Linguaggio SQL

Structured Query Language

Il linguaggio SQL

- per la definizione e la manipolazione dei dati
- supportato dalla totalità dei DBMS relazionali
- primo linguaggio **dichiarativo** progettato specificatamente per accesso e manipolazione di collezioni di dati
  - risultato di un'operazione specificato mediante condizioni sul contenuto dei dati
  - basato su calcolo (e algebra) relazionale
  - **Attenzione! grossa differenza: la semantica di una tabella non è un insieme, ma può contenere duplicati (multinsieme)**
- suo antesignano SEQUEL (IBM anni '70)
- standard
  - dal 1986 (SQL-1986)
  - più recente SQL:2008
  - due grosse revisioni precedenti: SQL2 (o SQL-92) e SQL:1999 (o SQL3), con caratteristiche object-relational

Il linguaggio SQL

- Sia DDL che DML
  - DDL (istruzioni per la definizione della struttura dei dati)
  - DML (istruzioni per l'interrogazione e l'aggiornamento dei dati)
- tutte le implementazioni di SQL prevedono inoltre comandi SDL (non standardizzati)
- comandi principali

operazione	DDL	DML
creazione	CREATE	INSERT
modifica	ALTER	UPDATE
cancellazione	DROP	DELETE
interrogazione		SELECT

DDL
Linguaggio di Definizione dei Dati

Tipi di dato

- I tipi di dato in SQL:2003 si suddividono in
  - tipi predefiniti
  - tipi user-defined
- ci concentreremo sui tipi predefiniti
- i tipi predefiniti sono suddivisi in tre categorie principali
  - tipi numerici
  - tipi carattere
  - tipi temporali
- i dettagli sui tipi
  - non li vediamo, li trovate sul libro di testo

Tipi numerici esatti

Valori interi

- **SMALLINT**
  - per risparmiare spazio quando si sa che i valori sono limitati
- **INTEGER**
- **BIGINT**
  - per memorizzare valori più grandi
- **Usualmente** INTEGER 32 bit, SMALLINT 16 bit, BIGINT 64 bit

Valori reali

numero complessivo di cifre

numero di cifre dopo la virgola

- NUMERIC [(**precisione**[,**scala**])]
  - NUMERIC (4,1) = [-999.9,999.9]
- DECIMAL [(**precisione**[,**scala**])]
  - Può essere implementato con precisione maggiore di quella richiesta
  - DECIMAL(4,1) ⊇ [-999.9,999.9]

Tipi numerici approssimati
Valori reali in virgola mobile

- **REAL**
  - singola precisione
- **DOUBLE PRECISION**
  - doppia precisione
- **FLOAT [(precisione)]**
  - precisione indicata
  - la precisione minima specificabile è 1, quella di default e la massima dipendono dalle specifiche implementazioni di SQL
- La precisione dipende dalla specifica implementazione di SQL
- La precisione singola deve in ogni caso essere minore della doppia

Tipi di dato carattere

- **CHARACTER/CHAR**
  - CHAR [(n)] = stringhe di **esattamente** n caratteri
  - stringhe di lunghezza inferiore ad n vengono completate con spazi fino a raggiungere lunghezza n
  - default CHAR = CHAR(1)
- **CHARACTER VARYING/ VARCHAR**
  - VARCHAR(n) = stringhe di caratteri con **lunghezza massima** n
- Per ogni valore di CHAR(n) si alloca la lunghezza predefinita, mentre per valori di VARCHAR si adottano strategie diverse
- I valori di tipo CHAR/VARCHAR vanno racchiusi tra singoli apici
  - es. ‘pulp fiction’, ‘pippo & sua zia’, ‘123’
  - case sensitive: ‘pippo’≠ ‘Pippo’

Tipi di dato temporali: Istantanei

- **DATE**
  - risoluzione di un giorno
  - formato dei valori

    DATE <<stringa che rappresenta data>>
    - in opportuno standard
    - es. DATE '27-Ott-1969' o DATE '1969-10-27'
- **TIME[(p)]**
  - p è l'eventuale numero di cifre frazionarie cui si è interessati
    - fino a 6, microsecondo
  - formato dei valori

    TIME 'hh:mm:ss[.nnnnnn]'
    - es. TIME '21:56:32.5'
- **TIMESTAMP[(p)] = DATE+TIME[(p)]**
  - formato dei valori

    TIMESTAMP <<stringa che rappresenta data>> 'hh:mm:ss[.nnnnnn]'
    - es. TIMESTAMP '27-Ott-1969 21:56:32.5'

Tipi di dato temporali: Intervalli

- **INTERVAL**
  - rappresenta una durata temporale
  - formato dei valori

    INTERVAL <<stringa che esprime durata>> <<unità di misura>>

    es. INTERVAL '3' YEAR

    INTERVAL <<stringa che esprime durata>> <<unità di misura>> TO <<unità di misura>>

    es. INTERVAL '36 22:30' DAY TO MINUTE
  - Se la granularità minima è il mese si parla di intervalli year-month
  - Altrimenti si parla di intervalli day-time (memorizzati in secondi)

TIPO DI DATO BOOLEAN

- Consiste di TRUE, FALSE, **UNKNOWN**
- UNKNOWN
  - rappresenta totale mancanza di conoscenza, equivale a NULL
  - si usa per la gestione dei confronti con valori nulli
    - se exp si valuta a null, i confronti con exp e i predicati atomici su di exp si valutano a UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Altri tipi di dato

- **CHARACTER LARGE OBJECT (CLOB)**
  - Per rappresentare sequenze di caratteri di elevate dimensioni (testo)
- **BINARY LARGE OBJECT (BLOB)**
  - per rappresentare sequenze di bit di elevate dimensioni (immagini)

CREAZIONE DI RELAZIONI

```
CREATE TABLE  Video
        (colloc     DECIMAL(4),
         titolo     VARCHAR(30),
         regista    VARCHAR(20),
         tipo       CHAR DEFAULT 'd');
```

Il valore di default deve appartenere al dominio

Sintassi base

- CREATE TABLE <nome relazione>
  (<specifica colonna> [,<specifica colonna>]*)
- specifica colonna ::= <**nome colonna**> <**dominio**>
  [**DEFAULT** <valore default>]

Creazione di relazioni - Vincoli

- I vincoli di integrità costituiscono un aspetto importante nella definizione dello schema di una base di dati
- in SQL è possibile specificare nella definizione di una relazione
  - obbligatorietà di colonne (NOT NULL)
  - chiavi (UNIQUE e PRIMARY KEY)
  - chiavi esterne (FOREIGN KEY)
  - vincoli CHECK (su colonna o su tupla)
    - li vedremo dopo aver presentato il linguaggio di interrogazione

OBBLIGATORIETÀ DI COLONNE

- Rispetto al modello relazionale il default è a rovescio
  se non si specifica esplicitamente la colonna può assumere valori nulli
- Per specificare che una colonna non può assumere valori nulli si aggiunge il vincolo NOT NULL nella specifica della colonna
  specifica colonna ::= <nome colonna> <dominio>
  [DEFAULT <valore default>]
  [NOT NULL]
- Esempio:

```
CREATE TABLE  Video
        (colloc     DECIMAL(4) NOT NULL,
         titolo      VARCHAR(30) NOT NULL,
         regista     VARCHAR(20) NOT NULL,
         tipo        CHAR DEFAULT 'd');
```

Se voglio posso assegnargli esplicitamente un NULL

Chiavi

- Chiavi primarie indicate dalla parola chiave PRIMARY KEY
  - per ogni tupla i valori degli attributi specificati sono non nulli e diversi da quelli di ogni altra tupla
- Chiavi alternative indicate dalla parola chiave UNIQUE
  - non esistano due tuple che condividono gli stessi valori non nulli per gli attributi, ma alcune colonne UNIQUE possono contenere valori nulli
- In una tabella è possibile specificare più chiavi UNIQUE ma una sola PRIMARY KEY
- Siccome una relazione SQL può contenere tuple duplicate può avere senso definire chiavi costituite da tutte le colonne di una relazione

Chiavi

- Alla fine della definizione della tabella si indicano le chiavi

  PRIMARY KEY (<lista nomi colonne>)

  o

  UNIQUE(<lista nomi colonne>)

- esempio:

```
CREATE TABLE Noleggio
    (colloc      DECIMAL(4),
     dataNol     DATE DEFAULT CURRENT_DATE,
     codCli      DECIMAL(4) NOT NULL,
     dataRest    DATE,
    PRIMARY KEY (colloc,dataNol),
    UNIQUE (colloc,dataRest));
```

dataRest può assumere valori nulli

+

i valori nulli non sono distinti

⇒

ci possono due noleggi correnti per lo stesso video

Chiavi

- Se la chiave è formata da una sola colonna si può far seguire la specifica della colonna da UNIQUE o PRIMARY KEY
- Esempio

```
CREATE TABLE  Video
      (colloc     DECIMAL(4) PRIMARY KEY,
       titolo      VARCHAR(30) NOT NULL,
       regista     VARCHAR(20) NOT NULL,
       tipo        CHAR DEFAULT 'd');
```

la specifica di PRIMARY KEY rende inutile la clausola NOT NULL

- Equivalente a

```
CREATE TABLE  Video
      (colloc     DECIMAL(4),
       titolo      VARCHAR(30) NOT NULL,
       regista     VARCHAR(20) NOT NULL,
       tipo        CHAR DEFAULT 'd',
      PRIMARY KEY (colloc));
```

CHIAVI ESTERNE

Clausola opzionale FOREIGN KEY del comando CREATE TABLE

chiave esterna della relazione che si sta definendo (relazione referente)

devono corrispondere ad una chiave della relazione riferita

```
FOREIGN KEY (<lista nomi colonne>)
    REFERENCES <nome relazione>(<lista nomi colonne riferite>)
    [ ON DELETE { NO ACTION |
        CASCADE | SET NULL | SET
DEFAULT}]
    [ ON UPDATE { NO ACTION |
        CASCADE | SET NULL | SET
DEFAULT}]
```

relazione riferita

```
CREATE TABLE Film
    (titolo   VARCHAR(30),
regista  VARCHAR(20),
anno     DECIMAL(4) NOT NULL,
genere   CHAR(15) NOT NULL,
valutaz  NUMERIC(3,2),
 PRIMARY KEY
(titolo,regista))
```

- non necessariamente gli stessi nomi
- i domini degli attributi corrispondenti devono essere compatibili
- si può (deve in caso di ambiguità) indicare **quali sono le colonne riferite**

```
CREATE TABLE  Video
    (colloc      DECIMAL(4) PRIMARY KEY,
 titolo      VARCHAR(30) NOT NULL,
 regista      VARCHAR(20) NOT NULL,
 tipo        CHAR NOT NULL DEFAULT 'd',
 FOREIGN KEY (titolo,regista)
      REFERENCES Film);
```

CHIAVI ESTERNE: SINGOLA COLONNA

Se la chiave esterna è costituita da un solo attributo si può far seguire la specifica della colonna da REFERENCES <nome relazione>

```
CREATE TABLE Cliente
      (codCli   DECIMAL(4) PRIMARY KEY,
       …)

CREATE TABLE Noleggio
      (colloc   DECIMAL(4) REFERENCES Video,
       dataNol  DATE DEFAULT CURRENT_DATE,
       codCli   DECIMAL(4) NOT NULL REFERENCES Cliente,
       dataRest DATE,
       PRIMARY KEY (colloc,dataNol),
       UNIQUE (colloc,dataRest));
```

CHIAVI ESTERNE: CLAUSOLA ON DELETE

- Per specificare le azioni da eseguire nel caso di cancellazione di una tupla **t** della tabella R_TA **riferita** tramite chiave esterna dalla tabella R_NTE
- le opzioni possibili sono
  - NO ACTION: **t** viene cancellata solo se non esiste alcuna tupla in R_NTE che vi fa riferimento
  - CASCADE: la cancellazione di **t** implica la cancellazione di tutte le tuple in R_NTE che vi fanno riferimento
  - SET NULL: la cancellazione di **t** implica che in tutte le tuple in R_NTE che vi fanno riferimento la chiave esterna viene posta a NULL (se ammesso)
  - SET DEFAULT: la cancellazione di **t** implica che in tutte le tuple in R_NTE che vi fanno riferimento la chiave esterna viene posta al valore di default specificato per le colonne che costituiscono la chiave esterna
- L'opzione di default è NO ACTION

Chiavi esterne: clausola ON UPDATE

- Per specificare le azioni da eseguire nel caso di modifica di una tupla **t** della tabella R_TA **riferita** tramite chiave esterna dalla tabella R_NTE
- le opzioni sono le stesse ed hanno lo stesso significato delle opzioni viste per la cancellazione

  Differenza

  l'opzione CASCADE assegna alla chiave esterna il nuovo valore di chiave di **t**
- l'opzione di default è NO ACTION anche per la modifica
- nel caso di inserimento o modifica nella tabella referente non è possibile specificare alcuna opzione e viene applicata sempre NO ACTION

Chiavi esterne

- Esempio videoteca

```
CREATE TABLE Film
    (titolo   VARCHAR(30),
     regista  VARCHAR(20),
     anno     DECIMAL(4) NOT NULL,
     genere   CHAR(15) NOT NULL,
     valutaz  NUMERIC(3,2),
     PRIMARY KEY (titolo,regista));

CREATE TABLE Cliente
    (codCli    DECIMAL(4) PRIMARY KEY,
     nome      VARCHAR(20) NOT NULL,
     cognome   VARCHAR(20) NOT NULL,
     telefono  CHAR(15) NOT NULL,
     dataN     DATE NOT NULL,
     residenza VARCHAR(30) NOT NULL,
     UNIQUE (nome,cognome,dataN));
```

Chiavi esterne

- Esempio videoteca (segue)

```
CREATE TABLE  Video
    (colloc      DECIMAL(4) PRIMARY KEY,
     titolo      VARCHAR(30) NOT NULL,
     regista     VARCHAR(20) NOT NULL,
     tipo        CHAR NOT NULL DEFAULT 'd',
     FOREIGN KEY (titolo,regista) REFERENCES Film
                          ON DELETE NO ACTION);
```

  - non viene permessa la cancellazione di informazioni relative a film se vi sono video contenenti quel film

Chiavi esterne

- Esempio videoteca (segue)

```
CREATE TABLE Noleggio
  (colloc   DECIMAL(4) REFERENCES Video
                ON DELETE CASCADE
                 ON UPDATE CASCADE,
   dataNol  DATE DEFAULT CURRENT_DATE,
   codCli   DECIMAL(4) NOT NULL REFERENCES Cliente
                ON DELETE CASCADE
                 ON UPDATE CASCADE,
   dataRest DATE,
   PRIMARY KEY (colloc,dataNol),
   UNIQUE (colloc,dataRest));
```

Chiavi esterne

- Esempio videoteca (segue)
  - se viene cancellato un cliente vengono cancellati anche tutti i noleggi da lui effettuati
  - se viene cancellato un video vengono cancellati anche tutti i noleggi di tale video
  - le modifiche di codice cliente e collocazione video vengono propagate ai relativi noleggi

CANCELLAZIONE DI RELAZIONI

relazione da cancellare

gestione relazioni referenti ad essa

- DROP TABLE <nome relazione> {RECTRICT | CASCADE};

la relazione viene cancellata solo se non è riferita da altri elementi dello schema della base di dati

la relazione e tutti gli elementi dello schema della base di dati che la riferiscono vengono cancellati

Nello schema della videoteca

- Film è riferita da Video
- Video è riferita da Noleggio

DROP TABLE Film
RESTRICT;
non ha effetto

DROP TABLE Film
CASCADE;
cancella Film
cancella Video
cancella Noleggio

Modifica di relazioni
variazioni su singola colonna

relazione da modificare

ALTER TABLE <nome relazione> <modifica>;

aggiunta

ADD [COLUMN] <colonna>

ALTER TABLE Film

ADD COLUMN studio VARCHAR(20);

- aggiunge in fondo alla relazione Film la colonna studio
- manca specifica del default ⇒ a tutte le tuple di Film viene assegnato il valore NULL per tale colonna

modifica del default

ALTER [COLUMN] <colonna>
{SET DEFAULT <valore default>
| DROP DEFAULT}

ALTER TABLE Video

ALTER COLUMN tipo SET DEFAULT 'v';

ALTER TABLE Video

ALTER COLUMN tipo DROP DEFAULT;

eliminazione

DROP [COLUMN] <colonna> {RESTRICT | CASCADE}

ALTER TABLE Film

DROP COLUMN RESTRICT valutaz;

Modifica di relazioni

- Come vedremo meglio in seguito, tra le modifiche possibili vi sono anche l'aggiunta e la modifica di vincoli
  - ADD CONSTRAINT <specifica vincolo>
  - DROP CONSTRAINT <nome vincolo>

- Esempio:

  ALTER TABLE Film

  ADD CONSTRAINT UNIQUE(studio,titolo,anno)
  - aggiunge il vincolo che uno stesso studio non produca due film con lo stesso titolo nello stesso anno