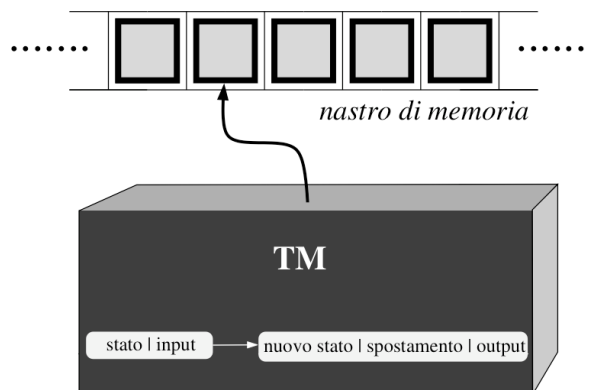


Testi di riferimento:

- A Plebe and M Monaca, Introduzione all'informatica delle conoscenze., Editori Riuniti University Press, 2010 (ISBN 3 9788864732152)
- "Excel 2019: Conoscere e utilizzare i fogli di calcolo", Edimatica, Apogeo editore, 2018 (ISBN: 9788850334537).

Turing Machine



Turing Machine

$$\downarrow$$

	0/1	0/1		
--	-----	-----	--	--

S	I	NS	NC	O
0	-	1	\Rightarrow	0
1	-	0	\Rightarrow	1

Turing Machine

$$\downarrow$$

0			
---	--	--	--

S	I	NS	NC	O
0	-	1	\Rightarrow	0
1	-	0	\Rightarrow	1

$$\downarrow$$

0	1		
---	---	--	--

S	I	NS	NC	O
0	-	1	\Rightarrow	0
1	-	0	\Rightarrow	1

$$\downarrow$$

0	1	0	
---	---	---	--

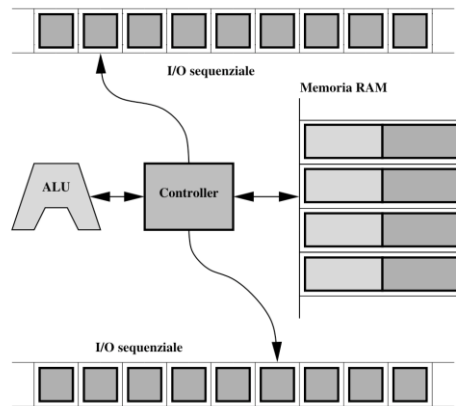
S	I	NS	NC	O
0	-	1	\Rightarrow	0
1	-	0	\Rightarrow	1

$$\downarrow$$

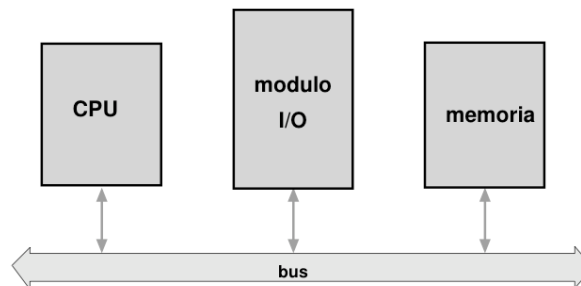
0	1	0	1
---	---	---	---

S	I	NS	NC	O
0	-	1	\Rightarrow	0
1	-	0	\Rightarrow	1

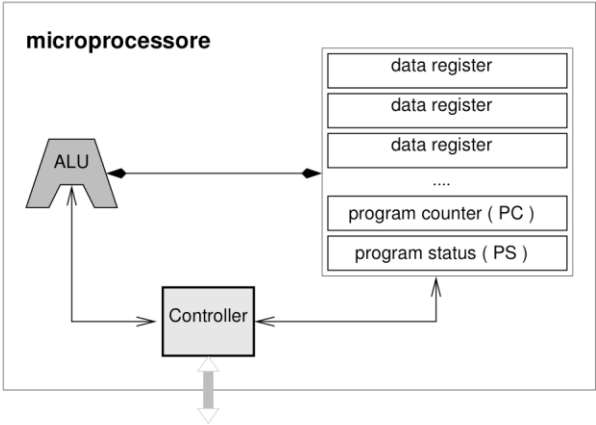
RAM Machine



CPU



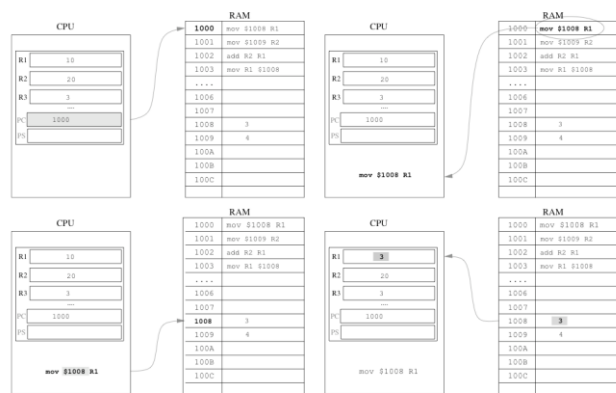
CPU



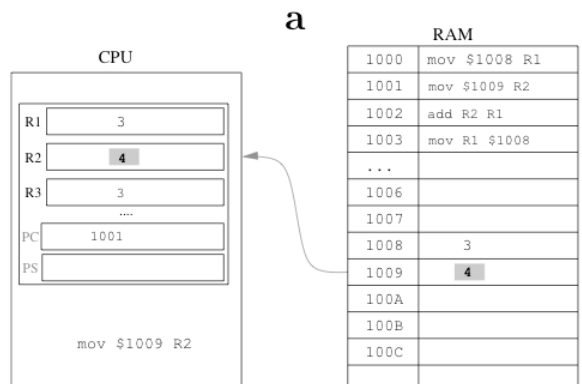
CPU

CPU		RAM	
R1	10	1000	mov \$1008 R1
R2	20	1001	mov \$1009 R2
R3	3	1002	add R2 R1
....		1003	mov R1 \$1008
PC	1000	
PS		1006	
		1007	
		1008	3
		1009	4
		100A	
		100B	
		100C	

CPU

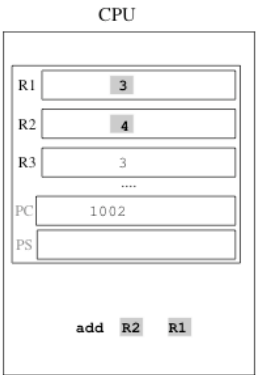


CPU



CPU

b



RAM

1000	mov \$1008 R1
1001	mov \$1009 R2
1002	add R2 R1
1003	mov R1 \$1008
...	
1006	
1007	
1008	3
1009	4
100A	
100B	
100C	

CPU

c

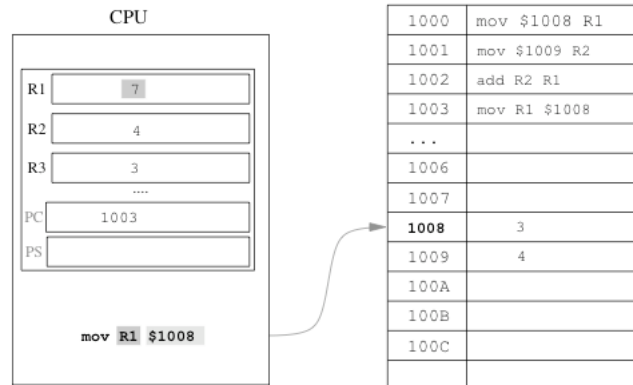


RAM

1000	mov \$1008 R1
1001	mov \$1009 R2
1002	add R2 R1
1003	mov R1 \$1008
...	
1006	
1007	
1008	3
1009	4
100A	
100B	
100C	

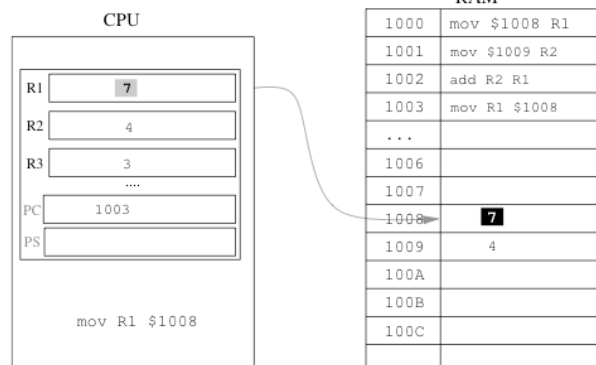
CPU

d



CPU

e



Risorse di una organizzazione

- Nello svolgimento di ogni attività, un'organizzazione deve disporre di determinate **risorse** per perseguire gli scopi che si è prefissa :
 - persone
 - denaro
 - materiali
 - **dati e informazioni**
- Ogni organizzazione dispone di un **sistema informativo** che organizza e gestisce un gran numero di **informazioni** rappresentate per mezzo di **dati**.
 - *I sistemi informativi esistono da molto prima dell'invenzione dei calcolatori.*
 - *Esempio : elenchi di utenze telefoniche, archivi anagrafici ...*

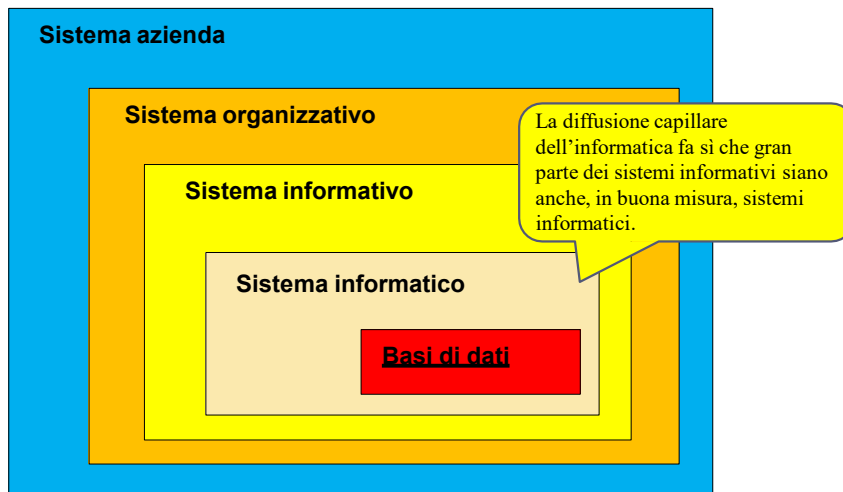
Funzioni di un Sistema Informativo

- Principali funzioni di un Sistema Informativo :
 - Raccolta e acquisizione delle informazioni;
 - archiviazione, conservazione delle informazioni;
 - elaborazione delle informazioni;
 - distribuzione, scambio di informazioni.

Il concetto di Sistema Informativo è indipendente da qualsiasi forma di automatizzazione!

- Un **Sistema Informatico** è quella porzione automatizzata del Sistema Informativo che gestisce le informazioni con tecnologia informatica.

Sistema Informatico



Dati e Informazioni

- Nelle *attività umane*, le informazioni vengono gestite in forme diverse:
 - idee informali, linguaggio naturale (scritto o parlato, formale o colloquiale), mente umana, carta, dispositivi elettronici, disegni, grafici, schemi, numeri e codici.
- Nei *sistemi informatici*, le **informazioni** vengono rappresentate per mezzo di **dati** :
 - **informazione** : notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.
 - **dato** : ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione. In informatica, un dato è un elemento di informazione costituiti da simboli che debbono essere elaborati.
 - Senza "interpretazione", i dati hanno poca utilità.
 - Se interpretati e correlati opportunamente, essi forniscono informazioni che consentono di arricchire la nostra conoscenza del mondo.

Base di Dati

□ Base di Dati :

(accezione generica) **Collezione di dati**, che tipicamente descrive le informazioni di interesse di una o più organizzazioni correlate.

(accezione specifica) **Collezione di dati** in memoria secondaria gestita da un apposito sistema software, chiamato **DBMS (Data Base Management System**, o Sistema di Gestione di Basi di Dati).

- Un DBMS è in grado di gestire e interrogare grandi collezioni di dati.
- I dati costituiscono una **risorsa strategica**, perché **più stabili nel tempo** di altre componenti (processi, tecnologie, ruoli umani).
 - *Ad esempio, i dati delle banche o delle anagrafi hanno una struttura sostanzialmente invariata da decenni, mentre le procedure che agiscono su di essi variano con una certa frequenza.*

Data Base Management System (DBMS)

□ **Sistema che gestisce collezioni di dati :**

- grandi
- persistenti
- condivise

garantendo

- privatezza
- affidabilità
- efficienza
- efficacia

□ *Alcuni prodotti software disponibili sul mercato :*

- *DB2*
- *Oracle*
- *SQLServer*
- *MySQL*
- *PostgreSQL*
- *Microsoft Access*

Esempio di rappresentazione dei dati

UNIVERSITA' DEGLI STUDI DI CHISSADOVE

Corso di Studi in Ingegneria Informatica

ORARIO DELLE LEZIONI PER L'ANNO ACCADEMICO 1999-2000

INSEGNAMENTO	Docente	Aula	Orario
Analisi matematica I	Luigi Neri	N1	8:00-9:30
Basi di dati	Piero Rossi	N2	9:45-11:15
Chimica	Nicola Mori	N1	9:45-11:30
Fisica I	Mario Bruni	N1	11:45-13:00
Fisica II	Mario Bruni	N3	9:45-11:15
Sistemi informativi	Piero Rossi	N3	8:00-9:30

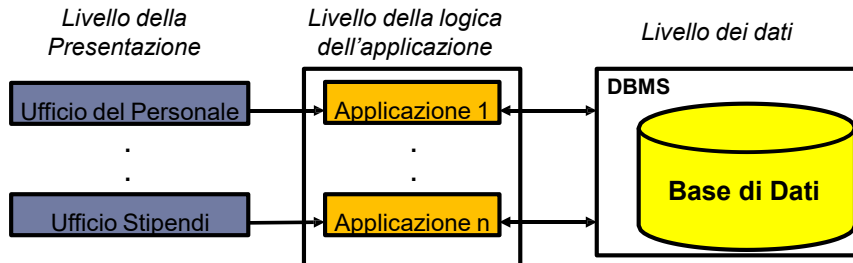
Orario di ricevimento dei docenti

DOCENTE	INSEGNAMENTI	
Mario BRUNI	Fisica I Fisica II	Martedì 10-12
Luigi NERI	Analisi matematica I	Lunedì 12-13
Piero ROSSI	Basi di dati Sistemi informativi	Giovedì 11-13
Nicola MORI	Chimica	Martedì 16-18

Le basi di dati sono...persistenti

- ☐ Hanno un tempo di vita indipendente dalle singole esecuzioni dei programmi che le utilizzano.
- ☐ In contrasto, i dati gestiti da un programma in memoria centrale hanno una vita che inizia e termina con l'esecuzione del programma.

Architettura a tre livelli del Sistema Informatico



- Una base di dati viene utilizzata dai vari uffici (e persone), ciascuno con le proprie competenze, attraverso **programmi diversi**.
- Il DBMS garantisce **integrazione e condivisione** coordinando i vari flussi di informazioni che giungono dalle varie sorgenti.

Modello dei Dati

- Insieme di costrutti utilizzati per organizzare i dati di interesse all'interno del DBMS e descriverne la dinamica.
- Come nei linguaggi di programmazione esistono meccanismi che permettono di definire nuovi tipi, così ogni modello dei dati prevede alcuni costruttori.
- Ad esempio, il **modello relazionale** prevede il **costruttore relazione**, che **permette di definire insiemi di record omogenei**
- Una relazione viene rappresentata per mezzo di una **tabella**, le cui righe rappresentano specifici record e le cui colonne corrispondono ai campi del record.

Il Modello Relazionale

- Una relazione viene rappresentata per mezzo di una **tabella**, le cui righe rappresentano specifici record e le cui colonne corrispondono ai campi del record.
- E' il più diffuso nell'ambito delle Basi di Dati. Sarà anche il nostro modello di riferimento.

Intestazione della tabella: nome della relazione

Nome	Cognome	Posizione	Squadra	Età
Tommaso	Rocchi	Attaccante	S.S.Lazio	31
Alessandro	Del Piero	Attaccante	Juventus F.C.	34
Francesco	Totti	Attaccante	A.S. Roma	32

Tupla o Record

Schemi e Istanze

- In ogni base di dati si distinguono :
 - lo **schema**, sostanzialmente **invariante nel tempo**, che ne descrive la struttura (**aspetto intensionale**); *nell'esempio*, le **intestazioni** delle tabelle :

Giocatore (Nome, Cognome, Posizione, Squadra, Età)

Squadra (Nome, Anno di Fondazione)

- l'**istanza**, costituita dai valori attuali, che possono cambiare molto rapidamente (**aspetto estensionale**); nell'esempio, il "**corpo**" di ciascuna tabella :

Tommaso	Rocchi	Attaccante	S.S. Lazio	31
Alessandro	Del Piero	Attaccante	Juventus F.C.	34
Francesco	Totti	Attaccante	A.S. Roma	32

Linguaggi delle Basi di Dati

- I DBMS sono caratterizzati dalla presenza di molteplici linguaggi per la gestione dei dati.
- L'accesso ai dati può avvenire:
 - con **linguaggi testuali interattivi (ad es. SQL)**.
 - con comandi (come quelli del linguaggio interattivo) immersi in un **linguaggio ospite (Java, C, Cobol, etc.)**.
 - con comandi (come quelli del linguaggio interattivo) immersi in un **linguaggio ad hoc**, con anche altre funzionalità (ad es. per grafici o stampe strutturate), anche con l'ausilio di strumenti di sviluppo (ad es. per la gestione di maschere).
 - con **interfacce user friendly** (senza linguaggio testuale).

SQL, un linguaggio interattivo

Corsi

Corso	Docente	Aula
Basi di dati	Rossi	DS3
Sistemi	Neri	N3
Reti	Bruni	N3
Controlli	Bruni	G

Aule

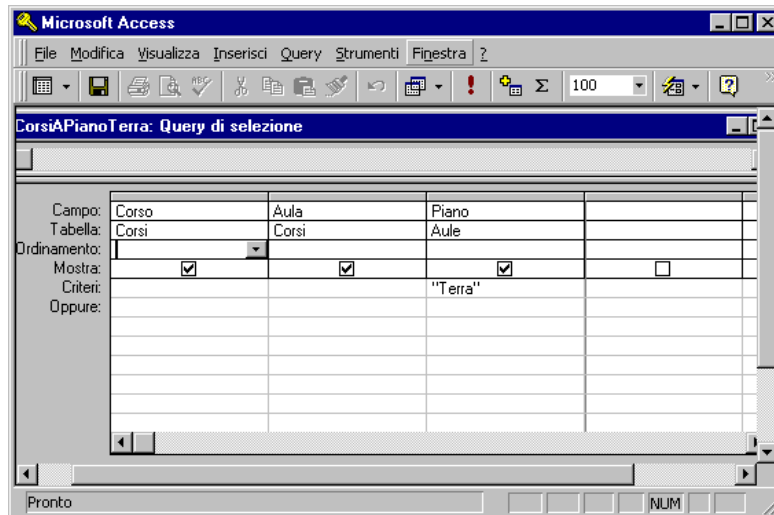
Nome	Edificio	Piano
DS1	OMI	Terra
N3	OMI	Terra
G	Pincherle	Primo

"Trovare i corsi tenuti in aule a piano terra"

```
SELECT Corso, Aula, Piano
FROM Aule, Corsi
WHERE Aule.Nome = Corsi.Aula
AND Aule.Piano = "Terra"
```

Corso	Aula	Piano
Sistemi	N3	Terra
Reti	N3	Terra

Interazione non testuale (MS Access)



Una distinzione (separazione fra dati e programmi)

- **data manipulation language (DML)**
 - per l'interrogazione e l'aggiornamento di (istanze di) basi di dati.
- **data definition language (DDL)**
 - per la definizione di schemi (logici, esterni, fisici) e altre operazioni del genere.

Un'operazione DDL

```
CREATE TABLE corsi (
  corso      CHAR(20),
  docente    CHAR(20),
  aula       CHAR(4))
```

Corso	Docente	Aula
Basi di dati	Rossi	DS3
Sistemi	Neri	N3
Reti	Bruni	N3
Controlli	Bruni	G

Vantaggi e Svantaggi dei DBMS

Pro

- dati come risorsa comune di un'organizzazione, a disposizione (con opportune forme di controllo) di tutte le sue componenti.
- schema dei dati come modello unificato e preciso della realtà di interesse per l'organizzazione.
- gestione centralizzata dei dati, con riduzione di ridondanze e incoerenze.
- indipendenza dei dati (favorisce lo sviluppo di applicazioni più flessibili e modificabili).
- integrità e sicurezza dei dati, attraverso *vincoli di integrità e controlli di accesso*.
- ripristino dai crash, proteggendo gli utenti dagli effetti dei guasti del sistema.

Vantaggi e Svantaggi dei DBMS

Contro

- I DBMS sono prodotti complessi e costosi. La loro introduzione comporta notevoli investimenti diretti (acquisto del prodotto) e indiretti (acquisizione delle risorse hardware e software necessarie, conversione delle applicazioni, formazione del personale).
- I DBMS forniscono una serie di servizi, necessariamente associati ad un costo. Se alcuni servizi non sono più necessari, è difficile scorporare quelli effettivamente richiesti dagli altri, e ciò può comportare una riduzione di prestazioni.
- Applicazioni con pochi utenti e senza necessità di accessi concorrenti possono essere realizzate più proficuamente con file ordinari piuttosto che con DBMS.

Sistemi di Basi di Dati

- **Base di Dati** : *Collezione di dati*, che tipicamente descrive le informazioni di interesse di una o più organizzazioni correlate.
- **DBMS (Database Management System)** : *Sistema software* in grado di memorizzare, gestire e interrogare grandi collezioni di dati.



Come vengono rappresentati i dati in un DBMS ?



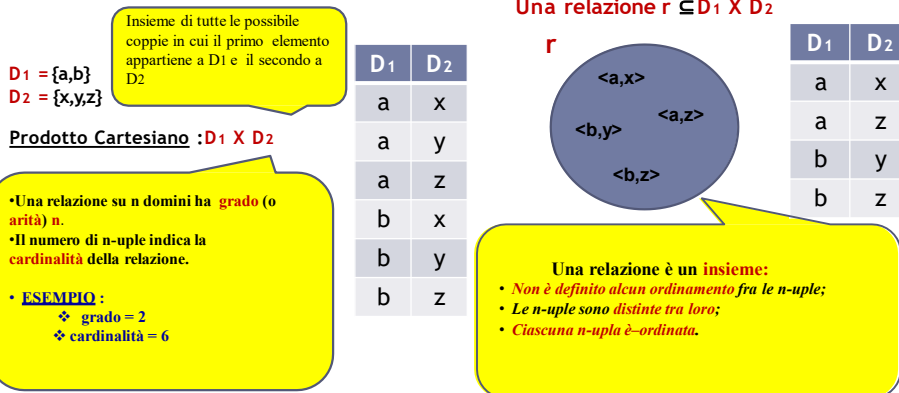
- **Modello dei dati** : *Collezione di costrutti* utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che risulti comprensibile ad un elaboratore. Il modello più diffuso è il **modello relazionale**.

Il Modello Relazionale

- È il modello più largamente usato.
 - Produttori: IBM, Microsoft, Oracle, etc.
 - Competitori recenti:
 - modello orientato agli oggetti;
 - altro modello emergente: XML.
- Disponibile come modello logico in DBMS reali a partire dal 1981 (non è facile realizzare l'indipendenza dei dati con efficienza e affidabilità!).
- Si basa sul concetto matematico di **relazione** (con una variante).
- Le relazioni provengono dalla teoria degli insiemi ed hanno una rappresentazione naturale per mezzo di **tabelle**.
 - La presenza contemporanea di questi due concetti, uno formale e l'altro intuitivo, è responsabile del grande successo ottenuto dal modello relazionale.
- Il modello è **"basato su valori"**: i riferimenti fra dati in strutture (relazioni) diverse sono rappresentati per mezzo dei valori stessi.

Relazione matematica

- D_1, D_2, \dots, D_n (n insiemi – detti **domini** della relazione - anche non distinti)
- Il **prodotto cartesiano** $D_1 \times D_2 \times \dots \times D_n$ è l'insieme di tutte le n-uple **ordinate** (d_1, d_2, \dots, d_n) tali che $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$
- Una relazione matematica su D_1, D_2, \dots, D_n è un **sottoinsieme del prodotto cartesiano** $D_1 \times D_2 \times \dots \times D_n$



Il Modello Relazionale

- Il costrutto di base per la descrizione dei dati è la **relazione**. Una relazione è sostanzialmente una **tabella**.
- A ciascun dominio è associato un **nome (attributo)**, unico nella relazione, che descrive il ruolo del dominio. Gli attributi sono usati come intestazione delle colonne (il cui ordinamento è irrilevante).
- Le righe della tabella rappresentano specifici **record** (o **tuple**) **diversifra loro**.

Nome	Cognome	Posizione	Squadra	Età
Tommaso	Rocchi	Attaccante	S.S.Lazio	31
Alessandro	Del Piero	Attaccante	Juventus F.C.	34
Francesco	Totti	Attaccante	A.S. Roma	32

Tabelle e Relazioni

- Una tabella rappresenta una relazione se :
 - i valori di ciascuna colonna sono fra loro omogenei (appartengono allo stesso dominio);
 - le righe sono diverse fra loro;
 - le intestazioni delle colonne (attributi) sono diverse tra loro.
- Inoltre, in una tabella che rappresenta una relazione :
 - l'ordinamento tra le righe è irrelevante;
 - l'ordinamento tra le colonne è irrelevante (struttura non posizionale).
- **Il modello relazionale è basato su valori** : i riferimenti fra dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle tuple.

SQL : Structured Query Language

- SQL non è un semplice linguaggio per le interrogazioni...
- ...ma si divide in 3 sotto-linguaggi principali:
 - **DDL (Data Definition Language)** : linguaggio che permette di creare\eliminare\modificare gli **oggetti** in un database (tabelle e viste). I comandi DDL definiscono la **struttura** del database.
 - **DML (Data Manipulation Language)** : linguaggio che permette di leggere\inserire\modificare\eliminare i **dati** di un database. Le **interrogazioni** in SQL appartengono a DML.
 - **DCL (Data Control Language)** : permette di fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi di DDL e DML su specifici oggetti/dati di un database.
- Altre componenti: **trigger** (azioni eseguite dal DBMS che soddisfano determinate condizioni), **SQL dinamico ed embedded**, **esecuzione client-server**, **gestione di transazioni**, **sicurezza**.

Sintassi di un'interrogazione SQL

```
SELECT [DISTINCT] listaAttributi  
FROM listaTabelle  
[WHERE condizione]
```

- Un'interrogazione SQL può essere valutata analizzando i comandi che la compongono nel seguente ordine :

1. **listaTabelle** = lista delle tabelle su cui calcolare il risultato.
 2. **condizione** = espressione booleana ottenuta combinando gli operatori di confronto (<, <=, =, <>, >=, >) e gli operatori logici AND, OR, NOT.
 3. **listaAttributi** = lista di attributi (presi dalle tabelle contenute in **listaTabelle**) che definiscono il risultato dell'interrogazione.
- **DISTINCT** è una parola chiave opzionale utile per specificare che il risultato dell'interrogazione **non deve contenere duplicati**.

1. Clausola **FROM**

Per formulare un'interrogazione che coinvolge tuple appartenenti a più di una tabella, si pone come argomento della clausola **FROM** l'insieme di tabelle alle quali si vuole accedere.

```
SELECT DISTINCT Impiegato  
FROM Impiegati, Reparti  
WHERE Impiegato = 'Neri'
```

Il risultato *parziale* consiste nel **prodotto cartesiano** delle tabelle elencate nella clausola **FROM**.

Impiegato	Codice
Rossi	A
Neri	B
Bianchi	B

Capo	Codice
Mori	A
Bruni	B

Impiegati Reparti

Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	B	Mori	A
Neri	B	Bruni	B
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

2.Clausola **WHERE**

```
SELECT DISTINCT Impiegato, Codice  
FROM Impiegati, Reparti  
WHERE Impiegato = 'Neri'
```

Impiegati		Reparti	
Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	B	Mori	A
Neri	B	Bruni	B
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

Sul **prodotto cartesiano** delle tabelle elencate nella clausola **FROM** verranno applicate le condizioni contenute nella clausola **WHERE**.

3.Clausola **SELECT**

La clausola **SELECT** specifica quali attributi faranno parte della tabella risultato.

```
SELECT DISTINCT Impiegato  
FROM Impiegati, Reparti  
WHERE Impiegato = 'Neri'
```

Impiegati		Reparti	
Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	B	Mori	A
Neri	B	Bruni	B
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

Il risultato di un'interrogazione SQL è un **multi-insieme**. Se si desidera che la tabella calcolata **non contenga duplicati**, si deve includere la parola chiave **DISTINCT**.

Impiegato
Neri
Neri

↓

Impiegato
Neri

Select con asterisco (*)

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Come argomento della clausola SELECT può anche comparire il carattere speciale * (**asterisco**), che rappresenta la selezione di tutti gli attributi delle tabelle elencate nella clausola FROM.

ESERCIZIO :Estrarre tutte le informazioni degli impiegati di cognome “Rossi”

```
SELECT *  
FROM Impiegato  
WHERE Cognome='Rossi'
```



Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Rossi	Direzione	80

Convenzioni sui nomi

- Per evitare ambiguità, ogni nome di attributo è composto da
NomeTabella.NomeAttributo
- Quando l'ambiguità non sussiste, si può omettere la parte
NomeTabella

```
SELECT persone.nome, persone.reddito  
FROM persone  
WHERE persone.età<30
```

si può scrivere come:

```
SELECT nome, reddito  
FROM persone  
WHERE età<30
```

NOT, AND, OR

La clausola **WHERE** consente di costruire un'espressione booleana combinando predicati semplici con gli operatori **AND**, **OR** e **NOT**.

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre il nome ed il cognome degli Impiegati che lavorano nel dipartimento Amministrazione ed hanno stipendio maggiore di 70

```
SELECT Nome,Cognome
FROM Impiegato
WHERE Dipart = 'Amministrazione'
AND StipAnn > 70
```



Nome	Cognome
------	---------

Operatore LIKE

SQL mette a disposizione un operatore **LIKE** per il confronto fra stringe.

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre gli Impiegati con un nome che comincia una "m" e che ha la coppia di caratteri "rc" in penultima posizione

```
SELECT *
FROM Impiegato
WHERE Nome LIKE 'm%rc_'
```

Il carattere **_** rappresenta un confronto con un carattere arbitrario, mentre **%** rappresenta un confronto con una stringa di lunghezza arbitraria (eventualmente nulla).

ORDER BY

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

Ordinamento ascendente per **Nome** e **Cognome**.
Prima vengono ordinati i valori contenuti in **Nome**.

ESERCIZIO :Estrarre nome e cognome degli impiegati che lavorano in Amministrazione, in ordine alfabetico di nome e cognome

```
SELECT Nome,Cognome
FROM Impiegato
WHERE Dipart='Amministrazione'
ORDER BY Nome, Cognome
```



Nome	Cognome
Giuseppe	Verdi
Mario	Rossi
Paola	Rosati

Successivamente, per tuple che hanno lo stesso valore in **Nome**, si ordinano i **Cognome**

Operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre il numero di Impiegati del dipartimento Produzione

```
SELECT count(*)
FROM Impiegato
WHERE Dipart = 'Produzione'
```



count(*)
2

count : operatore aggregato di conteggio. Conta quante tuple soddisfano le condizioni inserite nella clausola WHERE.

L'operatore SUM

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre la somma degli Stipendi del dipartimento Amministrazione

```
SELECT SUM(StipAnn)
FROM Impiegato
WHERE Dipart = 'Amministrazione'
```



```
SUM(StipAnn)
125
```

L'operatore MAX

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre il massimo stipendio tra quelli degli impiegati che lavorano in un dipartimento con sede a Milano

```
SELECT MAX(StipAnn)
FROM Impiegato, Dipartimento D
WHERE Dipart = D.Nome and Città='Milano'
```



```
MAX(StipAnn)
80
```

Combinare operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre gli stipendi minimo, massimo e medio fra quelli di tutti gli impiegati

```
SELECT MAX(StipAnn),
       MIN(StipAnn),
       AVG(StipAnn)
FROM Impiegato
```



MAX(StipAnn)	MIN(StipAnn)	AVG(StipAnn)
80	36	51

GROUP BY

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre la somma degli stipendi degli impiegati che lavorano nello stesso dipartimento

```
SELECT Dipart, SUM(StipAnn)
FROM Impiegato
GROUP BY Dipart
```



Dipart	SUM(StipAnn)
Amministrazione	125
Produzione	82
Distribuzione	45
Direttore	53

Clausola HAVING

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO :Estrarre i dipartimenti che spendono più di 100mila euro in stipendi

```
SELECT Dipart,SUM(StipAnn) AS  
Sommastipendi  
FROM Impiegato  
GROUP BY Dipart  
HAVING SUM(StipAnn)>100
```

Dipart	Sommastipendi
Amministrazione	125
Direzione	153



Sintassi Completa di un'interrogazione SQL

```
SELECT [DISTINCT] lista-select  
FROM lista-from  
[WHERE condizione]  
[GROUP BY lista gruppo]  
[HAVING qualificazione gruppo]  
[ORDER BY AttrDiOrdinamento]
```

SQL : Structured Query Language

- SQL non è un semplice linguaggio per le interrogazioni...
- ...ma contiene 3 sotto-linguaggi :
 - **DDL (Data Definition Language)** : linguaggio che permette di creare\eliminare\modificare gli oggetti in un database.
 - i comandi DDL permettono la definizione dello schema di una base di dati (*livello intensionale*).
 - **DML (Data Manipulation Language)** : linguaggio che permette di leggere\inserire\modificare\eliminare i dati di un database.
 - i comandi DML permettono di interrogare e modificare un'istanza di una base di dati (*livello estensionale*).
 - **DCL (Data Control Language)** : permette di gestire gli utenti ed i permessi.

SQL : Alcune Notazioni

- Notazione utilizzata per specificare la sintassi dei comandi:
 - Le parentesi quadre [] indicano che il termine contenuto al suo interno è opzionale, cioè può non comparire o comparire una sola volta.
 - Le parentesi graffe { } indicano che il termine racchiuso può non comparire o essere ripetuto un numero arbitrario di volte.
 - Le barre verticali | indicano che deve essere scelto solo uno tra i termini separati dalle barre.
 - Le parentesi tonde () dovranno essere intese sempre come termini del linguaggio SQL e non come simboli per la definizione della grammatica.

Creazione di una tabella

- L'istruzione più importante del DDL di SQL è:

CREATE TABLE

- definisce la struttura di uno **schema di relazione** (specificandone gli **attributi** e un insieme - eventualmente vuoto - di **vincoli**).
- crea un'istanza vuota dello schema.
- Il nome della tabella può essere formato da lettere o numeri, ma il primo carattere deve essere sempre una lettera.
- **Sintassi :**

```
CREATE TABLE NomeTabella (  
    NomeAttributo Dominio [Vincoli]  
    ....  
    {NomeAttributo Dominio [Vincoli] }  
    {Vincolo di tabella }  
)
```

Deve essere **diverso** dai nomi degli altri attributi **nella stessa tabella**.

Deve essere **diverso** dai nomi delle altre tabelle **nello stesso database**.

CREATE TABLE, esempio

```
CREATE TABLE NomeTabella (  
    NomeAttributo Dominio {Vincoli }  
    ....  
    {NomeAttributo Dominio {Vincoli } }  
    {Vincoli di tabella }  
)
```

```
CREATE TABLE Impiegato (  
    Matricola char(6) PRIMARY KEY,  
    Nome varchar(20) NOT NULL,  
    Cognome varchar(20) NOT NULL,  
    Dipart varchar(15),  
    Stipendio numeric(9) DEFAULT 0,  
    Salario real,  
    FOREIGN KEY(Dipart) REFERENCES Dipartimento(NomeDip),  
    UNIQUE(Cognome, Nome))
```

ATTENZIONE

- Una tabella in SQL è definita come un **multinsieme** di tuple.
- In particolare, se una tabella non ha una *primary key* o un insieme di attributi definiti come *chiave*, allora potranno comparire due tuple uguali nella tabella; ne segue che:
 - una tabella SQL non è in generale una relazione matematica.
- Se invece una tabella ha una *primary key* o un insieme di attributi definiti come *chiave*, allora non potranno mai comparire nella tabella due tuple uguali; per questo,
 - è indispensabile definire almeno una primary key per ogni tabella SQL.

► 6
1

Basi di Dati

Domini

- **Domini elementari** (predefiniti)
 - **Carattere**: singoli caratteri o stringhe, anche di lunghezza variabile
 - **Bit**: singoli booleani
 - **Numerici**: esatti e approssimati
 - **Data, ora, intervalli di tempo**
 - Introdotti in SQL:1999
 - **Boolean**
 - **BLOB, CLOB** (binary/character large object): per grandi immagini e testi
- **Domini definiti dall'utente** (semplici, ma riutilizzabili)

SQL : Domini Predefiniti

□ **Carattere** : singoli caratteri o stringhe

- **char(n) o character(n)** – stringhe di lunghezza fissa.
 - *n è il numero massimo di caratteri che si vogliono memorizzare.*
- **varchar(n)** – stringhe di lunghezza variabile.
 - la lunghezza di un campo definito con dominio CHAR(n) è **precisamente di dimensione n bytes**, indipendentemente dal dato inserito.
 - la lunghezza di un campo definito con dominio VARCHAR(n) **assume la dimensione del dato inserito + 1 byte di prefisso**.

Il DBMS elimina automaticamente i caratteri "non utilizzati" di troppo.

Possibile rappresentazione dei domini *char* e *varchar* in un calcolatore.

Valore	CHAR(4)	Spazio Richiesto	VARCHAR(4)	Spazio Richiesto
"	' '	4 byte	"	1 byte
'ab'	'ab '	4 byte	'ab'	3 bytes
'abcd'	'abcd'	4 byte	'abcd'	5 bytes

SQL : Domini Predefiniti

□ **Numerici** : valori numerici esatti

- **int (o smallint, bigint) [UNSIGNED]** – interi di lunghezza fissa.
- utili nei casi in cui non interessa avere una rappresentazione della parte frazionaria.
- lo standard non fissa la precisione dei tipi numerici, che dipende dalla rappresentazione interna del calcolatore.

□ **Numerici** : valori numerici esatti con eventuale parte frazionaria.

- **numeric (o numeric(p) o numeric(p,s))** : valori numerici esatti (anche negativi) utilizzabili se interessa una rappresentazione della parte frazionaria.
 - **p** rappresenta il numero massimo di cifre definibili. **s** rappresenta la *scala*, ovvero il numero di cifre che devono comparire dopo la virgola (con $s \leq p$).
 - **Esempio:**
 - **numeric(3,1)** : numeri da -99,9 a +99,9
 - **numeric(3,2)** : numeri da -9,99 a +9,99
 - **numeric(3)** : numeri da -999 a +999

Di default vale **SIGNED**.

UNSIGNED permette la rappresentazione dei soli interi positivi.

SQL : Domini Predefiniti

- **Numerici** : valori numerici approssimati (utili per rappresentare grandezze fisiche).
 - **float** (**real** è equivalente), (**float(m,d)**, **double**, **double(m,d)**) [**UNSIGNED**]: numeri in virgola mobile.
 - **m** rappresenta la precisione – numero di cifre – dedicate alla mantissa. La mantissa è un valore frazionario.
 - **d** è la precisione dell'esponente. L'esponente è un numero intero.
 - **FLOAT** è a "**precisione singola**", **DOUBLE** sono invece a "**precisione doppia**".
 - Per entrambi i tipi di dato l'uso di UNSIGNED disabilita i valori negativi, ma non ha effetto sui valori massimi positivi memorizzabili
- **Esempio:**
 - **0.17E16** rappresenta il valore $1,7 \times 10^{15}$
 - **-0.4E-6** rappresenta il valore -4×10^{-7}
- **bit, bit(n)** : sequenza fissa di n bit (valori appartenenti all'insieme {0,1})
 - **Esempio:** **bit(2)** : {00},{01},{10},{11}

Utili per rappresentare *flag*, che specificano se l'oggetto rappresentato da una tupla possiede o meno una certa proprietà.

SQL : Domini Predefiniti

- **Data e Ora** :
 - **date** :
 - ammette i campi ('anno-mese-giorno') nel formato ('**aaaa-mm-gg**').
 - rappresentabili tutte le date da '1000-01-01' (1° gennaio 1000) a '9999-12-31' (31 dicembre 9999).
 - **Esempio:**
 - **INSERT into table name_table VALUES ('2009-03-26')**
 - **time** :
 - contiene un valore di tempo ('ore:minuti:secondi') nel formato ('**hh:mm:ss**').
 - **timestamp** :
 - Visualizzato nel formato ('**AAAAMMGGhhmmss**').
 - Comodo per memorizzare il momento dell'inserimento o aggiornamento di record, in quanto può essere assegnato automaticamente.

ATTENZIONE
agli apici

Valore di default:
timestamp corrente

Domini introdotti in SQL:1999

□ **boolean :**

- utilizzato per rappresentare i valori booleani (restrizione del dominio bit) *true* o *false*.

□ **BLOB e CLOB :**

- permettono di rappresentare oggetti di grandi dimensioni, costituiti da una sequenza arbitraria di valori binari (*blob* – “*binary large object*”) o di caratteri (*clob* – “*character large object*”).
- il sistema garantisce solo di memorizzarne il valore, ma **non permette che il valore venga utilizzato come criterio di selezione per le interrogazioni.**
- utili per gestire contenuti di grandi dimensioni (semi-strutturati o multimediali – immagini, documenti, video) e la loro fruizione richiede l'uso di applicazioni specifiche.

Vincoli di Integrità

- Un **vincolo di integrità** descrive le condizioni che ogni istanza legale di una relazione deve soddisfare.
 - *Limita* i dati che possono essere memorizzati in un'istanza della base di dati.
 - Inserimenti/cancellazioni/aggiornamenti che violano i vincoli di integrità non sono permessi.
 - Possono essere usati per garantire la semantica dell'applicazione, o per prevenire inconsistenze.
- A. **Vincoli intrarelazionali:** vincoli che coinvolgono una sola relazione (*not null, unique, primary key, default*).
- Esistono anche i **vincoli di dominio**, che specificano che i valori dei campi devono essere sempre del tipo corretto.
- B. **Vincoli interrelazionali:** vincoli di integrità referenziale (*foreign key*).



Vincoli predefiniti sugli attributi

```
CREATE TABLE NomeTabella(  
    NomeAttributo Dominio [Vincoli]  
    ....  
    {NomeAttributo Dominio [Vincoli] }  
    {Vincolo di tabella }  
)
```

- ❑ **[NOT NULL | NULL]** - In assenza del vincolo, di default può contenere valori NULL.
- ❑ **[DEFAULT valore]** - usato per impostare un valore di default, utile quando in un inserimento il valore dell'attributo non viene specificato. In assenza del vincolo, si suppone come default il valore NULL.
- ❑ **[UNIQUE | PRIMARY KEY]** - UNIQUE rappresenta un attributo che non può contenere valori duplicati (una chiave); PRIMARY KEY indica la chiave primaria, che oltre a non ammettere duplicati non può contenere valori NULL.
- ❑ **[REFERENCES Nome_Tabella [(Nome_Attributo)]]** - Permette di definire un vincolo di chiave esterna su *Nome_Attributo* verso la chiave primaria di *Nome_Tabella*.



Vincoli predefiniti sulla tabella

```
CREATE TABLE NomeTabella(  
    NomeAttributo Dominio [Vincoli]  
    ....  
    {NomeAttributo Dominio [Vincoli] }  
    {Vincolo di tabella }  
)
```

- ❑ **[PRIMARY KEY (nome_attributo1, nome_attributo2,...)]** - Permette di definire una chiave primaria per la tabella sfruttando un certo insieme di attributi.
- ❑ **[UNIQUE (nome_attributo1, nome_attributo2,...)]** - Permette di definire una chiave candidata per la tabella sfruttando un certo insieme di attributi.
- ❑ **[FOREING KEY (nome_att1,nome_att2,...) REFERENCES nome_tabella [(nome_att1,nome,att2,...)]]** - Permette di definire vincoli di chiave esterna su più attributi.



Chiavi Candidate e Primarie

□ Vincoli di chiave

- Insieme minimo di campi tale che in ogni istanza legale due tuple distinte siano diverse nei valori di tali campi.
- **UNIQUE** rappresenta la definizione di (super)chiave, cioè di un insieme di campi che non può contenere valori duplicati.
 - viene fatta un'eccezione per il valore NULL, che può comparire su righe diverse senza violare il vincolo.
- **PRIMARY KEY** indica la chiave primaria (scelta tra le *chiavi candidate*) che, oltre a non ammettere duplicati, non può contenere valori NULL.
- **è FONDAMENTALE definire una e una sola PRIMARY KEY per ogni relazione.**
 - per evitare la possibilità di avere due tuple identiche nella relazione.
 - perché il DBMS può creare un indice con i campi della chiave primaria come campi di ricerca.

Chiavi Candidate e Primarie

Si crea la tabella persona con i seguenti vincoli:

- **nome** è una PRIMARY KEY, perciò non ammette duplicati e non può assumere valori NULL.
- **cognome** può assumere valori NULL ed ha un valore di DEFAULT.
- **età** non ammette duplicati, può assumere valori NULL ed ha un valore di DEFAULT.

```
CREATE TABLE Persona(  
  nome varchar(20) PRIMARY KEY,  
  cognome varchar(20) DEFAULT 'pippo',  
  età int UNIQUE DEFAULT 0  
)
```

Chiavi Candidate e Primarie

PRIMARY KEY e **UNIQUE**
definite per più attributi.

```
CREATE TABLE NomeTabella(  
    NomeAttributo1 Dominio [Vincoli],  
    NomeAttributo2 Dominio [Vincoli],  
    NomeAttributo3 Dominio [Vincoli],  
    NomeAttributo4 Dominio [Vincoli],  
    PRIMARY KEY(NomeAttributo1, NomeAttributo2),  
    UNIQUE(NomeAttributo3, NomeAttributo4)  
)
```

Inserimento dei dati in una tabella

□ Sintassi :

```
INSERT INTO NomeTabella  
    [(attributo_1,...,attributo_n)]  
    VALUES (valore 1,...,valore n)
```

Esempio: cominciamo con il creare una relazione SQL

```
CREATE TABLE Ricoveri (  
    Paziente CHAR(4),  
    Inizio DATE DEFAULT ('0000-00-00') ,  
    Fine DATE,  
    Reparto CHAR  
)
```

Ricoveri

Paziente	Inizio	Fine	Reparto
----------	--------	------	---------

Inserimento dei dati in una tabella

- Per ogni colonna deve essere specificato un valore corrispondente del giusto dominio.
- Se non viene inserita nessuna lista di colonne, allora deve essere dato un valore (seguendo l'ordine originale di definizione degli attributi nello schema relazionale) per ogni colonna della relazione.
- **ESEMPIO :**

```
INSERT INTO Ricoveri(Paziente, Inizio, Fine, Reparto)
VALUES('A102','2008-06-02','2008-06-05','A')
```

```
INSERT INTO Ricoveri
VALUES('B444','B')
```

Ricoveri

Paziente :CHAR(4)	Inizio :Date	Fine :Date	Reparto :CHAR
A102	02/05/2008	05/06/2008	A

Cancellazione di tuple

- **cancellare una o più tuple da una tabella**

```
DELETE FROM NomeTabella
[WHERE Condizione]
```

- **ESEMPIO :**

- **Cancellare tutte le righe della tabella Dipartimento**

```
DELETE FROM Dipartimento
```

Se l'argomento della clausola WHERE non viene specificato, il comando cancella tutte le righe della tabella.

- **Cancellare tutte le righe della tabella Dipartimento relative al settore Produzione**

```
DELETE FROM Dipartimento
WHERE Settore='Produzione'
```

La clausola WHERE può contenere interrogazioni annidate.

Aggiornamento di tuple

- **Aggiornare una o più tuple di una tabella**

```
UPDATE NomeTabella  
SET Attributo = Espressione {, Attributo = Espressione}  
[WHERE Condizione]
```

La clausola WHERE viene applicata per prima e determina i campi da modificare. La clausola SET determina i nuovi valori.

- **ESEMPIO:**
- **Incrementare di 5 lo stipendio dei dipendenti afferenti al settore Amministrazione**

```
UPDATE Dipendente SET Stipendio = Stipendio + 5  
WHERE Settore = 'Amministrazione'
```

Dipendente

ID	Nome	Stipendio	Settore
0	Marco	200	Direzione
1	Paola	300	Amministrazione



ID	Nome	Stipendio	Settore
0	Marco	200	Direzione
1	Paola	305	Amministrazione

Creazione/Cancellazione di schemi

- **creazione di un nuovo database**

```
create database NomeDatabase
```

- **cancellazione di un database**

```
drop database NomeDatabase [restrict|cascade]
```

- **cancellazione di una tabella**

```
drop table NomeTabella [restrict|cascade]
```

- L'opzione **restrict** (di default) specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti. Uno schema non viene rimosso se contiene tabelle o altri oggetti. Una tabella non viene rimossa se possiede delle righe o se è presente in qualche definizione di tabella o vista.
- Con l'opzione **cascade** tutti gli oggetti specificati devono essere rimossi.

```
DROP TABLE Persone cascade
```

Distrugge la relazione Persone. Le informazioni sullo schema e le tuple vengono cancellate.

Modifica di tabelle

- **aggiungere colonne ad una tabella**

```
ALTER TABLE NomeTabella  
  ADD COLUMN NomeColonna  
              Dominio [Vincoli]
```

- **eliminare colonne da una tabella**

```
ALTER TABLE NomeTabella  
  DROP NomeColonna
```

Modifica di tabelle

- **aggiungere\eliminare un valore di default da una tabella**

```
ALTER TABLE NomeTabella  
  ALTER COLUMN NomeAttributo  
    <SET DEFAULT Valore di default | DROP DEFAULT>
```

- **eliminare un vincolo unique da una tabella**

```
ALTER TABLE NomeTabella  
  DROP UNIQUE (NomeColonna1,...,NomeColonnaN)
```