

ISIS PAOLO SARPI
San Vito al Tagliamento
prof. Cristian Virgili

SQL

Lezione 1

Structured Query Language

Il linguaggio SQL (Structured Query Language, Linguaggio di interrogazione strutturato) consente di interagire con i dati presenti nelle basi di dati.

SQL può essere suddiviso in quattro sotto linguaggi.

- **DDL**-Data Definition Language (Linguaggio di definizione dei dati): è la parte del linguaggio SQL che comprende tutti i comandi preposti alla creazione di tavole e alla definizione dei dati in esse contenuti. Mediante questo linguaggio si definiscono/modificano le strutture delle relazioni dello schema della base di dati (livello intensionale).
- **DML**-Data Manipulation Language (Linguaggio di manipolazione dei dati): è la parte del linguaggio contenente tutti i comandi per la modifica dei dati contenuti nelle tavole, in termini di aggiunta, eliminazione, modifica, aggiornamento dei dati (livello estensionale).

Structured Query Language

- **DCL**-Data Control Language: appartengono a questa sezione di SQL tutti i comandi che rendono possibile la gestione dei permessi di accesso alle risorse del database.
- **QL**-Query Language: rappresenta il fulcro di quasi tutte le attività legate ai database, in quanto i comandi appartenenti a questa sezione di SQL consentono l'interrogazione, il raggruppamento, il conteggio e l'ottenimento di prospetti personalizzati dei dati presenti nelle varie tavole dei database.

SQL

Le tabelle relazionali sono insiemi e le righe delle tabelle possono essere considerate come elementi dell'insieme.

Le operazioni che si possono eseguire sugli insiemi possono essere, pertanto, applicate anche alle tabelle relazionali;

SQL

Le operazioni fondamentali sono sette: **selezione; proiezione; intersezione; congiunzione; differenza; prodotto; unione.**

Il linguaggio di manipolazione dei dati (DML) permette di realizzare operazioni e interrogazioni anche molto complesse mediante il costrutto SELECT

Il costrutto SELECT

```
SELECT <elenco attributi>  
FROM <elenco tabelle>  
WHERE <condizione>
```

Il costrutto SELECT

<elenco attributi>: è un elenco dei nomi degli attributi i cui valori devono essere reperiti dall'interrogazione;

<elenco tavelle>: è un elenco dei nomi delle relazioni necessarie per eseguire l'interrogazione;

<condizione>: è un'espressione condizionale che identifica la condizione che devono avere le tuple desiderate dall'interrogazione

Esempio

Il costrutto DISTINCT e ORDER BY

La clausola DISTINCT è un modificatore del comando SELECT.
Permette di evitare duplicazioni dei risultati

SELECT DISTINCT titolo FROM LIBRI;

Possiamo anche richiedere un ordinamento

SELECT titolo FROM LIBRI ORDER BY titolo ASC; (DESC)

Gli operatori di confronto

campo | espressione

op. di confronto

campo | espressione | costante

Operatore	Significato
=	Uguale
≠	Diverso
<	Minore
>	Maggiore
≤	Minore o uguale
≥	Maggiore o uguale
Like	Confronto su stringhe
Between	Compreso tra due valori

Between e Like

L'operatore Between verifica se l'espressione è maggiore o uguale al valore minimo e minore o uguale al valore massimo.

La sintassi è la seguente.

```
espressione Between valore minimo AND valore massimo
```

L'operatore Like è usato quando si devono esprimere riferimenti a insiemi di caratteri. Tali riferimenti si ottengono utilizzando uno dei caratteri % (percento) oppure ? (punto di domanda), detti wildcard o caratteri jolly. Il carattere % indica “qualsiasi carattere”, mentre il carattere ? indica “qualsiasi carattere, uno per ciascun punto di domanda”.

La sintassi dei due operatori è la seguente.

```
LIKE carattere%
```

DISTINCT

La clausola Distinct serve a **non ripetere nei risultati della SELECT** quelli con lo stesso valore.

```
SELECT DISTINCT nome  
FROM alunni
```

GROUP BY

La clausola GROUP BY di SQL è utilizzata nelle operazioni di SELECT al fine di raggruppare i valori identici presenti in una o più colonne.

GROUP BY

```
SELECT colonna1, colonna2, ...
      FROM nome_tabella
      WHERE condizione
GROUP BY colonna1, colonna2, ...
```

```
SELECT nome FROM alunni
      Group by nome
```

Le funzioni di aggregazione

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una tabella; a tale scopo si utilizzano delle funzioni speciali che restituiscono un solo valore, e come tali concorrono a creare un'unica riga.

COUNT

La funzione COUNT conta il numero di righe presenti in una tabella, la cardinalità di una relazione.

```
SELECT COUNT(Codice) As 'Alunni Classe IIA'  
      FROM Alunni  
     WHERE Classe = 'IIA';
```

SUM

La funzione SUM restituisce la somma di tutti i valori contenuti in una colonna, naturalmente di tipo numerico, specificata come argomento della funzione.

```
SELECT SUM (Stipendio) AS 'Totale_Stipendi'  
FROM Dipendenti;
```

AVG

La funzione AVG (dall'inglese Average) calcola la media aritmetica dei valori numerici contenuti in una determinata colonna di una tabella

```
SELECT AVG(Voto)  
      FROM Alunni  
     WHERE Classe = 'IIA';
```

MIN e MAX

Le funzioni MIN e MAX restituiscono rispettivamente il valore minimo e il valore massimo tra i valori della colonna, anche di tipo carattere, specificata come argomento della funzione

```
SELECT MIN(Voto) AS 'Voto  
Minimo'  
FROM Alunni  
WHERE Classe = 'IIA';
```

```
SELECT MAX(Cognome)  
FROM Alunni  
WHERE Classe = 'IIA';
```

HAVING

HAVING , utilizzata assieme a GROUP BY, consente di impostare un filtro sui valori raggruppati.

SELECT ...
GROUP BY attributo
HAVING condizione

SELECT Città, AVG(Età)
FROM Studenti GROUP BY Città
HAVING AVG(Età)<21

DDL

Creazione di un DB

CREATE DATABASE *nome_database*;

Eliminare un DB

DROP DATABASE *nome_database*;

DDL/2

Creazione di una TABELLA

```
CREATE TABLE nomeTabella (  
    nomeAttributo1 tipo [valoreDefault][vincoli],  
    nomeAttributo2 tipo [valoreDefault][vincoli], ... );
```

dove per ciascun attributo si deve indicare il nome e il suo dominio e, facoltativamente, un eventuale valore di default e i vincoli ai quali deve sottostare.

ESEMPIO

```
CREATE TABLE alunni(  
matricola char(4) PRIMARY KEY,  
nome char(20),  
cognome char(40),  
indirizzo char(80)  
)
```

ELIMINAZIONE DI UNA TABELLA

DROP TABLE nome_tabella;

ATTENZIONE ai vincoli di integrità

DDL 3

TIPI DI DATI

Numerici:

NUMERI INTERI

possono essere di dimensioni differenti, dipendenti dalla rappresentazione interna del calcolatore.

– INTEGER, INT, SMALLINT, BIGINT

È anche possibile indicare l'assenza di segno con la clausola UNSIGNED.

Per i numeri interi l'aggiunta della keyword AUTO_INCREMENT consente di creare dei campi numerici che si autoincrementano a ogni nuovo inserimento nella tabella, particolarmente utilizzati nelle chiavi artificiali.

– INTEGER AUTO_INCREMENT

– SMALLINT AUTO_INCREMENT

Numeri reali

Numeri reali con dimensionamento fisso: è possibile definire attributi numerici in cui vengono specificati il numero di cifre totali e le eventuali cifre decimali:

- **NUMERIC(n,d)** oppure **DECIMAL(n,d)**

Esempio

- **NUMERIC(4)**// numeri interi tra -9999 e + 9999
- **NUMERIC(4.1)** // numeri reali tra -999.9 e + 999.9
- **NUMERIC(4.2)** // numeri reali tra -99.99 e + 99.99Numeri reali a virgola mobile
- **FLOAT** oppure **REAL** // precisione singola
- **DOUBLE** // precisione doppia

STRINGHE DI CARATTERI

CHAR(n): definisce attributi con lunghezza **fissa** di n caratteri;

VARCHAR(n): definisce attributi con lunghezza **variabile** con un numero massimo n di caratteri.

Nel tipo VARCHAR l'occupazione totale di memoria richiede un byte aggiuntivo utilizzato dal DBMS come prefisso.

STRINGHE DI BIT

Le stringhe di bit possono essere di lunghezza fissa oppure variabile, con dimensione minima di un bit:

BIT (n): definisce attributi con lunghezza fissa di n bit;

BIT VARYING (n): definisce attributi con lunghezza variabile con un numero massimo n di bit

DATE

DATE: ha 10 posizioni composta da tre parti, cioè ha componenti ‘AAAA-MM-GG’;

TIME: ha almeno 8 posizioni strutturato con componenti ‘hh:mm:ss’;

TIMESTAMP: comprende entrambi, cioè ha componenti ‘AAAAMMGGhhmmss’, estremamente importante per aggiungere le marche temporali fondamentali per tutti i record che richiedono una validazione con data certa.

BOOLEANI

BOOLEAN, che può assumere i valori TRUE e FALSE.

BLOB e CLOB

Domini e tipi di dati per documenti di grandi dimensioni (introdotti in SQL-3) Sono stati anch'essi introdotti con la versione SQL-3 per soddisfare le esigenze di avere nei record componenti multimediali in modo da rappresentare oggetti di grandi dimensioni costituiti da una sequenza arbitraria di valori binari o di caratteri:

- **BLOB** // binary large object
- **CLOB** // character large object

Su questi tipi di dati NON è però permesso effettuare alcun tipo di operazione oltre la memorizzazione del dato, cioè non possono essere effettuati confronti o elaborazioni mediante l'SQL, operazioni che richiedono linguaggi specifici in base alla natura dell'oggetto. Indicazione dei vincoli di base in SQL

Vincoli di chiave e di integrità referenziale intrarelazionale

Valori di default, NULL, NOT NULL, CHECK, UNIQUE

Innanzitutto, per ciascun attributo è possibile indicare quale deve essere il suo valore di DEFAULT, che può essere sia un valore qualunque del suo dominio, sia il valore NOT NULL: dove non è specificato il valore di default è NULL

```
CREATE TABLE Interrogazioni(  
ID INT NOT NULL AUTO_INCREMENT,  
ID_alunno INT NOT NULL //chiave esterna,  
data_interrogazione date,  
voto INT default 1 CHECK(value >=1 and value <=10  
)
```

unique

Oltre all'indicazione del valore su di un attributo è anche possibile porre il vincolo di unicità con la clausola **UNIQUE**, riferita a un singolo campo oppure a più campi: ciò permette di definire attributi che identificano la tupla che, quindi, diventano una chiave candidata per quella tabella.

```
CREATE TABLE dipartimento(  
    nome_dipartimento char(10) UNIQUE  
)
```

```
CREATE TABLE docenti(  
    cognome char(30),  
    nome char(20),  
UNIQUE (nome, cognome)  
)
```

Primary Key

```
CREATE TABLE docenti(  
ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
nome char(20) NOT_NULL //chiave esterna,  
cognome char(30),  
stipendio numeric(9) default 0  
)
```

A differenza di UNIQUE e NOT NULL che possono essere definiti su più attributi della stessa tabella, **il vincolo PRIMARY KEY deve essere unico nella tabella**

Primary Key

CREATE TABLE Interrogazioni(

ID_studente INT,

ID_materia INT,

data_interrogazione date,

voto INT default 1

PRIMARY KEY (id_studente, id_materia,data_interrogazione)

)

A differenza di UNIQUE e NOT NULL che possono essere definiti su più attributi della stessa tabella, **il vincolo PRIMARY KEY deve essere unico nella tabella**

Vincoli Interrelazini (Foreign Key)

I vincoli interrelazionali sono quelli esistenti tra due differenti relazioni e, sostanzialmente, riguardano le chiavi esterne; le due relazioni si chiamano rispettivamente:

esterna: tabella contenente l'insieme delle chiavi, dove è definito l'attributo;

interna: tabella contenente la chiave FOREIGN KEY (FK) di collegamento alla tabella esterna, che prende appunto il nome di chiave esterna.

Ci sono due notazioni che permettono di definire questo vincolo e utilizzano le clausole **REFERENCES** e **FOREIGN KEY**; a seconda del caso, il vincolo di integrità referenziale riguarda uno o più di un attributo delle tabelle interne.

Vincoli Interrelazini (Foreign Key)

Nella situazione che segue mettiamo in relazione i **dipartimenti** (tabella interna) con i **docenti** (esterna).

The diagram illustrates a foreign key relationship. A blue box labeled "chiave esterna" (foreign key) has two arrows pointing down to the "dipartimento" column in the "docenti" table and the "nome_dipartimento" column in the "dipartimenti" table.

docenti			
ID_docente	cognome	nome	dipartimento
123	Russo	Antonio	Informatica
132	Verdi	Gino	Fisica
321	Rossi	Mario	Matematica

dipartimenti	
nome_dipartimento	
Informatica	
Fisica	
Matematica	

Tabella interna

Tabella esterna

La codifica è la seguente:

```
1 CREATE TABLE dipartimenti(  
2   nome_dipartimento CHAR(15) PRIMARY KEY  
3 )
```

```
1 CREATE TABLE docenti(  
2   ID_docente INT NOT NULL PRIMARY KEY,  
3   nome CHAR(20) NOT NULL,  
4   cognome CHAR(20) NOT NULL,  
5   dipartimento CHAR(15) REFERENCES dipartimenti(nome_dipartimento)  
6 )  
7
```

Vincoli Interrelazionali (Foreign Key)



La codifica è la seguente:

```
1 CREATE TABLE corsi(
2   classe INT,
3   sezione CHAR(4),
4   nome_corso CHAR(25),
5   PRIMARY KEY(classe, sezione)
6 )
```

```
1 CREATE TABLE studenti(
2   matricola CHAR(20) PRIMARY KEY,
3   nome VARCHAR(20),
4   cognome VARCHAR(20),
5   data_nascita DATE,
6   classe INT,
7   sez CHAR(4),
8   FOREIGN KEY(classe, sez) REFERENCES corsi(classe, sez)
9 )
```

Cancellazione di una istanza di un attributo

Nel ciclo di vita del database potrebbero essere effettuate azioni sui dati di una tabella esterna, cioè in una tupla i dati potrebbero essere modificati (**UPDATE**) oppure potrebbe essere cancellato l'intero record (**DELETE**).

Se, per esempio, venisse cancellato il dipartimento di fisica, quale conseguenza ne deriverebbe per la **chiave esterna** presente nella tabella **docenti**?

docenti			
ID_docente	cognome	nome	dipartimento
123	Russo	Antonio	Fisica
132	Verdi	Gino	Informatica
321	Rossi	Mario	Informatica

Tabella interna

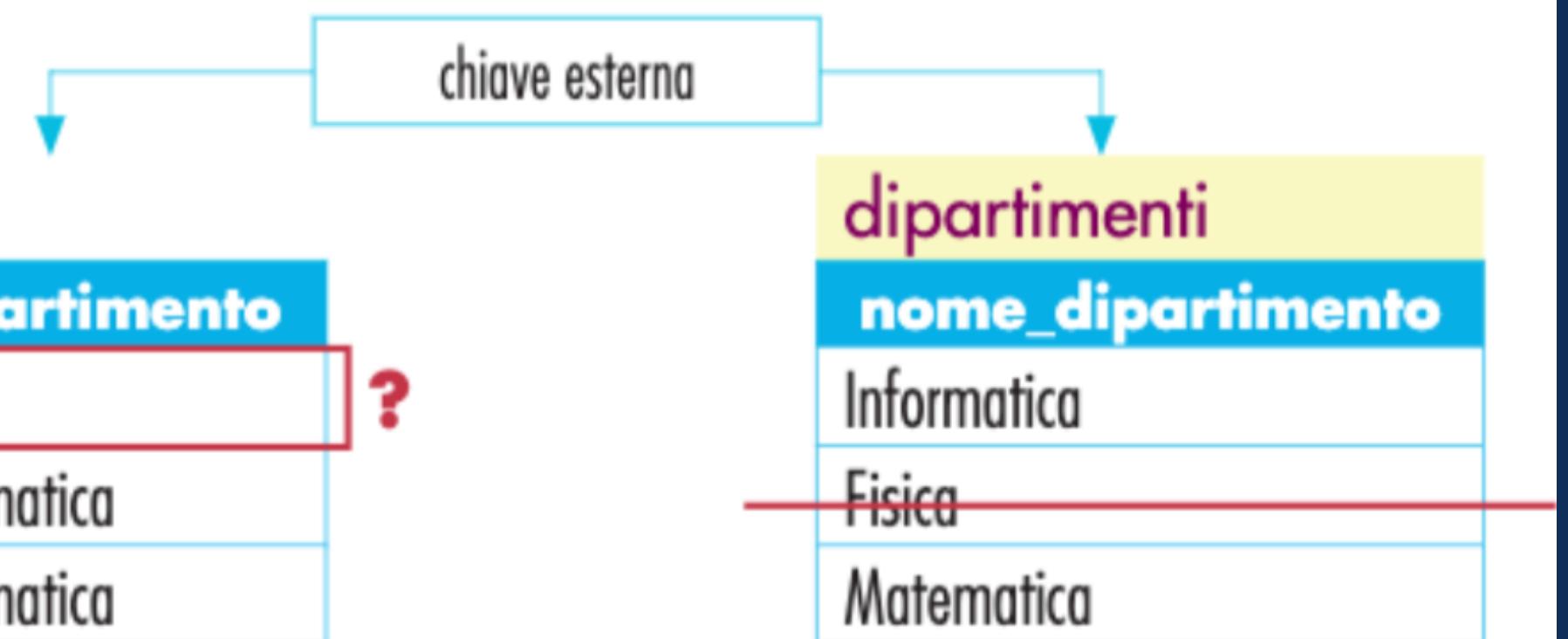


Tabella esterna

Cancellazione di una istanza di un attributo

Nel DBMS è possibile impostare un'operazione alternativa da effettuare in caso di violazione del vincolo di integrità referenziale nella tabella interna, scegliendola tra le seguenti opzioni:

SET NULL: pone uguale a NULL la chiave esterna;

CASCADE: esegue la medesima operazione (UPDATE/DELETE) sui record connessi;

SET DEFAULT: ripristina i valori di default;

NO ACTION: nessuna azione

Cancellazione di una istanza di un attributo

```
1 CREATE TABLE docenti(
2   ID_docente INT NOT NULL PRIMARY KEY,
3   nome CHAR(20) NOT NULL,
4   cognome CHAR(20) NOT NULL,
5   dipartimento CHAR(15) REFERENCES dipartimenti(nome_dipartimento)
6     ON DELETE SET DEFAULT ON UPDATE CASCADE
7 )
```

```
1 CREATE TABLE studenti(
2   matricola CHAR(20) PRIMARY KEY,
3   nome VARCHAR(20),
4   cognome VARCHAR(20),
5   data_nascita DATE,
6   classe INT,
7   sez CHAR(4),
8   FOREIGN KEY(classe, sez) REFERENCES corsi(classe, sez)
9     ON DELETE SET DEFAULT ON UPDATE SET DEFAULT
10 )
```

A. Cancellazione con vincolo SET NULL

Effettuando la cancellazione del record **Fisica** nella tabella **dipartimenti** automaticamente il **DBMS** provvede ad aggiornare tutti i riferimenti a esso presenti nelle chiavi esterne sostituendo in essi il valore **NULL**.

docenti			
ID_docente	cognome	nome	dipartimento
123	Russo	Antonio	NULL
132	Verdi	Gino	Informatica
321	Rossi	Mario	Informatica

Tabella interna

chiave esterna

dipartimenti	
nome_dipartimento	
Informatica	
Fisica	
Matematica	

Tabella esterna

B. Cancellazione/modifica con vincolo CASCADE

In questo caso effettuando la cancellazione del record **Fisica** nella tabella **dipartimenti** automaticamente il **DBMS** provvede a cancellare tutti i record che hanno una chiave esterna che contiene il valore rimosso.

docenti			
ID_docente	cognome	nome	dipartimento
123	Russo	Antonio	Fisica
132	Verdi	Gino	Informatica
321	Rossi	Mario	Informatica

Tabella interna

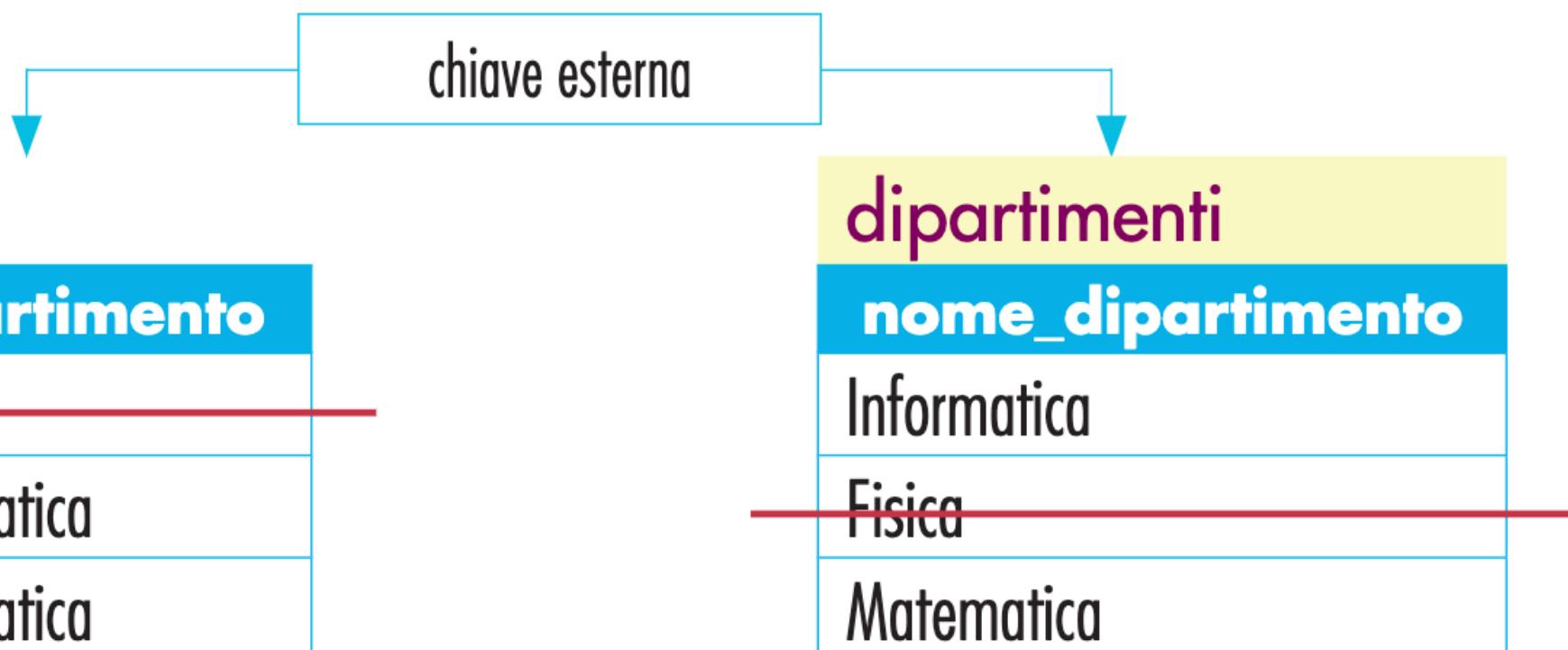


Tabella esterna

C. Cancellazione con vincolo SET DEFAULT

In questo caso, effettuando la cancellazione del record **Fisica** nella tabella **dipartimenti**, automaticamente il **DBMS** provvede a inserire in tutti i record che hanno una chiave esterna che contiene il valore rimosso un valore definito all'atto della creazione della tabella: per gli attributi alfanumerici spesso è il campo vuoto, cioè “”.

docenti			
ID_docente	cognome	nome	dipartimento
123	Russo	Antonio	—
132	Verdi	Gino	Informatica
321	Rossi	Mario	Informatica

Tabella interna

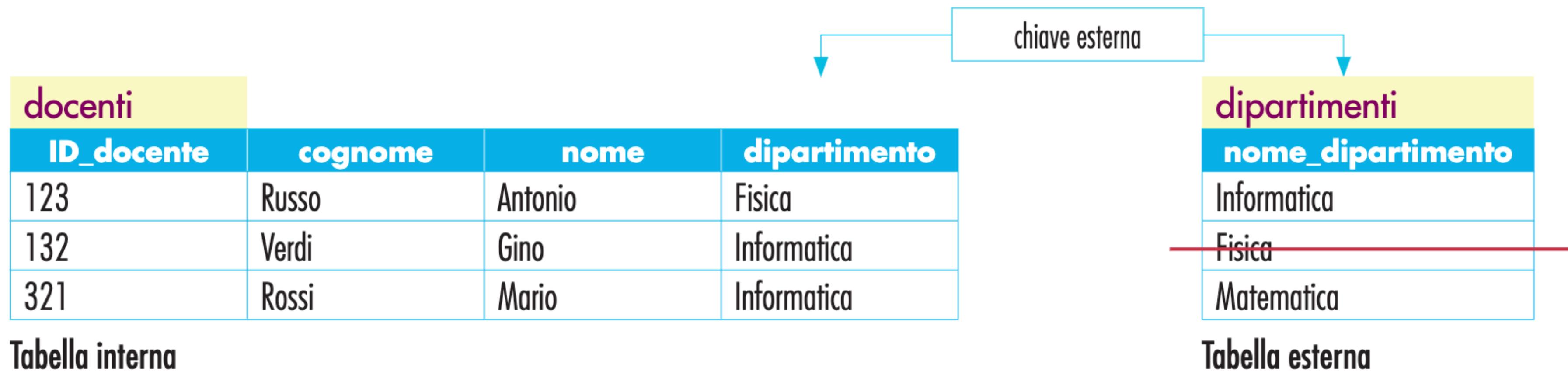
chiave esterna

dipartimenti	
nome_dipartimento	
Informatica	
Fisica	
Matematica	

Tabella esterna

D. Cancellazione con vincolo NO ACTION

In questo caso non vengono fatti controlli sulle altre tabelle, lasciandole invariate, quindi la cancellazione del record **Fisica** nella tabella **dipartimenti** non provoca modifiche nella tabella interna.

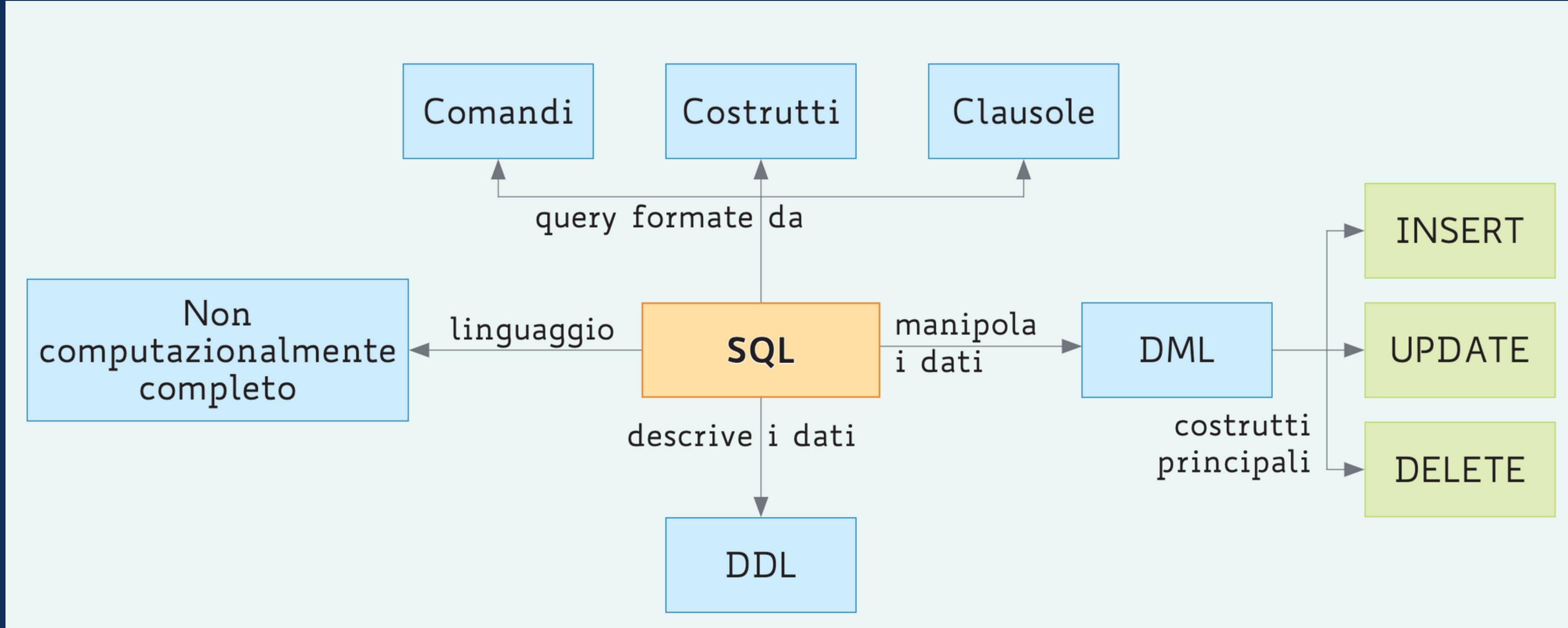


UPDATE

Per le operazione di aggiornamento/inserimento verranno applicate apposite le regole di inserzione:

- Inserzione dipendente
- Inserzione automatica
- Inserzione di default
- Inserzione nulla
- nessun effetto

DML Data Manipulation Language



DML Data Manipulation Language

■ Modifiche agli schemi e alle tabelle con il DDL

Il linguaggio **DDL** viene anche utilizzato per portare successive modifiche agli schemi e alle tabelle, per esempio con la primitiva **ALTER** per effettuare **modifiche**, **DROP** per effettuare **cancellazioni** e **ADD** per **aggiungere** colonne/vincoli.

```
ALTER (schema|table) <nome_elemento>          /* operazioni di modifica */  
DROP (schema|table|column) <nome_elemento>        /* operazioni di eliminazione */  
ADD column <nome_attributo>                      /* aggiunta di un attributo */  
ADD constraint <definizione_vincolo>            /* aggiunta di un vincolo */
```

Le operazioni di **modifica** su di una colonna della tabella (**ALTER**) riguardano la possibilità di:

- modificare il tipo di dato;
- cambiare la dimensione del campo;
- aggiungere il **constraint NOT NULL**.

Queste istruzioni vengono effettuate digitandole direttamente al prompt di comandi oppure nelle apposite sezioni **SQL** presenti in ogni interfaccia grafica di gestione del **DMBS**.

Riprendiamo la tabella **dipartimenti** e modifichiamo la dimensione del campo **nome_dipartimento**.

```
1 ALTER TABLE dipartimenti  
2   CHANGE COLUMN nome_dipartimento nome_dipartimento CHAR(20) NOT NULL FIRST;  
3
```

```
1 ALTER TABLE studenti  
2   CHANGE COLUMN cognome cognome VARCHAR(20) NOT NULL AFTER nome;
```

```
1 ALTER TABLE docenti  
2   DROP COLUMN data_nascita          /* cancella la colonna */  
3
```

È possibile cancellare il campo solo se su di esso non sono presenti **constraint**.

Aggiungiamo ora una colonna alla tabella **docenti**.

```
1 ALTER TABLE docenti  
2   ADD COLUMN data_nascita DATE NOT NULL;      /* aggiunge la colonna */
```

Aggiungiamo due attributi **BOOLEAN** alla tabella **studenti**.

```
1 ALTER TABLE studenti
2   ADD COLUMN ripetente TINYINT(1) DEFAULT 0      /* aggiunge la colonna BOOL */
```

```
1 ALTER TABLE studenti
2   ADD COLUMN maggiorenne TINYINT(1) DEFAULT 0      /* aggiunge la colonna BOOL */
```

Possiamo inoltre rimuovere un'intera tabella con la seguente istruzione.

```
1 DROP TABLE studenti
2
```

DML

I costrutti principali sono i seguenti:

INSERT: per effettuare le operazioni di inserimento di nuovi record;

DELETE: per effettuare l'eliminazione di record;

UPDATE: per effettuare le operazioni di aggiornamento del contenuto dei record presenti;

SELECT: per effettuare le operazioni di ricerca ed estrazione dei dati

INSERIMENTO DATI

INSERT INTO tabella [(lista attributi)] **VALUES** (lista valori)

ESEMPIO

```
CREATE TABLE docenti(  
ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
nome char(20) NOT_NULL //chiave esterna,  
cognome char(30),  
stipendio numeric(9) default 0  
)
```

INSERT INTO docenti (cognome, nome, dipartimento) VALUES
(**'Virgili'**,**'Cristian'**,**'Informatica'**)

ELIMINAZIONE DATI

DELETE FROM tabella **WHERE** <condizione>;

ESEMPIO

DELETE FROM docenti **WHERE** ID=1;

Per la cancellazione di tuple delle tabelle esterne valgono le regole definite per rispettare i vincoli di integrità referenziale.

AGGIORNAMENTO DATI

UPDATE tabella SET attributo = valore o espressione;

ESEMPIO

UPDATE docenti SET nome='Pietro' WHERE ID=1;

Per l'aggiornamento di tuple delle tabelle esterne valgono le regole definite per rispettare i vincoli di integrità referenziale.

