EPICODE

Progettazione Logica e implementazione fisica di un DB

Agenda

- SQL
- Traduzione logica di un modello concettuale
- Implementazione fisica di una base dati

Cosa imparerai

- ✔ I primi costrutti DDL del linguaggio SQL
- ✔ Implementare tabelle in SQL
- ✔ Garantire l’integrità referenziale

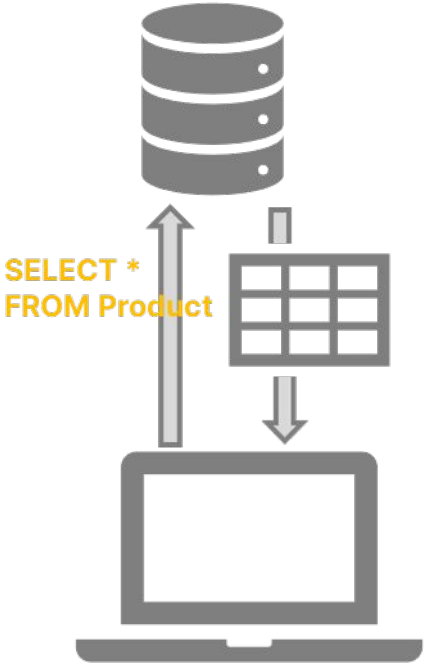
SQL

- SQL è l'acronimo di Structured Query Language.
- SQL è un linguaggio progettato per interrogare basi dati relazionali.
- SQL è stato ideato dall'informatico statunitense Donald Chamberline nei laboratori dell'IBM (1974).

- SQL è un linguaggio dichiarativo: si specifica cosa si vuole ottenere cioè l'output desiderato, mentre 'dietro le quinte' un motore di elaborazione sviluppa un piano di esecuzione per recuperare i risultati!

Le istruzioni SQL possono essere classificate nelle seguenti categorie:

- **DDL – Data Definition Language**, comprende le istruzioni utili a definire (creare, modificare, eliminare) oggetti (tabelle per esempio).
- **DML – Data Manipulation Language**, comprende le istruzioni per interrogare e modificare i dati all'interno di tabelle.
- **DCL – Data Control Language**, comprende le istruzioni per attribuire o revocare permessi agli utenti che accedono alla base dati (permessi di scrittura e/o lettura).



SQL: DATA TYPE

NUMERIC Data Type

Type	Storage (Bytes)	Minimum Value Signed	Maximum Value Signed
INT	4	-2147483648	2147483647
SMALLINT	2	-32768	32767
TINYINT(MYSQL)	1	-128	127
TINYINT (SQL SERVER)	1	0	255

SQL: DATA TYPE

NUMERIC Data Type

DECIMAL

- Consente di memorizzare valori numerici esatti.
- Utile quando è necessario garantire la precisione esatta.
- Il DECIMAL è implementato dichiarando una precisione *p* e una scala *s*.
- Precisione *p* e scala *s* indicando rispettivamente il numero totale di digit che devono essere archiviati e il numero di cifre memorizzabili dopo il separatore decimale (in altre parole, *p* = numero totale di digit e *s* = numero di decimali).

Ad esempio, un campo DECIMAL(5,2) consente di archiviare valori come 125.55; in generale, un campo DECIMAL(5,2) può memorizzare un range di valori che va da -999.99 a 999.99

SQL: DATA TYPE

CHAR e VARCHAR Data Type

CHAR

- CHAR sta per CHARACTER è utilizzato per la gestione di stringhe alfanumeriche a lunghezza fissa (fixed-lenght string).
- Occupa 1 bytes per ogni carattere.
- Il data type CHAR è dichiarato con una lunghezza che indica il numero massimo di caratteri ammissibili.
- Lo spazio occupato è quello dichiarato a prescindere dal numero di carattere effettivi (... fixed-lenght!).
- Ha senso utilizzare il CHAR quando ci si aspetta che i valori archiviati dalla colonna abbiano la stessa lunghezza.

Ad esempio se si dichiarasse un campo CHAR(5), le stringhe ‘ABC’ e ‘ABCDE’ occuperebbero entrambe 5 bytes

SQL: DATA TYPE

```
CREATE TABLE TestChar (CharColumn CHAR(5));
INSERT INTO TestChar (CharColumn)VALUES ('ABC'), ('ABCDE');
SET sql_mode = 'PAD_CHAR_TO_FULL_LENGTH';
SELECT char_length(CharColumn) FROM TestChar;
```

SQL: DATA TYPE

CHAR e VARCHAR Data Type

VARCHAR

- VARCHAR sta per VARIABLE CHARACTER è utilizzato per la gestione di stringhe alfanumeriche a lunghezza variabile (variable-lenght string).
- Occupa 1 bytes per ogni carattere.
- Il data type VARCHAR è dichiarato con una lunghezza che indica il numero massimo di caratteri ammissibili.
- Lo spazio occupato è dato dal numero di carattere effettivi.
- Ha senso utilizzare il VARCHAR quando ci si aspetta che i valori archiviati dalla colonna NON abbiano la stessa lunghezza.

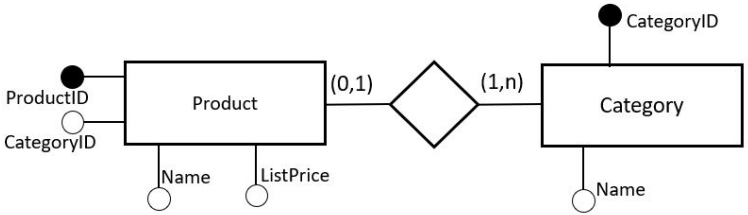
Ad esempio se si dichiarasse un campo VARCHAR(5), le stringhe ‘ABC’ e ‘ABCDE’ occuperebbero rispettivamente 3 e 5 bytes.

SQL: DATA TYPE

```
CREATE TABLE TestVARChar (VARCharColumn VARCHAR(5));
INSERT INTO TestVARChar (VARCharColumn)VALUES ('ABC'), ('ABCDE’);
SELECT char_length(VARCharColumn)
FROM TestVARChar;
```

SQL

Consideriamo il seguente schema E/R



.. e la sua modellazione logica

Product

ProductID	Name	ListPrice	CategoryID
706	HL Road Frame	1431,5	18
715	Jersey, L	49,99	25
836	ML Road Frame	594,83	18
875	Racing Socks, L	8,99	27
877	Blade	null	null
881	Classic Jersey, S	53,99	25

* → 1

Category

CategoryID	Name
18	Road Frames
25	Jerseys
27	Socks

L'implementazione fisica cioè la creazione delle tabelle potrebbe essere la seguente:

```
CREATE TABLE Product (
ProductID INT
, ProductName VARCHAR(25)
, ListPrice DECIMAL(10,2)
, CategoryID INT)

CREATE TABLE Category (
CategoryID INT
, CategoryName VARCHAR(25))
```

La sintassi generale è quindi:

```
CREATE TABLE <table_name> (
<column1> datatype
, <column2> datatype
, <column3> datatype
, ...)
```

Attenzione l'implementazione non è completa (mancano le relazioni.. ci arriviamo a breve!!)

SQL

.. Popoliamo le tabelle

```
CREATE TABLE Product (
ProductID INT
, ProductName VARCHAR(25)
, ListPrice DECIMAL(10,2)
, CategoryID INT)

CREATE TABLE Category (
CategoryID INT
, CategoryName VARCHAR(25))
```

```
INSERT INTO Product VALUES
(706, 'HL Road Frame', '1431.5', '18')
, (715, 'Jersey, L', '49.99', '26')
, (706,'ML Road Frame', '594.83', '18’)

INSERT INTO Category VALUES
(18, 'Road Frames')
,(25, 'Jersey')
,(27, 'Socks')
```

Product

ProductID	Name	ListPrice	CategoryID

Category

CategoryID	Name

Violazione del vincolo di PK: valori duplicati

Violazione del vincolo di FK

Product

ProductID	Name	ListPrice	CategoryID
706	HL Road Frame	1431.50	18
715	Jersey, L	49.99	26
706	ML Road Frame	594.83	18

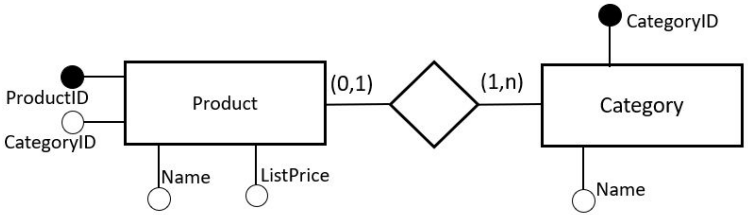
Category

CategoryID	Name
18	Road Frames
25	Jersey
27	Socks

È possibile inserire valori duplicati nel campo che dovrebbe essere PK.
È possibile inserire valori inconsistenti nel campo che dovrebbe invece garantire l’integrità referenziale! La base implementata non è affidabile e non consente di gestire correttamente le operazione di data entry (non modella correttamente la realtà!)

SQL

Consideriamo il seguente schema E/R



.. e la sua modellazione logica

Product

ProductID	Name	ListPrice	CategoryID
706	HL Road Frame	1431,5	18
715	Jersey, L	49,99	25
836	ML Road Frame	594,83	18
875	Racing Socks, L	8,99	27
877	Blade	null	null
881	Classic Jersey, S	53,99	25

* → 1

Category

CategoryID	Name
18	Road Frames
25	Jerseys
27	Socks

L'implementazione fisica cioè la creazione delle tabelle potrebbe essere la seguente:

```
CREATE TABLE Product (
ProductID INT
, ProductName VARCHAR(25)
, ListPrice DECIMAL(10,2)
, CategoryID INT
, CONSTRAINT PK_Product PRIMARY KEY (ProductID)
, CONSTRAINT FK_Category_Product FOREIGN KEY (CategoryID)
REFERENCES Category (CategoryID))

CREATE TABLE Category (
CategoryID INT
, CategoryName VARCHAR(25)
, CONSTRAINT PK_Category PRIMARY KEY (CategoryID))
```

SQL

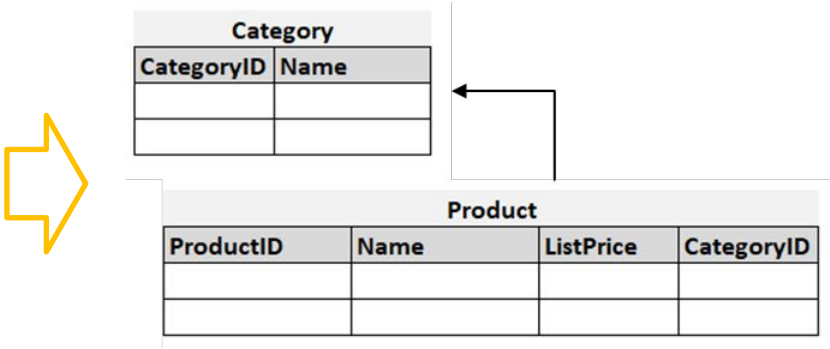
.. Popoliamo le tabelle

```
CREATE TABLE Product (
ProductID INT
, ProductName VARCHAR(25)
, ListPrice DECIMAL(10,2)
, CategoryID INT
, CONSTRAINT PK_Product PRIMARY KEY (ProductID)
, CONSTRAINT FK_Category_Product FOREIGN KEY (CategoryID)
REFERENCES Category (CategoryID))

CREATE TABLE Category (
CategoryID INT
, CategoryName VARCHAR(25)
, CONSTRAINT PK_Category PRIMARY KEY (CategoryID))
```

```
INSERT INTO Product VALUES
(706, 'HL Road Frame', '1431.5', '18')
, (715, 'Jersey, L', '49.99', '26')
, (706,'ML Road Frame', '594.83', '18’)

INSERT INTO Category VALUES
(18, 'Road Frames')
,(25, 'Jersey')
,(27, 'Socks')
```




```
Msg 547, Level 16, State 0, Line 16
The INSERT statement conflicted with the FOREIGN KEY
constraint "FK_Category_Product". The conflict occurred in
database "sample", table "dbo.Category", column
'CategoryID'.

Msg 2627, Level 14, State 1, Line 16
Violation of PRIMARY KEY constraint 'PK_Product'. Cannot
insert duplicate key in object 'dbo.Product'. The
duplicate key value is (706).
```

SQL

LET'S TAKE A LOOK!

EPICODE

GRAZIE

Epicode