Sviluppi dell'Informatica dellaTelematica
a.a. 2008-2009

il linguaggio SQL

Atzeni-Ceri-Paraboschi-Torlone, Basi di dati, Capitolo 4: SQL

avviso

- Giovedi pomeriggio (26/3 ore 16-18) inizio laboratorio di Sviluppi
- Presso aula informatica grande

Il linguaggio SQL consente:

Creazione di una base di dati

definizione di schemi di *relazioni* (tabelle) e rispettive instanziazioni (popolamento)

Interrogazione di una base di dati

derivazione di *nuove relazioni* (tabelle o views) a partire da quelle esistenti

due fasi:
definizione dei dati (schemi, domini e vincoli)
operazioni sui dati (istanze e queries)

SQL

Originariamente

***S**tructured **Q**uery **L**anguage*

- **linguaggio con varie funzionalità, contiene:**
  - **sia il *Data Definition Language (DDL)***
  - **sia il *Data Management Language (DML)***
- **ne esistono varie versioni**

Nota: le slides illustrano gli aspetti essenziali, i dettagli sono disponibili nel libro di Atzeni et alii.

SQL: "storia"

- prima proposta SEQUEL, 1974
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto"

SQL: prima fase

DEFINIZIONE DEI DATI

Definizione dei dati in SQL

Sintassi usata nella grammatica delle istruzioni SQL

- Parentesi angolari **<>**, isolano un termine della sintassi
- Parentesi quadre **[,]** indicano un termine opzionale (compare 0 o max 1 volta)
- Parentesi graffe **{,}** indicano un termine opzionale (compare 0 o più volte)
- Le barre **|** indicano che deve essere scelto uno tra i termini separati dalle barre

NOTA: Le parentesi tonde **(,)** vanno intese come simboli del linguaggio SQL e non come simboli per la definizione della grammatica

Definizione dei dati in SQL

- Istruzione CREATE TABLE
  - definisce uno schema di relazione e ne crea un'istanza vuota
  - specifica attributi, domini e vincoli

Esempio:

```
Create table NomeTabella
 (
 {Attributo Dominio [Default] [Vincoli]}
 [AltriVincoli]
 )
```

CREATE TABLE: semplice esempio

```
Create table dipartimento
(
  NomeDip      char(15) primary key,
  Sede         char(20) not null
)
```

NOTA: non si usano accenti

CREATE TABLE: esempio complesso

attributi

vincoli

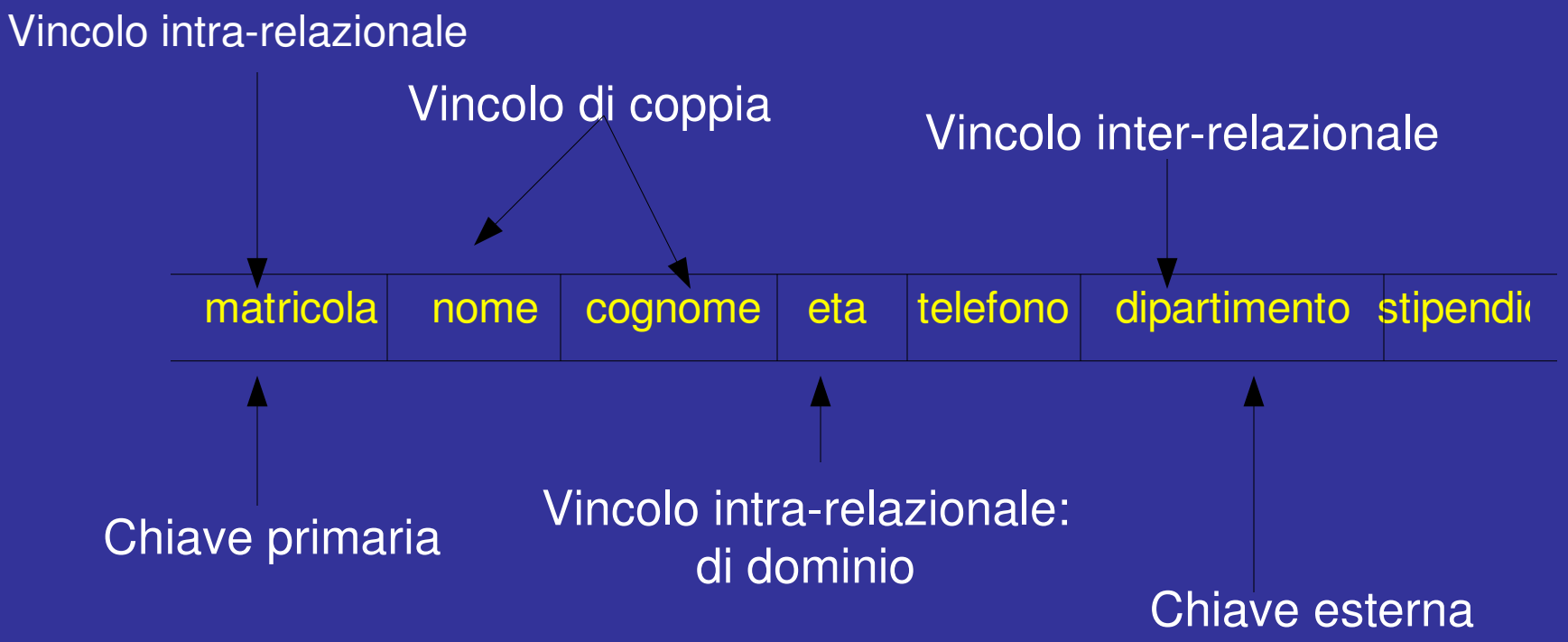
```
CREATE TABLE  Impiegato
(
   Matricola          CHARACTER(6)          PRIMARY KEY,
   Nome               VARCHAR(20)           NOT NULL,
   Cognome            VARCHAR(20)           NOT NULL,
   Eta                SMALLINT,
   Telefono           NUMERIC(10)           UNIQUE,
   Dipart             CHAR(15),
   Stipendio          DECIMAL(6,2)          DEFAULT (0000,00),

   PRIMARY KEY        (Matricola),
   FOREIGN KEY        (Dipart)              REFERENCES
                                            Dipartimento(NomeDip)
                                            ON DELETE SET NULL
                                            ON UPDATE CASCADE,

   UNIQUE             (Cognome, Nome),
   CHECK(ETA>0)
)
```

Schema della tabella creata

- **vincoli:**



Domini

- Domini **elementari** predefiniti
- Domini **definiti dall'utente** :
  semplici, ma riutilizzabili

Domini elementari (predefiniti)

- **Carattere** : singoli caratteri o stringhe, anche di lunghezza variabile

  Esempio:

  ```
  character [varying](20)[character set Courier]
  ```

- **Booleani** : 0, 1
- **Numerici** : esatti e approssimati
- **Bit** : singoli booleani o stringhe
- **Data, Ora, Intervalli di tempo…**

Domini definiti: esempio
CREATE DOMAIN

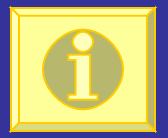
```
CREATE DOMAIN Voto
  AS SMALLINT DEFAULT NULL
  CHECK ( value >=18 AND value <= 30 )
```

Vincoli intra-relazionali

- **NOT NULL**
- **UNIQUE** (definisce le chiavi)
- **PRIMARY KEY**:
  - chiave primaria,
  - una sola,
  - implica **NOT NULL**,
  - può essere composta da più attributi

UNIQUE e PRIMARY KEY

- due forme:
  - nella definizione di un attributo, se questo forma da solo la chiave
  - come elemento separato



CREATE TABLE, esempio

```
CREATE TABLE Impiegato
(
   Matricola        CHAR(6) PRIMARY KEY,
   Nome             CHAR(20) NOT NULL,
   Cognome          CHAR(20) NOT NULL,
   Dipart           CHAR(15),
   Stipendio        NUMERIC(9) DEFAULT 0,
   FOREIGN KEY(Dipart) REFERENCES
                    Dipartimento(NomeDip),
   UNIQUE(Cognome, Nome)
 )
```

PRIMARY KEY, alternative

Matricola CHAR(6) PRIMARY KEY

OPPURE

```
create table Impiegato
(
Matricola CHAR(6),
... ,
PRIMARY KEY (Matricola)
)
```

CREATE TABLE, esempio

```
CREATE TABLE Impiegato
(
   Matricola        CHAR(6) PRIMARY KEY,
   Nome             CHAR(20) NOT NULL,
   Cognome          CHAR(20) NOT NULL,
   Dipart           CHAR(15),
   Stipendio        NUMERIC(9) DEFAULT 0,
   FOREIGN KEY(Dipart) REFERENCES
                    Dipartimento(NomeDip),
   UNIQUE(Cognome,Nome)
 )
```

Chiave Primaria nella tabella Dipartimento

Chiavi su più attributi, attenzione

```
...
Nome        CHAR(20) NOT NULL,
Cognome     CHAR(20) NOT NULL,
UNIQUE      (Cognome,Nome)
```

(una chiave primaria!)

```
...

Nome        CHAR(20) NOT NULL UNIQUE,
Cognome     CHAR(20) NOT NULL UNIQUE,
```

(due chiavi primarie!)

Non è la stessa cosa!

Vincoli inter-relazionali

- **REFERENCES e FOREIGN KEY permettono di definire vincoli di integrità referenziale :**
  - **per singoli attributi**
  - **su più attributi**

- **(... è possibile definire politiche di reazione alla violazione)**

Auto

Prov	Numero	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

Vigile

Matricola	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Infrazione

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

CREATE TABLE: esempio

```
CREATE TABLE Infrazioni
(
   Codice      CHAR(6)      PRIMARY KEY,
   Data        DATE         NOT NULL ,
   Vigile      INTEGER      NOT NULL
                            REFERENCES Vigile(Matricola),
                            on delete set null
   Prov        CHAR(2) ,
   Numero      CHAR(6) ,
   FOREIGN KEY(Prov, Numero)
                            REFERENCES Auto(Prov , Numero)
                            on update cascade
)
```

Modifiche degli schemi

ALTER TABLE

(consente di modificare lo schema e gli attributi di una tabella)

Esempio

alter table Impiegato add column NumUfficio char(20)

DROP TABLE

(consente di rimuovere componenti, schemi o domini, di una tabella)

Esempio

drop table Impiegato

SQL: seconda fase

OPERAZIONI SUI DATI

SQL: operazioni sui dati

***Modifica* dei dati:**

- **INSERT**
- **DELETE**
- **UPDATE**

***Interrogazione* dei dati:**

- **SELECT**

Operazioni di aggiornamento

operazioni di

- ***inserimento*** : **insert**
- ***eliminazione*** : **delete**
- ***modifica*** : **update**

- **su di una o più ennuple di una relazione**
- **sulla base di una condizione (che può coinvolgere anche altre relazioni)**

Inserimento: popolamento tabelle

```
INSERT INTO Tabella ( Attributo_1,…,Attributo_n )
       VALUES ( Valore_1,…,Valore_n )
```

Esempi

```
INSERT INTO Persone ( Nome, Eta, Reddito )
       VALUES ( 'Pino', 25, 52 )

INSERT INTO Persone ( Nome, Reddito )
       VALUES ( 'Lino', 55 )
```

Inserimento , commenti

- l'ordinamento degli attributi (se presente) e dei valori è significativo
- le due liste debbono avere lo stesso numero di elementi
- se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione, secondo l'ordine con cui sono stati definiti
- se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un valore nullo (che deve essere permesso) o un valore di default

Eliminazione di ennuple

DELETE FROM Tabella [**WHERE** Condizione]

Esempio semplice:
Cancellare le persone con meno di 35 anni

DELETE FROM Persone
WHERE Eta < 35

Esempio complesso:
cancellare tutte le persone che non sono padri

```
DELETE FROM Paternita
WHERE Figlio NOT in ( SELECT Nome
                      FROM Persone )
```

Eliminazione, commenti

- elimina le ennuple che soddisfano la condizione
- può causare (se i vincoli di integrità referenziale sono definiti con politiche di reazione cascade) eliminiazioni da altre relazioni
- ricordare: se la condizione *where* viene omessa, si intende where true

Modifica di ennuple

```
UPDATE NomeTabella
SET  Attributo = < Espressione |
                   SELECT ... |
                   NULL |
                   DEFAULT >
       [ WHERE Condizione ]
```

```
UPDATE Persone
SET Reddito = 45
WHERE Nome = 'Piero'
```

```
UPDATE Persone
SET Reddito = Reddito * 1.1
WHERE Eta < 30
```