Laboratorio di Basi di Dati

Turni T3 e T4

a.a. 2018/2019

Ruggero Pensa - Fabiana Vernero

In questa lezione

- Interrogazione dei dati - SQL come DML
- Prime semplici query

Interrogazione dei dati
SQL come DML

Caratteristiche di SQL come linguaggio di interrogazione

- SQL è un linguaggio dichiarativo:
  - L'**utente** definisce ***cosa*** vuole ottenere, il **DBMS** determina ***come*** ottenerlo.
    - L'algebra relazionale, invece, è procedurale: l'interrogazione specifica i passi da compiere per estrarre le informazioni.
  - Una query SQL viene analizzata dal *query optimizer* del DBMS e trasformata in modo da essere eseguita in modo molto efficiente con un linguaggio procedurale interno.
    - Quando si scrivono query SQL, quindi, si possono in gran parte ignorare gli aspetti di efficienza.

Sintassi

- La struttura essenziale di una query SQL è la seguente (introdurremo le variazioni di volta in volta):

select *ListaAttributi* ← target list (o clausola "select")
from *ListaTabelle* ← clausola "from"
[**where** *Condizione*]; ← clausola "where"

Significato dell'interrogazione

- L'interrogazione restituisce una tabella:
  - Considerando il prodotto cartesiano delle tabelle della clausola "*from*".
  - Selezionando le righe che soddisfano la condizione della clausola "*where*" (opzionale).
  - Dando in output i valori degli attributi elencati nella target list ("*select*").

Database di esempio

S

SNum	SName	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

S=Supplier (fornitore)
P=Parts (parti)
QTY=quantity

P

PNum	PName	Color	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SP

SNum	PNum	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

nut=dado, bolt=bullone, screw=vite, cam=camma, cog=ruota dentata

Forme di una query SQL

- Forma più semplice:
  - Mostra tutte le informazioni di tutti i fornitori.

SQL	select * from S;
Calcolo su tuple con dichiarazioni di range	{ s.* | s(S) }
Algebra relazionale	S

Forme di una query SQL

- Proiezione:
  - Mostra nome e città di tutti i fornitori.

SQL	select SName, City from S;
Calcolo su tuple con dichiarazioni di range	{ s.(SName, City) | s(S) }
Algebra relazionale	$\pi_{SName,City}(S)$

Forme di una query SQL

- Selezione:
  - Mostra nome e città di tutti i fornitori che hanno status maggiore di 20

SQL	select SName, City from S where Status > 20;
Calcolo su tuple con dichiarazioni di range	{ s.(SName, City) | s(S) | s.Status > 20 }
Algebra relazionale	$\pi_{SName,City}(\sigma_{Status>20}(\pi_{SName,City,Status}(S)))$ *oppure* $\pi_{SName,City}(\sigma_{Status>20}(S))$

Prime semplici query

Espressioni nella clausola select

- Nella clausola `select` possono comparire espressioni sul valore degli attributi di ciascuna riga selezionata.
  - Esempio: estrarre il peso in grammi delle viti.

```
select PName, Color, Weight*1000
from P
where PName='Screw';
```

PName	Color	Weight*1000
Screw	Blue	17000
Screw	Red	14000

Espressioni nella clausola where - 1

- La clausola `where` ammette come argomento un'espressione booleana costruita combinando predicati semplici con gli operatori **`and`**, **`or`** e **`not`**.
- Gli operatori ammessi sono =, <>, <, >, <=, >=
  - Possono essere usati anche per le stringhe
- Inoltre, con le stringhe si può usare l'operatore **`like`**.

Espressioni nella clausola where - 2

- Elencare tutti i fornitori di Parigi o di Atene con Status di almeno 20.

```
select SName, City
from S
where Status >= 20 and (City = 'Paris' or City = 'Athens');
```

SName	City
Blake	Paris
Adams	Athens

- Per non sbagliare con le precedenze, usare sempre le parentesi.

Espressioni nella clausola where - 3

- L'operatore **`like`** permette di confrontare una stringa con un'altra stringa in cui possono comparire i seguenti caratteri speciali:

  _ (trattino basso) = un carattere qualsiasi.

  % (percentuale) = una sequenza di lunghezza arbitraria (eventualmente anche zero) di caratteri arbitrari.

- Esempio: l'espressione '*ab%ba_*' è soddisfatta sia da '*abcdefghilmbac*' che da '*abbar*', ma non da '*abba*'.
- Solitamente il confronto è case sensitive.

Espressioni nella clausola where - 4

- Esempio 1: elencare tutte le parti che iniziano per "C":

```
select * from P where PName like 'C%';
```

PNum	PName	Color	Weight	City
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

- Esempio 2: elencare tutti i fornitori il cui nome contiene la lettera "a" seguita, in qualsiasi posizione, dalla lettera "k":

```
select * from S
where SName like '%a%k%';
```

SNum	SName	Status	City
S3	Blake	30	Paris
S4	Clark	20	London

Espressioni nella clausola where - 5

- Esempio 3: elencare tutti i fornitori il cui nome contiene la lettera “a” seguita, dopo un carattere qualsiasi, dalla lettera “k”:

```
select * from S
where SName like '%a_k%';
```

SNum	SName	Status	City
S4	Clark	20	London

Join

- Per formulare interrogazioni che coinvolgono più tabelle occorre effettuare un ***join***, cioè "congiungere" le tabelle.
  - È un'operazione fondamentale: di norma in un database relazionale le informazioni sono registrate in più tabelle che occorre "congiungere".
- La congiunzione avviene sui valori in comune tra le tabelle.

Join in SQL

- In SQL per effettuare un join occorre(*):
  1. Elencare le tabelle di interesse nella clausola "from".
  2. Definire nella clausola "where" le condizioni necessarie per mettere in relazione fra loro gli attributi di interesse.

(*)In alternativa, è possibile usare la parola riservata "join", che vedremo dopo.

Forme di una query SQL

- Selezione da più tabelle:
  - Mostra nome e città dei fornitori con le relative quantità delle parti fornite e nome e colore delle parti.

SQL	select SName, S.City, Qty, PName, Color from S, SP, P where S.SNum=SP.SNum and P.PNum=SP.PNum;
Calcolo su tuple con dichiarazioni di range	{ s.(SName, City), sp.(Qty), p.(PName, Color) | s(S), p(P), sp(SP) | s.SNum=sp.SNum ∧ p.PNum=sp.PNum }
Algebra relazionale	$\pi_{SName,S.City,Qty,PName,Color}(S \bowtie SP \bowtie P)$ (*) Nota: il natural join nella query in algebra relazionale è veramente corretto?

SName	S.City	Qty	PName	Color
Smith	London	300	Nut	Red
…	…	…	…	…
Clark	London	400	Cam	Blue

Uso di alias - 1

- Per migliorare la leggibilità (o evitare ambiguità), è possibile usare degli *alias*, cioè associare , in una query, un altro nome a una tabella:

```
select T1.Attributo1, …, TM.AttributoN
from Tabella1 T1, …, TabellaM TM
[where Condizione]
```

- In modo simile è possibile rinominare gli attributi nella target list:

```
select Attributo1 NuovoNome1, …, AttributoN
NuovoNomeN
from Tabella1, …, TabellaM
[where Condizione]
```

Uso di alias - 2

- Esempio: riscrittura della query precedente

```
select SName NomeFornitore, Fornitore.City
CittàFornitore, Qty QuantitàFornita, PName NomeParte,
Color ColoreParte
from S Fornitore, SP Fornitura, P Parte
where Fornitore.Snum = Fornitura.Snum      and
Parte.Pnum = Fornitura.PNum;
```

NomeFornitore	CittàFornitore	QuantitàFornita	NomeParte	ColoreParte
Smith	London	300	Nut	Red
…	…	…	…	…
Clark	London	400	Cam	Blue

Join in SQL

- **Attenzione:** se si omette la condizione di join, si ottiene il prodotto cartesiano: ogni riga di una tabella viene messa in corrispondenza con ogni riga dell'altra tabella.

SQL	select S.SName, S.City, SP.Qty, P.PName, P.Color from S, SP, P;
Calcolo su tuple con dichiarazioni di range	{ s.(SName, City), sp.(Qty), p.(PName, Color) | s(S), p(P), sp(SP) }
Algebra relazionale	$\pi_{SName,S.City,Qty,PName,Color}(S \times SP \times P)$

Dimenticare le condizioni del join è un errore grave!

Forme di una query SQL

- Selezione e join:
  - Mostra nome e città dei fornitori che hanno fornito viti rosse (red screws).

SQL	select S.SName, S.City from S, SP, P where S.SNum=SP.SNum and P.PNum=SP.Pnum and P.PName='Screw' and P.Color='Red';
Calcolo su tuple con dichiarazioni di range	{ s.(SName, City) | s(S), p(P), sp(SP) | s.SNum=sp.SNum ∧ p.PNum=sp.PNum ∧ p.PName='Screw' ∧ p.Color='Red' }
Algebra relazionale	$\pi_{SName,S.City}(\sigma_{PName='Screw' \wedge Color='Red'}(S \bowtie SP \bowtie P))$ oppure $\pi_{SName,S.City}(S \bowtie SP \bowtie \sigma_{PName='Screw' \wedge Color='Red'}(P))$

Come impostare una query SQL - 1

- Clausola from:
  - Dove sono contenute le informazioni? (range list)
- Clausola where:
  - Quali condizioni devono soddisfare le informazioni? (formula)
- Clausola select:
  - Quali informazioni voglio mostrare? (target list)

Come impostare una query SQL - 2

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw).

Come impostare una query SQL - 3

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)
- Domande da porsi:
  - Dove sono contenute le informazioni?
    - Status, nome e città dei fornitori: Tabella *S*
    - Nome della parte: Tabella *P*
    - Il fatto che un fornitore abbia fornito un prodotto: Tabella *SP*
- Traduzione in SQL:

```
from S, P, SP
```

Come impostare una query SQL - 4

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)
- Domande da porsi:
  - Quali condizioni devono soddisfare le informazioni?
    - Le forniture in SP devono essere collegate ai fornitori in S tramite il codice SNum e alle parti in P tramite il codice PNum
    - I fornitori devono essere di Londra
    - La parti fornite devono essere le viti (Screw)
- Traduzione in SQL

```
where SP.SNum=S.SNum and SP.PNum=P.PNum and
S.City='London' and P.PName='Screw'
```

Come impostare una query SQL - 5

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)
- Domande da porsi:
  - Quali informazioni voglio mostrare?
    - Status e nome del fornitore
- Traduzione in SQL:

```
select Status, SName
```

Come impostare una query SQL - 6

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)
- Mettiamo insieme i pezzi:
  - clausola select

    ```
    select Status, SName
    ```
  - clausola from

    ```
    from S, P, SP
    ```
  - clausola where

    ```
    where SP.SNum=S.SNum and SP.PNum=P.PNum and
    S.City='London' and P.PName='Screw'
    ```

Come impostare una query SQL - 7

- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)
- Query risultante:

```
select Status, SName
from S, P, SP
where SP.SNum=S.SNum and SP.PNum=P.PNum and
S.City='London' and P.PName='Screw';
```

Status	SName
20	Smith
20	Smith
20	Clark

Gestione dei duplicati - 1

- Riprendiamo la query seguente
  - Elencare solo i nomi di tutti i fornitori di Londra che hanno fornito viti (Screw)

```
select SName
from S, P, SP
where SP.SNum=S.SNum and SP.PNum=P.PNum and
S.City='London' and P.PName='Screw'
```

- Qual è il risultato?

SName
Smith
Smith
Clark

Gestione dei duplicati - 2

- Differenza tra Algebra Relazionale e SQL:
  - In Algebra Relazionale una tabella viene vista come una *relazione* dal punto di vista matematico.
    - *Insieme di tuple* necessariamente diverse tra loro.
  - In SQL si possono avere nelle tabelle risultanti righe uguali, ovvero righe con gli stessi valori per tutti gli attributi.
    - Per motivi di efficienza i duplicati vengono conservati a meno di richiederne esplicitamente la rimozione.

Gestione dei duplicati - 3

- Come posso impedire di visualizzare duplicati?
  - Faccio seguire *select* dalla parola chiave *distinct*.
    - *distinct* elimina le *righe* duplicate.
- Elencare lo status e il nome di tutti i fornitori di Londra che hanno fornito viti (Screw)

```
select distinct Status, SName
from S, P, SP
where SP.SNum=S.SNum and SP.PNum=P.PNum
and S.City='London' and P.PName='Screw';
```

Status	SName
20	Smith
20	Clark

Gestione dei valori nulli - 1

- Un campo che vale *null* significa: l'attributo non è applicabile alla riga, o il valore è sconosciuto, o non si sa nulla.
- *null* è diverso rispetto a 0 (zero), stringa vuota, blank.
- La presenza di *null* e la logica a tre valori implicano che il confronto di un valore con null sia sempre unknown, quindi scrivere *Attributo = null* o *Attributo <> null* nella clausola *where* dà risultati inaspettati.
- Per questo motivo SQL offre il predicato *is null*, che invece è sempre o true o false:

Attributo is [not] null

Gestione dei valori nulli - 2

- Esempio 1: Selezionare le parti il cui attributo indicante il colore è nullo:

```
select * from P where Color is null;
```

- Esempio 2: Selezionare le parti il cui attributo indicante la città non è nullo:

```
select * from P where City is not null;
```

Gestione dei valori nulli - 3

- Prendiamo in considerazione una versione leggermente modificata della tabella *S*.

S

SNum	SName	Status	City
S1	Smith	(null)	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Gestione dei valori nulli - 3

- Con la query

```
select * from S where Status=10
```

cosa succede alle righe per cui l'attributo *Status* è **nullo**?

SNum	SName	Status	City
S2	Jones	10	Paris

Non vengono selezionate.

Gestione dei valori nulli - 4

- E con la query

```
select * from S where Status<>10
```
?

SNum	SName	Status	City
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Non vengono selezionate.

Gestione dei valori nulli - 5

- E con la query

```
select * from S where Status =10 or Status<>10?
```

SNum	SName	Status	City
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Non vengono selezionate.

- Nonostante la formula sembri una tautologia, non seleziona tutte le righe

Gestione dei valori nulli - 6

- Invece

```
select * from S where Status=10 or
Status<>10 or Status is null
```

seleziona tutte le righe

SNum	SName	Status	City
S1	Smith	(null)	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Gestione dei valori nulli - 7

- Cosa succede con

  `select * from S where Status=Status`

  nella seguente tabella?

SNum	SName	Status	City
S1	Smith	(null)	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Gestione dei valori nulli - 8

- A volte è utile interpretare i valori nulli come 0.
- La funzione *coalesce* accetta un elenco di parametri e restituisce il primo parametro non nullo (o NULL se sono tutti nulli).
- Per esempio visualizziamo il nome e lo status del fornitore riportando 0 se lo status è nullo:

```
select SName, coalesce(Status,0)
from S;
```

SName	Status
Smith	0
Jones	10
Blake	30
Clark	20
Adams	30

Ordinamento dei risultati - 1

- Per ordinare i risultati di una query si usa la clausola *order by:*

```
order by NomeAttributo1 [asc | desc],
NomeAttributo2 [asc | desc], …
```

- Le righe vengono ordinate in base al primo attributo in ordine crescente `asc` ("*ascending*") o decrescente `desc` ("*descending*").
- Le righe con valori uguali del primo attributo vengono ordinate in base al secondo attributo e così via.
- I valori nulli vengono messi tutti insieme all'inizio o alla fine (a seconda del DBMS).
- `asc` può essere lasciato sottinteso.

Ordinamento dei risultati - 2

- Elencare tutte le parti in ordine di peso crescente

```
select * from P order by Weight;
```

- Elencare tutti i fornitori ordinati in base allo status (decrescente) e al nome (crescente)

```
select * from S order by Status desc, SName;
```

- Elencare tutte le parti rosse in ordine di peso decrescente

```
select *
from P
where Color='Red'
order by Weight desc;
```

Commenti

- Nelle query complesse può essere utile inserire commenti.
- I commenti su un riga sola devono iniziare con -- (meno meno):

```
-- Questo è un commento
```

- Per i commenti su più righe si usa la notazione del C:

```
/* Questo
    è
    un
    commento */
```