lezione 19

Un linguaggio per le basi di dati relazionali

Il **linguaggio SQL** (*Structured Query Language*), di tipo non procedurale (o dichiarativo), è divenuto ormai da tempo il linguaggio standard per creare, manipolare e interrogare database relazionali.

Il linguaggio SQL assolve alle funzioni di:

- **DDL** (*Data Definition Language*), che prevede le istruzioni per definire la struttura delle relazioni della base di dati. Serve, quindi, a creare tabelle, vincoli, viste e così via;
- **DML** (*Data Manipulation Language*), che prevede le istruzioni per manipolare i dati contenuti nelle diverse tabelle; in particolare, permette inserimenti, cancellazioni e modifiche delle righe delle tabelle, nonché di effettuare interrogazioni sulle basi di dati;
- **DCL** (*Data Control Language*), che prevede istruzioni per controllare il modo in cui le operazioni vengono eseguite; consente di gestire il controllo degli accessi per più utenti e i permessi per gli utenti autorizzati.

SQL non è un linguaggio case-sensitive, pertanto le istruzioni possono essere scritte utilizzando, indifferentemente, caratteri maiuscoli o minuscoli. Ricordiamo, però, che esistono versioni per le quali è previsto un controllo di tipo case-sensitive per i nomi delle tabelle e per quelli delle variabili, mentre per i comandi non si fa differenza.
Le istruzioni vengono generalmente separate dal *";"* ma anche questo non accade per tutte le versioni. In questa unità, per garantire la migliore leggibilità, utilizzeremo caratteri maiuscoli per le parole chiave e per le tabelle.

Gli **identificatori** utilizzati per i nomi delle tabelle e degli attributi devono:

- avere una lunghezza massima di 18 caratteri;
- iniziare con una lettera;
- contenere come unico carattere speciale l'underscore *"_"*, che in molte versioni, comunque, può creare problemi quando si usano le espressioni regolari.

Nella terminologia SQL:

- le relazioni sono chiamate **tabelle**;
- le t-uple sono chiamate **righe** o **registrazioni**;
- gli attributi sono le **colonne** delle tabelle.

Per riferirsi a un attributo di una tabella si utilizza la seguente sintassi:

```
<NOMETABELLA>.<NomeAttributo>
```

I **tipi di dato** (o **domini** della tabella) utilizzabili per gli attributi sono riepilogati nella tabella, riportata a pagina seguente (occorre precisare che in alcune versioni di SQL tali tipi potrebbero essere differenti).
Le **costanti stringa** sono rappresentabili usando, indifferentemente, apici singoli (' ') o doppi (" "). Sono pertanto valide entrambe le stringhe seguenti:

"Stringa valida" 'Stringa valida'

Nelle **espressioni** possono essere usati i seguenti **operatori**:

- aritmetici (+, −, /, *);
- relazionali [<, >, <= (minore o uguale), >= (maggiore o uguale), <> (diverso)];
- logici (AND, OR, NOT).

Tipo	Descrizione	Esempi e range di variabilità
CHARACTER o CHAR	Singolo carattere.	"C", "5", "%", "c"
CHARACTER(N) o CHAR(N)	Stringa di caratteri di lunghezza fissa pari a N caratteri. CHAR(1) corrisponde a CHARACTER o CHAR.	N varia da 1 a 15000 Esempio di costante: "rty56"
CHARACTER VARYNG(N) o VARCHAR(N)	Stringa di caratteri di lunghezza variabile con dimensione massima pari a N.	VARCHAR(5) Esempio di costante: "rty56"
BIT	Singolo bit; definisce il tipo booleano.	0 (corrisponde a false o No) o 1 (corrisponde a true o SI)
BIT(N)	Stringa di bit di lunghezza fissa pari a N.	BIT(5) Esempio di costante: "01101"
BIT VARYNG(N)	Stringa di bit di lunghezza variabile avente dimensione massima pari a N.	BIT VARING(7) Esempio di costante: "0011011"
INTEGER o INT	Numero intero con precisione superiore a SMALLINT (generalmente occupano 4 byte, ma dipende dall'implementazione).	Generalmente da -2147483648 a +2147483647
INTEGER(N) o INT(N)	Numero intero con precisione N (numero massimo di cifre che il numero può contenere). Anche se non standard, è supportato dalla maggior parte delle implementazioni SQL.	INT(5)
SMALLINT	Numero intero con precisione inferiore a INTEGER (generalmente occupano 2 byte in memoria).	Generalmente da -32768 a +32767
DECIMAL(P,D) o DEC(P,D)	Numero reale in fixed point, con un numero massimo di cifre prima del punto decimale pari a P, e un numero massimo di cifre dopo il punto decimale pari a D. Le dimensioni massime di P e D sono definite dall'implementazione.	DECIMAL(6,2) Esempi di costanti sono: 100.3, +0.7, -.3
REAL	Numero reale in floating point con precisione definita dall'implementazione. Generalmente vale 7 per la mantissa.	Da 1E-38 a 1E+38. Esempi di costanti sono: 10E4 +24.7E-3 -7.897E+12
FLOAT	Numero reale in floating point con precisione definita dall'implementazione. Generalmente vale 15 per la mantissa.	Da 1E-38 a 1E+38. Stessi esempi dei REAL per le costanti
FLOAT(P)	Numero reale in floating point con precisione P per la mantissa.	La dimensione di P è definita dall'implementazione generalmente P varia da 1 a 45
DOUBLE PRECISION	Numero reale in floating point, con precisione per la mantissa doppia rispetto alla precisione di un FLOAT.	L'esponente generalmente varia da -127 a +128
DATE	Data nel formato "AAAA/MM/GG" oppure "AAAA-MM-GG".	"2006/12/25" "2006-12-25"
TIME	Ora nel formato ora, minuti, secondi e millisecondi.	"10:34:25:42"
TIMESTAMP	Data e orario nel formato anno, mese, giorno, ora, minuti, secondi e millisecondi.	"2006/12/25 10:34:25:42"

lezione 20

Istruzioni del DDL di SQL

I comandi del **DDL** di SQL creano o modificano lo schema di una base di dati.

Creare e selezionare un nuovo database

Per creare un nuovo database utilizzeremo il comando **CREATE DATABASE**, la cui sintassi è la seguente:

```
CREATE DATABASE <NomeDatabase> [AUTHORIZATION <Proprietario>];
```

Questo comando crea un nuovo database di nome <NomeDatabase> ed, eventualmente, specifica in <Proprietario> il nome dell'utente proprietario, cioè dell'utente che possiede i privilegi di accesso e che, come tale, è l'unico a poter svolgere determinate azioni sul database. Se non viene specificata la clausola **AUTHORIZATION**, si suppone che il nome del proprietario sia quello dell'utente collegato in quel momento. Per esempio:

```
CREATE DATABASE Negozio;
```

crea una nuovo database di nome *Negozio*. Il comando:

```
CREATE DATABASE Negozio AUTHORIZATION Venditore;
```

crea un nuovo database di nome *Negozio* e assegna i privilegi di accesso all'utente *Venditore*.
Per impartire i comandi SQL per uno specifico database occorre prima selezionare quest'ultimo, utilizzando il comando **USE**, la cui sintassi è:

```
USE <NomeDatabase>;
```

dove <NomeDatabase> è il nome del database creato con l'istruzione **CREATE DATABASE**. Nel nostro caso avremo:

```
USE Negozio;
```

Creare una tabella e i vincoli di integrità

Per creare una tabella si utilizza la seguente sintassi:

```
CREATE TABLE <NOMETABELLA>
   (<Attributo1> <Tipo1> [<VincoloAttributo1>],
   <Attributo2> <Tipo2> [<VincoloAttributo2>],
   ....
   <AttributoN> <TipoN> [<VincoloAttributoN>],
   [<VincoloTabella>]);
```

Nella definizione di una tabella sono presenti vincoli:

- per un singolo attributo;
- per un gruppo di attributi, detti anche **vincoli di ennupla**;
- per l'integrità referenziale.

I vincoli per un singolo attributo (detti anche **vincoli di dominio**) impostano limitazioni da specificare sui valori di un singolo attributo. Possono essere impostati attraverso le seguenti clausole:

- **NOT NULL**: se presente, richiede che il corrispondente attributo debba necessariamente avere un valore e, quindi, non può rimanere "non specificato" (valore *NULL*). Per esempio:

```
Stipendio DECIMAL(8,3) NOT NULL
```

- **DEFAULT** <ValoreDiDefault>: assegna all'attributo il valore di default (cioè predefinito) specificato in <ValoreDiDefault>. Per esempio:

```
Pensionato BIT DEFAULT 0
```

- **CHECK**(<Condizione>): serve per specificare un vincolo qualsiasi che riguarda il valore di un attributo. Per esempio:

```
CHECK(Stipendio > 1000)
```

impedisce che il valore di *Stipendio* sia inferiore a 1000.
All'interno di **CHECK** è possibile utilizzare, oltre agli operatori di confronto, anche i seguenti operatori:

<Attributo> **IN** (<Valore1>, ..., <ValoreN>)	Richiede che il valore di <Attributo> sia tra quelli specificati da: <Valore1>, ..., <ValoreN>
<Attributo> **BETWEEN** <Min> **AND** <Max>	Richiede che il valore di <Attributo> sia compreso tra i valori <Min> e <Max>
<Attributo> **NOT BETWEEN** <Min> **AND** <Max>	Richiede che il valore di <Attributo> non sia compreso tra i valori <Min> e <Max>
<Attributo> **LIKE** <Espressione1>	Richiede che il valore di <Attributo> assuma il formato specificato da <Espressione1>
<Attributo> **NOT LIKE** <Espressione1>	Richiede che il valore di <Attributo> non assuma il formato specificato da <Espressione1>

Per esempio, per richiedere che l'attributo *Stipendio* sia uguale a uno dei valori 1500, 2000, 2500 e 3000, scriveremo:

```
CHECK(Stipendio IN (1500, 2000, 2500, 3000))
```

Per specificare che l'attributo *Stipendio* sia compreso tra 1500 e 3000 scriveremo:

```
CHECK(Stipendio BETWEEN 1500 AND 3000)
```

Per specificare che l'attributo *CodiceArticolo* inizi con *Cod* scriveremo:

```
CHECK(CodiceArticolo LIKE "Cod%")
```

Il carattere "%" (**carattere jolly**) rappresenta una sequenza di zero o più caratteri.
Consideriamo la seguente relazione:

```
AZIENDA(CodAzienda, RagioneSociale, Fatturato, NumDipendenti)
```

Creiamo la relativa tabella imponendo i seguenti vincoli:

- i valori degli attributi *CodAzienda* e *RagioneSociale* devono essere sempre presenti;
- il valore di default del *Fatturato* è 1000000;
- il numero dei dipendenti deve essere compreso tra 5 e 200.

Scriveremo:

```
CREATE TABLE AZIENDA
(
   CodAzienda          CHAR(5)     NOT NULL,
   RagioneSociale      CHAR(30)    NOT NULL,
   Fatturato           INT(9)      DEFAULT 1000000,
   NumDipendenti       INT(5),
   CHECK(NumeroDip BETWEEN 5 AND 200)
);
```

Vincoli su attributi

Se i valori degli operatori **IN**, **BETWEEN**, **NOT BETWEEN** sono di tipo stringa, occorrerà racchiuderli tra apici o virgolette. Nel seguito, li utilizzeremo entrambi.

lezione 21

Vincoli di ennupla e di integrità

Per comprendere questi vincoli ci serviremo della precedente relazione AZIENDA, e della nuova relazione DIPENDENTE:

AZIENDA(CodAzienda, RagioneSociale, Fatturato, NumDipendenti)
DIPENDENTE(CodDip, Cognome, Nome, DataNascita, DataAssunzione, Livello, StipendioLordo, *CodAzienda*)

Vincoli di ennupla

I vincoli di ennupla impostano limitazioni da specificare sui valori di più attributi della stessa tabella. Possono essere impostati con le seguenti clausole:

- **PRIMARY KEY**(<Attributo1>, ..., <AttributoN>): indica le colonne che fanno parte della chiave primaria. Per esempio, per la tabella AZIENDA avremo:

```
PRIMARY KEY(CodAzienda)
```

e per la tabella DIPENDENTE:

```
PRIMARY KEY(CodDip)
```

stop

Gli attributi che compaiono nella clausola **PRIMARY KEY** devono anche essere dichiarati NOT NULL.

- **UNIQUE**(<Attributo1>, ..., <AttributoN>): indica che i valori degli attributi specificati in <Attributo1>, ..., <AttributoN> (che non formano una chiave primaria) devono essere necessariamente distinti all'interno della tabella (formano cioè una chiave candidata). Per esempio, considerando gli attributi *Cognome*, *Nome* e *DataAssunzione* della tabella DIPENDENTE, possiamo scrivere:

```
UNIQUE(Cognome, Nome, DataAssunzione)
```

- **CHECK**(<Condizione>): specifica un qualsiasi vincolo che riguarda il valore di più attributi della tabella. Per esempio, considerando gli attributi *DataNascita* e *DataAssunzione* della tabella DIPENDENTE possiamo scrivere:

```
CHECK(DataNascita < DataAssunzione)
```

Vincoli di integrità referenziale e politiche di violazione

I vincoli di integrità referenziale (analizzati nell'unità precedente) possono essere dichiarati con la seguente clausola:

```
FOREIGN KEY(<Attributo1>, ..., <AttributoN>)
REFERENCES <NOMETABELLA>(<Attr1>, ..., <AttrN>)
[[ON DELETE | ON UPDATE] CASCADE | SET NULL | SET DEFAULT | NO ACTION]
```

Le colonne <Attributo1>, ..., <AttributoN> rappresentano la chiave esterna e corrispondono alle colonne <Attr1>, ..., <AttrN> che formano la chiave primaria della tabella <NOMETABELLA>.
Non è indispensabile citare le colonne <Attr1>, ..., <AttrN> quando la chiave primaria della tabella <NOMETABELLA> è costituita da una sola colonna.

La parte successiva è relativa al tipo di politica da seguire in caso di violazione del vincolo referenziale. I tipi possibili sono: **RESTRICT**, **CASCADE**, **SET NULL**, **SET DEFAULT**, **NO ACTION**.
Al momento della **cancellazione** di un attributo interessato da un vincolo referenziale (cosa che si specifica attraverso la clausola ON DELETE), si ha il seguente comportamento, a seconda del tipo di politica specificato:

- con l'opzione **RESTRICT** l'azione si limita alla riga corrispondente;
- con l'opzione **CASCADE** vengono cancellate le righe corrispondenti in tutte le tabelle correlate;
- con l'opzione **SET NULL** vengono impostate a NULL le righe corrispondenti;
- con l'opzione **SET DEFAULT** le righe corrispondenti vengono impostate al valore di default;
- con l'opzione **NO ACTION** non viene eseguita alcuna azione. È questa l'impostazione di default se non viene specificata la clausola ON DELETE.

Al momento di una **modifica** di un attributo interessato da un vincolo referenziale (cosa che si specifica attraverso la clausola **ON UPDATE**), si ha il seguente comportamento, a seconda del tipo di politica specificato:

- con l'opzione **RESTRICT** l'azione si limita alla riga corrispondente;
- con l'opzione **CASCADE** tutte le righe corrispondenti vengono impostate con lo stesso nuovo valore;
- con l'opzione **SET NULL** le righe corrispondenti vengono impostate a NULL;
- con l'opzione **SET DEFAULT** le righe corrispondenti vengono impostate al valore di default;
- con l'opzione **NO ACTION** non viene eseguita alcuna azione. È questa l'impostazione di default, se non viene specificata la clausola ON UPDATE.

Esaminiamo, ora, le istruzioni SQL per la creazione delle due tabelle AZIENDA e DIPENDENTE, tenendo conto delle nuove clausole introdotte in questa lezione:

```
CREATE TABLE AZIENDA
(
   CodAzienda          CHAR(5)              NOT NULL,
   RagioneSociale      CHAR(30)             NOT NULL,
   Fatturato           DECIMAL(9,2)         DEFAULT 1000000.00,
   NumDipendenti       INT(5),
   PRIMARY KEY(CodAzienda),
   CHECK(NumeroDip BETWEEN 5 AND 200)
);
```

Vincolo di ennupla

```
CREATE TABLE DIPENDENTE
(
   CodDip              CHAR(6)              NOT NULL,
   Cognome             CHAR(30)             NOT NULL,
   Nome                CHAR(20)             NOT NULL,
   DataNascita         DATE,
   DataAssunzione      DATE,
   Livello             CHAR(1)              DEFAULT '6',
   StipendioLordo      DECIMAL(8,3)         NOT NULL,
   CodAzienda          CHAR(5)              NOT NULL,
   PRIMARY KEY(CodDip),
   UNIQUE(Cognome, Nome, DataAssunzione),
   CHECK(DataAssunzione > DataNascita),
   FOREIGN KEY(CodAzienda)          REFERENCES AZIENDA(CodAzienda)
         ON DELETE RESTRICT
         ON UPDATE CASCADE
);
```

Vincolo di integrità referenziale