un metallo al di sopra di una certa temperatura critica, gli spin degli atomi assumono direzioni e versi casuali (un metallo ferromagnetico perde le sue proprietà magnetiche); il raffreddamento del metallo al di sotto della temperatura critica porta tutti gli spin verso una posizione di equilibrio (minima energia) ferromagnetico o non.

Nei vetri di spin il raffreddamento congela, invece, gli spin in uno stato caotico, pertanto, a bassa temperatura, non esiste uno stato di minima energia, ma molti minimi relativi.
Un raffreddamento lento permette di raggiungere un minimo ottimale (più vicino alla energia

minima assoluta).

Uno dei problemi che si possono presentare nell' addestramento di una rete neurale è quello della caduta in un minimo locale.

Uno degli espedienti che sono stati sperimentati per superare tale problema è quello di adottare una legge di tipo probabilistico del neurone, tale che le variazioni di attivazione vengano accettate con un certo grado di probabilità crescente nel tempo.

Facciamo un esempio che dia un immagine "fisica" del problema: immaginate di avere un piano di plastica deformato con "valli" e "monti" di varie estensioni e profondità, e di porvi sopra una pallina con l'obbiettivo di farla cadere nella valle più profonda (minimo assoluto). Molto probabilmente la pallina cadrà nella valle più vicina al punto in cui la abbiamo messa che potrebbe non essere la valle più profonda.

Il processo di simulated annealing consiste nel muovere il piano di plastica in modo che la pallina lo percorra superando monti e valli e diminuire la intensità del movimento gradualmente fino a zero.
Molto probabilmente, la pallina rimarrà nella valle più profonda dalla quale, a un certo punto, il movimento non sarà più sufficiente a farla uscire.

Nella nostra rete tutto questo non viene realizzato con un vero algoritmo probabilistico, ma si utilizza una legge di attivazione a sigmoide modificata che contiene un parametro "temperatura":

A = 1/(1+e**P/t)
A = valore di uscita del neurone
P = valore di attivazione del neurone(sommatoria degli ingressi)
t = temperatura

Nella cosiddetta Macchina di Boltzmann(*), che fa uso di una legge di attivazione probabilistica, la temperatura viene normalmente fatta diminuire gradualmente nel tempo.

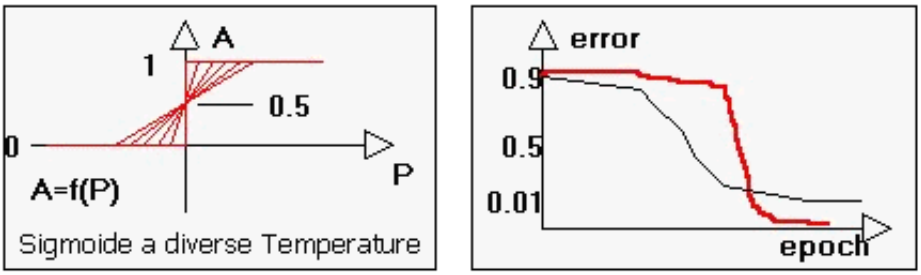
In una rete EBPN la temperatura viene correlata alla differenza tra errore della rete e target error in modo che diminuisca con il diminuire di questa differenza.
In principio la sigmoide ha una pendenza molto lieve garantendo risposte di output poco intense a variazioni di input anche elevate; in seguito la sigmoide si avvicina sempre più al gradino per cui il neurone ha risposte intense a piccole variazioni di input (fig. 2).

Con questo sistema, in questo specifico programma, è possibile avere risultati molto interessanti nella risoluzione di certi problemi e avere risultati negativi con altri tipi di problemi. Ciò dipende, effettivamente, da come si presenta la "superficie" ottimale dei pesi per la risoluzione del problema stesso.

Utilizzando questo processo, si può notare un mantenimento più lungo di livello di errore molto elevato, che talvolta (problema adatto a questo tipo di soluzione), scende a valori estremamente bassi in tempi rapidissimi (fig.3).

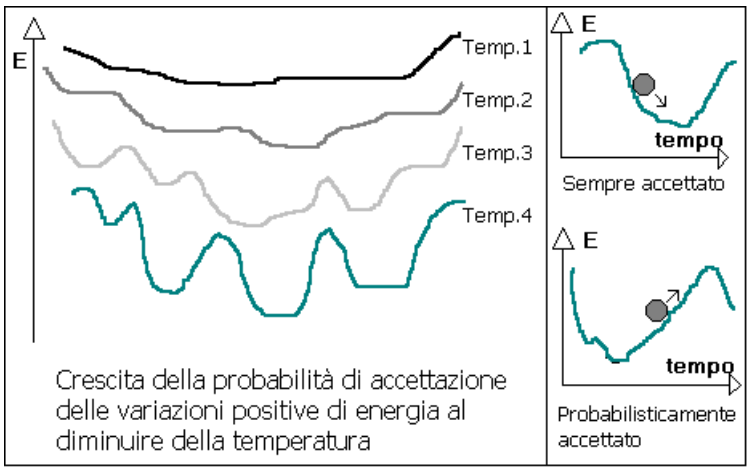
Bisogna precisare che, la Macchina di Boltzmann nulla ha in comune con questa rete e, il principio del Simulated Annealing è stato, in questo contesto, applicato con modalità differenti.

Figure 2 e 3




(*) Macchina di Boltzmann*: Si tratta di una rete neurale derivata dalla rete di Hopfield che, attraverso opportune formule di apprendimento, converge verso uno stato di equilibrio compatibile con i vincoli del problema. Può essere una memoria associativa ma è impiegata, in particolare, per la risoluzione di problemi di ottimizzazione, in cui la fase di apprendimento si basa sulla dimostrazione (di Hopfield) che, alla rete è associata una funzione "energia" da minimizzare rispettando i vincoli del problema (ottimizzazione).*
La Macchina di Boltzmann è sostanzialmente una rete di Hopfield in cui i neuroni adottano una legge di attivazione di tipo probabilistico: se da uno stato S(t) la rete evolve verso uno stato S(t+1) allora, questo nuovo stato viene sicuramente accettato se la energia della rete è diminuita (o è rimasta invariata), altrimenti viene accettato, ma solo con una certa probabilità che, aumenta al diminuire della temperatura (fig.4). Questo sistema consente il raggiungimento di una soluzione molto vicina all ottimo assoluto nei problemi di ottimizzazione, secondo la teoria precedentemente esposta. La nostra rete non è una memoria associativa ma una rete multistrato "feed-forward" (propagazione del segnale da input verso output attraverso gli strati intermedi), utilizzata principalmente per estrazione di funzioni matematiche non conosciute da esempi input/ output (si può comunque utilizzare anche per problemi di classificazione e ottimizzazione). Questa rete utilizza un algoritmo di apprendimento basato sulla retropropagazione degli errori e non sulla evoluzione verso stati energetici bassi.
La applicazione del Simulated Annealing su questa rete è stata realizzata in via sperimentale con risultati soddisfacenti nella soluzione di diversi problemi; la filosofia di trasporto di questa teoria su questo tipo di rete (non potendosi basare su stati energetici) è stata quella di fornire i neuroni di una legge di attivazione a sigmoide variabile in funzione della temperatura: i neuroni si attivano (a parità di segnali di ingresso) maggiormente alle basse temperature (questo comporta anche una differente attività di modifica dei pesi delle connessioni sinaptiche da parte dell algoritmo di error_back_propagation alle diverse temperature).

Figura 4



# Esempi di Report

***Report Standard da Griglia ·***

***Stampa diretta Grafico ·***

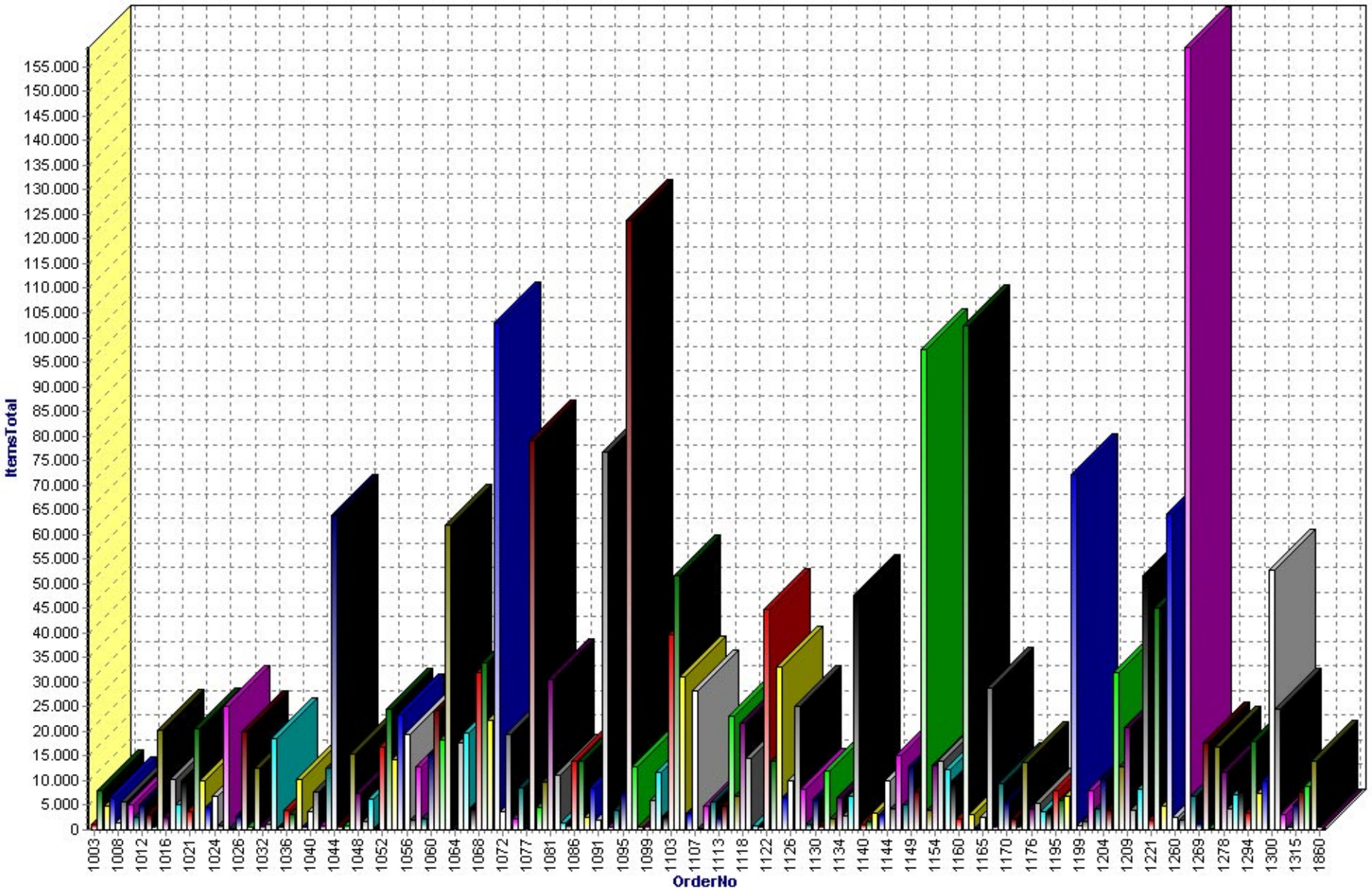
***Pagine Dati ·***

***Report Master/Detail ·***

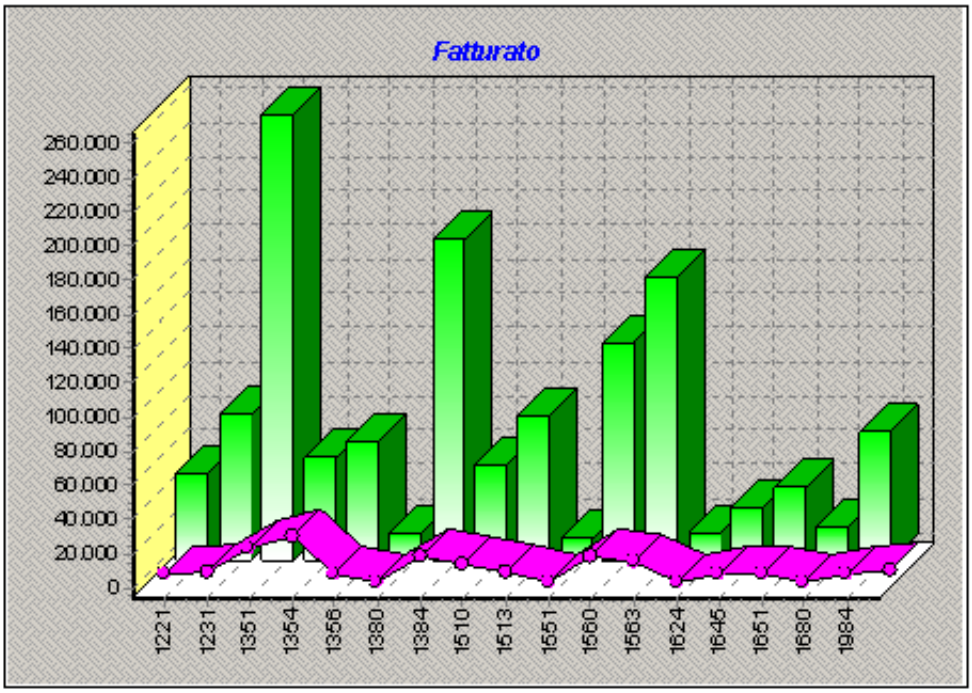
***Nota***:
Iin questo documento sono state incluse soltanto le immagini Bitmap a bassa risoluzione dei report originali. Pertanto la qualità degli esempi è molto inferiore a quella degli stampati ottenibili utilizzando Open DB. Il programma produce report alla massima risoluzione consentita dalla stampante perchè i dati per la stampa vengono generati in formato vettoriale e non bitmap.

# DEMOData-Ordini.Orders

| OrderNo | CustNo | SaleDate | ShipDate | EmpNo | ShipVIA | Terms | PaymentMethod | ItemsTotal | TaxRate | AmountPaid |
|---|---|---|---|---|---|---|---|---|---|---|
| 1003 | 1351 | 12/04/1988 | 03/05/1988 12.00.00 | 114 | UPS | FOB | Credit | 1250 | 4,5 | 0 |
| 1004 | 2156 | 17/04/1988 | 18/04/1988 | 145 | DHL | FOB | Check | 7885 | 0 | 7885 |
| 1005 | 1356 | 20/04/1988 | 21/01/1988 12.00.00 | 110 | UPS | FOB | Visa | 4807 | 0 | 4807 |
| 1007 | 1384 | 01/05/1988 | 02/05/1988 | 45 | US Mail | FOB | Visa | 6500 | 0 | 6500 |
| 1008 | 1510 | 03/05/1988 | 04/05/1988 | 12 | US Mail | Net 30 | Visa | 1449,5 | 0 | 0 |
| 1009 | 1513 | 11/05/1988 | 12/05/1988 | 71 | US Mail | Net 30 | COD | 5587 | 0 | 0 |
| 1010 | 1551 | 11/05/1988 | 12/05/1988 | 46 | UPS | Net 30 | COD | 4996 | 0 | 4996 |
| 1011 | 1560 | 18/05/1988 | 19/05/1988 | 5 | UPS | Net 30 | COD | 2679,85 | 0 | 2679,85 |
| 1012 | 1563 | 19/05/1988 | 20/05/1988 | 118 | UPS | Net 30 | Credit | 5201 | 0 | 5201 |
| 1013 | 1624 | 25/05/1988 | 26/05/1988 | 134 | Emery | Net 30 | Credit | 3115 | 0 | 3115 |
| 1014 | 1645 | 25/05/1988 | 26/05/1988 | 144 | Emery | Net 30 | Credit | 134,85 | 0 | 134,85 |
| 1015 | 1651 | 25/05/1988 | 26/05/1988 | 71 | Emery | FOB | MC | 20321,75 | 0 | 20321,75 |
| 1016 | 1680 | 02/06/1988 | 03/06/1988 | 65 | UPS | FOB | AmEx | 2605 | 0 | 0 |
| 1017 | 1984 | 12/06/1988 | 13/06/1988 | 28 | DHL | FOB | Check | 10195 | 0 | 0 |
| 1018 | 2118 | 18/06/1988 | 19/06/1988 | 118 | DHL | FOB | Check | 5256 | 0 | 0 |
| 1020 | 2156 | 24/06/1988 | 25/06/1988 | 61 | DHL | Net 30 | Credit | 9955 | 0 | 9955 |
| 1021 | 2163 | 24/06/1988 | 25/06/1988 | 52 | UPS | Net 30 | Credit | 3719 | 0 | 3719 |
| 1019 | 2135 | 24/06/1988 | 25/06/1988 | 114 | DHL | Net 30 | Credit | 20602 | 0 | 0 |
| 1022 | 2163 | 30/06/1988 | 01/07/1988 | 46 | UPS | Net 30 | Credit | 10064,65 | 0 | 10064,65 |
| 1023 | 1221 | 01/07/1988 | 02/07/1988 | 5 | UPS | Net 30 | Check | 4674 | 0 | 4674 |
| 1024 | 3151 | 02/07/1988 | 03/07/1988 | 2 | UPS | Net 30 | Check | 6897 | 0 | 6897 |
| 1025 | 1510 | 03/07/1988 | 04/07/1988 | 8 | DHL | Net 30 | AmEx | 930 | 0 | 930 |

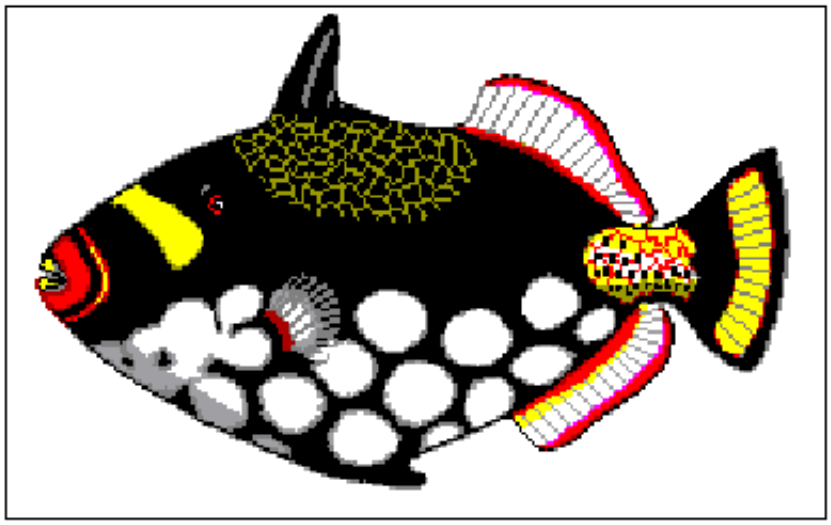

ItemsTotal
155.000
150.000
145.000
140.000
135.000
130.000
125.000
120.000
115.000
110.000
105.000
100.000
95.000
90.000
85.000
80.000
75.000
70.000
65.000
60.000
55.000
50.000
45.000
40.000
35.000
30.000
25.000
20.000
15.000
10.000
5.000
0
1003
1008
1012
1016
1021
1024
1026
1032
1036
1040
1044
1048
1052
1056
1060
1064
1068
1072
1077
1081
1086
1091
1095
1099
1103
1107
1113
1118
1122
1126
1130
1134
1140
1144
1149
1154
1160
1165
1170
1176
1195
1199
1204
1209
1221
1260
1269
1278
1294
1300
1315
1860
OrderNo

# Esempio di Report creato con Open DB




| Cliente | Fatturato | Media |
|---|---|---|
| 1221 | 51450,8 | 8575 |
| 1231 | 85643,6 | 9516 |
| 1351 | 260325,8 | 23666 |
| 1354 | 59660,05 | 29830 |
| 1356 | 69364,7 | 7707 |
| 1380 | 16612,5 | 3322 |
| 1384 | 186984,95 | 18698 |
| 1510 | 55224,55 | 13806 |
| 1513 | 84556,4 | 9395 |
| 1551 | 12223,25 | 3056 |
| 1560 | 126889,35 | 18127 |
| 1563 | 165245,45 | 15022 |
| 1624 | 16142 | 4036 |
| 1645 | 31399,65 | 7850 |
| 1651 | 42302,6 | 8461 |
| 1680 | 20196,8 | 4039 |
| 1984 | 74705,05 | 8301 |
| 2118 | 80892,2 | 10112 |
| 2135 | 0 | 0 |
| 2156 | 36541,9 | 7308 |

Fatturato per il Cliente 1221 **51450,8**



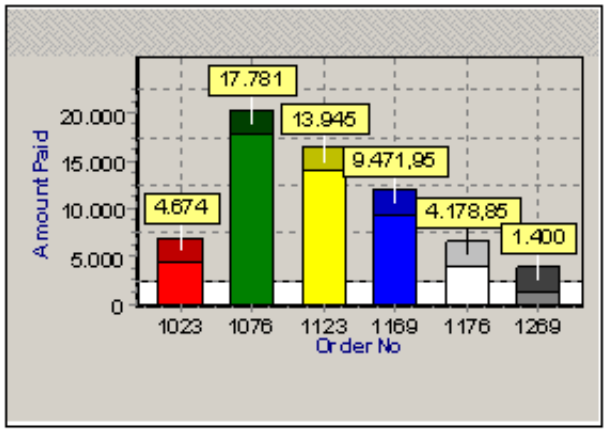
Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.

Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes rocked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering."

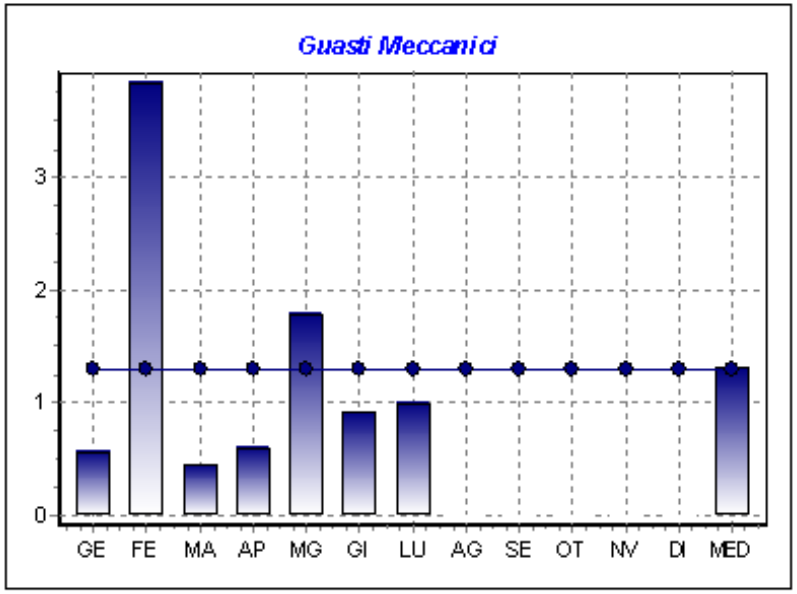
Not edible.

Range is Indo-Pacific and East Africa to Somoa.

| OrderNo | CustNo | SaleDate | ShipDate | AmountPaid |
|---|---|---|---|---|
| 1023 | 1221 | 01/07/1988 | 02/07/1988 | 4674 |
| 1076 | 1221 | 16/12/1994 | 26/04/1989 | 17781 |
| 1123 | 1221 | 24/08/1993 | 24/08/1993 | 13945 |
| 1169 | 1221 | 06/07/1994 | 06/07/1994 | 9471,95 |
| 1176 | 1221 | 26/07/1994 | 26/07/1994 | 4178,85 |
| 1269 | 1221 | 16/12/1994 | 16/12/1994 | 1400 |

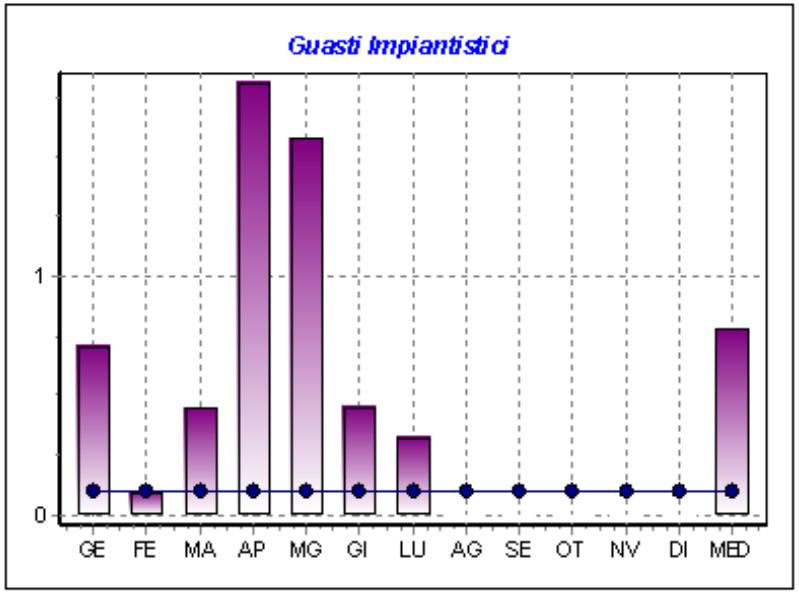


## 100 LINEA TAGLIO SVILUPPI PR. 107



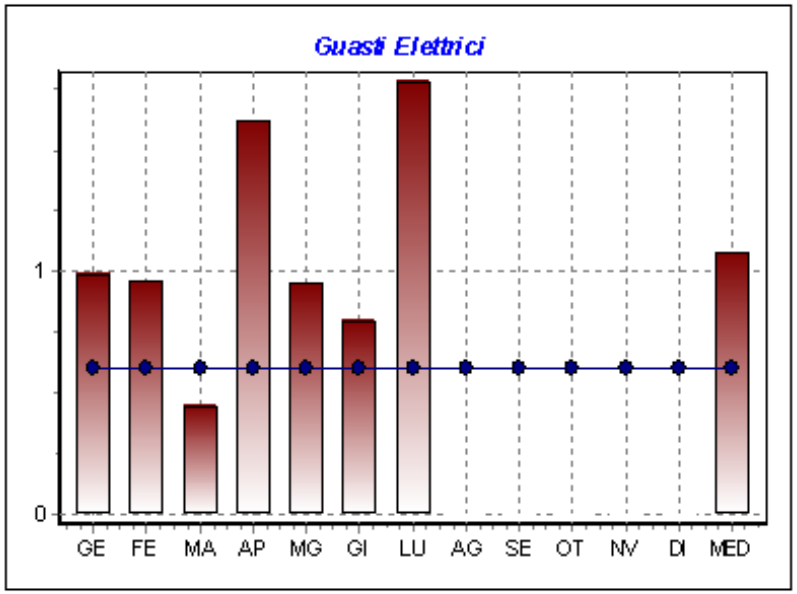

| | |
|---|---|
| Gen | 0,57 |
| Feb | 3,84 |
| Mar | 0,45 |
| Apr | 0,61 |
| Mag | 1,79 |
| Giu | 0,92 |
| Lug | 1 |
| Ago | 0 |
| Set | 0 |
| Ott | 0 |
| Nov | 0 |
| Dic | 0 |

**Med** 1,31
**Bgt** 1,3



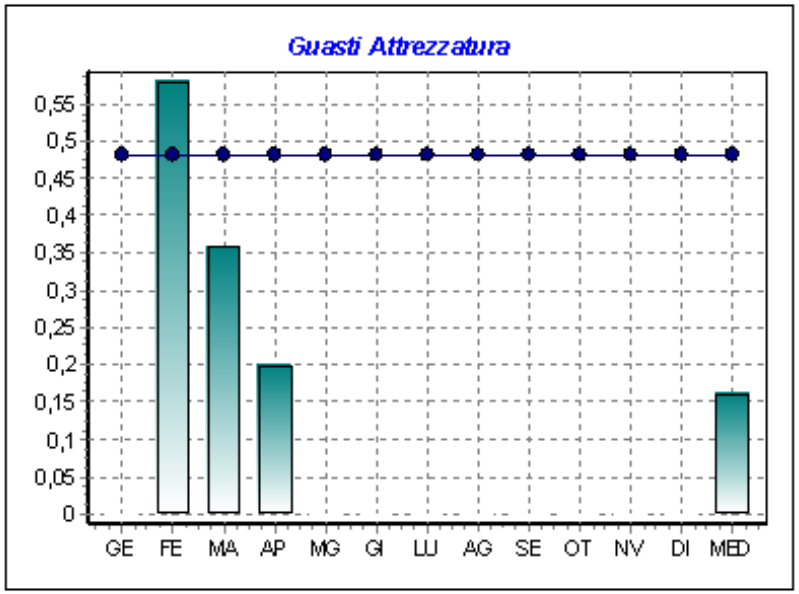

| | |
|---|---|
| Gen | 0,71 |
| Feb | 0,1 |
| Mar | 0,45 |
| Apr | 1,82 |
| Mag | 1,58 |
| Giu | 0,46 |
| Lug | 0,33 |
| Ago | 0 |
| Set | 0 |
| Ott | 0 |
| Nov | 0 |
| Dic | 0 |

**Med** 0,78
**Bgt** 0,1




| | |
|---|---|
| Gen | 0,99 |
| Feb | 0,96 |
| Mar | 0,45 |
| Apr | 1,62 |
| Mag | 0,95 |
| Giu | 0,8 |
| Lug | 1,78 |
| Ago | 0 |
| Set | 0 |
| Ott | 0 |
| Nov | 0 |
| Dic | 0 |

**Med** 1,08
**Bgt** 0,6




| | |
|---|---|
| Gen | 0 |
| Feb | 0,58 |
| Mar | 0,36 |
| Apr | 0,2 |
| Mag | 0 |
| Giu | 0 |
| Lug | 0 |
| Ago | 0 |
| Set | 0 |
| Ott | 0 |
| Nov | 0 |
| Dic | 0 |

**Med** 0,16
**Bgt** 0,48

# Ordini Cliente

| | |
|---|---|
| ID Cliente | 1221 |
| Società | Kauai Dive Shoppe |
| Indirizzo | 4-976 Sugarloaf Hwy |
| Città | Kapaa Kauai |
| N. Telefono | 808-555-0269 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1023 | 01/07/1988 | UPS | Net 30 | Check | 4674 | 4674 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 1313 | Regulator System | 5 | 0 |
| 2 | 2383 | Depth/Pressure Gauge | 4 | 0 |
| 3 | 1330 | Alternate Inflation Regulator | 10 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1076 | 16/12/1994 | UPS | FOB | Visa | 17781 | 17781 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 11221 | Remotely Operated Video Sy | 4 | 0 |
| 2 | 11518 | Tow able Video Camera (B&W | 4 | 0 |
| 3 | 1946 | Second Stage Regulator | 1 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1123 | 24/08/1993 | UPS | Net 30 | Check | 13945 | 13945 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 1314 | Second Stage Regulator | 16 | 0 |
| 2 | 1314 | Second Stage Regulator | 24 | 0 |
| 3 | 1986 | Depth/Pressure Gauge Const | 2 | 0 |
| 4 | 2630 | Wrist Band Thermometer (F) | 2 | 0 |
| 5 | 5324 | Chisel Point Knife | 2 | 0 |
| 6 | 5349 | Flashlight | 3 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1169 | 06/07/1994 | UPS | FOB | Credit | 9471,95 | 9471,95 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 912 | Underw ater Diver Vehicle | 3 | 0 |
| 2 | 1330 | Alternate Inflation Regulator | 4 | 0 |
| 3 | 2657 | Wrist Band Thermometer (C) | 9 | 0 |
| 4 | 2954 | Dive Computer | 4 | 0 |
| 5 | 5356 | Medium Stainless Steel Knife | 2 | 0 |
| 6 | 7654 | Halogen Flashlight | 1 | 0 |
| 7 | 9318 | 71.4 cu ft Tank | 1 | 0 |
| 8 | 9354 | 75.8 cu ft Tank | 1 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1176 | 26/07/1994 | UPS | FOB | Visa | 4178,85 | 4178,85 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 2619 | Navigation Compass | 2 | 0 |
| 2 | 2648 | Depth/Pressure Gauge (Anal | 1 | 0 |
| 3 | 9316 | 95.1 cu ft Tank | 12 | 0 |
| 4 | 11635 | Camera and Case | 1 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1269 | 16/12/1994 | UPS | FOB | Credit | 1400 | 1400 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 5378 | Divers Knife and Sheath | 20 | 0 |

| | |
|---|---|
| *ID Cliente* | 1231 |
| *Società* | Unisco |
| *Indirizzo* | PO Box Z-547 |
| *Città* | Freeport |
| *N. Telefono* | 809-555-3915 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1060 | 28/02/1989 | US Mail | FOB | Check | 15355 | 15355 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 5324 | Chisel Point Knife | 45 | 0 |
| 2 | 5349 | Flashlight | 8 | 0 |
| 3 | 11518 | Tow able Video Camera (B&W | 5 | 0 |
| 4 | 12301 | Boat Tow able Metal Detector | 5 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1073 | 15/04/1989 | US Mail | Net 30 | MC | 19414 | 19414 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 2954 | Dive Computer | 8 | 0 |
| 2 | 11221 | Remotely Operated Video Sy | 6 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1102 | 06/06/1992 | UPS | FOB | Credit | 2844 | 2844 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 1328 | Regulator System | 3 | 0 |
| 2 | 2350 | Compass Console Mount | 2 | 0 |
| 3 | 9312 | 60.6 cu ft Tank | 4 | 0 |
| 4 | 9318 | 71.4 cu ft Tank | 4 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1160 | 01/06/1994 | US Mail | FOB | Check | 2206,85 | 2206,85 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 1364 | Second Stage Regulator | 1 | 0 |
| 2 | 1390 | First Stage Regulator | 2 | 0 |
| 3 | 2613 | Dive Computer | 2 | 0 |
| 4 | 5324 | Chisel Point Knife | 4 | 0 |
| 5 | 7612 | Krypton Flashlight | 3 | 0 |
| 6 | 9354 | 75.8 cu ft Tank | 4 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1173 | 16/07/1994 | US Mail | Net 30 | MC | 54 | 54 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 2630 | Wrist Band Thermometer (F) | 3 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1178 | 02/08/1994 | DHL | FOB | Credit | 5511,75 | 5511,75 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 900 | Dive kayak | 1 | 0 |
| 2 | 1390 | First Stage Regulator | 2 | 0 |
| 3 | 5324 | Chisel Point Knife | 1 | 0 |
| 4 | 7612 | Krypton Flashlight | 4 | 0 |
| 5 | 11652 | Video Light | 1 | 0 |
| 6 | 12301 | Boat Tow able Metal Detector | 4 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1202 | 06/10/1994 | UPS | FOB | Credit | 4205 | 4205 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 3326 | Front Clip Stabilizing Vest | 12 | 0 |
| 2 | 9316 | 95.1 cu ft Tank | 2 | 0 |
| 3 | 9318 | 71.4 cu ft Tank | 1 | 0 |

| Ordine | Data | Metodo | Termini | Tipo Pag. | N. Items | Tot. Pagato |
|---|---|---|---|---|---|---|
| 1278 | 23/12/1994 | DHL | FOB | Credit | 11568 | 11568 |

| Dettaglio | CD Parte | Descrizione | Quant. | Sconto |
|---|---|---|---|---|
| 1 | 3316 | Stabilizing Vest | 4 | 0 |
| 2 | 5356 | Medium Stainless Steel Knife | 3 | 0 |
| 3 | 9312 | 60.6 cu ft Tank | 4 | 0 |