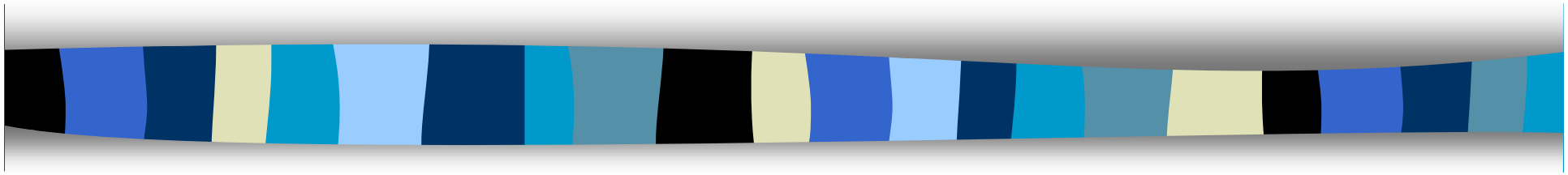
INFORMATICA



Prof. MARCO CASTIGLIONE

Istituto Tecnico Statale tito Acerbo - PESCARA

Il linguaggio SQL

Il linguaggio SQL

INTRODUZIONE

Il linguaggio SQL

STRUCTURED QUERY LANGUAGE

- **DDL** (*Data Definition Language*)
  Permette la creazione di un database (articolazione, correlazioni e vincoli) mediante un apposito linguaggio.

- **DML** (*Data Manipulation Language*)
  Permette la manipolazione (inserimento, variazione e cancellazione) dei dati attraverso un linguaggio.

- **QL** (*Query Language*)
  Permette l'estrazione (interrogazione) dei dati mediante un apposito linguaggio.

Il linguaggio SQL

STORIA

- **Fine anni ‘70**
  IBM, linguaggio per i software DBMS prodotti dalla IBM.
- **Standard**
  1986 standard ANSI.
  1987 standard ISO.
  1992 SQL2 ISO 9075.
  1999 SQL3 ISO 9075-2.

- **SQL**
  Oggi utilizzati in tutti i prodotti DBMS come linguaggio di comandi (Oracle, Informix, MySQL, SQLServer, Access…)

Il linguaggio SQL

SIMBOLOGIA

- **Simboli previsti**
  - Caratteri alfabetici.
  - Cifre decimali.
  - Operatori aritmetici.
  - Operatori di confronto.
  - Operatori logici (AND, OR, NOT).
  - Altri caratteri.
- **Notazione**
  NomeTabella.NomeAttributo
  NB. Nome della tabella può essere omesso se non ci sono ambiguità.

Il linguaggio SQL

TIPI STANDARD

BOOLEAN	Valore logico	True, False
CHARACTER(n)	Stringa di lunghezza n	n da 1 a 15000
DATE	Data nella forma MM/GG/AA	
TIME	Ora nella forma HH:MM	
INTEGER(p)	Numero intero con precisione *p*	*p* da 1 a 45
SMALLINT	Numero intero con precisione 5	da -32768 a 32767
INTEGER	Numero intero con precisione 10	da -2.147.483.648 a 2.147.483.647
DECIMAL(p,s)	Numero decimale con precisione *p* e *s* cifre decimali	*p* da 1 a 45 e *s* da 0 a *p*
REAL	Numero reale con mantissa di precisione 7	valore 0 oppure valore assoluto da 1E-38 a 1E+38
FLOAT (o DOUBLE PRECISION)	Numero reale con mantissa di precisione 15	valore 0 oppure valore assoluto da 1E-38 a 1E+38
FLOAT(p)	Numero reale con mantissa di precisione *p*	*p* da 1 a 45

Il linguaggio SQL

TIPI STANDARD

- **Abbreviazioni**
  CHAR(n) = CHARACTER(n)
  INT(n) = INTEGER(n)
  DEC(p,s) = DECIMAL(p,s)

- **Stringhe**
  Delimitatore di stringa ‘ o “.

- **NULL**
  Valore non disponibile o non definito.
  Non è uguale a 0 per dati numerici o “” stringa vuota.
  Negli ordinamenti crescenti compare prima degli altri valori.

Il linguaggio SQL

INTEGRITA' REFERENZIALE

- **Definizione**
  Insieme di regole che garantiscono l'integrità dei dati nelle tabelle collegate tramite la chiave esterne: eliminano gli errori dovuti a inserimento, cancellazione e aggiornamento dei dati.

- **Regole pratiche**
  1 - Non è possibile inserire un valore della chiave esterna se tale valore non esiste nella tabella primaria.
  2 - Non è possibile eliminare un record nella tabella primaria se esistono record legati ad esso nella tabella correlata.
  3 - Non è possibile modificare valori della chiave primaria se esistono record legati ad essa nella tabella correlata.

Il linguaggio SQL

DDL

Il linguaggio SQL

CREAZIONE DI UNA TABELLA

- **CREATE TABLE** NomeTabella
  (NomeCampo **Tipo+Caratteristiche,**
  Unique (NomeCampo**),**
  Primary Key (NomeCampo**),**
  Foreign Key (NomeCampo**) references** NomeTab(Campo)
  clausole di integrità**);**

- **Caratteristiche**
  not null valore non nullo
  default 'valore' calore di default
- **Integrità**
  ON DELETE set null|cascade|no action
  ON UPDATE set null|cascade|no action

Il linguaggio SQL

ESEMPIO

```
CREATE TABLE Dipartimenti
    (Codice          char(5),
     Descrizione     char(20) not null,
     Sede            char(20),
     Manager         smallint,
     Primary Key (Codice),
     Unique (Descrizione),
     Foreign Key (Manager) references Impiegati(ID)
             On Delete set null
             On Update cascade);
```

Il linguaggio SQL

MODIFICA DI UNA TABELLA

- **Inserimento campo**
  ALTER TABLE NomeTabella **ADD** NomeCampo Tipo**;**
- **Eliminazione campo**
  ALTER TABLE NomeTabella **DROP** NomeCampo**;**

- **Creazione indice**
  CREATE (UNIQUE) INDEX NomeIndice
  ON NomeTabella(ElencoCampi)**;**
- **Eliminazione indice**
  DROP INDEX NomeIndice **ON** NomeTabella**;**

- **Eliminazione Tabella**
  DROP TABLE NomeTabella**;**

Il linguaggio SQL

DML

Il linguaggio SQL

MANIPOLAZIONE DEI DATI

- **INSERT INTO** NomeTabella (NomeCampi) **VALUES** (valori);
- **UPDATE** NomeTabella **SET** Campo=Valore **WHERE** ???**;**
- **DELETE FROM** NomeTabella **WHERE** ???**;**

```
INSERT INTO Impiegati
(ID, Nome, Cognome, Residenza, Stipendio, Dipartimento)
VALUES(20,'Mario', 'Rossini','Caserta',31500, 'Mag');
```

```
UPDATE Impiegati
SET Dipartimento = 'Prod'
WHERE Matricola = 20;
```

```
DELETE FROM Impiegati
WHERE ID = 20;
```

Il linguaggio SQL

QL

Il linguaggio SQL

COMANDO SELECT

- **SELECT** **Campi**
- **FROM** **Tabelle**
- **WHERE** **Condizioni**

- **Predicati**
  ALL tutte le righe
  DISTINCT elimina le duplicazioni

```
SELECT ALL *
FROM Impiegati
WHERE Residenza = 'Milano';
```

```
SELECT DISTINCT Residenza
FROM Impiegati;
```

Il linguaggio SQL

ALIAS

- **Tabelle / Campi**
  Ci consente di rinominare tabelle e campi, ottenendo nomi più compatti o più significativi.

```
SELECT I.Cognome, I.Nome FROM Impiegati AS I;
SELECT DataN AS Nascita FROM Studenti;
```

Il linguaggio SQL

OPERAZIONI RELAZIONALI

- **Selezione**

σ_P Impiegati Selezione di Impiegati per Stipendio < 31000

```
SELECT *
FROM Impiegati
WHERE Stipendio < 31000;
```

- **Proiezione**

Π_L Impiegati Proiezione di Impiegati su Cognome, Nome, ID

```
SELECT Cognome, Nome, ID
FROM Impiegati
```

Il linguaggio SQL

OPERAZIONI RELAZIONALI

- **Congiunzione - JOIN**

$Impiegati_{FK} \bowtie Dipartimenti_{PK}$ Congiunzione di Impiegati su **Dipartimento** e Dipartimenti su **Codice**

```
SELECT *
FROM  Impiegati, Dipartimenti
WHERE Dipartimento = Codice;
```

```
SELECT *
FROM  Impiegati, Dipartimenti
WHERE Impiegati.Dipartimento = Dipartimenti.Codice;
```

```
SELECT Impiegati.*, Dipartimenti.*
FROM Dipartimenti INNER JOIN Impiegati ON
     Dipartimenti.Codice = Impiegati.Dipartimento;
```

Il linguaggio SQL

ESEMPIO JOIN

- Elenco dei dipendenti che lavorano in una sede di Roma con i dati relativi a Cognome e Nome del dipendente e Descrizione del dipartimento.

1. σ_P Dipartimenti — Temp1=Selezione di Dipartimenti per **Sede = 'Roma'**
2. Temp1 $_{PK}\bowtie$ Impiegati $_{FK}$ — Temp2=Congiunzione di Temp1 su **Codice** e Impiegati su **Dipartimento**
3. Π_L Temp2 — Proiezione di Temp2 su **Cognome, Nome, Descrizione**

```
SELECT Cognome, Nome, Descrizione
FROM Impiegati, Dipartimenti
WHERE Dipartimento=Codice AND
      Sede = 'Roma';
```

```
SELECT Impiegati.Cognome, Impiegati.Nome,
       Dipartimenti.Descrizione
FROM Impiegati, Dipartimenti
WHERE Impiegati.Dipartimento = Dipartimenti.Codice AND
      Dipartimenti.Sede = 'Roma';
```

Il linguaggio SQL

JOIN ESTERNE

- **Left Join**

```
SELECT Impiegati.Nome, Impiegati.Cognome, Dipartimenti.Descrizione
FROM Impiegati LEFT JOIN Dipartimenti
     ON Dipartimenti.Codice = Impiegati.Dipartimento;
```

- **Right Join**

```
SELECT Impiegati.Nome, Impiegati.Cognome, Dipartimenti.Descrizione
FROM Impiegati RIGHT JOIN Dipartimenti
     ON Dipartimenti.Codice = Impiegati.Dipartimento;
```

Il linguaggio SQL

OPERAZIONI INSIEMISTICHE

- **Unione**

```
(SELECT Impiegati.Nome, Impiegati.Cognome, Dipartimenti.Descrizione
 FROM Impiegati LEFT JOIN Dipartimenti ON Dipartimenti.Codice =
                                          Impiegati.Dipartimento)
 UNION
(SELECT Impiegati.Nome, Impiegati.Cognome, Dipartimenti.Descrizione
 FROM Impiegati RIGHT JOIN Dipartimenti ON Dipartimenti.Codice =
                                           Impiegati.Dipartimento);
```

- **Intersezione**
- **Differenza**

```
T1 INTERSECT T2;        Intersezione
T1 EXCEPT T2;           Differenza
```

ATTENZIONE: in Access non sono ammesse.

Il linguaggio SQL

FUNZIONI DI AGGREGAZIONE

- **COUNT**
  Numero di record.

```
SELECT COUNT (*)
FROM Impiegati;
SELECT COUNT (Dipartimento)
FROM  Impiegati;
```

```
SELECT COUNT(DISTINCT Residenza)
FROM  Impiegati
WHERE Dipartimento = 'Prod';
```

- **SUM**
  Somma valori relativi ad un campo.

```
SELECT SUM (Stipendio)
FROM  Impiegati
WHERE Dipartimento = 'Mkt';
```

- **AVG, MIN, MAX**
  Valore medio,
  minimo e massimo.

```
SELECT AVG(Stipendio)
FROM  Impiegati, Dipartimenti
WHERE Dipartimento = Codice AND
      Sede = 'Torino';
```

```
SELECT MIN(Stipendio), MAX(Stipendio)
FROM  Impiegati;
```

Il linguaggio SQL

ORDINAMENTO E RAGGRUPPAMENTO

- **ORDER BY**
  Clausola per l'ordinamento crescente (ASC) e decrescente (DES).

```
SELECT Cognome, Stipendio
FROM Impiegati
ORDER BY Stipendio DESC, Cognome;
```

- **GROUP BY**
  Raggruppamento di valori omogenei per effettuare somme o conteggi.

```
SELECT Dipartimento, COUNT(ID), SUM(Stipendio),
FROM Impiegati
GROUP BY Dipartimento;
```

Il linguaggio SQL

ESEMPIO RAGGRUPPAMENTO

```
SELECT Dipartimento, COUNT(ID), SUM(Stipendio),
FROM Impiegati
GROUP BY Dipartimento;
```

Dipartimento	ID	Stipendio
	5	28300
Amm	12	61000
Amm	16	29000
Direz	13	85000
Direz	19	39000
Mag	6	25000
Mag	18	61500
Mkt	17	52600
Prod	1	32000
Prod	10	65000
Prod	11	41000
R&S	14	57000

Dipartimento	Count(ID)	Sum(Stipendio)
	1	28300
Amm	2	90000
Direz	2	124000
Mag	2	86500
Mkt	1	52600
Prod	3	138000
R&S	1	57000

Il linguaggio SQL

CONDIZIONI SUI RAGGRUPPAMENTI

- **HAVING**
  E' al clausola utilizza per inserire criteri nei raggruppamenti.
  Prima verifica delle condizioni WHERE.

```
SELECT Dipartimento, COUNT(ID), SUM(Stipendio),
FROM Impiegati
GROUP BY Dipartimento
HAVING COUNT(ID) > 2;
```

Dopo aggregazione con i criteri HAVING.

```
SELECT Descrizione, COUNT(*), SUM(Stipendio)
FROM Impiegati, Dipartimenti
WHERE Impiegati.Dipartimento = Dipartimenti.Codice AND
      Sede = 'Torino'
GROUP BY Descrizione
HAVING COUNT(*) > 1;
```

Il linguaggio SQL

UTILIZZO DELLE CLAUSOLE DEL COMANDO **SELECT**

- **Nota Bene**
  - Solo SELECT e WHERE sono obbligatorie.
  - L'ordine di utilizzo è quello riportato, SELECT precede FROM, HAVING precede ORDER BY.

  - Ordine di elaborazione d parte del DBMS: FROM=>WHERE=>GROUP BY=>HAVING=> SELECT=>ORDER BY

SELECT	Elenco colonne da mostrare
FROM	Tabelle da cui estrarre le righe
WHERE	Condizioni sulle congiunzioni o sulle righe estratte
GROUP BY	Campi da considerare per i raggruppamenti
HAVING	Condizioni sui raggruppamenti
ORDER BY	Ordinamenti sulle colonne elencate nella clausola SELECT

Il linguaggio SQL

CONDIZIONI DI RICERCA

- **Operatori logici**
  AND, OR e NOT.

- **BETWEEN**
  Valori compresi in un intervallo.

```
SELECT Cognome, Nome, Residenza
FROM Impiegati
WHERE Stipendio BETWEEN 30000 AND 45000;
```

Corrisponde all'utilizzo degli operatori logici:

```
Stipendio>=30000 AND Stipendio<=45000
```

Il linguaggio SQL

CONDIZIONI DI RICERCA

- **IN**
  Valore compreso in un elenco riportato.

```
SELECT *
FROM Impiegati
WHERE Residenza IN ('Torino','Venezia','Palermo');
```

  Corrisponde all'utilizzo degli operatori logici:

```
Residenza ='Torino' OR Residenza ='Venezia' OR Residenza ='Palermo'
```

- **IS NULL / IS NOT NULL**
  Ricerca di un valore NULL.

```
SELECT Cognome, Nome
FROM Impiegati
WHERE Stipendio IS NOT NULL;
```

Il linguaggio SQL

CONDIZIONI DI RICERCA

- **LIKE**
  Per attributi di tipo carattere consente di utilizzare caratteri jolly.
  - _ (underline) indica un carattere in una precisa posizione.
  - **%** indica una sequenza di caratteri in una precisa posizione.

  LIKE 'xyz%': tutte le stringhe che iniziano con xyz.
  LIKE '%xyz': tutte le stringhe che terminano con xyz.
  LIKE '%xyz%': tutte le stringhe che contengono xyz.
  LIKE '_xyz': tutte le stringhe di 4 caratteri che terminano con xyz.

```
SELECT Cognome, Dipartimento
FROM Impiegati
WHERE Cognome LIKE 'R%';
```

Il linguaggio SQL

CONDIZIONI DI RICERCA

- **ANY**
  Da utilizzare in una clausola WHERE con risultato VERO se il confronto è vero per almeno uno dei valori.

```
SELECT Cognome, Nome
FROM Impiegati
WHERE Dipartimento = 'Amm' AND
      Stipendio > ANY (  SELECT Stipendio
                         FROM Impiegati
                         WHERE Dipartimento = 'Mkt');
```

- **ALL**
  Da utilizzare in una clausola WHERE con risultato VERO se il confronto è vero per tutti i valori.

```
SELECT Cognome, Nome
FROM Impiegati
WHERE Dipartimento = 'Amm' AND
      Stipendio > ALL ( SELECT Stipendio
                        FROM Impiegati
                        WHERE Dipartimento = 'Mkt');
```

Il linguaggio SQL

VISTE LOGICHE

- **CREATE VIEW**
  Consente di creare una vista, in modo da operare su una parte dei dati come fosse una tabella del database.

```
CREATE VIEW PersonaleDipartimenti
    (Dipart,NumDipendenti,TotStipendi) AS
    SELECT Dipartimento, Count(*), SUM(Stipendio)
    FROM Impiegati
    GROUP BY Dipartimenti;
```

```
SELECT AVG(NumDipendenti)
FROM PersonaleDipartimenti;
```

- **DROP VIEW**
  Elimina una vista precedentemente creata.

```
    DROP VIEW ImpiegatiCapi;
```

Il linguaggio SQL

BIBLIOGRAFIA

- Agostino Lorenzi, Enrico Cavalli, ***Informatica: le base di dati e il linguaggio SQL – Access, MySQL, database in rete***, Edizioni ATLAS, Bergamo, 2007.