26-10-2022

CREARE TABELLE

Per creare una tabella si usa l'istruzione CREATE TABLE. La sintassi è la seguente:

```
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)
)
```

dove:

- `PRIMARY KEY` indica **la chiave primaria (un solo attributo)** e implica *`NOT NULL`*. Per specificare la chiave primaria **con più attributi** basta specificarla alla fine e non accanto all'attributo
- `NOT NULL` indica che tale attributi non può avere record nulli;
- `REFERENCES` indica che è una **chiave esterna** (un solo attributo) che si riferisce ad un'attributo di un'altra tabella;
- `FOREIGN KEY`, indica la chiave esterna (più attributi), ed è seguita da *`REFERENCES`*
- `NUMERIC(9)` indica il numero avrà esattamente 9 cifre;
- `UNIQUE` definisce chiavi. Inoltre non ci possono essere duplicati nella colonna o gruppi di colonne;
- `CHECK` specifica i valori ammissibili; esempio: `Voto INTEGER NOT NULL CHECK (18 >= Voto AND Voto <=31)`; `CHECK Condizione` si può usare anche per attributi diversi della stessa n-upla;
- `DEFAULT(Costante|NULL)` assegna quel valore di default per ogni inserimento

Esempi:

1. Chiave esterna su un solo attributo: `Vigile INTEGER NOT NULL REFERENCES Vigili(Matricola)`
2. `FOREIGN KEY(Provincia, Numero) REFERENCES Auto(Provincia, Numero)`
   - **FOREIGN KEY** = quali attributi...
   - **REFERENCES** = fa riferimento a... Tabella(chiave primaria)

TIPI DI VARIABILI

I tipi di variabili più frequenti sono:

- `CHAR(n)` stringhe di esattamente n caratteri. Se sono di più, la stringa viene tagliata. Se sono di meno allora si fa un "*padding*" per arrivare a n, cioè si "*aggiungono degli 0*"
- `VARCHAR(n)` stringhe di lunghezza variabile ma al massimo n, dove n è opzionale. (*massimo specificabile 255 in SQL*). Se non si specifica nulla è come se si specifica `VARCHAR(255)`.
- `INTEGER` interi
- `REAL` reali
- `NUMERIC (p,s)` p cifre di cui s decimali
- `FLOAT(p)` es. 0.17E16. **Non permette di rappresentare tutti i possibili numeri con la virgola.** Sotto un determinato numero il computer li considera 0 (questo limite si chiama *epsilon-macchina*)
- `DATE,TIME` per date ed ore.

DOMINI ELEMENTARI

Per creare un dominio si utilizza l'istruzione `CREATE DOMAIN`.
Essa definisce un dominio (semplice), utilizzabile in definizioni di relazioni, anche con vincoli e valori di default.

> Sostanzialmente serve per **creare tipi personalizzati** con i relativi vincoli (creare **scorciatoie**) che si usano spesso per non invocare sempre un controllo costante. Quindi si utilizzano i domini primari.

Per esempio:

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

Vincoli di integrità

1. Permette di garantire che il database sia valido.
2. Devono essere sempre soddisfatti altrimenti la transazione fallisce.
3. Riguardano i valori ammissibili dagli attributi di una tupla oppure l'utente può opzionalmente definire delle azioni (correttive) da intraprendere per ripristinare l'integrità
   Esistono due tipi di vincoli:
   Vincoli Intrarelazionali: nell'ambito della stessa relazione;
   Vincoli Referenziali (o *Interrelazionali*): tra diverse relazioni (cioè chiavi esterne).

I vincoli di integrità vengono controllati durante le **tre possibili operazioni di modifica SQL**:

- `INSERT`, `DELETE` e `UPDATE`.

ALTRI COMANDI

- `ALTERTABLE` serve per aggiungere qualcosa alla definizione della tabella.
- `ON DELETE` cosa avviene quando nella tabella riferimento si cancella un record. Esso è seguito da:
  - `NO ACTION` rifiuta l'operazione;
  - `CASCADE`, cancella anche tutto quello che è associato alle n-uple;
  - `SET NULL`, mette a NULL tutte le multe associate al vigile (se può avere valori *NULL*)
  - `RESTRICT` impedisce l'eliminazione

  Per esempio:
  `Vigile INTEGER NOT NULL REFERENCES Vigili(Matricola) ON DELETE {NO ACTION,CASCADE,SET NULL}`

Generalizzando

```
CREATE TABLE Clienti (
        CodiceCliente CHAR(3) UNIQUE NOT NULL,
        Nome CHAR(30) NOT NULL,
        Citta' CHAR(30) NOT NULL,
        Sconto INTEGER NOT NULL CHECK(Sconto>0 AND Sconto<100),
        PRIMARY KEY pk_Clienti(CodiceCliente)
)

CREATE TABLE Agenti (
        CodiceAgente CHAR(3) UNIQUE NOT NULL,
        Nome CHAR(30) NOT NULL,
        Zona CHAR(8) NOT NULL,
        Supervisore CHAR(3),
        Commissione INTEGER)
        PRIMARY KEY pk_Agenti(CodiceAgente),
        CHECK (Supervisore = CodiceAgente OR Supervisore IS NULL
)

CREATE TABLE Ordini(
        NumOrdine CHAR(3) NOT NULL,
        CodiceCliente CHAR(3) NOT NULL,
        CodiceAgente CHAR(3) NOT NULL,
        Data CHAR(8) NOT NULL,
        Prodotto CHAR(3) NOT NULL,
        Ammontare INTEGER NOT NULL
        CHECK (Ammontare > 100)
        PRIMARY KEY pk-Ordini (NumOrdine)
        FOREIGN KEY fk_ClienteOrdine (CodiceCliente)
                REFERENCES Clienti ON DELETE NO ACTION
        FOREIGN KEY fk_AgenteOrdine (CodiceAgente)
                REFERENCES Agenti ON DELETE NO ACTION
)
```