LINGUAGGIO SQL

Introduzione

Il linguaggio SQL (*Structured Query Language*) è un linguaggio di tipo non procedurale o di tipo dichiarativo e rappresenta lo standard per l'interrogazione delle basi di dati relazionali.

In realtà la dicitura "interrogazione" che si usa di solito e' piuttosto riduttiva dal momento che esso contiene, oltre ai costrutti di interrogazione (QML - Query Manipulation Language), anche quelli di definizione dei dati (DDL - Data Definition Language) e quelli per la loro manipolazione (DML - Data Manipulation Language).

In particolare il termine Structured (strutturato) di SQL è riferito al fatto che si possono inserire comandi all'interno di altri comandi; più in generale si parla di query innestate.

Come spesso succede nel mondo dell' informatica anche in questo caso e' nato uno **"standard"** che avrebbe permesso a tutti di parlare la stessa lingua, ed appena e' nato sono nati così tanti dialetti che non parlano l' uno con l'altro. Lo standard si chiama **ANSI SQL (Structured Query Language della American National Standard Institute)**, e ovviamente sebbene TUTTI dicano che il loro linguaggio e' ANSI Compliant (compatibile con l' ansi SQL) in realtà tutti hanno differenze sostanziali l' uno con l'altro.
I più diffusi linguaggi SQL oggi sono (ovviamente) quelle dei più diffusi database relazionali, cioè:

- Oracle SQL
- Transact - SQL
- PostgreSQL
- MySQL
- SQLInformix
- DBII SQL
- Interbase

Non potendo fare un corso su ognuno di essi (in primis perché molti non li conosco) farò riferimento all' ANSI SQL.
Le differenze fra le varie versioni dipendono essenzialmente dai prodotti software, tutti in genere si rifanno allo standard de facto adottato nel 1992 detto **SQL2** o **SQL92,** e sono riscontrabili dai manuali stessi.

Una estensione a tale standard è l' **SQL3** che implementa nuove caratteristiche come la ricorsione e le funzionalità per il trattamento degli oggetti.

Utilizzo di SQL

Il linguaggio SQL può essere usato in modalità stand-alone o in modalità embedded (linguaggio ospite).

- In modalità stand alone le istruzioni possono essere inviate al RDBMS in modalità interattiva o batch (da eseguire in sequenza). In questo caso l'interazione con l'utente avviene direttamente con l'interprete SQL che può mettere a disposizione oltre alla classica riga di comando, menu, finestre, e icone per guidare l'utente nella costruzione di query. In tale caso si parla di ambiente Query Builder.
- In modalità embedded SQL può essere utilizzato all'interno di altri linguaggi di programmazione, chiamati linguaggi ospite, come **C**, **Pascal**, **Java**, o di linguaggi di scripting quali **PHP**, **JSP** o **ASP**, offrendo la possibilità agli sviluppatori di poter interagire direttamente con il RDBMS.
  Nel caso in cui il linguaggio ospite non possiede le primitive per l'interfacciamento al DBMS, si ricorre a interfacce standard chiamate **ODBC** (Open DataBase Connectivity). Ogni produttore software di DBMS fornisce, non sempre gratuitamente, i driver ODBC necessari al collegamento.

Le funzioni di SQL

Il linguaggio assolve alle funzioni di:

- ***Data Definition Language (DDL)*** che prevede le istruzioni per la definizione o la modifica della struttura del database relazionale: creare modificare ed eliminare database e relativi schemi;i inoltre è possibile specificare vincoli, sia a livello di tupla (o "riga") che a livello di tabella, definire nuovi domini, oltre a quelli predefiniti.

  Le istruzioni più importanti sono:

ALTER INDEX	Modifica i parametri dell' indice
ALTER TABLE	Modifica la struttura di una tabella
ALTER VIEW	Modifica una vista
CREATE INDEX	Crea un indice
CREATE TABLE	Crea una tabella
CREATE VIEW	Crea una vista
DROP INDEX	Elimina un indice
DROP TABLE	Elimina una tabella
DROP VIEW	Elimina una vista
RENAME	Rinomina un oggetto
TRUNCATE	Cancella irrimediabilmente tutte le righe di una tabella

- Questi comandi, vista la loro potenza e pericolosità, sono spesso resi eseguibili a pochi utenti, tra cui il Data Base Administrator (DBA), che si occupa di installare il software del database, di creare e attivare il DB, di farne periodicamente il backup, di migliorare le performance del DB, di creare e gestire gli utenti che vi accedono e, in generale, di supervisionare l'andamento della base dati.
- ***Data Manipulation Language (DML)*** che prevede le istruzioni per la manipolazione dei dati.
  Interrogare il database; inserire, aggiornare o eliminare i dati memorizzati nelle tabelle.

  Le istruzioni più importanti sono:

SELECT	Seleziona dati da una o più tabelle
DELETE	Elimina i dati da una tabella secondo alcune condizioni.
INSERT	Inserisce nuovi dati in una tabella
UPDATE	Modifica i dati di una o più righe di una tabella

- ***Data Control Language (DCL)*** che prevede le istruzioni per abilitare o inibire l'accesso ai dati , sia in senso di impostazione della politica di condivisione degli accessi, sia attribuzione dei privilegi sugli oggetti del database.

  I comandi di controllo della sicurezza permettono di gestire gli accessi al DB e sono:

GRANT	Fornisce un privilegio a un utente o a un gruppo di utenti
REVOKE	Toglie un privilegio a uno o piu' utenti

  I comandi di controllo della transazione permettono di gestire le modifiche operate dai comandi
  di Data Manipulation e sono:

COMMIT	Rende permanenti le modifiche operate dall'inizio della transazione corrente
ROLLBACK	Annulla le modifiche operate dall'inizio dell'operazione o dall'ultimo savepoint e riporta i dati alle condizioni iniziali
SAVEPOINT	Stabilisce un punto oltre il quale e' possibile effettuare un rollback
LOCK TABLE	Impedisce l'accesso ad una tabella agli altri utenti
SET TRANSACTION	Stabilisce determinate proprietà per la transazione corrente

I comandi DDL – ALTER

Alter index

Modifica alcuni parametri dell' indice.

Sintassi:

ALTER INDEX indice STORAGE clausola

Parametri:

indice: il nome dell' indice da creare.

Esempio:
ALTER INDEX utenti STORAGE(200K);

Alter table

Modifica la struttura di una tabella; usando Alter Table e' possibile:

· Aggiungere e modificare una colonna.
· Aggiungere un constraint.
· Modificare i parametri di dimensionamento della tabella.
· Abilitare o disabilitare i constraint.

Sintassi:

ALTER TABLE tabella

[ADD {colonna tipo_dato[, ...] | constraint definizione[, ...]}]

[MODIFY colonna tipo_dato[, ...]]

[DROP CONSTRAINT constraint]

[PCTFREE intero] [PCTUSED intero] [STORAGE clausola]

Parametri:

tabella: il nome della tabella che deve essere modificata.

colonna: il nome della colonna che deve essere modificata o aggiunta.

costraint: il nome del costraint che deve essere aggiunta o tolto.

Esempi:

ALTER TABLE cantanti add(categoria decimal(2));

ALTER TABLE cantanti modify (cognome varchar(20));

I comandi DDL – CREATE

Create table

Crea una nuova tabella con un dato nome nel database corrente , specificando la definizione delle colonne ed, eventualmente, permette di riempire tale tabella con il risultato di una interrogazione da altra tabella.

Sintassi:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] <nome tabella>

[(create_definizione,...)]
[table_opzioni]
[select_statement]

create_definizione:

col_name *tipo* [NOT NULL | NULL]
[DEFAULT default_value]
[AUTO_INCREMENT]

[PRIMARY KEY] [reference_definition]
or PRIMARY KEY (index_col_name,...)
or KEY [index_name] (index_col_name,...)
or INDEX [index_name] (index_col_name,...)
or UNIQUE [INDEX] [index_name] (index_col_name,...)
or FULLTEXT [INDEX] [index_name] (index_col_name,...)
or [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name,...)
[reference_definition]
or CHECK (expr)

tipo:

TINYINT [(length)] [UNSIGNED] [ZEROFILL]
or SMALLINT [(length)] [UNSIGNED] [ZEROFILL]
or MEDIUMINT [(length)] [UNSIGNED] [ZEROFILL]
or INT [(length)] [UNSIGNED] [ZEROFILL]
or INTEGER [(length)] [UNSIGNED] [ZEROFILL]
or BIGINT [(length)] [UNSIGNED] [ZEROFILL]
or REAL [(length,decimals)] [UNSIGNED] [ZEROFILL]
or DOUBLE [(length,decimals)] [UNSIGNED] [ZEROFILL]
or FLOAT [(length,decimals)] [UNSIGNED] [ZEROFILL]
or DECIMAL (length,decimals) [UNSIGNED] [ZEROFILL]
or NUMERIC (length,decimals) [UNSIGNED] [ZEROFILL]
or CHAR (length) [BINARY]
or VARCHAR (length) [BINARY]
or DATE
or TIME
or TIMESTAMP
or DATETIME
or TINYBLOB
or BLOB
or MEDIUMBLOB
or LONGBLOB
or TINYTEXT
or TEXT

or MEDIUMTEXT
or LONGTEXT
or ENUM(value1,value2,value3,...)
or SET(value1,value2,value3,...)

index_col_name:

col_name [(length)]

reference_definition:

REFERENCES tbl_name [(index_col_name,...)]
[MATCH FULL | MATCH PARTIAL]
[ON DELETE reference_option]
[ON UPDATE reference_option]

reference_option:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_opzioni:

TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MYISAM }
or AUTO_INCREMENT = #
or AVG_ROW_LENGTH = #
or CHECKSUM = {0 | 1}
or COMMENT = "string"
or MAX_ROWS = #
or MIN_ROWS = #
or PACK_KEYS = {0 | 1}
or PASSWORD = "string"
or DELAY_KEY_WRITE = {0 | 1}
or ROW_FORMAT= { default | dynamic | fixed | compressed }
or RAID_TYPE= {1 | STRIPED | RAID0 } RAID_CHUNKS=# RAID_CHUNKSIZE=#
or UNION = (table_name,[table_name...])
or DATA DIRECTORY="directory"
or INDEX DIRECTORY="directory"

select_statement:

[IGNORE | REPLACE] SELECT ... (UNA ISTRZIONE SELECT VALIDA)

SHOW

SHOW fornisce informazioni sui database, tabelle, colonne, or informazioni di stato del server.
Se viene utilizzata la clausola Like wild , i valori utilizzabili sono "*" e "_" .

SHOW DATABASES [LIKE wild]

| SHOW [OPEN] TABLES [FROM db_name] [LIKE wild]

| SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE wild]

| SHOW INDEX FROM tbl_name [FROM db_name]

| SHOW TABLE STATUS [FROM db_name] [LIKE wild]

| SHOW STATUS [LIKE wild]

| SHOW VARIABLES [LIKE wild]

| SHOW LOGS

| SHOW [FULL] PROCESSLIST

| SHOW GRANTS FOR user

| SHOW CREATE TABLE table_name

| SHOW MASTER STATUS

| SHOW MASTER LOGS

| SHOW SLAVE STATUS

I comandi DML

Select:

è uno dei comandi più importanti del linguaggio SQL, usato per selezionare dati da una o piu' tabelle.

Si osservi che la clausola SELECT realizza l'operatore di proiezione dell'algebra relazionale, non quello di selezione; quest'ultimo è realizzato dalla clausola WHERE.

Sintassi:

SELECT [ALL|DISTINCT] { * | lista_colonne }

FROM lista_tabelle

[WHERE condizione]

[GROUP BY lista_colonne [HAVING condizione]]

[{UNION|INTERSECT|MINUS} select ...]

[ORDER BY {lista_colonne|posizione} [ASC|DESC]]

[FOR UPDATE OF lista_colonne]

Parametri:

***:** tutte le colonne dell'oggetto.

ALL: restituisce tutte le righe selezionate.

DISTINCT: agisce su tutte le colonne e le righe della select, eliminando le combinazioni duplicate.

lista_colonne: una lista separata da virgole di colonne da selezionare.

lista_tabelle: una lista di tabelle o viste.

condizione: una lista booleana di condizioni usata per discriminare la ricerca delle righe.

Insert

Aggiunge righe a una tabella.

Sintassi:

INSERT INTO tabella [lista_colonne] {VALUES lista_valori|subquery}

Parametri:

tabella: nome della tabella in cui inserire le righe.

lista_colonne: lista delle colonne che devono essere coinvolte nell'inserimento.

lista_valori: lista di valori da inserire, nello stesso ordine delle colonne specificate oppure, se le colonne non sono specificate, nello stesso ordine di creazione delle colonne stesse.

subquery: il risultato della selezione sarà inserito nella tabella

Esempio:

INSERT INTO cantanti (cognome, nome)
SELECT first_name, second_name FROM rockers WHERE first_name LIKE 'S%';

Delete:

Elimina righe da una tabella o da una vista (solo se e' costruita su una sola tabella).

Sintassi:

DELETE [FROM] tabella [WHERE condizione]

Parametri:

tabella: il nome della tabella da cui eliminare le righe.

condizione: una lista booleana contenente i valori per cui eliminare le righe; se non viene specificata nessuna condizione, la DELETE eliminerà tutte le righe della tabella.

Esempio:

DELETE FROM città WHERE nazione = 'SPAGNA';

Update:

Modifica il contenuto di una tabella o di una vista (solo se e' costruita su una sola tabella).

Sintassi:

UPDATE tabella SET col = expr [,col = expr] [WHERE condizione]

Parametri:

tabella: il nome della tabella da modificare.

col: il nome della colonna.

expr: il valore da assegnare.

condizione: l'eventuale elenco delle condizioni per cui applicare la modifica.

Esempio:
UPDATE registrazioni SET tipo_supporto = 'CD' WHERE to_char (date,'YYYY') > '1990';

La clausola WHERE

Nella parte che segue il WHERE del comando SELECT può essere inserita una qualsiasi espressione booleana (cioè una qualsiasi espressione che, risolta, dia come risultato uno dei due valori booleani (Vero, Falso)) che sia calcolabile, ovviamente nel risultato della query verranno inserite tutte e sole le righe della tabella che rendono vera l'espressione che segue la clausola WHERE (*Where Condition*).

Es: Sia la tabella Dipendenti:

Cognome	Nome	Età	Data_assunzione
Rossi	Mario	40	10/5/1995
Verdi	Giorgina	20	1/3/1999
Bianchi	Paolo	50	1/6/1975
Gialli	Loredana	35	24/9/1989
Rossi	Giorgio	46	15/7/1979

La query *SELECT * FROM Dipendenti WHERE True* ritorna tutta la tabella, mentre la query *SELECT * FROM Dipendenti WHERE False* non ritorna nulla.

La query *SELECT Cognome FROM Dipendenti WHERE Età>35 AND (Cognome='Rossi' OR Nome Like 'Paolo') AND Età<>46* ritorna solamente le seguenti righe:

Cognome	Nome	Età	Data_assunzione
Rossi	Mario	40	10/5/1995
Bianchi	Paolo	50	1/6/1975

Dopo la *where condition* possono essere presenti ancora alcuni comandi:

```
GROUP BY
HAVING
ORDER BY {Campo1, Campo2, ...} [ASC | DESC]
```

Il più semplice è ovviamente ORDER BY, che serve per ordinare una tabella rispetto ad uno o più campi, le parole chiave opzionali in fondo significano il tipo di ordinamento: ASC (o niente) sta per ascendente, DESC sta per discendente.
Vediamo subito con un esempio come funziona:

Es: Con la tabella dell'esempio precedente, la query

SELECT * FROM Dipendenti WHERE Età<=40 ORDER BY Cognome DESC

ritorna

Cognome	Nome	Età	Data_assunzione
Verdi	Giorgina	20	1/3/1999
Rossi	Mario	40	10/5/1995
Gialli	Loredana	35	24/9/1989

Se invece non avessimo messo alcunché in fondo o avessimo messo ASC la tabella sarebbe stata ordinata in ordine crescente rispetto al Cognome.

Si può ordinare anche rispetto a più campi, i campi che vengono dopo il primo sono usati per l'ordinamento nel caso in cui i valori del primo sono uguali:

SELECT * FROM Dipendenti ORDER BY Cognome, Età

Cognome	Nome	Età	Data_assunzione
Bianchi	Paolo	50	1/6/1975
Gialli	Loredana	35	24/9/1989
Rossi	Mario	40	10/5/1995
Rossi	Giorgio	46	15/7/1979
Verdi	Giorgina	20	1/3/1999

Le funzioni di aggregazione
Uno dei più utili comandi all'interno della clausola WHERE è sicuramente il GROUP BY, che serve per aggregare i dati secondo alcuni criteri; nelle query con questa clausola si possono usare funzioni che permettono di calcolare medie, somme, massimi, minimi etc; per esempio la funzione COUNT effettua il conteggio delle righe restituite, mentre la funzione AVG effettua la media dei valori dell'intervallo.

Nella parte dopo la parola chiave SELECT, che deve contenere i nomi dei campi da estrarre, possono essere messi solo campi contenuti nella clausola GROUP BY o funzioni di calcolo di valori su intervalli (perché? Sai spiegarlo?)

Es: Prendiamo ora invece la tabella Progetti:

Nome_progetto	Ore_uomo	Ambito
fax	400	Comunicazioni
mail	700	Comunicazioni
contabilità	3500	Gestionale
stipendi	7000	Gestionale
SMTP	400	Comunicazioni
POP	7000	Comunicazioni
inventario	10500	Gestionale

La query

SELECT Ambito, Count(*) As TOT, AVG(Ore_uomo) AS MEDIA
FROM Progetti GROUP BY Ambito

Ritorna

Ambito	TOT	MEDIA
Comunicazioni	4	2125
Gestionale	3	7000

La funzione AVG effettua la media sui valori contenuti nel campo Ore_uomo, raggruppati secondo il valore contenuto nel campo Ambito.

La clausola HAVING pone delle condizioni sulle clausole di gruppo, per esempio per considerare solo i raggruppamenti che hanno più di un certo numero di elementi:

Es: con la stessa tabella dell'esempio precedente, la query

```
SELECT Count(*) As NUM, Ore_uomo
FROM Progetti
GROUP BY Ore_uomo
HAVING Count(*)>1
```

NUM	Ore_uomo
2	400
2	7000

Che si riferiscono ovviamente alla 1°, 4°, 5° e 6° riga della tabella.