Laboratorio n°8 del 12-12-2017

Ing. Dario Cogliati

SQL

- Il nome sta per *Structured Query Language*
- Le interrogazioni SQL sono dichiarative
  - l'utente specifica quale informazione è di suo interesse, ma non come estrarla dai dati
- Le interrogazioni vengono tradotte dall'ottimizzatore (query optimizer) nel linguaggio procedurale interno al DBMS
- Il programmatore si focalizza sulla leggibilità, non sull'efficienza
- È l'aspetto più qualificante delle basi di dati relazionali

Interrogazioni SQL

- Le interrogazioni SQL hanno una struttura `select-from-where`
- Sintassi:
  `select` *AttrEspr* {, *AttrEspr*}
  `from` *Tabella* {, *Tabella*}
  [`where` *Condizione*]
- Le tre parti della query sono chiamate:
  - clausola `select` / target list
  - clausola `from`
  - clausola `where`
- La query effettua il prodotto cartesiano delle tabelle nella clausola `from`, considera solo le righe che soddisfano la condizione nella clausola `where` e per ogni riga valuta le espressioni nella `select`
- Sintassi completa:
  `select` *AttrEspr* [[`as`] *Alias*] {, *AttrEspr* [[`as`] *Alias*] }
  `from` *Tabella* [[`as`] *Alias*] {, *Tabella* [[`as`] *Alias*] }
  [`where` *Condizione*]

Esempio: gestione degli esami universitari

Studente

MATR	NOME	CITTA'	CDIP
123	Carlo	Bologna	Inf
415	Alex	Torino	Inf
702	Antonio	Roma	Log

Esame

MATR	COD-CORSO	DATA	VOTO
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

Corso

COD-CORSO	TITOLO	DOCENTE
1	matematica	Barozzi
2	informatica	Meo

Proiezioni

SELECT Nome, Cdip

FROM STUDENTE

è una tabella con

- **schema :**
  gli attributi Nome e Cdip (grado <=)
- **istanza :**
  la restrizione delle tuple sugli attributi Nome e CDip (cardinalità <=)

Nome	CDip
Carlo	Inf
Alex	Inf
Antonio	Log

Proiezione

SELECT *

FROM STUDENTE

Prende tutte le colonne della tabella STUDENTE

Duplicati

- In SQL, le tabelle prodotte dalle interrogazioni possono contenere più righe identiche tra loro
- I duplicati possono essere rimossi usando la parola chiave `distinct`

```
Select distinct CDip
from Studente
```

CDip
Inf
Log

```
select CDip
from Studente
```

CDip
Inf
Inf
Log

Selezione

```
SELECT *
FROM STUDENTE
WHERE Nome='Alex'
```

È una tabella con

- schema: lo stesso schema di STUDENTE (grado =)
- istanza: le tuple di STUDENTE che soddisfano il predicato di selezione (cardinalità <=)

Matr	Nome	Città	CDip
415	Alex	Torino	Inf

Sintassi

espressione booleana di predicati semplici

operazioni booleane :

- **AND (P1 AND P2)**
- **OR (P1 OR P2)**
- **NOT (P1)**

predicati semplici :

- **TRUE, FALSE**
- **termine comparatore termine**

comparatore :

- **=, <>, <, <=, >, >=**

termine :

- **costante, attributo**
- **espressione aritmetica di costanti e attributi**

Sintassi della clausola WHERE

- Espressione booleana di predicati semplici (come in algebra)
- Estrarre gli studenti di informatica originari di Bologna:

```
select *
from Studente
where CDip = 'Inf' and Città = 'Bologna'
```

- Estrarre gli studenti originari di Bologna o di Torino:

```
select *
from Studente
where Città = 'Bologna' or Città = 'Torino'
```

  - Attenzione: estrarre gli studenti originari di Bologna e originari di Torino

Espressioni booleane

- Estrarre gli studenti originari di Roma che frequentano il corso in Informatica o in Logistica:

```
select *
from Studente
where Città = 'Roma' and
      (CDip = 'Inf' or
       CDip = 'Log')
```

- Risultato:

Matr	Nome	Città	CDip
702	Antonio	Roma	Log

Gestione dei valori nulli

- I valori nulli rappresentano tre diverse situazioni:
  - un valore non è applicabile
  - un valore è applicabile ma sconosciuto
  - non si sa se il valore è applicabile o meno
- Per fare una verifica sui valori nulli:

  Attributo **`is [ not ] null`**

```
select *
from Studente
where CDip = 'Inf' or CDip <> 'Inf'
```

è equivalente a:

```
select *
from Studente
where CDip is not null
```

Esempio di Selezione

```
SELECT *
FROM STUDENTE
WHERE (Città='Torino') OR ((Città='Roma') AND NOT (CDip='Log'))
```

MATR	NOME	CITTA'	C-DIP
123	Carlo	Bologna	Inf
415	Alex	Torino	Inf
702	Antonio	Roma	Log

Selezione e proiezione

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Alex	Torino	Inf
702	Antonio	Roma	Log

- Estrarre il nome degli studenti iscritti al diploma in informatica?

```
SELECT Nome
FROM STUDENTE
WHERE CDip='Inf'
```

NOME
Carlo
Alex

where

- Alcuni predicati aggiuntivi:
  - **between:**

```
Data between 1-1-90 and 31-12-99
```

  - **like:**

```
CDip like 'Lo%'
Targa like 'MI_777_8%'
```

Prodotto Cartesiano

R , S

è una tabella (priva di nome) con

- **schema :**

  gli attributi di R e S
  (grado(RxS)= grado(R)+grado(S))

- **istanza :**

  tutte le possibili coppie di tuple di R e S
  (card(RxS)=card(R)*card(S))

Esempio

R(A,B)

A	B
a	1
b	3

S(C,D)

C	D
c	1
d	5
a	2

```
Select *
FROM R,S
```

R,S (A,B,C,D)

A	B	C	D
a	1	c	1
a	1	d	5
a	1	a	2
b	3	c	1
b	3	d	5
b	3	a	2

Prodotto cartesiano con condizione di JOIN

```
SELECT *
FROM STUDENTE , ESAME
WHERE STUDENTE.Matr=ESAME.Matr
```

JOIN

```
FROM STUDENTE JOIN ESAME
                ON STUDENTE.Matr=ESAME.Matr
```

è equivalente alla seguente espressione (operatore derivato):

```
FROM STUDENTE , ESAME
WHERE STUDENTE.Matr=ESAME.Matr
```

attributi omonimi sono resi non ambigui usando la notazione “puntata”:

ESAME.Matr

STUDENTE.Matr

JOIN

FROM STUDENTE JOIN ESAME

ON STUDENTE.Matr=ESAME.Matr

produce una tabella con

- **schema: la concatenazione degli schemi di STUDENTE e ESAME**
- **istanza: le tuple ottenute concatenando quelle tuple di STUDENTE e di ESAME che soddisfano il predicato**

STUDENTE. Matr	Nome	Città	CDip	ESAME. Matr	Cod-Corso	Data	Voto
123	Carlo	Bologna	Inf	123	1	7-9-97	30
123	Carlo	Bologna	Inf	123	2	8-1-98	28
702	Antonio	Roma	Log	702	2	7-9-97	20

Sintassi del predicato di JOIN

espressione congiuntiva di predicati semplici:

ATTR1 comp ATTR2

ove ATTR1 appartiene a TAB1

ATTR2 appartiene a TAB2

comp: =, <>, <, <=, >, >=

Interrogazione semplice con 2 tabelle

Estrarre il nome degli studenti di "Logistica" che hanno preso almeno un 30

```
select distinct Nome
from Studente, Esame
where Studente.Matr = Esame.Matr
  and CDip = 'Log' and Voto = 30
```

NOME
Carlo

Interrogazione semplice 2 tabelle

Estrarre il nome degli studenti di "Logistica" che hanno preso sempre 30

```
select distinct Nome
from Studente, Esame
where Studente.Matr = Esame.Matr
   and CDip = 'Log' and Voto = 30
```

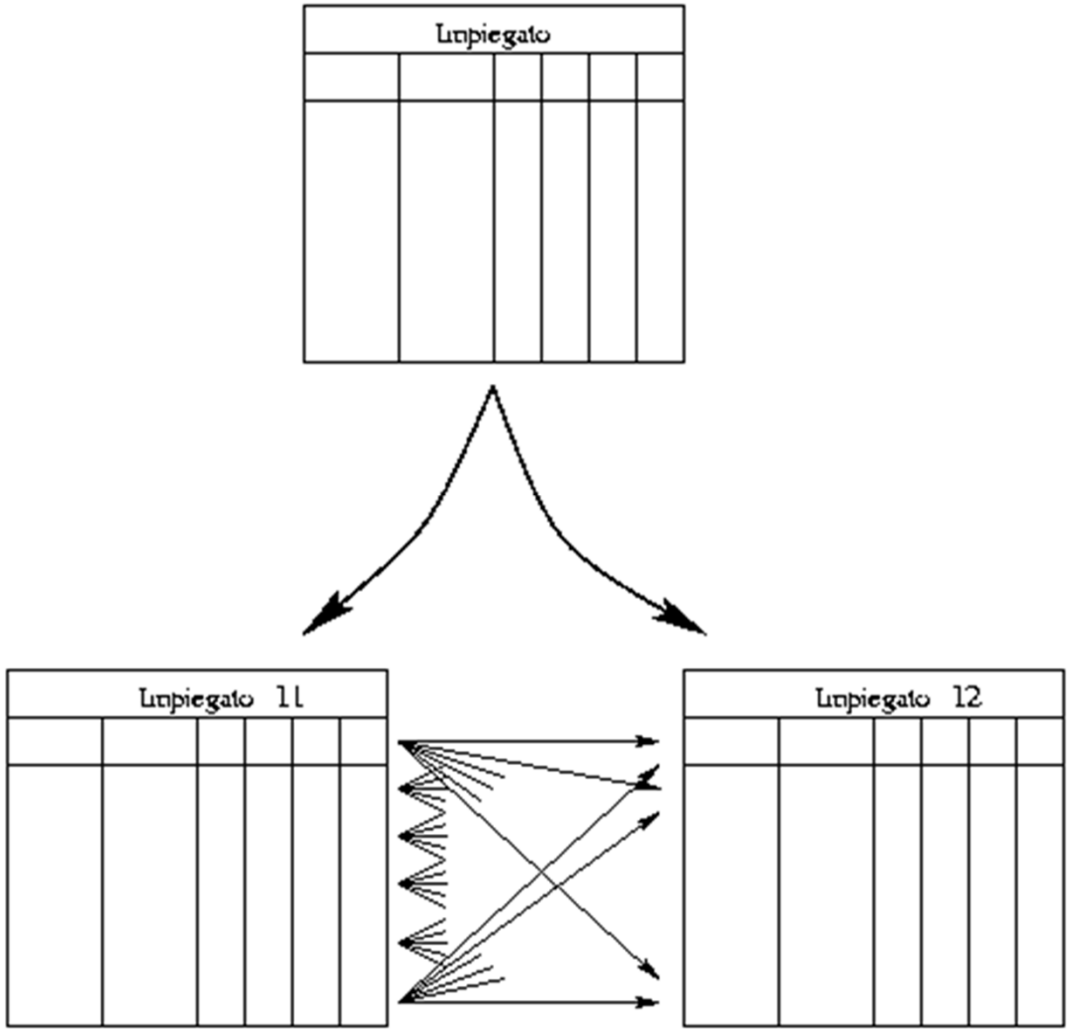
Join in SQL

- SQL ha una sintassi per i join, li rappresenta esplicitamente nella clausola `from`:

`select` ***AttrEspr*** {, ***AttrEspr***}
`from` ***Tabella*** { [***TipoJoin***] **`join`** ***Tabella*** **`on`** ***Condizioni*** }
[**`where`** ***AltreCondizioni***]

- *TipoJoin* può essere `inner`, `right` [`outer`], `left` [`outer`] oppure `full` [`outer`], consentendo la rappresentazione dei join esterni

Variabili in SQL



Interrogazione semplice con variabili relazionali

Chi sono i dipendenti di Giorgio?

Impiegato

Matr	Nome	DataAss	Salario	MatrMgr
1	Piero	1-1-95	3 M	2
2	Giorgio	1-1-97	2,5 M	null
3	Giovanni	1-7-96	2 M	2

Chi sono i dipendenti di Giorgio?

```
select X.Nome,X.MatrMgr,Y.Matr,Y.Nome
from Impiegato as X, Impiegato as Y
where X.MatrMgr = Y.Matr
   and Y.Nome = 'Giorgio'
```

X.Nome	X.MatrMgr	Y.Matr	Y.Nome
Piero	2	2	Giorgio
Giovanni	2	2	Giorgio

Ordinamento

- La clausola `order by`, che compare in coda all'interrogazione, ordina le righe del risultato
- Sintassi:

  `order by` *AttributoOrdinamento* [`asc` | `desc`]
  {, *AttributoOrdinamento* [`asc` | `desc`] }

- Le condizioni di ordinamento vengono valutate in ordine
  - a pari valore del primo attributo, si considera l'ordinamento sul secondo, e così via

Es. gestione Ordini

CODCLI	INDIRIZZO	P-IVA

Ordine

CODORD	CODCLI	DATA	IMPORTO

Dettaglio

CODORD	CODPROD	QTA

Prodotto

CODPROD	NOME	PREZZO

Istanza di ordine

Ordine

CODORD	CODCLI	DATA	IMPORTO
1	3	1-6-97	50.000.000
2	4	3-8-97	8.000.000
3	3	1-9-97	5.500.000
4	1	1-7-97	12.000.000
5	1	1-8-97	1.500.000
6	3	3-9-97	27.000.000

Query con ordinamento

```
select  *
from Ordine
where Importo > 100.000
order by Data
```

CODORD	CODCLI	DATA	IMPORTO
1	3	1-6-97	50.000.000
4	1	1-7-97	12.000.000
5	1	1-8-97	1.500.000
2	4	3-8-97	8.000.000
3	3	1-9-97	1.500.000
6	3	3-9-97	5.500.000

Funzioni Aggregate

- Il risultato di una query con funzioni aggregate dipende dalla valutazione del contenuto di un insieme di righe
- Cinque operatori aggregati:
  - `count` cardinalità
  - `sum` sommatoria
  - `max` massimo
  - `min` minimo
  - `avg` media

COUNT

- `count` restituisce il numero di righe o valori distinti; sintassi:

  `count(<*|[distinct|all]`*ListaAttributi*`>)`

- Estrarre il numero di ordini:

```
select count(*)
from Ordine
```

- Estrarre il numero di valori distinti dell'attributo CodCli per tutte le righe di Ordine:

```
select count(distinct CodCli)
from Ordine
```

- Estrarre il numero di righe di Ordine che posseggono un valore non nullo per l'attributo CodCli:

```
select count(all CodCli)
from Ordine
```

SUM, MIN , MAX, AVG

- Sintassi:
  < `sum` | `max` | `min` | `avg` > ([`distinct` | `all`] *AttrEspr*)

- L'opzione `distinct` considera una sola volta ciascun valore
  - utile solo per le funzioni `sum` e `avg`

- L'opzione `all` considera tutti i valori diversi da *null*

Query con massimo

- **Estrarre l'importo massimo degli ordini**

```
select max(Importo) as MaxImp
from Ordine
```

MaxImp
50.000.000

Query con sommatoria

- **Estrarre la somma degli importi degli ordini relativi al cliente numero 1**

```
select sum(Importo) as SommaImp
from Ordine
where CodCliente = 1
```

SommaImp
13.500.000

Funzioni aggregate con Join

- Estrarre l'ordine massimo tra quelli contenenti il prodotto con codice 'ABC' :

```
select max(Importo) as MaxImportoABC
from Ordine, Dettaglio
where Ordine.CodOrd = Dettaglio.CodOrd and
      CodProd = 'ABC'
```

Query con raggruppamento

- Nelle interrogazioni si possono applicare gli operatori aggregati a sottoinsiemi di righe
- Si aggiungono le clausole
  - **`group by`** (raggruppamento)
  - **`having`** (selezione dei gruppi)

```
select …
from …
where …
group by …
having …
```

Query con raggruppamento

- Estrarre la somma degli importi degli ordini successivi al 10-6-97 per quei clienti che hanno emesso almeno 2 ordini

```
select CodCli, sum(Importo)
from Ordine
where Data > 10-6-97
group by CodCli
having count(*) >= 2
```

Passo 1 : Valutazione where

CodOrd	CodCli	Data	Importo
2	4	3-8-97	8.000.000
3	3	1-9-97	5.500.000
4	1	1-7-97	12.000.000
5	1	1-8-97	1.500.000
6	3	3-9-97	27.000.000

Passo 2 : Raggruppamento

- si valuta la clausola **`group by`**

CodOrd	CodCli	Data	Importo
4	1	1-7-97	12.000.000
5	1	1-8-97	1.500.000
3	3	1-9-97	1.500.000
6	3	3-9-97	5.500.000
2	4	3-8-97	8.000.000

Passo 3 : calcolo degli aggregati

- si calcolano **`sum(Importo)`** e **`count(*)`** per ciascun gruppo

CodCli	sum(Importo)	count(*)
1	13.500.000	2
3	32.500.000	2
4	5.000.000	1

Passo 4 : estrazione dei gruppi

- si valuta il predicato `count(*) >= 2`

CodCli	sum (Importo)	count(*)
1	13.500.000	2
3	32.500.000	2
~~4~~	~~5.000.000~~	~~1~~

Passo 5: produzione del risultato

CodCli	sum(Importo)
1	13.500.000
3	32.500.000

Query con group by

- **Query scorretta:**

```
select Importo
from Ordine
group by CodCli
```

- **Query scorretta:**

```
select O.CodCli, count(*), C.Città
from Ordine O join Cliente C
     on (O.CodCli = C.CodCli)
group by O.CodCli
```

- **Query corretta:**

```
select O.CodCli, count(*), C.Città
from Ordine O join Cliente C
     on (O.CodCli = C.CodCli)
group by O.CodCli, C.Città
```

Where o having???

- Soltanto i predicati che richiedono la valutazione di funzioni aggregate dovrebbero comparire nell'argomento della clausola `having`

- Estrarre i dipartimenti in cui lo stipendio medio degli impiegati che lavorano nell'ufficio 20 è maggiore di 25:

```
select Dipart
from Impiegato
where Ufficio = '20'
group by Dipart
having avg(Stipendio) > 25
```

Query nidificate

- Nella clausola **`where`** e nella clausola **`having`** possono comparire predicati che:
  - confrontano un attributo (o un'espressione sugli attributi) con il risultato di una query SQL; sintassi:

    AttrExpr Operator < **`any`** | **`all`** > *SelectSQL*

    - **`any`**: il predicato è vero se almeno una riga restituita dalla query *SelectSQL* soddisfa il confronto
    - **`all`**: il predicato è vero se tutte le righe restituite dalla query *SelectSQL* soddisfano il confronto
    - *Operator*: uno qualsiasi tra =, <>, <, <=, >, >=
- La query che appare nella clausola **`where`** e nella clausola **`having`** è chiamata query nidificata
- Nelle query nidificate posso usare variabili definite esternamente

Uso di ANY e ALL

```
select CodOrd
from Ordine
where Importo > any
        select Importo
        from Ordine
```

```
select CodOrd
from Ordine
where Importo >= all
        select Importo
        from Ordine
```

COD-ORD	IMPORTO
1	50
2	300
3	90

ANY	ALL
F	F
V	V
V	F

Query nidificate con ANY

- Estrarre gli ordini di prodotti con un prezzo superiore a 100

```
select distinct CodOrd
from Dettaglio
where CodProd = any(select CodProd
                    from Prodotto
                    where Prezzo > 100)
```

- Equivalente a (senza query nidificata)

```
select distinct CodOrd
from Dettaglio D, Prodotto P
where D.CodProd = P.CodProd
    and Prezzo > 100
```

Query nidificate con ANY

- Estrarre i prodotti ordinati assieme al prodotto avente codice 'ABC'
  - con una query nidificata:

```
select distinct CodProd
from Dettaglio
where CodProd<>'ABC' and CodOrd = any
          (select CodOrd
           from Dettaglio
           where CodProd = 'ABC')
```

  - senza query nidificata, a meno di duplicati:

```
select distinct D1.CodProd
from Dettaglio D1, Dettaglio D2
where D1.CodOrd = D2.CodOrd and
D1.CodProd<>'ABC' and D2.CodProd='ABC'
```

Negazione con query nidificate

- Estrarre gli ordini che non contengono il prodotto 'ABC':

```
select distinct CodOrd
from Ordine
where CodOrd <> all (select CodOrd
                     from Dettaglio
                     where CodProd = 'ABC')
```

Query nidificate

- *AttrExpr Operator* < **`in`** | **`not in`** > *SelectSQL*
  - **`in`**: il predicato è vero se almeno una riga restituita dalla query *SelectSQL* e' presente nell'espressione
  - **`not in`**: il predicato è vero se nessuna riga restituita query e' presente nell'espressione

Query IN e NOT IN

- L'operatore `in` è equivalente a `= any`

```
select CodProd
from Dettaglio
where CodOrd in
          (select CodOrd
           from Dettaglio
           where CodProd = 'ABC')
```

- L'operatore `not in` è equivalente a `<> all`

```
select distinct CodOrd
from Ordine
where CodOrd not in (select CodOrd
                     from Dettaglio
                     where CodProd = 'ABC')
```

Max con query nidificate

- Gli operatori aggregati `max` (e `min`) possono essere espressi tramite query nidificate
- Estrarre l'ordine con il massimo importo
  - Con una query nidificata, usando `max`:

```
select CodOrd
from Ordine
where Importo in (select max(Importo)
                  from Ordine)
```

  - con una query nidificata, usando all:

```
select CodOrd
from Ordine
where Importo >= all (select Importo
                      from Ordine)
```

ESERCIZIO : Aeroporti

AEROPORTO (Citta, Nazione, NumPiste)

VOLO (IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)

AEREO (TipoAereo, NumPasseggeri, QtaMerci)

QUERY 1

Trovare le città con un aeroporto di cui non è noto il numero di piste

```
SELECT Città
FROM Aeroporto
WHERE NumPiste IS NULL
```

Query 2

Di ogni volo misto (merci e passeggeri) estrarre il codice e i dati relativi al trasporto

```
SELECT IdVolo, NumPasseggeri, QtaMerci
FROM VOLO AS V, AEREO AS A
WHERE V.TipoAereo = A.TipoAereo and
                    NumPasseggeri > 0 and QtaMerci > 0
```

Query 3

Le nazioni di partenza e arrivo del volo AZ274

```
SELECT A1.Nazione, A2.Nazione
FROM (AEROPORTO A1 JOIN VOLO
        ON A1.Citta=CittaArr)
      JOIN AEROPORTO A2
        ON CittaPar=A2.Citta
WHERE IdVolo= 'AZ274'
```

Query 4

Trovare l'aeroporto italiano con il maggior numero di piste (soluzione corretta)

Ad esempio si può usare una query annidata

```
SELECT Citta, NumPiste
FROM       AEROPORTO
WHERE    Nazione='Italia' and
                NumPiste = (SELECT max(numPiste)
                                FROM AEROPORTO
                                WHERE Nazione='Italia' )
```

Query 5

Trovare l'aeroporto italiano con il maggior numero di piste

```
SELECT   Citta, max(NumPiste)
FROM          AEROPORTO
WHERE    Nazione = 'Italia'
GROUP BY Citta
```