Tutorato di Base di Dati
Lezione 4

Andrea Gasparetto

Il linguaggio SQL (Structured Query Language)

- Le implementazioni nei vari DBMS relazionali commerciali
  - includono funzionalità non previste dallo standard
  - non includono funzionalità previste dallo standard
  - implementano funzionalità previste dallo standard ma in modo diverso
  - vengono usati dei “dialetti” dell’SQL

Il linguaggio SQL (cont.)

- Linguaggio dichiarativo basato su Calcolo Relazionale su Ennuple e Algebra Relazionale
  - relazioni -> tabelle
  - ennuple -> record/righe
  - attributi -> campi/colonne
- Le tabelle possono avere righe duplicate (una tabella è un multinsieme), per
  - efficienza: eliminare i duplicati costa (n log(n))
  - flessibilità:
    - può essere utile vedere i duplicati
    - possono servire per le funzioni di aggregazione (es. media)

Il linguaggio SQL (cont.)

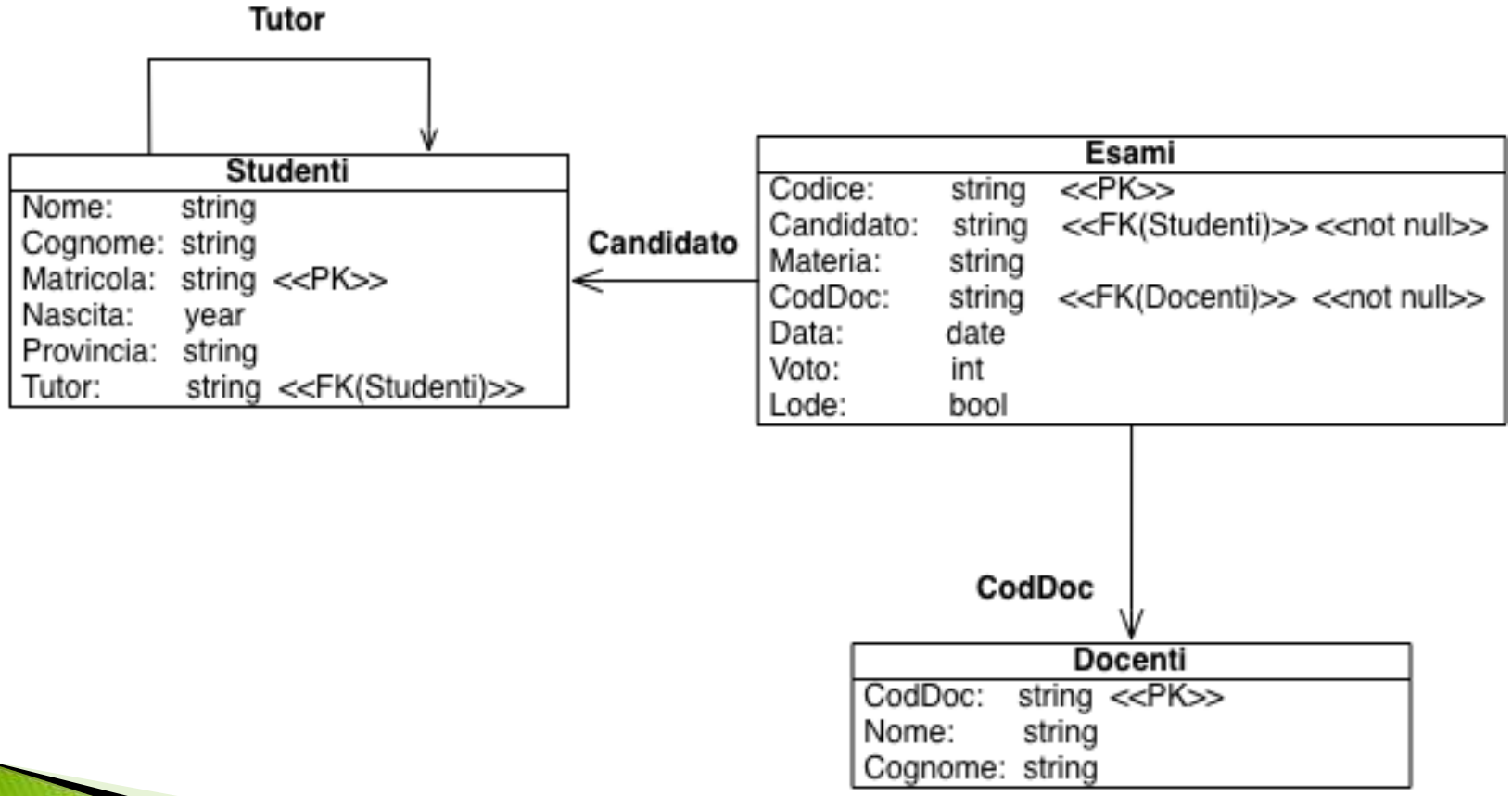
- Il linguaggio comprende
  - DML (Data Manipulation Language)

    ricerche e/o modifiche interattive -> interrogazioni

    o query
  - DDL (Data Definition Language)
    definizione (e amministrazione) della base di dati
  - uso di SQL in altri linguaggi di programmazione

Un assaggio ...

Consideriamo lo schema relazionale



Un assaggio ...

```
SELECT   s.Nome, s.Cognome, e.Data
 FROM     Studenti s JOIN Esami e
          ON (s.Matricola = e.Candidato)
 WHERE    e.Materia='BD' AND e.Voto=30
```

```
SELECT     s.Nome AS Nome,
           YEAR(CURDATE()) - s.Nascita AS Età,
           0 AS NumeroEsami
 FROM      Studenti s
 WHERE     NOT EXISTS
                 (SELECT   *
                  FROM     Esami e
                  WHERE    s.Matricola =
                              e.Candidato)
```

Il comando SELECT

- Il comando base dell'SQL:
  - **SELECT** [**DISTINCT**] *Attributi*
    FROM *Tabelle*
    [**WHERE** *Condizione*]

    Tabelle ::= *Tabella* [*Ide*] {, *Tabella* [*Ide*] }

- Condizione può essere una combinazione booleana (AND, OR, NOT) di (dis)uguaglianze tra attributi (=, <, <=, ...) ... ma anche molto altro.

Il comando SELECT (cont.)

- Semantica: prodotto + restrizione + proiezione.
  - **SELECT** *
    FROM R1, ..., Rn

    $R_1 \times \ldots \times R_n$

  - **SELECT** *
    FROM R1, ..., Rn
    WHERE C

    $\sigma_C(R_1 \times \ldots \times R_n)$

  - **SELECT DISTINCT** A1, ..., An
    FROM R1, ..., Rn
    WHERE C

    $\pi_{A_1,\ldots,A_n}(\sigma_C(R_1 \times \ldots \times R_n))$

Esempi Elementari

- ```
  SELECT *
  FROM   Studenti;
  ```

- ```
  SELECT *
  FROM   Esami
  WHERE  Voto > 26;
  ```

- ```
  SELECT DISTINCT Provincia
  FROM Studenti;
  ```

- ```
  SELECT *
  FROM   Studenti, Esami;
  ```

Esempi: Proiezione

- Trovare il nome, la matricola e la provincia degli studenti:

```
SELECT Nome, Matricola, Provincia
FROM   Studenti
```

Nome	Matricola	Prov
Paolo	71523	VE
Anna	76366	PD
Chiara	71347	VE

Esempi: Restrizione

- Trovare tutti i dati degli studenti di Venezia:

```
SELECT   *
FROM     Studenti
WHERE    Provincia = 'VE';
```

Nome	Cognome	...	Provincia	...
Paolo	Verdi	...	VE	...
Chiara	Scuri	...	VE	...

- Trovare nome, matricola e anno di nascita degli studenti di Venezia (Proiezione+Restrizione):

```
SELECT   Nome, Matricola, Nascita
FROM     Studenti
WHERE    Provincia = 'VE';
```

Nome	Matricola	Nascita
Paolo	71523	1989
Chiara	71346	1987

Prodotto e Giunzioni

- Tutte le possibili coppie (Studente, Esame):

```
SELECT  *
FROM    Studenti, Esami
```

- Tutte le possibili coppie (Studente, Esame sostenuto dallo studente):

```
SELECT *
FROM   Studenti, Esami
WHERE  Matricola = Candidato
```

- Nome e data degli esami per studenti che hanno superato l'esame di BD con 30:

```
SELECT  Nome, Data
FROM    Studenti, Esami
WHERE   Materia='BD' AND Voto=30
        AND Matricola = Candidato
```

Qualificazione: notazione con il punto

- Se si opera sul prodotto di tabelle con attributi omonimi occorre qualificarli, ovvero identificare la tabella alla quale ciascun attributo si riferisce
- Notazione con il Punto. Utile se si opera su tabelle diverse con attributi aventi lo stesso nome

`Tabella.Attributo`

  - Es. generare una tabella che riporti `Codice`, `Nome`, `Cognome` dei docenti e `Codice` degli esami corrispondenti

```
SELECT   Docenti.CodDoc, Docenti.Nome, Docenti.Cognome,
         Esami.Codice
FROM     Esami, Docenti
WHERE    Docenti.CodDoc = Esami.CodDoc
```

Qualificazione: notazione con il punto e alias

- Alias
  - Si associa un identificatore alle relazioni in gioco
  - Essenziale se si opera su più copie della stessa relazione (-> associazioni ricorsive!)
  - Es. generare una tabella che contenga cognomi e matricole degli studenti e dei loro tutor

```
SELECT s.Cognome, s.Matricola, t.Cognome, t.Matricola
FROM   Studenti s, Studenti t
WHERE  s.Tutor = t.Matricola
```

- La qualificazione è sempre possibile e può rendere la query più leggibile

Alias e query ricorsive

- Gli alias permettono di avere 'ricorsività' a un numero arbitrario di livelli.
  Esempio:

```
Persone (Id, Nome, Cognome, IdPadre, Lavoro)
PK(Id), IdPadre FK(Persone)
```

```
SELECT n.Nome, n.Cognome,
       f.Nome, f.Cognome
FROM   Persone n,
       Persone p,
       Persone f
WHERE  f.IdPadre = p.Id AND
       p.IdPadre = n.Id AND
       n.Lavoro = f.Lavoro
```

Cognome e nome delle persone (e dei nonni) che fanno lo stesso lavoro dei nonni

Lista degli attributi

- `Attributi ::= *`
  `| Expr [[AS] Nome] {, Expr [[AS] Nome] }`

  `Expr AS Nome`: dà il nome `Nome` alla colonna ottenuta come risultato dell'espressione **`Expr`**
- usato per rinominare attributi o più comunemente per dare un nome ad un attributo calcolato

```
SELECT Nome, Cognome, YEAR(CURDATE())-Nascita
AS Età
FROM   Studenti
WHERE  Provincia='VE'
```

- Nota: Un attributo A di una tabella "`R x`" si denota come: `A` oppure `R.A` oppure `x.A`

Sintassi delle espressioni

- Le espressioni possono includere operatori aritmetici (o altri operatori e funzioni sui tipi degli attributi) o funzioni di aggregazione

  - ```
    Expr ::= [Ide.]Attributo | Const
    |   ( Expr )   |   [-] Expr [Op Expr]
    |   COUNT(*)
    |   AggrFun ( [DISTINCT] [Ide.]Attributo)
    ```

  - ```
    AggrFun ::= SUM   |   COUNT   |   AVG   |   MAX   |   MIN
    ```

- NB: si usano tutte funzioni di aggregazione (-> produce un'unica riga) o nessuna.
- Le funzioni di aggregazione NON possono essere usati nella clausola WHERE

ESEMPI: funzioni di aggregazione

- Numero di elementi della relazione Studenti

```
SELECT  count(*)
FROM    Studenti
```

- Anno di nascita minimo, massimo e medio degli studenti:

```
SELECT    min(Nascita), max(Nascita), avg(Nascita)
FROM      Studenti
```

- è diverso da

```
SELECT    min(Nascita), max(Nascita), avg(DISTINCT Nascita)
FROM      Studenti
```

- Nota: non ha senso ... (vedi `GROUP BY`)

```
SELECT    Candidato, avg(Voto)
FROM      Esami
```

ESEMPI: funzioni di aggregazione

- Numero di Studenti che hanno un Tutor

```
SELECT   count(Tutor)
FROM     Studenti
```

- Numero di studenti che fanno i Tutor

```
SELECT   count(DISTINCT Tutor)
FROM     Studenti
```

Clausola FROM (reprise)

- Le tabelle si possono combinare usando:
  - "," (prodotto): `FROM T1,T2`
  - Giunzioni di vario genere

- `Tabelle ::= Tabella [Ide] {, Tabella [Ide] } |`

  `Tabella Giunzione Tabella`
  `[ USING (Attributi) | ON Condizione ]`

`Giunzione ::= [CROSS|NATURAL] [LEFT|RIGHT|FULL] JOIN`

Clausola FROM: giunzioni

- **CROSS JOIN**
  realizza il prodotto

```
SELECT *
FROM   Esami CROSS JOIN Docenti
```

- **NATURAL JOIN**
  è il join naturale

```
SELECT *
FROM   Esami NATURAL JOIN Docenti;
```

Clausola FROM: giunzioni

- **JOIN ... USING** *Alcuni attributi comuni*
  come il natural join, ma solo sugli attributi comuni elencati

- **JOIN ... ON Condizione**
  effettua il join su di una condizione (ad es. che indica quali valori devono essere uguali)

```
SELECT *
FROM   Studenti s JOIN Studenti t
                  ON s.Tutor = t.Matricola;
```

Clausola FROM: giunzioni

- **LEFT, RIGHT, FULL**
  se precedono JOIN, effettuano la corrispondente giunzione esterna
- Esempio: Esami di tutti gli studenti, con nome e cognome relativo, elencando anche gli studenti che non hanno fatto esami

```
SELECT Nome, Cognome, Matricola, Data, Materia
FROM   Studenti s LEFT JOIN Esami e
       ON s.Matricola = e.Candidato;
```

Clausola FROM: giunzioni

- Risultato

Nome	Cognome	Matricola	Data	Materia
Chiara	Scuri	71346	NULL	NULL
Giorgio	Zeri	71347	NULL	NULL
Paolo	Verdi	71523	2006-07-08	BD
Paolo	Verdi	71523	2006-12-28	ALG
Paolo	Poli	71576	2007-07-19	ALG
Paolo	Poli	71576	2007-07-29	FIS
Anna	Rossi	76366	2007-07-18	BD
Anna	Rossi	76366	2007-07-08	FIS

Nota: compaiono anche ennuple corrispondenti a studenti che non hanno fatto esami, completate con valori nulli.

Clausola ORDER BY

- Inserendo la clausola

  ORDER BY *Attributo* [**DESC|ASC**] {, Attributo [**DESC|ASC**] }

  si può far sì che la tabella risultante sia ordinata, secondo gli attribuuti indicati (ordine lessicografico) in modo crescente (ASC) [default] o decrescente (DESC): e.g.

```
SELECT Nome,Cognome
FROM   Studenti
WHERE  Provincia='VE'
ORDER BY Cognome DESC, Nome DESC
```

Operatori insiemistici

- SQL comprende operatori insiemistici (`UNION`, `INTERSECTION` ed `EXCEPT`) per combinare i risultati di tabelle con colonne di ugual nome e ugual tipo
- Es: Nome e cognome degli studenti di Venezia e di quelli che hanno preso più di 28 in qualche esame

```
SELECT Nome, Cognome
FROM   Studenti
WHERE  Provincia='VE'
   UNION
SELECT Nome, Cognome
FROM   Studenti JOIN Esami ON (Matricola=Candidato)
WHERE  Voto>28;
```

Operatori insiemistici

- Se le tabelle sulle quali operare hanno attributi con lo stesso tipo, ma con nome diverso, si possono rinominare con AS
- Esempio: Le matricole degli studenti che non sono tutor

```
SELECT  Matricola
FROM    Studenti
   EXCEPT
SELECT  Tutor AS Matricola
FROM    Studenti
```

Operatori insiemistici

- Effettuano la rimozione dei duplicati, a meno che non sia esplicitamente richiesto il contrario con l'opzione `ALL`

- Es: Nome e cognome degli studenti di Venezia che hanno preso più di 28 in qualche esame

```
SELECT Nome, Cognome
FROM   Studenti
WHERE  Provincia='VE'
   INTERSECTION ALL
SELECT Nome, Cognome
FROM   Studenti JOIN Esami ON (Matricola=Candidato)
WHERE  Voto>28;
```

Il valore NULL

- Il valore di un campo di un'ennupla può mancare per varie ragioni
  - attributo non applicabile
  - attributo non disponibile
  - ...
- SQL fornisce il valore speciale **NULL** per tali situazioni.
- La presenza di `NULL` introduce dei problemi:
  - la condizione "`Matricola=9`" è vera o falsa quando la Matricola è `NULL`?

    è vero `NULL=NULL`?

    Cosa succede degli operatori `AND`, `OR` e `NOT`?

Il valore NULL

- Dato che NULL può avere diversi significati
  - NULL=0 non è né vero, né falso, ma unknown
  - anche NULL=NULL è unknown
- Occorre una logica a 3 valori (vero, falso e unknown).

p	¬p
T	F
F	T
U	U

p	q	p∧q	p∨q
T	T	T	T
T	F	F	T
T	U	U	T
F	T	F	T
F	F	F	F
F	U	F	U
U	T	U	T
U	F	F	U
U	U	U	U

Il valore NULL

- Va definita opportunamente la semantica dei costrutti. Ad es.

  SELECT ... **FROM** ...
  WHERE *COND*

  restituisce solo le ennuple che rendono vera la condizione *COND*.

- Necessario un predicato per il test di nullità

  `Expr` **IS** [**NOT**] `NULL`

  è vero se `Expr` (non) è `NULL`

- Nota che `NULL=NULL` vale `NULL`!!

- Nuovi operatori sono utili (es. giunzioni esterne)

Il valore NULL: Esempio

- Gli studenti che non hanno Tutor

```
SELECT *
FROM   Studenti
WHERE  Tutor IS NULL
```

Nome	Cognome	Matricola	Nascita	Provincia	Tutor
Giorgio	Zeri	71347	1987	VE	NULL
Paolo	Verdi	71523	1986	VE	NULL

- Cosa ritorna?

```
SELECT *
FROM   Studenti
WHERE  Tutor = NULL
```

Altre condizioni: between

- Su valori numerici
  - **WHERE** *Expr* **BETWEEN** *Expr* **AND** *Expr*

  - ```
    SELECT *
    FROM   Studenti
    WHERE  Matricola BETWEEN 71000 AND 72000;
    ```

```
+---------+---------+-----------+---------+-----------+-------+
| Nome    | Cognome | Matricola | Nascita | Provincia | Tutor |
+---------+---------+-----------+---------+-----------+-------+
| Chiara  | Scuri   | 71346     |    1985 | VE        | 71347 |
| Giorgio | Zeri    | 71347     |    1987 | VE        | NULL  |
| Paolo   | Verdi   | 71523     |    1986 | VE        | NULL  |
| Paolo   | Poli    | 71576     |    1988 | PD        | 71523 |
+---------+---------+-----------+---------+-----------+-------+
```
```