



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il Linguaggio SQL-DDL

Basi di Dati

Corso di Laurea in Informatica per il Management

Alma Mater Studiorum - Università di Bologna

Prof. Marco Di Felice

Dipartimento di Informatica – Scienza e Ingegneria

marco.difelice3@unibo.it



Linguaggi per DBMS

Il **modello relazionale** definisce i **concetti generali** ed i **vincoli** per modellare e strutturare i dati di una certa applicazione.

D. Come implementare il modello relazionale in un RDBMS?

D.1 Come costruire lo schema del DB?

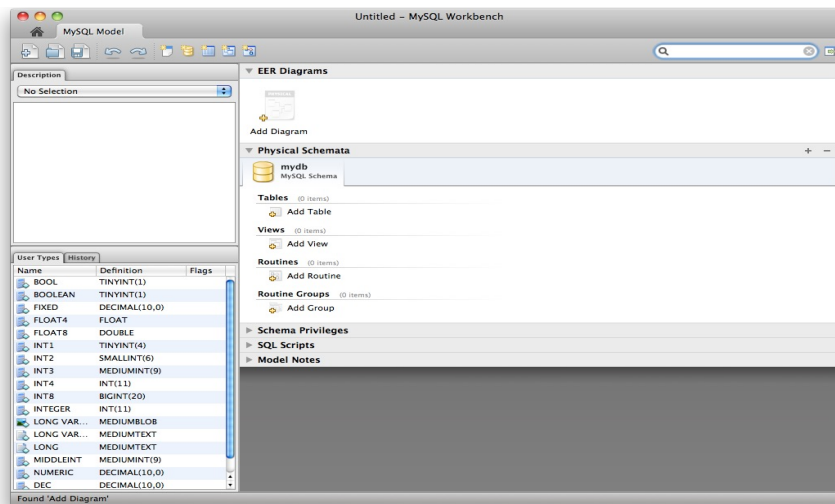
D.2 Come manipolare le istanze?

R. Attraverso opportuni **linguaggi** data-oriented!

Linguaggi per DBMS

- **LINGUAGGI** supportati dai **RDBMS**

1. Interfacce grafiche



- Creare un nuovo DB
- Creare tabelle
- Definire vincoli
- Inserire istanze
- Rimuovere istanze
- Cercare istanze
- ...

Linguaggi per DBMS

- **LINGUAGGI** supportati dai **RDBMS**

2. Linguaggi basati sulle proprietà algebrico/logiche

- **Calcolo relazionale** sui domini
- **Algebra relazionale**

$$\Pi_{A_1 A_2 \dots A_n} (\sigma_{Condizione} (T_1 \bowtie T_2 \bowtie \dots \bowtie T_m))$$

Il Linguaggio SQL

- **LINGUAGGI** supportati dai **RDBMS**

3. SQL (*Structured Query Language*)

Diverse **versioni** del linguaggio:

- SQL-86 → Costrutti base
- SQL-89 → (**SQL1**) Integrità referenziale
- SQL-92 (**SQL2**) → SQL Interattivo, sistema tipi
- SQL:1999 (**SQL3**) → Modello ad oggetti
- SQL:2003 (**SQL3**) → Nuove parti: SQL/JRT, SQL/XML
- SQL:2006 (**SQL3**) → Estensione di SQL/XML
- SQL:2008 (**SQL3**) → Lievi aggiunte
-

<http://troels.arvin.dk/db/rdbms/>

Il Linguaggio SQL

SQL è un linguaggio per basi di dati basate sul **modello relazionale**.

Valgono i concetti generali del modello relazionale visto fin qui,
ma con qualche differenza:

- Si parla di **tabelle** (e non relazioni).
- Il risultato di un'operazione sui dati può restituire una tabella con **righe duplicate**.
- Il sistema dei **vincoli** è **più espressivo**.
- Il vincolo di **integrità referenziale** (chiave esterna) è **meno stringente**.

Il Linguaggio SQL

- RDBMS possono implementare **sottoinsiemi differenti** del linguaggio SQL standard, o usare **keyword differenti** per implementare la stessa operazione.

ES. Comando di **EXCEPT**

RDMBS	Supportato (SI/NO)	Differenze
ORACLE	SI	Si chiama MINUS
MS SQL SERVER	SI	Come nello standard
DB2	SI	Come nello standard
SQLite	SI	Come nello standard
MYSQL	NO	Non supportato

Il Linguaggio SQL

- RDBMS possono implementare "dialetti" del linguaggio SQL che ne estendono le funzionalità attraverso le **estensioni procedurali**.

RDMBS	Nome estensione
ORACLE	PL/SQL
MS SQL SERVER	Transact-SQL
DB2	SQL PL
PostgreSQL	PL/pgSQL
MYSQL	MYSQL (like the DBMS)
...	...

NOTA. In queste slide (e nelle prossime), faremo riferimento ai costrutti del linguaggio SQL2 standard...

Il Linguaggio SQL

Due **componenti** principali:



- **DDL** (*Data Definition Language*)
Contiene i costrutti necessari per la creazione e modifica dello **schema** della base di dati.
- **DML/DQL** (*Data Manipulation/Query Language*)
Contiene i costrutti per le interrogazioni e di inserimento, eliminazione e modifica di **dati**.

Il Linguaggio SQL

Esempi di costrutti:

- Creazione/rimozione di database
- Creazione/rimozione/aggiornamento di tabelle
- Creazione/rimozione/aggiornamento di vincoli
- ...

} **SQL**
DDL

- Creazione/rimozione/aggiornamento di istanze (righe)
- Ricerca (query) di istanze
- ...

} **SQL**
DML/
DQL

SQL: DDL

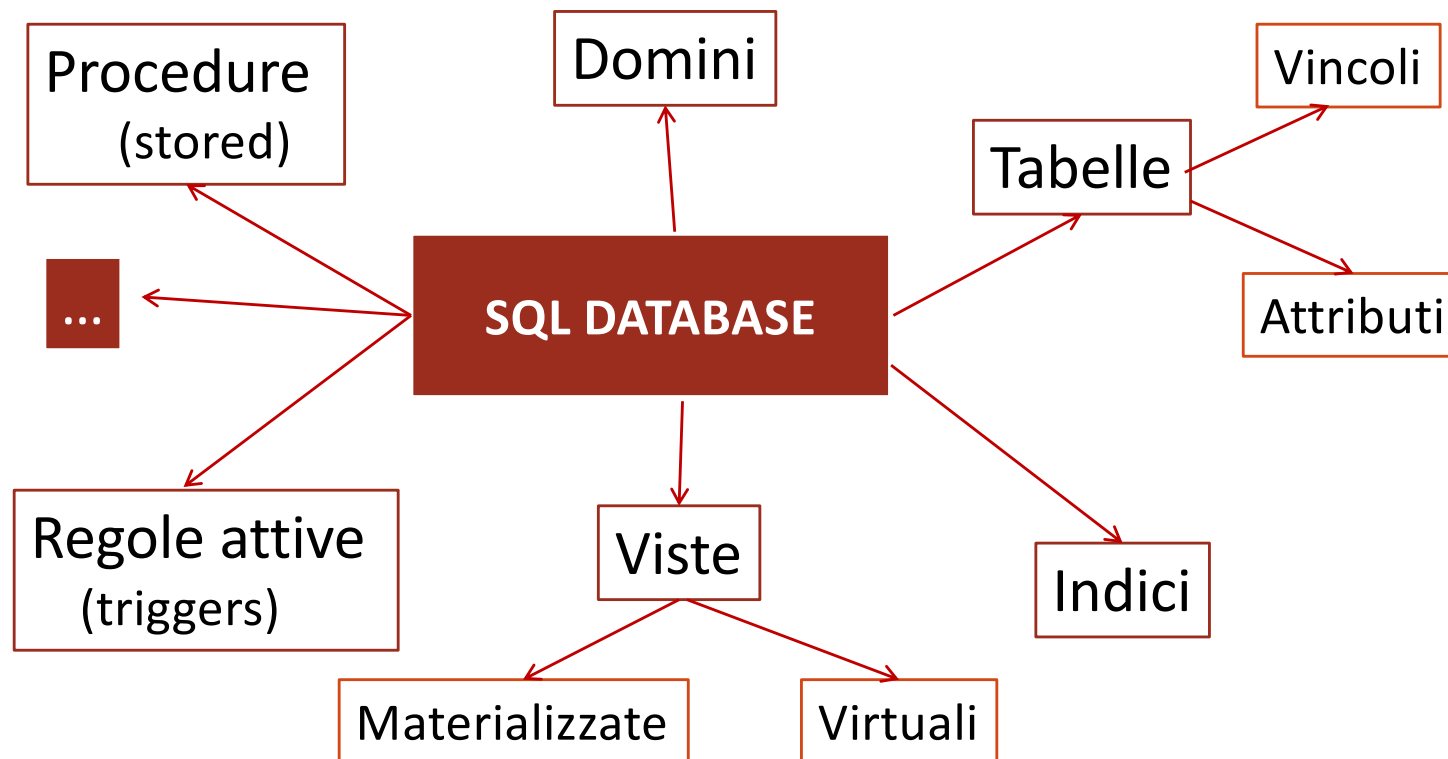
Tramite il costrutto `create database`, è possibile costruire uno **schema** di una base di dati (ossia il collettore di tabelle/viste/etc).

```
create database [if not exists] NomeDB  
drop database [if exists] NomeDB
```

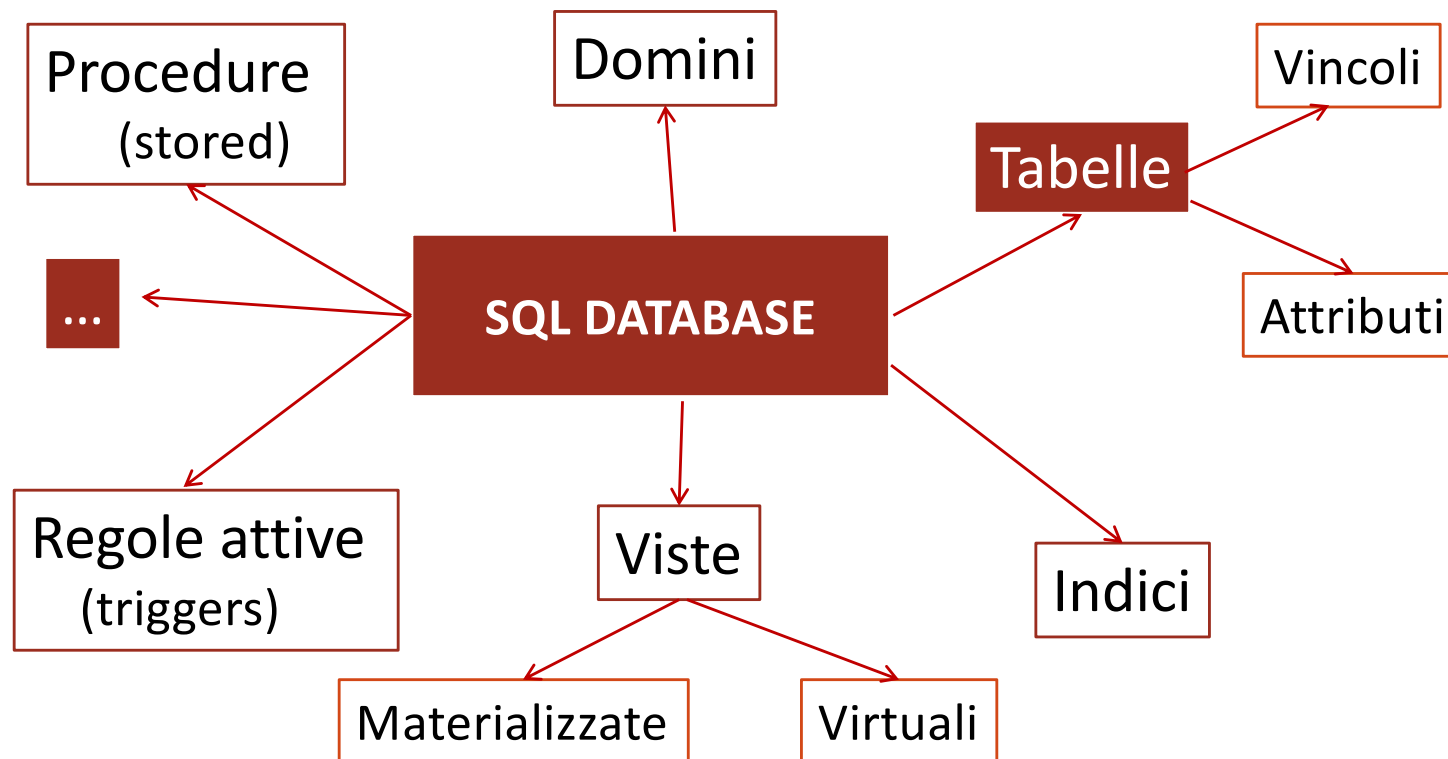
- `if exists` verifica l'esistenza o meno del database nel DBMS.

```
CREATE DATABASE IF NOT EXISTS DB-UNIBO  
DROP DATABASE IF EXISTS DB-UNIBO
```

SQL: DDL



SQL: DDL



SQL: DDL

Tramite il costrutto **create table**, è possibile costruire una **tabella** all'interno dello schema.

```
create table NomeTabella (  
    nomeAttributo1 Dominio [ValDefault][Vincoli],  
    nomeAttributo2 Dominio [ValDefault][Vincoli],  
    ...  
);
```

Per ciascun attributo, è possibile specificare, oltre al **nome** e **dominio**, un **valore di default** e i **vincoli**.

SQL: Domini elementari

In SQL, è possibile associare i seguenti **domini** (**elementari**) agli attributi di uno schema.

- Caratteri
- Tipi numerici esatti
- Tipi numerici approssimati
- Istanti temporali
- Intervalli temporali
- ...

SQL: Domini elementari

Il dominio **character** consente di rappresentare singoli caratteri o stringhe di lunghezza max fissa.

character/char [**varying**][(Lunghezza)]

Lunghezza non specificata → Singolo carattere

Es. specificare una stringa di max 20 caratteri.

- character varying (20)
- varchar (20)

SQL: Domini elementari

I tipi **numerici esatti** consentono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata.

- `numeric [(Precisione[, Scala])]`
- `decimal [(Precisione[, Scala])]`
- `integer`
- `smallint`

Es. `numeric(4,2)` → Intervallo [-99.99:99.99]

SQL: Domini elementari

La keyword **auto_increment** consente di creare dei campi numerici che si auto-incrementano ad ogni nuovo inserimento nella tabella.

- integer **auto_increment**
- smallint **auto_increment**

SQL: Domini elementari

I tipi **numerici approssimati** consentono di rappresentare valori reali con rappresentazione in virgola mobile.

- `float [(Precisione)]`
- `real`
- `double precision`

Es. `float(5)` → Mantissa di lunghezza 5.

SQL: Domini elementari

I **domini temporali** consentono di rappresentare informazioni temporali o intervalli di tempo.

- `date [(Precisione)]`
- `time [(Precisione)]`
- `timestamp`

Es. `time (2)` → 21:03:04
`time (4)` → 21:03:04:34

SQL: Domini elementari

I **domini temporali** consentono di rappresentare informazioni temporali o intervalli di tempo.

- interval PrimaUnità [to UltimaUnità]

Es. interval month to second

Il dominio boolean consente di rappresentare valori di verità (**true/false**).

SQL: Domini elementari

I **domini blob e clob** consentono di rappresentare oggetti di grandi dimensioni come sequenza di valori binari (blob) o di caratteri (clob) .

- La **dimensione massima** del file dipende dalle specifiche implementazioni (es. MySQL 4GB).
- Non è possibile specificare interrogazioni sui dati in base al valore del dominio ...

SQL: Domini elementari

Tramite il costrutto domain, l'utente può costruire un proprio **dominio di dati** a partire dai domini elementari.

```
create domain NomeDominio as TipoDati  
    [Valore di default]  
    [Vincolo] (vedremo dopo)
```

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

SQL: DDL

CORSI

Corso	Codice	NumeroOre	DataInizio
Basi di dati	0121	80	26/09/2012

```
CREATE TABLE CORSI (  
    CORSO VARCHAR(20),  
    CODICE VARCHAR(4),  
    NUMEROORE SMALLINT,  
    DATAINIZIO DATE  
);
```


SQL: DDL

Per ciascun dominio o attributo, è possibile specificare un **valore di default** attraverso il costrutto default.

```
default [valore | user | null]
```

- valore indica un **valore del dominio**.
- user è **l'id dell'utente** che esegue il comando.
- null è il **valore null**.

SQL: DDL

CORSI

Corso	Codice	NumeroOre	DataInizio
Basi di dati	0121	80	26/09/2012

```
CREATE TABLE CORSI (  
    CORSO VARCHAR(20),  
    CODICE VARCHAR(4),  
    NUMEROORE SMALLINT DEFAULT 40,  
    DATAINIZIO DATE  
);
```


SQL: DDL

CORSI

Corso	Codice	NumeroOre	DataInizio
Basi di dati	0121	80	26/09/2012

```
CREATE DOMAIN ORELEZIONE AS SMALLINT DEFAULT 40
```

```
CREATE TABLE CORSI (  
  CORSO VARCHAR(20),  
  CODICE VARCHAR(4),  
  NUMEROORE ORELEZIONE,  
  DATAINIZIO DATE  
);
```



SQL: DDL

Per ciascun dominio o attributo, è possibile definire dei **vincoli** che devono essere rispettati da tutte le istanze di quel dominio o attributo.

- Vincoli **intra-relazionali**
 - vincoli **generici di ennupla**
 - vincolo **not null**
 - vincolo **unique**
 - vincolo **primary key**
- Vincoli **inter-relazionali**
 - vincolo **references**

SQL: DDL

Mediante la clausola **check** è possibile esprimere vincoli di **ennupla** arbitrari.

NomeAttributo ... **check** (Condizione)

VOTO SMALLINT **CHECK((VOTO>=18) and (VOTO<=30))**

- Il vincolo viene valutato **ennupla per ennupla**.
- E' possibile creare vincoli più complessi mediante le **asserzioni**.

SQL: DDL

IMPIEGATI

Codice	Nome	Cognome	Ufficio
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
  CODICE SMALLINT CHECK (CODICE >=0),  
  NOME VARCHAR(30),  
  COGNOME VARCHAR(30),  
  UFFICIO CHARACTER  
);
```

SQL: DDL

Il **vincolo** `not null` indica che il **valore null non è ammesso** come valore dell'attributo.

Es. `NUMEROORE SMALLINT NOT NULL`

- In caso di inserimento, l'attributo deve essere specificato, a meno che non sia stato specificato un valore di default diverso dal valore null.

Es. `NUMEROORE SMALLINT DEFAULT 40`

SQL: DDL

Il **vincolo** `unique` impone che l'attributo/attributi su cui sia applica non presenti valori comuni in righe differenti → ossia che l'attributo/i sia una **superchiave della tabella**.

Due sintassi:

- `Attributo Dominio [ValDefault] unique`
Se la superchiave è **un solo attributo**.
- `unique(Attributo1, Attributo2, ..)`
Se la superchiave è **composta da piu' attributi**.

SQL: DDL

IMPIEGATI

Codice	Nome	Cognome	Ufficio
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT UNIQUE,  
    NOME VARCHAR(30),  
    COGNOME VARCHAR(30),  
    UFFICIO CHARACTER  
)
```

SQL: DDL

Violazione del vincolo di chiave!

IMPIEGATI

Codice	Nome	Cognome	Ufficio
145	Michele	Micheli	B
145	Giovanni	Di Giovanni	B
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT UNIQUE,  
    ...  
)
```

SQL: DDL

IMPIEGATI

NON sono violazioni del vincolo di chiave! NULL<>NULL

Codice	Nome	Cognome	Ufficio
NULL	Michele	Micheli	B
NULL	Giovanni	Di Giovanni	B
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT UNIQUE,  
    ...  
)
```

SQL: DDL

Esempio: Superchiave composta da **due attributi**.

IMPIEGATI

Codice	Nome	Cognome	Ufficio
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT NOT NULL,  
    UFFICIO CHARACTER NOT NULL,  
    UNIQUE(CODICE, UFFICIO)  
)
```

SQL: DDL

IMPIEGATI

Codice	Nome	Cognome	Ufficio
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT NOT NULL UNIQUE,  
    UFFICIO CHARACTER NOT NULL UNIQUE,  
)
```

ATTENZIONE, I DUE CODICI **NON** SONO EQUIVALENTI!!! (perche'?)

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT NOT NULL,  
    UFFICIO CHARACTER NOT NULL,  
    UNIQUE(CODICE, UFFICIO)  
)
```

SQL: DDL

Il **vincolo** primary key impone che l'attributo/attributi su cui sia applica non presenti valori comuni in righe differenti e non assuma valori NULL → ossia che l'attributo/i sia una **chiave primaria**.

Due sintassi:

- Attributo Dominio [ValDefault] primary key
Se la chiave è **un solo attributo**.
- primary key(Attributo1, Attributo2, ..)
Se la chiave è **composta da piu' attributi**.

SQL: DDL

Il **vincolo primary key** impone che l'attributo/attributi su cui sia applica non presenti valori comuni in righe differenti e non assuma valori NULL → ossia che l'attributo/i sia una **chiave primaria**.

IMPORTANTE: A differenza di unique e not null che possono essere definiti su più attributi della stessa tabella, il vincolo **primary key** deve apparire una sola volta per tabella.

SQL: DDL

Esempio: Chiave composta da **due attributi**.

IMPIEGATI

Codice	Nome	Cognome	Ufficio
123	Marco	Marchi	A

```
CREATE TABLE IMPIEGATI (  
    CODICE SMALLINT NOT NULL,  
    UFFICIO CHARACTER NOT NULL,  
    PRIMARY KEY (CODICE, UFFICIO)  
)
```


SQL: DDL

Esempio: Chiave composta da **un solo attributo**.

MAGAZZINO			
Articolo	Tipo	Marca	Prezzo
1	PC	HP	19000

```
CREATE TABLE IMPIEGATI (  
    ARTICOLO INTEGER NOT NULL  
    AUTO_INCREMENT PRIMARY KEY,  
    ...  
)
```

SQL: DDL

I vincoli references e foreign key consentono di definire dei **vincoli di integrità referenziale** tra i valori di un attributo nella tabella in cui è definito (**tabella interna**) ed i valori di un attributo in una seconda tabella (**tabella esterna**).

NOTA: L'attributo/i cui si fa riferimento nella tabella esterna deve/devono essere soggetto/i al vincolo unique.

SQL: DDL

I vincoli references e foreign key consentono di definire dei **vincoli di integrità referenziale** tra i valori di un attributo nella tabella in cui e' definito (**tabella interna**) ed i valori di un attributo in una seconda tabella (**tabella esterna**).

CORSI			ESAMI		
Nome	<u>Codice</u>	IdDocente	<u>Corso</u>	<u>Matricola</u>	Voto
Basi di dati	0121	00	0121	4324235245	30L
Programmazione	1213	01	1213	4324235245	25
Sistemi Operativi	1455	02	1213	9854456565	18

SQL: DDL

CORSI

Nome	<u>Codice</u>	IdDocente
Basi di dati	0121	00
Programmazione	1213	01
Sistemi Operativi	1455	02

ESAMI

<u>Corso</u>	<u>Matricola</u>	Voto
0121	4324235245	30L
1213	4324235245	25
1213	9854456565	18

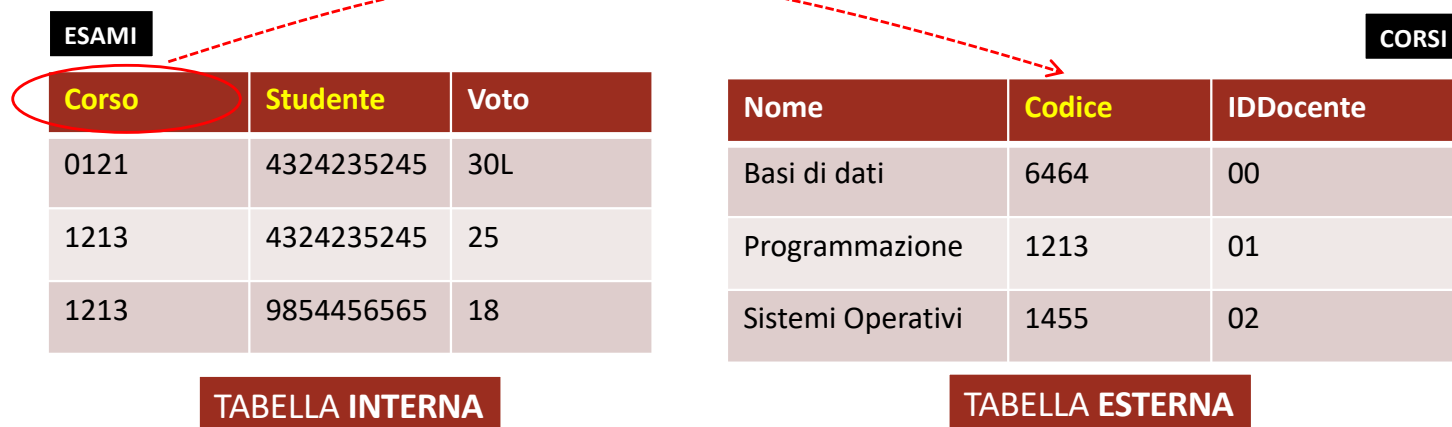
```
CREATE TABLE ESAMI (  
  CORSO VARCHAR(4) REFERENCES CORSI(CODICE)  
  STUDENTE VARCHAR(20),  
  PRIMARY KEY(CORSO, MATRICOLA),  
  ...  
)
```

SQL: DDL

Il costrutto `foreign key` si utilizza nel caso il **vincolo di integrità referenziale** riguardi più di un attributo delle tabelle interne/esterne.

```
CREATE TABLE STUDENTE {  
    MATRICOLA CHARACTER(20) PRIMARY KEY,  
    NOME VARCHAR(20),  
    COGNOME VARCHAR(20),  
    DATANASCITA DATE,  
    FOREIGN KEY(NOME,COGNOME,DATANASCITA) REFERENCES  
    ANAGRAFICA(NOME,COGNOME,DATA)  
};
```

SQL: DDL



D. Che accade se un valore nella tabella esterna viene **cancellato** o viene **modificato**?

R. Il vincolo di integrità referenziale nella tabella interna potrebbe non essere più **valido**! Cosa fare?

SQL: DDL



D. Che accade se un valore nella tabella esterna viene **cancellato** o viene **modificato**?

R. Il vincolo di integrità referenziale nella tabella interna potrebbe non essere più **valido**! Cosa fare?

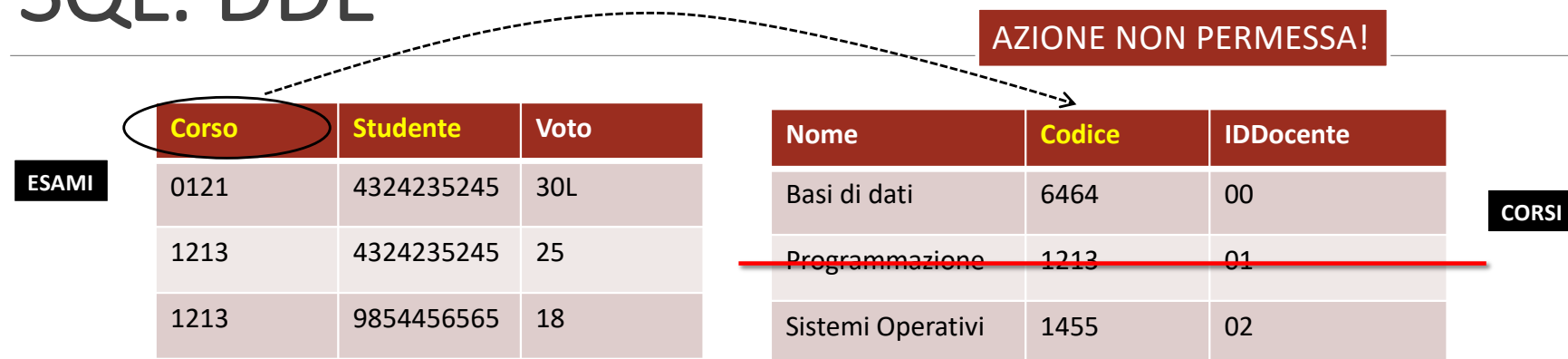
SQL: DDL

E' possibile associare **azioni specifiche da eseguire sulla tabella interna** in caso di violazioni del vincolo di integrità referenziale.

```
on (delete | update)
(cascade | set null | set default | no action)
```

- cascade → elimina/aggiorna righe (della tabella interna)
- set null → setta i valori a null
- set default → ripristina il valore di default
- no action → non consente l'azione (sulla tabella esterna)

SQL: DDL



```
CREATE TABLE ESAMI (  
  CORSO VARCHAR(4) REFERENCES CORSI(CODICE)  
    ON DELETE NO ACTION  
    ON UPDATE CASCADE  
  STUDENTE VARCHAR(20),  
  PRIMARY KEY(CORSO, STUDENTE),  
  ...  
)
```

SQL: DDL

ESAMI	Corso	Studente	Voto				CORSI
	0121	4324235245	30L	Nome	Codice	IDDocente	
	0001	4324235245	25	Basi di dati	6464	00	
	0001	9854456565	18	Programmazione	1213	01	
				Sistemi Operativi	1455	02	

```
CREATE TABLE ESAMI (  
  CORSO VARCHAR(4) DEFAULT 0001  
    REFERENCES CORSI(CODICE)  
    ON DELETE SET DEFAULT  
    ON UPDATE CASCADE  
  STUDENTE VARCHAR(20),  
  PRIMARY KEY(CORSO, STUDENTE),  
  ...  
)
```

SQL: DDL

ESAMI			CORSI		
Corso	Studente	Voto	Nome	Codice	IDDocente
0121	4324235245	30L	Basi di dati	6464	00
NULL	4324235245	25	Programmazione	1213	01
NULL	9854456565	18	Sistemi Operativi	1455	02

ERRORE!

```
CREATE TABLE ESAMI (  
  CORSO VARCHAR(4) REFERENCES CORSI(CODICE)  
    ON DELETE SET NULL  
    ON UPDATE CASCADE  
  STUDENTE VARCHAR(20),  
  PRIMARY KEY(CORSO, STUDENTE),  
  ...  
)
```

SQL: DDL

E' possibile **modificare** gli schemi di dati precedentemente creati tramite le primitive di alter (**modifica**) e drop (**cancellazione**).

drop (schema|domain|table|view) NomeElemento

drop (restrict|cascade) NomeElemento

alter NomeTabella

 alter column NomeAttributo

 add column NomeAttributo

 drop column NomeAttributo

 add constraint DefVincolo

...