SQL e algebra relazionale

Il linguaggio SQL consente una maggiore espressività dell'algebra relazionale grazie a costrutti che permettono:

- di definire *join più sofisticati*
- di *ordinare le righe* dei risultati delle interrogazioni in base a criteri definiti dall'utente
- di applicare *operatori aggregati* per ottenere conteggi, minimi, massimi, medie e somme.
- di *raggruppare le righe* della tabella-risultato prima di applicare gli operatori aggregati

Inoltre è possibile:

- applicare operatori insiemistici
- nidificare le interrogazioni

Join esplicito

In SQL-2 è possibile dichiarare esplicitamente la presenza di un join in una operazione di selezione

SQL

```
select I.Nome, Cognome, Citta
from Impiegato as I
        inner join
     Dipartimento as D
        on Dipartimento = D.Nome
```

L'`inner join` coincide con il theta-join. Altri tipi di join consentiti sono:
`right` [`outer`] `join`: il risultato ha tutte le tuple della tabella di destra, eventualmente completate con valori nulli
`left` [`outer`] `join`: lo stesso per la tabella di sinistra
`full` [`outer`] `join`: tutte le tuple si riflettono nel risultato

Esempio di join esplicito

Date le relazioni:

Guidatore(Nome, Cognome, NroPatente)
Automobile(Targa, Marca, Modello, NroPatente)

Domanda

Trovare i guidatori con le automobili associate mantenendo tutti i guidatori della tabella

SQL

```
select *
from Guidatore left join
        Automobileon
       (Guidatore.NroPatente =
        Automobile.NroPatente)
```

Costrutto `order by`

Sintassi

"`order by`" **Attributo**
[< "`asc`" | "`desc`" >]
{ "`,`" **Attributo**
[< "`asc`" | "`desc`" >] }

Domanda

Ordinare per stipendio e, secondariamente, per cognome il personale amministrativo

SQL

```
select Cognome, Nome,
       Stipendio
from Impiegato
where Dipartimento like 'Amm%'
order by Stipendio desc,
         Cognome
```

Risultato

Cognome	Nome	Stipendio
Rossi	Mario	45
Borroni	Paola	40
Verdi	Giuseppe	40

Costrutto **count**

E' un operatore aggregato che permette di "contare" le tuple del risultato di una interrogazione

Sintassi 1

"`count (*)`"

Domanda

Determinare il numero degli impiegati del dipartimento 'Produzione'

SQL

```
select count(*)
from Impiegato
where Dipartimento =
        'Produzione'
```

Risultato

2

Costrutto **count** (2)

Sintassi 2

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

Domanda 1

Trovare il numero di valori diversi per gli stipendi degli impiegati

SQL 1

```
select count (distinct Stipendio)
from Impiegato
```

Domanda 2

Trovare il numero delle persone che hanno sia il cognome che il nome specificato

SQL 2

```
select count (all Nome,Cognome)
from Impiegato
```

Altri operatori aggregati

Ammettono tutti una singola sintassi:

Sintassi

Operatore “(” [< “distinct” | “all” >] **AttrEspr** “)”

Dove **Operatore** può essere: `sum`, `max`, `min` o `avg`, con ovvio significato

Esempio 1

```
select sum(Stipendio)
from Impiegato
where Dipartimento='Produzione'
```

Esempio 2

```
select  max(Stipendio),
        avg(Stipendio),
        min(Stipendio)
from Impiegato
```

Interrogazione errata

Si consideri la seguente interrogazione:

Interrogazione errata

```
select Cognome, Nome,
       max(Stipendio)
from Impiegato, Dipartimento
where Impiegato.Dipartimento =
       Dipartimento.Nome and
       citta = 'Milano'
```

Cosa sarebbe successo se al posto di `max` ci fosse stato `sum`?

Non si possono mischiare nella stessa *target list* operatori aggregati ed espressioni a livello di riga, come per esempio il nome di un attributo (unica eccezione: il costrutto `group by`)

Costrutto `group by`

Ha lo scopo di applicare gli operatori aggregati su sottoinsiemi di righe

Esempio

```
select Dipartimento,
        sum(Stipendio)
                as SommaStipendi
from Impiegato
group by Dipartimento
```

Risultato

Dipartimento	SommaStipendi
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

`group by`: semantica

Innanzi tutto viene fatta una proiezione sugli attributi interessati dagli operatori aggregati e dal costrutto `group by`

Risultato intermedio

Dipartimento	Stipendio
Amministrazione	45
Produzione	36
Amministrazione	40
Distribuzione	45
Direzione	80
Direzione	73
Amministrazione	40
Produzione	46

E' necessario che eventuali attributi della target list non oggetto di aggregazione compaiano nella `group by`

`group by`: semantica (2)

Quindi si raggruppano le tuple della relazione in base al valore degli attributi della `group by`

Risultato intermedio

Dipartimento	Stipendio
Amministrazione	45
Amministrazione	40
Amministrazione	40
Produzione	36
Produzione	46
Distribuzione	45
Direzione	80
Direzione	73

`group by`: semantica (3)

Infine si applica l'operatore aggregato su ogni gruppo e si rinominano le colonne se richiesto

Risultato finale

Dipartimento	SommaStipendi
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

L'operatore aggregato produce un solo valore

Interrogazione errata

Interrogazione errata

```
select Dipartimento,
       count(*), Citta
from Impiegato, Dipartimento
where Impiegato.Dipartimento =
       Dipartimento.Nome
group by Dipartimento
```

L'attributo Citta *compare nella target list ma non è oggetto di aggregazione e non compare nella* `group by`

Interrogazione giusta

```
select Dipartimento,
       count(*), Citta
from Impiegato, Dipartimento
where Impiegato.Dipartimento =
       Dipartimento.Nome
group by Dipartimento, Citta
```

`group by` con predicati

Esempio

```
select Dipartimento,
        sum(Stipendio)
                as SommaStipendi
from Impiegato
group by Dipartimento
having sum(Stipendio) > 100
```

Risultato

Dipartimento	SommaStipendi
Amministrazione	125
Direzione	153

Generalmente, per chiarezza, solo i predicati contenenti operatori aggregati si mettono nella clausola `having`*, gli altri si mettono nella clausola* `where`

Esempio di **having**

Domanda

Trovare i dipartimenti per cui la media degli stipendi degli impiegati che lavorano nell'ufficio 20 è superiore a 25 milioni

SQL

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

Risultato

Dipartimento
Produzione
Amministrazione

Sintassi della select

Considerando tutti i costrutti introdotti finora l'istruzione `select` ha la seguente sintassi:

Sintassi

"`select`" **ListaAttributiOEspr**
"`from`" **ListaTabelle**
["`where`" **CondizioniSemplici**]
["`group by`"
ListaAttributiRaggruppamento
["`having`" **CondAggregate**]]
["`order by`"
ListaAttributiOrdinamento]

Operatori insiemistici

Sintassi

IstruzioneSelect { < “`union`” |
“`intersect`” |
“`exept`” >
[“`all`”]
IstruzioneSelect }

`union`, `intersect` e `exept` corrispondono rispettivamente ad unione intersezione e differenza tra gli insiemi ottenuti con le istruzioni select

Gli operatori insiemistici eliminano di default di duplicati. Se si vogliono conservare i duplicati occorre inserire la particella `all`

Operatori insiemistici: esempio

Domanda

Tutte le stringhe che compaiono come nomi o cognomi degli impiegati

SQL

```
select Nome
from Impiegato
        union
select Cognome
from Impiegato
```

Operatori insiemistici: altro esempio

Domanda

Tutte le stringhe, anche ripetute, che compaiono come nomi o cognomi degli impiegati dei dipartimenti diversi dall'amministrazione

SQL

```
select Nome
from Impiegato
where Dipartimento <>
        'Amministrazione'
        union all
select Cognome
from Impiegato
where Dipartimento <>
        'Amministrazione'
```

Interrogazioni nidificate

Domanda

Trovare gli impiegati che lavorano in dipartimenti situati a Firenze

SQL

```
select *
from Impiegato
where Dipartimento =
  any (select Nome
        from Dipartimento
        where Citta = 'Firenze')
```

Si possono utilizzare gli operatori di confronto (=,<>,<,>,<=,>=) con le particelle `any` ed `all` (il valore di default è `any`)

in particolare:

`= any` si può sostituire con `in`

`<> all` si può sostituire con `not in`

Formulazioni alternative

Domanda

Trovare gli impiegati che hanno lo stesso nome di un impiegato del dipartimento produzione

SQL

```
select I1.Nome
from Impiegato as I1,
        Impiegato as I2
where I1.Nome = I2.Nome and
          I2.Dipartimento =
                    'Produzione'
```

SQL

```
select Nome
from Impiegato
where Nome = any (
  select Nome
  from Impiegato
  where Dipartimento =
                    'Produzione')
```

Formulazioni alternative (2)

Domanda

Trovare il nome dei dipartimenti in cui non lavorano persone di cognome ‘Rossi’

SQL

```
select Nome
from Dipartimento
            except
select Dipartimento
from Impiegato
where Cognome = ‘Rossi’
```

SQL

```
select Nome
from Dipartimento
where Nome <> all (
  select Dipartimento
  from Impiegato
  where Cognome = ‘Rossi’)
```

Formulazioni alternative (3)

Domanda

Trovare il dipartimento in cui lavora l'impiegato che percepisce lo stipendio maggiore

SQL

```
select Dipartimento
from Impiegato
where Stipendio = (
  select max (Stipendio)
  from Impiegato
)
```

SQL

```
select Dipartimento
from Impiegato
where Stipendio >= all (
  select Stipendio
  from Impiegato
)
```

Operatore logico **exists**

Data la relazione:
Persona(CF, Nome, Cognome, Citta)

Domanda

Trovare le persone che hanno degli omonimi

SQL

```
select *
from Persona as P
where exists (
  select *
  from Persona P1
  where P1.Nome = P.Nome and
   P1.Cognome = P.Cognome and
   P1.CF <> P.CF
)
```

Inserimento di righe

Sintassi

“insert into” **NomeTabella**
[“(” **ListaAttributi** “)”]
< “values (” **ListaValori** “)” |
“(” **IstruzioneSelect** “)” >

Esempio

```
insert into
  Dipartimento (Nome, Citta)
values
  (‘Produzione’, ‘Torino’)
```

Esempio

```
insert into
  ProdottiMilanesi (
  select Codice, Descrizione
  from Prodotto
  where LuogoProd = ‘Milano’)
```

Cancellazione di righe

Sintassi

“delete from” **NomeTabella**
[“where” **Condizione**]

Es.

```
delete from Dipartimento
where Nome = ‘Produzione’
```

Esempio

```
delete from Dipartimento
where Nome not in (
  select Dipartimento
  from Impiegato
)
```

Es.

```
delete from Dipartimento
```

Cancella tutte le righe (ma lo schema della relazione rimane)

Modifica di righe

Sintassi

“update” **NomeTabella**
“set” **Attributo** “=”
< **Espressione** | **Select** |
“null” | “default” >
{ “,” **Attributo** “=”
< **Espressione** | **Select** |
“null” | “default” > }
[“where” **Condizione**]

Esempio

```
update Impiegato
set Stipendio = Stipendio * 1.1
where Dipartimento =
        ‘Amministrazione’
```

Vincoli espressi con **check**

Sintassi

“check (” **Condizione** “)”

Esempio

```
create table Impiegato (
Matricola char(6) primary key,
Cognome char(20),
Nome char(20),
Dipartimento char(20),
Superiore char(6),

check ( Dipartimento = (
  select Dipartimento
  from Impiegato as I
  where I.Matricola = Superiore))
)
```

Asserzioni

Le asserzioni sono vincoli definiti a livello di schema tramite il costrutto `check`

Sintassi

“`create assertion`”
NomeAsserzione
“`check (`” **Condizione** “`)`”

Esempio

```
create assertion
AlmenoUnImpiegato
check ( 1 <= (
  select count(*)
  from Impiegati)
)
```

Vincola la relazione Impiegati ad avere almeno un elemento

Comando `alter`

`alter domain`

“alter domain” **NomeDominio**
< “set default” **ValoreDef** |
“drop default” |
“add constraint” **DefVincolo** |
“drop constraint”
NomeVincolo >

`alter table`

“alter table” **NomeTabella**
< “alter column” **NomeAttr**
< “set default” **ValoreDef** |
“drop default” > |
“add constraint” **DefVincolo** |
“drop constraint”
NomeVincolo |
“add column” **DefAttr** |
“drop column” **NomeAttr** >

Comando `drop`

Sintassi

```
"drop" < "schema" |
         "domain" |
         "table" |
         "assertion" > Nome
[ < "restrict" | "cascade" > ]
```

L'opzione `restrict` è quella di default e impone che il comando non venga eseguito in presenza di oggetti *non vuoti*

Con l'opzione `cascade`:

- Quando si rimuove un dominio, tutti gli attributi che usano quel dominio rimangono associati alla stessa definizione di tipo
- Quando si rimuove una tabella, tutti i riferimenti a quella tabella vengono rimossi

Gestione di indici

Non fanno parte dello standard SQL

Sintassi

"create" ["unique"] "index"
NomeIndice "on"
NomeTabella
"(" **ListaAttributi** ")"

Sin.

"drop index" **NomeIndice**

Esempio

```
create index
NomeCittaIdx
on Impiegato(Cognome,Citta)
```

L'ordine degli attributi è significativo

Es.

```
drop index NomeCittaIdx
```

Esercizio

Dato il seguente schema:

Aeroporto(Citta, Nazione, NunPiste)

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

Aereo(TipoAereo, NumPasseggeri,QtaMerci)

Scrivere le interrogazioni SQL che permettono di determinare:

1) Le città con un aeroporto di cui non è noto il numero di piste
2) Le nazioni da cui parte e arriva il volo con codice AZ274
3) I tipi di aereo usati nei voli che partono da Torino

Esercizio (continua)

4) I tipi di aereo e il corrispondente numero di passeggeri usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile visualizzare solamente il tipo
5) Le città da cui partono voli internazionali
6) Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente
7) Il numero di voli internazionali che partono il giovedì da Napoli
8) Il numero di voli internazionali che partono ogni settimana da città italiane (farlo in due modi: facendo comparire o meno nel risultato gli aeroporti senza voli internazionali)
9) Le città francesi da cui partono più di venti voli alla settimana diretti in Italia

Esercizio (fine)

10) Gli aeroporti italiani che hanno solo voli interni. Rappresentare questa interrogazione in quattro modi:
    - con operatori insiemistici
    - con un'interrogazione nidificata con operatore not in
    - con un'interogazione nidificata con operatore not exists
    - con l'outer join e l'operatore di conteggio.

    Esprimere l'interogazione anche in algebra relazionale.
11) Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri
12) Il massimo numero di passeggeri che possono arrivare in un aeroporto italiano dalla Francia di giovedì (se vi sono più voli occorre sommare i passeggeri)