Introduzione a SQL

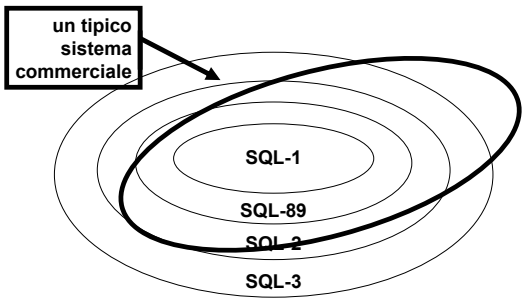
SQL

- Il nome sta per *Structured Query Language*
- Più che un semplice linguaggio di query: si compone di una parte DDL e di una DML
  - DDL: definizione di domini, tabelle, indici, autorizzazioni, viste, vincoli, procedure, trigger
  - DML: linguaggio di query, linguaggio di modifica, comandi transazionali
- Storia:
  - Prima proposta: SEQUEL (IBM Research, 1974)
  - Prima implementazione commerciale in SQL/DS (IBM, 1981)

Standardizzazione di SQL

- La standardizzazione è stata cruciale per la diffusione di SQL (nell'ambito di ANSI e ISO)
  - Dal 1983, standard de facto
  - Prima versione ufficiale nel 1986 (SQL-1), rivista nel 1989 (SQL-89)
  - Seconda versione nel 1992 (SQL-2 o SQL-92)
  - Terza versione nel 1999 (SQL-3 o SQL:1999)
- In SQL-92 si distinguono tre livelli:
  - Entry SQL (più o meno equivalente a SQL-89)
  - Intermediate SQL
  - Full SQL
- La maggior parte dei sistemi è conforme al livello Entry e offre delle estensioni proprietarie per le funzioni avanzate

Potere espressivo di standard e sistemi commerciali



Definizione degli schemi in SQL

Definizione di schemi

- Uno *schema* è una collezione di oggetti:
  - domini, tabelle, indici, asserzioni, viste, privilegi
- Uno schema ha un nome e un proprietario
- Sintassi:

```
create schema [ NomeSchema ]
  [ [ authorization ] Autorizzazione ]
  { DefinizioneElementoSchema }
```

Domini

- I domini specificano i valori ammissibili per gli attributi
  - Simili ai meccanismi di definizione dei tipi dei linguaggi di programmazione
- Due categorie
  - Elementari (predefiniti dallo standard, elementary o built-in)
    - SQL-2 prevede 6 famiglie
  - Definiti dall'utente (user-defined)

Domini elementari, 1

- Caratteri
  - Caratteri singoli o stringhe
  - Le stringhe possono avere lunghezza variabile
  - Possono usare una famiglia di caratteri (character set) diversa da quella di default (es., Latin, Greek, Cyrillic, etc.)
  - Sintassi:
    `character` [`varying`] [(*Lunghezza*)]
    [`character set` *FamigliaCaratteri*]
  - Si possono usare le alternative più compatte `char` e `varchar`, rispettivamente per `character` e `character varying`
  - Esempi:
    - `char(6)`
    - `varchar(50)`

Domini elementari, 2

- Bit
  - Valori booleani (vero/falso), singoli o in sequenza (la sequenza può essere di lunghezza variabile)
  - Sintassi:
    `bit` [`varying`] [(*Lunghezza*)]
    Esempi: bit(100), varbit(680)
- Domini numerici esatti
  - Valori esatti, interi o con una parte razionale
  - 4 alternative:
    `numeric` [(*Precisione* [, *Scala*])]
    `decimal` [(*Precisione* [, *Scala*])]
    `integer`
    `smallint`

Domini elementari, 3

- Domini numerici approssimati
  - Valori reali approssimati
  - Basati su una rappresentazione a virgola mobile: mantissa + esponente
    `float` [(*Precisione*)]
    `real`
    `double precision`

Domini elementari, 4

- Istanti temporali
  - Ammettono dei campi
    `date` (campi `month, day, year`)
    `time` [(*Precisione*)] [`with time zone`]: (campi `hour, minute, second`)
    `timestamp` [(*Precisione*)] [`with time zone`]
    con timezone si hanno i due ulteriori campi `timezone_hour` e `timezone_minute`
    - Esempio: timestamp(4) with time zone 2-30-2004 3-13-42.0564 5-30
- Intervalli temporali
  `interval` *PrimaUnitàDiTempo* [`to` *UltimaUnitàDiTempo*]
  - Le unità di tempo sono divise in 2 gruppi:
    - year, month
    - day, hour, minute, second
  - Esempi:
    - `interval year to month`
    - `interval second`

Domini elementari, 5

- Nuovi domini semplici introdotti in SQL:1999
  - `Boolean`
  - `BLOB` Binary Large OBject
  - `CLOB` Character Large Object
- SQL:1999 introduce anche dei costruttori (`REF`, `ARRAY`, `ROW`; vanno al di là del modello relazionale e non ne parliamo)

Domini definiti dagli utenti

- Paragonabile alla definizione dei tipi nei linguaggi di programmazione: si definiscono i valori ammissibili per un oggetto
- Un dominio è caratterizzato da
  - nome
  - dominio elementare
  - valore di default
  - insieme di vincoli (constraint)
- Sintassi:
  `create domain` *NomeDominio* `as` *DominioElementare*
  [*ValoreDefault*] [*Constraints*]
- Esempio:
  `create domain Voto as smallint default null`
- Rispetto ai linguaggi di programmazione
  + vincoli, valori di default, domini di base più ricchi
  - costruttori assenti (solo ridenominazione di domini)

Valori di default per il dominio

- Definiscono il valore che deve assumere l'attributo quando non viene specificato un valore durante l'inserimento di una tupla
- Sintassi:
  `default` < *ValoreGenerico* | `user` | `null` >
- *ValoreGenerico* rappresenta un valore compatibile con il dominio, rappresentato come una costante o come un'espressione
- `user` è la login dell'utente che effettua il comando

Il valore "null"

`null`
è un valore polimorfico (che appartiene a tutti i domini) col significato di valore non noto

- il valore esiste in realtà ma è ignoto al database
  (es.: data di nascita)
- il valore è inapplicabile
  (es.: numero patente per minorenni)
- non si sa se il valore è inapplicabile o meno
  (es.: numero patente per un maggiorenne)

Definizione dei domini applicativi

```
create domain PrezzoQuotidiani
as decimal(3)
     default 0,90
     not null
```

Definizione di tabelle

- Una tabella SQL consiste di:
  - un insieme ordinato di attributi
  - un insieme di vincoli (eventualmente vuoto)
- Comando `create table`
  - definisce lo schema di una relazione, creandone un'istanza vuota
- Sintassi:
  `create table` *NomeTabella*
  (
  NomeAttributo Dominio [*ValoreDiDefault*] [*Constraints*]
  {, *NomeAttributo Dominio* [*ValoreDiDefault*] [*Constraints*] }
  [*AltriConstraints*]
)

Esempio di **`create table`** (1)

```
create table Studente
  (  Matr      character(6) primary key,
     Nome      varchar(30) not null,
     Città     varchar(20),
     CDip      char(3) )
```

Esempio di **`create table`** (2)

```
create table Esame
 (  Matr       char(6),
    CodCorso   char(6),
    Data       date  not null,
    Voto       smallint not null,
    primary key(Matr,CodCorso) )

 create table Corso
 (  CodCorso char(6) primary key,
    Titolo   varchar(30) not null,
    Docente  varchar(20) )
```

Vincoli intra-relazionali

- I vincoli sono condizioni che devono essere verificate da ogni istanza della base di dati
- I vincoli intra-relazionali coinvolgono una sola relazione (distinguibili ulteriormente a livello di tupla o di tabella)
  - `not null` (su un solo attributo; a livello di tupla)
  - `unique`: permette la definizione di chiavi candidate (opera quindi a livello di tabella); sintassi:
    - per un solo attributo:
      `unique`, dopo il dominio
    - per diversi attributi:
      `unique`(*Attributo* {, *Attributo* })
  - `primary key`: definisce la chiave primaria (una volta per ogni tabella; implica *not null*); sintassi come per `unique`
  - `check`: descritto più avanti (può rappresentare vincoli di ogni tipo)

Esempi di vincoli intra-relazionali

- Ogni coppia di attributi Nome e Cognome identifica univocamente ogni tupla

```
Nome character(20) not null,
Cognome character(20) not null,
unique(Nome,Cognome)
```

- Si noti la differenza con la seguente definizione (più restrittiva):

```
Nome character(20) not null unique,
Cognome character(20) not null unique,
```

Integrita' referenziale

Esempio: studente - esame

Studente

Matr	
123	
415	
702	

Esame

Matr	
123	
123	
702	

Il problema degli orfani

Studente

Matr	
~~123~~	
415	
702	

orfani:
tuple che restano prive di padre a causa di cancellazioni
e modifiche della tabella padre

Esame

Matr	
~~123~~	
~~123~~	
702	

Gestione degli orfani

- Le reazioni operano sulla tabella interna, in seguito a modifiche alla tabella esterna
- Le violazioni possono essere introdotte (1) da aggiornamenti (update) dell'attributo cui si fa riferimento oppure (2) da cancellazioni di tuple
- Reazioni previste:
  - `cascade`: propaga la modifica
  - `set null`: annulla l'attributo che fa riferimento
  - `set default`: assegna il valore di default all'attributo
  - `no action`: impedisce che la modifica possa avvenire
- La reazione può dipendere dall'evento; sintassi:
  `on <delete|update>`
  `<cascade|set null|set default|no action>`

Gestione degli orfani: cancellazione

Cosa succede degli esami se si cancella uno studente?

- **`cascade`**
  si cancellano anche gli esami dello studente
- **`set null`**
  si pone a null la matricola dei relativi esami
- **`set default`**
  si pone al valore di default la matricola dei relativi esami
- **`no action`**
  si impedisce la cancellazione dello studente

Gestione degli orfani: modifica

Cosa succede degli esami se si modifica la matricola di uno studente?

- **`cascade`**
  si modifica la matricola degli esami dello studente
- **`set null`**
  si pone a null la matricola dei relativi esami
- **`set default`**
  si pone al valore di default la matricola dei relativi esami
- **`no action`**
  si impedisce la modifica della matricola dello studente

Sintassi per l'integrità referenziale

- Gli attributi descritti come foreign key nella tabella figlio devono presentare valori presenti come valori di chiave nella tabella padre
- `references` e `foreign key` per l'integrità referenziale; sintassi:
  - per un solo attributo
    `references` dopo il dominio
  - per diversi attributi
    `foreign key` (*Attributo* {, *Attributo* })
    `references` …

Definizione: nella tabella figlio

```
create table Esame
   ( ....
     ....
     foreign key Matr
       references Studente
        on delete cascade
        on update cascade )
```

Definizione: nella tabella figlio

```
create table Esame
   ( Matr char(6) references Studente
        on delete cascade
        on update cascade ,
     ……)
```

E' lecito essere figli di più padri

```
create table Esame
( ....
    primary key(Matr,CodCorso)
    foreign key Matr
      references Studente
        on delete cascade
        on update cascade
    foreign key CodCorso
      references Corso
        on delete no action
        on update no action )
```

Una istanza scorretta

Matr	Nome	Città	CDip
123			
415			
702			

Esame

Matr	Cod Corso	Data	Voto
123	1	7-9-97	30
123	2	8-1-98	28
~~123~~	~~2~~	~~1-8-97~~	~~28~~
702	2	7-9-97	20
~~702~~	~~1~~	~~NULL~~	~~NULL~~
~~714~~	~~1~~	~~7-9-97~~	~~28~~

viola la chiave

viola il NULL

viola la integrità referenziale

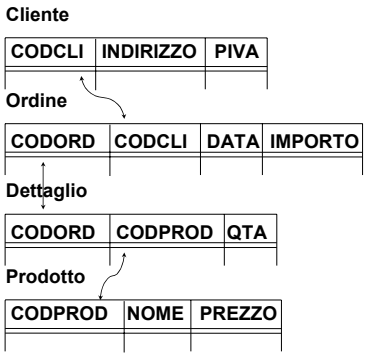
Una istanza corretta

Matr	Nome	Città	CDip
123			
415			
702			

Esame

Matr	Cod Corso	Data	Voto
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

Esempio: gestione ordini



Definizione della tabella Cliente

```
create table Cliente
 ( CodCli    char(6)  primary key,
   Indirizzo char(50),
   PIva      char(12) unique )
```

Definizione della tabella Ordine

```
create table Ordine
(  CodOrd   char(6)  primary key,
   CodCli   char(6)  not null
                     default='999999',
   Data     date,
   Importo  integer,
    foreign key CodCli
      references Cliente
         on delete set default
         on update set default)
```

Definizione della tabella Dettaglio

```
create table Dettaglio
(  CodOrd  char(6),
   CodProd char(6),
   Qta     smallint,
    primary key(CodOrd,CodProd)
    foreign key CodOrd
      references Ordine
        on delete cascade
        on update cascade
    foreign key CodProd
      references Prodotto
        on delete no action
        on update no action)
```

Definizione della tabella Prodotto

```
create table Prodotto
 ( CodProd   char(6)  primary key,
   Nome      char(20),
   Prezzo    smallint )
```

Creazione di indici

• Indici: meccanismi di accesso efficiente ai dati

```
create index
es.: create  index DataIx
       on Ordine(Data)

create unique index
es.: create unique index OrdKey
       on Ordine(CodOrd)
```

Modifica degli schemi

Modifiche degli schemi

- Necessarie per garantire l'evoluzione della base di dati a fronte di nuove esigenze
- Ci sono due comandi SQL appositi:
  - `alter` (`alter domain` ..., `alter table` …)
    modifica oggetti persistenti
  - `drop`
    `drop` < `schema` | `domain` | `table` | `view` | `assertion` >
    NomeComponente [`restrict` | `cascade`]
    cancella oggetti dallo schema

Modifica degli oggetti DDL

- **`alter`**
  - Si applica su domini e tabelle

```
es.: alter table Ordine
     add column NumFatt char(6)

es.: alter table Ordine
     alter column Importo
     add default 0

es.: alter table Ordine
     drop column Data
```

Cancellazione degli oggetti DDL

- **`drop`**
  - si applica su domini, tabelle, indici, view, asserzioni, procedure, trigger

    es.: **`drop table Ordine`**

    es.: **`drop index DataIx`**

- Opzioni `restrict` e `cascade`
  - **`restrict`**: impedisce `drop` se gli oggetti comprendono istanze
  - **`cascade`**: applica `drop` agli oggetti collegati

Cataloghi relazionali

- Il catalogo contiene il dizionario dei dati (data dictionary), ovvero la descrizione della struttura dei dati contenuti nel database
- È basato su una struttura relazionale
  - Il modello relazionale è senz'altro noto agli utenti del sistema
  - Il sistema è in grado di gestire tabelle in modo efficiente
  - Ogni sistema usa rappresenta il catalogo tramite il proprio modello dei dati (es.: sistemi ad oggetti avranno un catalogo con schema a oggetti)
- Lo standard SQL-2 organizza il catalogo su due livelli
  - **`Definition_Schema`** (composto da tabelle, non vincolante)
  - **`Information_Schema`** (composto da viste, vincolante)

Information Schema

- Nell'Information_Schema compaiono viste come:
  - Domains
  - Domain_Constraints
  - Tables
  - Views
  - Columns
  - ....
- SQL-2 prevede 23 viste

La vista Columns

- Ad esempio, la vista Columns può avere uno schema con attributi:
  - Table_Name
  - Column_Name
  - Ordinal_Position
  - Column_Default
  - Is_Nullable

  (più molti altri)

Riflessivita' del catalogo

- Il catalogo è normalmente riflessivo (le strutture del catalogo sono descritte nel catalogo stesso)
- Ogni comando DDL viene quindi realizzato da opportuni comandi DML che operano sullo schema della base di dati
- Non per questo il DDL è inutile!
- Il DDL permette di descrivere gli oggetti dello schema in modo
  - affidabile
  - consistente
  - efficiente
  - portabile
- Il catalogo non deve MAI essere modificato direttamente