Funzionalità del DBMS relazionale

- Funzioni per
  - definizione della base di dati
  - inserimento / rimozione /aggiornamento di informazioni
    - *deve soddisfare i vincoli!*
  - Interrogazione

SQL (Structured Query Language)

- SQL: Linguaggio standard per creazione e interrogazione di DB
- Vediamo solamente come formulare interrogazioni con SQL
- un'interrogazione produce come risultato una tabella
- Clausola Base

  SELECT Lista di attributi

  FROM Elenco relazioni

  WHERE Condizione

Query

- Selezionare Autore e Titolo dei libri editi da Einaudi

```
SELECT Autore, Titolo
FROM   Libri
WHERE Casa_ed = ”Einaudi";
```

- Sezionare Autore e Titolo dei libri di Feltrinelli pubblicati a partire dal 1990

```
SELECT Autore, Titolo
FROM   Libri
WHERE (Casa_ed = "Feltrinelli") and
        (Anno_ed >=1990);
```

- Sleziona **tutti i dati** dei libri della Feltrinelli e Dell’Einaudi

```
SELECT  *
FROM   Libri
WHERE (Casa_ed = "Feltrinelli") or
(Casa_ed = "Einaudi");
```

Provare:

- Trova i libri editi da Einaudi o Bompiani, pubblicati dopo 1980
  (Autore,Titolo,Casa_ed,Anno_ed)
- Trova i libri editi da Einaudi o editi da Bompiani dopo 1980

Ridenomina degli Attributi

- E’ possible ridenominare i nomi degli attributi in una query mediante l’operatore **as:**

Esempio

```
SELECT Titolo AS Libro, Autore AS Scrittore
FROM Libri
WHERE Casa_ed="Einaudi";
```

Esempi: query parametriche

- Trova tutti i libri presenti in biblioteca, dato il nome dell'autore. Il nome dell'autore viene introdotto dall'utente

```
SELECT *
  FROM   Libri
  WHERE  Autore =  [dimmi il nome dell'autore] ;
```

- In esecuzione:

```
    > dimmi il nome dell' autore
Alessandro Manzoni
```

Matching approssimato

- L’operatore **like** permette di confrontare il valore di un attributo con un valore specificato in modo incompleto
- in ACCESS Si usa con gli operatori **?** e *****

(In SQL puro _ e %)

- ? indica un carattere qualsiasi
- *una sequenza di caratteri qualunque

Esempio

```
SELECT Autore, Titolo
FROM   Libri
WHERE Autore like “*Man*”;
```

- Seleziona tutti i libri in cui il nome dell’autore *contiene* “Man”

Esempio con like e parametri

```
SELECT *
 FROM libri
  WHERE  Autore like "*"& [dimmi l'autore]  & "*";
```

- Seleziona tutti i libri in cui il nome dell’autore contiene la stringa introdotta dall’utente
- & e’ l’operatore di concatenazione tra stringhe

Funzioni aggregate

Funzioni aggregate: i valori dipendono da più tuple)

- COUNT: conta il numero delle tuple selezionate
- SUM: somma i valori di un attributo delle tuple selezionate
- MAX: fornisce il massimo valore di un attributo delle tuple selezionate
- MIN: fornisce il minimo valore di un attributo delle tuple selezionate
- AVG: fornisce la media dei valori di un attributo delle tuple selezionate

ESEMPI

- Contare i libri presenti in biblioteca editi da Feltrinelli

```
SELECT Count(*) as TOTALE
FROM   Libri
WHERE Casa_ed = "Feltrinelli";
```

- Qual è il prezzo del più costoso libro presente in biblioteca?

```
SELECT Max(Prezzo)
FROM   Libri;
```

- Calcolare il costo totale dei libri presenti in biblioteca, scritti da Umberto Eco

```
SELECT Sum (Prezzo)  as TotaleEco
FROM   Libri
       WHERE Autore = "Umberto Eco"
```

Raggruppamento e operatori aggregati

- Raggruppare le tuple in base a qualche attributo
- applicare le funzioni aggregate a ciascun raggruppamento

Raggruppamento

GROUP BY:

- Contare quanti libri ci sono di ciascun editore

```
SELECT Casa_ed, Count(*) as NumCopie
FROM   Libri
       GROUP BY Casa_ed;
```

- Per ogni editore, indicare il costo totale dei libri presenti in biblioteca

```
SELECT Casa_ed, Sum (Prezzo) as Totali
FROM   Libri
GROUP BY Casa_ed;
```

Raggruppamento (2)

- Per ogni autore, contare i libri presenti in biblioteca editi da Einaudi, ed indicarne il costo totale

```
SELECT Autore, Count(*) as Numero, Sum (Prezzo) as Valori
FROM   Libri
WHERE Casa_Ed = “Einaudi”
GROUP BY Autore;
```

Raggruppamento (3)

- La clausola **HAVING** consente di imporre una condizione sul risultato di una funzione aggregata

- Indicare il numero di libri presenti in biblioteca degli editori di cui sono presenti almeno 3 libri

```
SELECT Casa_ed, Count(*) AS Num_copie
FROM libri
GROUP BY Casa_ed
HAVING Count(*) >=3;
```

Ordinamento dei risultati

- Si può chiedere che le tuple del risultato siano ordinate in base ai valori dei campi: ORDER BY
- Es. Restituire l'elenco dei libri in catalogo della Bompiani, secondo l'ordine alfabetico degli autori, per anno di edizione decrescente

```
SELECT *
FROM LIBRI
WHERE Casa_ed = "Bompiani"
ORDER BY Autore ASC, Anno_ed DESC;
```

Valori Unici

- SQL restituisce una tabella che contiene tutte le righe che soddisfano una certa condizione, puo' contenere duplicati
- Per eliminare i duplicati si premette la parola chiave **distinct**
- Esempio elenca i libri per autore e titolo senza ripetizioni (ignorando copie multiple e differenti edizioni)

```
SELECT DISTINCT Autore, Titolo
FROM libri
ORDER BY Autore ASC;
```

Query con piu' tabelle - join

- Join: combinare le tuple di più tabelle i cui valori per attributi correlati soddisfano una condizione di confronto (caso più semplice: sono uguali)
- Il join di due relazioni è il sottoinsieme del loro prodotto cartesiano specificato dalla condizione di selezione

Relazioni tra tabelle e Join

- Le relazioni tra tabelle sono espresse da valori comuni di attributi correlati

```
SELECT Utenti.Nome, Libri.Titolo, Prestiti.Data_restituzione
FROM libri, prestiti, utenti
WHERE
Prestiti.Cod_utente=Utenti.Cod_utente
AND Prestiti.N_inv=Libri.N_inv;
```

Libri presi in prestito da un utente il cui nome è un parametro

```
SELECT Utenti.Nome, Libri.Titolo
FROM libri, prestiti, utenti
WHERE (Utenti.Nome like "*"& dimmi l'Utente
   & "*")
   AND Prestiti.Cod_utente=Utenti.Cod_Utente
   AND Prestiti.N_inv=Libri.N_inv;
```

Altro Esempio

- Esempio Seleziona gli studenti e gli esami che hanno sostenuto con i rispettivi titoli e voti:

```
SELECT Studenti.Nome, Studenti.Cognome,
    Corsi.Titolo, Esami.Voto. Esami.Lode
FROM Studenti, Corsi, Esami
WHERE Corsi.CodiceCorso=Esami.CodiceCorso
and Studenti.Matricola = Esami.Matricola;
```

INNER JOIN (Access)

- Modo alternativo per specifcare JOIN:

```
SELECT (Attributi)
FROM Tabella1, Tabella2
WHERE Tabella1.At1 = Tabella2.At2
```

- diventa

```
SELECT (Attributi)
FROM Tabella1 INNER JOIN Tabella2
ON Tabella1.At1 = Tabella2.At2
```

INNER JOIN (Access)

SELECT (Attributi)

FROM Tabella1, Tabella2, Tabella3

WHERE Tabella1.ATX = Tabella2.ATY and Tabella2.ATZ = Tabella3.ATW

- diventa

SELECT (Attributi)

FROM (Tabella1 INNER JOIN Tabella2
ON Tabella1.At1 = Tabella2.At2)
INNER JOIN Tabella3
ON Tabella2.ATZ = Tabella3.ATW)

INNER JOIN

- oppure

```
SELECT (Attributi)
FROM Tabella1 INNER JOIN
        (Tabella2 INNER JOIN Tabella3
            ON Tabella2.ATZ = Tabella3.ATW)
        ON Tabella1.At1 = Tabella2.At2
```

INNER JOIN (Access)

```
SELECT Studenti.Nome, Studenti.Cognome,
   Corsi.Titolo, Esami.Voto,Esami.Lode
FROM
(Studenti INNER JOIN Esami
ON Studenti.Matricola = Esami.Matricola)
INNER JOIN Corsi
ON Corsi.CodiceCorso = Esami. CodiceCorso
```

INNER JOIN (Alternativa)

```
SELECT Studenti.Nome, Studenti.Cognome,
   Corsi.Titolo, Esami.Voto,Esami.Lode
FROM
Studenti INNER JOIN
(Corsi  INNER JOIN Esami ON
   Corsi.CodiceCorso = Esami. CodiceCorso)
ON Studenti.Matricola = Esami.Matricola
```

Per ogni studente determina quanti esami ha sostenuto
(ordina per num. decrescente di esami sostenuti)

```
SELECT Studenti.Nome, Studenti.Cognome,
   Count(*) AS Esami_sostenuti
FROM Studenti, Esami
WHERE Studenti.Matricola = Esami.Matricola
GROUP BY Studenti.Nome, Studenti.Cognome
ORDER BY Count(*) DESC;
```

Versione con INNER JOIN

```
SELECT Studenti.Nome, Studenti.Cognome,
   Count(*) AS Esami_sostenuti
FROM Studenti INNER JOIN Esami ON
   Studenti.Matricola = Esami.Matricola
GROUP BY Studenti.Nome, Studenti.Cognome
ORDER BY Count(*) DESC;
```

Join e Aggregati esempi:

1. Per ogni corso determina il numero di studenti che ne hanno sostenuto l'esame
2. Per ogni studente determina la media dei voti
3. Elenca gli studenti che hanno una media >= 27
4. Elenca gli studenti che hanno sostenuto almeno due esami

(3 e 4 possono essere fatte riutilizzando e modificando altre query gia' fatte)

1:Soluzione

```
SELECT  Corsi. Titolo,  Corsi.CodiceCorso ,
   Count(*) AS EsamiSostenuti
FROM Esami, Corsi
WHERE
Esami.CodiceCorso = Corsi.CodiceCorso
GROUP BY  Corsi.Titolo ,  Corsi.CodiceCorso;
```

2:Soluzione

```
SELECT Studenti.Nome, Studenti.Cognome,
   Avg(Esami.Voto) AS Media
FROM Studenti, Esami
WHERE Studenti.Matricola=Esami.Matricola
GROUP BY Studenti.Nome, Studenti.Cognome;
```

Seleziona gli esami sostenuti da uno studente (parametro)

```
SELECT Corsi.Titolo, Corsi.CodiceCorso AS Codice,
    Corsi.Docente
FROM Studenti, Esami, Corsi
WHERE Corsi.CodiceCorso=Esami.CodiceCorso
And Studenti.Matricola=Esami.Matricola
And Studenti.Cognome Like "*" & [inserisci studente] & "*";
```

Select annidate

- E’possibile utilizzare la tabella risultante da una query come condizione di selezione per un’altra query

Esempi:

- seleziona il libro con il prezzo massimo
- seleziona gli autori i cui autori di libri compaiono in una tabella ma non in un’altra
- Seleziona i libri (autore, titolo pubblicati da una casa editrice ma non da un’altra)

Select annidate

- SELECT INTERNA produce tabella
- SELECT ESTERNA usa tale tabella come condizione
- RISULTATO è solo la tabella prodotta dalla select esterna

Seleziona libro con il prezzo massimo

```
SELECT Autore, Titolo, Prezzo
FROM libri
WHERE Prezzo in
(SELECT Max(Prezzo)
FROM libri);
```

Libri pubblicati da entrambi Garzanti e Bompiani

```
SELECT Titolo
FROM libri
WHERE Casa_ed = ”Bompiani";
```

Produce una tabella, chiamiamola LibriBompiani

```
SELECT Autore, Titolo
FROM libri
WHERE Casa_ed = "Garzanti" and Titolo  In (select
    Titolo from LibriBompiani);
```

Esempio intersezione con parametri

Seleziona Autore e Titolo dei libri pubblicati sia da casa1 e da casa2

```
SELECT Autore, Titolo
FROM libri
WHERE Casa_ed = [casa1] and Titolo in
  (select Titolo from libri where Casa_ed =
  [casa2] );
```

Esempio differenza

Seleziona Autore e Titolo, pubblicati da Casa che pubblica , ma non da Casa che non pubblica

```
SELECT Autore, Titolo
FROM libri
WHERE Casa_ed = [Casa che pubblica] and
  Titolo not in (select Titolo from libri where
  Casa_ed = [Casa che non pubblica] );
```