Le basi di dati

- Definizioni
- Modello relazionale e SQL

Base di dati

E' una collezione di dati utilizzata per rappresentare le informazioni di interesse in un sistema informativo

- I dati hanno vita più lunga delle procedure che operano su di essi

Gestione di una base di dati

- Organizzazione e conservazione dei dati
- Operazioni per la gestione dei dati

DBMS

Data Base Management System è un sistema software per gestire basi di dati

Caratteristiche

grandi dimensioni: uso della memoria secondaria
condivisione: l'accesso di più utenti a dati comuni si riduce la ridondanza dei dati e la possibilità di inconsistenze
persistenza - i dati rimangono memorizzati
affidabilità - capacità di non perdere i dati in caso di malfunzionamento (*backup* e *recovery*)
privatezza - ciascun utente è riconosciuto da un *username* e una *password* e ha autorizzazione solo per certe operazioni
efficienza - capacità di svolgere le operazioni in tempo ragionevole e con risorse di calcolo e memoria accettabili

Modelli, schemi ed istanze

- Modello dei dati: strumento per organizzare i dati e descriverne la struttura
  - relazionale, gerarchico, reticolare, ad oggetti...
- Schema: struttura delle informazioni (sostanzialmente invariabile)

RUBRICA(Nome,Cognome,Indirizzo,Telefono)

- Istanza (stato): valori effettivi contenuti nella base di dati

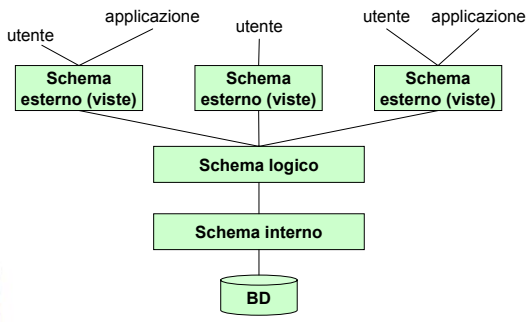
Mario	Rossi	Via Verdi 5	0577234567
Marco	Bianchi	Via Righi 73	0577456783
Anna	Dati	Via Romeo	0578345234

Architettura ANSI/SPARC: schemi

Esistono tre tipi di rappresentazione dei dati:

- Schema logico
  descrizione dell'intera base di dati nel modello logico "principale" del DBMS
- Schema interno (o fisico)
  rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione
- Schema esterno
  - descrizione di parte della base di dati in un modello logico
  - definisce "viste" parziali, anche in modelli diversi da quello logico

Architettura standard (ANSI/SPARC) a tre livelli per DBMS



Indipendenza dei dati

- indipendenza fisica: il livello logico e quello esterno sono indipendenti da quello fisico
  - una relazione è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica
  - la realizzazione fisica può cambiare senza che debbano essere modificati i programmi
- indipendenza logica: il livello esterno è indipendente da quello logico
  - aggiunte o modifiche alle viste non richiedono modifiche al livello logico
  - modifiche allo schema logico che lascino inalterato lo schema esterno sono trasparenti

Modello relazionale

Il modello relazionale Codd, 1970

- Modello logico
  - Definisce come sono organizzati i dati e non come sono poi memorizzati e gestiti dal sistema informatico
  - Per accedere ai dati non è necessario conoscere le strutture fisiche con cui sono realizzati!
- Adottato dalla maggior parte dei DBMS in commercio
- Costrutto di base: **Relazione – Tabella**
  - Il concetto di relazione proviene dalla matematica mentre quello di tabella è intuitivo

Prodotto cartesiano

- Consideriamo gli insiemi dei nomi e dei numeri telefonici dei dipendenti

D_1
Mario Rossi
Luca Verdi
Anna Bianchi

D_2
2345
2367
2378
2356

D_1 x D_2	
Mario Rossi	2345
Mario Rossi	2367
Mario Rossi	2378
Mario Rossi	2356
Luca Verdi	2345
Luca Verdi	2367
Luca Verdi	2378
Luca Verdi	2356
Anna Bianchi	2345
Anna Bianchi	2367
Anna Bianchi	2378
Anna Bianchi	2356

- Il prodotto cartesiano $D_1 x D_2$ è l'insieme di tutte le coppie ordinate (NOME,TELEFONO)
  - contiene tutte le possibili associazioni fra gli elementi degli insiemi

Relazioni

- Una relazione matematica sugli insiemi D_1 e D_2 (domini) è un sottoinsieme del prodotto cartesiano $D_1 \times D_2$

La rubrica contiene solo alcune delle possibili coppie (NOME,TELEFONO)

Mario Rossi	2345
Luca Verdi	2367
Luca Verdi	2378
Anna Bianchi	2356

In generale

- Si possono considerare *n* domini non necessariamente distinti $D_1, D_2, \ldots, D_n$
- Il prodotto cartesiano $D_1 x D_2 x \ldots x D_n$ contiene tutte le possibile *n*-uple $(v_1, v_2, .., v_n)$ dove v_i è uno dei possibili valori presenti nel dominio D_i
- Una relazione sui domini $D_1, D_2, \ldots, D_n$ è un sottoinsieme del prodotto cartesiano $D_1 x D_2 x \ldots x D_n$
  - Il numero delle componenti del prodotto (*n*) è detto grado della relazione
  - il numero di n-uple della relazione è la cardinalità della relazione

Un esempio

D_1 = Squadre di Serie A D_2 = Squadre di Serie A

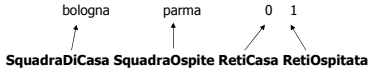
D_3 = Numero D_4 = Numero

Relazione: risultati delle partite della II giornata del campionato

Lazio	Inter	1	1
Fiorentina	Roma	3	1
Milan	Bari	0	0
Bologna	Parma	0	1
Udinese	Napoli	2	1
Lecce	Perugia	1	1
Reggina	Juventus	3	0
Atalanta	Brescia	2	2
Verona	Vicenza	0	0

Relazioni con attributi

- I nomi associati ai domini si dicono attributi e descrivono il ruolo rappresentato dominio nella relazione
- Ogni n-upla stabilisce un legame fra valori
  - il significato dei valori dipende dall'ordine con cui sono elencati nell'n-upla



- Si può associare un nome alle componenti delle n-uple:
  - il nome dell'attributo corrispondente
  - si parla di tuple

tuple

- Data una tupla posso estrarne il valore degli attributi

SquadraDiCasa	SquadraOspitata	RetiCasa	RetiOspitata
Fiorentina	Roma	3	1

t[**SquadraOspitata**] = Roma

t[**SquadraDiCasa**] = Fiorentina

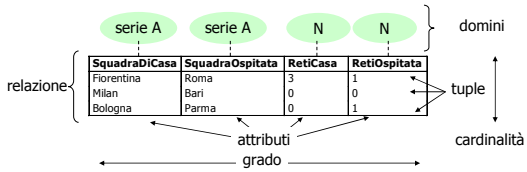
- Si possono estrarre anche insiemi di attributi ottenendo tuple

t[**SquadraDiCasa**,**SquadraOspitata**]=

SquadraDiCasa	SquadraOspitata
Fiorentina	Roma

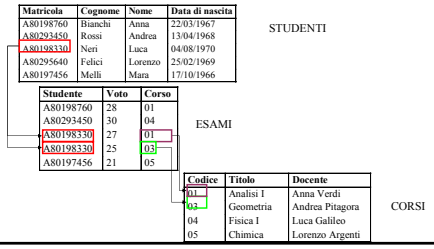
tuple e relazioni

- Una relazione su X è un insieme di tuple su X



Relazioni e basi di dati

- Una base di dati è in genere costituita da più relazioni
- Si possono creare corrispondenze fra tuple di due relazioni
  - le corrispondenze si ottengono fra tuple di relazioni diverse aventi gli stessi valori su un insieme assegnato di attributi



Schemi per basi di dati

- Uno schema di relazione è costituito dal nome della relazione e da un insieme di attributi $X=\{A_1, A_2, \ldots, A_n\}$

STUDENTI(Matricola,Cognome,Nome,Data di Nascita)

- Uno schema di base di dati è un insieme di schemi di relazioni con nomi diversi:

$\mathbf{R}=\{R_1(X_1), R_2(X_2), \ldots, R_m(X_m)\}$

R = { STUDENTI(Matricola,Cognome,Nome,Data di Nascita),
ESAMI(Studente,Voto,Corso),
CORSI(Codice,Titolo,Docente) }

Informazione incompleta

- Può accadere che il valore di un attributo non sia disponibile per tutte le tuple
  - il valore è sconosciuto, è realmente inesistente o è senza informazione (non si sa se esiste o meno)
- Si introduce un valore nullo (NULL) che denota l'assenza di informazione sul valore dell' attributo per una data tupla

Codice	Nome	Indirizzo	PIVA
0001	Carlo Berti	Via Roma 6	NULL
0034	BCD Spa	Via Verdi 4	04554303
0101	A&G Srl	Viale Morgagni 16	NULL

inesistente

sconosciuto

Vincoli di integrità

- Occorre che le tuple rappresentino informazioni corrette per l'applicazione
  - In generale occorre che i valori assegnati agli attributi in ciascuna tupla soddisfino una serie di vincoli
- Es: Valori nulli

Non ammesso!

Matricola	Cognome	Nome	Data di nascita
NULL	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968
A80197456	Melli	Mara	NULL

STUDENTI

Può essere accettabile

Alcuni campi non possono assumere valori nulli!

Vincoli intrarelazionali

Sono vincoli che coinvolgono il valore degli attributi all'interno di una stessa relazione

Studente	Voto	Lode	Corso
A80198760	37		01
A80293450	30	L	04
A80198330	22		01
A80198330	25	L	03

Valore non ammesso

Combinazione non ammessa

Non ci possono essere due studenti con matricola uguale!

Matricola	Cognome	Nome	Data di nascita
A80198760	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968
A80198760	Felici	Lorenzo	25/02/1969
A80197456	Melli	Mara	17/10/1966

Vincoli interrelazionali

- Sono vincoli che coinvolgono più relazioni
  - Le corrispondenze fra i dati in relazioni diverse si stabiliscono per mezzo dei valori di chiavi delle tuple

Matricola	Cognome	Nome
672	Rossi	Mario
223	Verdi	Francesco
532	Belli	Carlo

AGENTI

Codice	Data	Agente	Art	Prov	Numero
174655	25-12-1999	672	44	FI	G03635
156732	13-05-2000	223	12	SI	734563
456345	17-08-1999	672	44	GR	234322

INFRAZIONI

AUTO

Prov	Numero	Proprietario	Indirizzo
FI	G03635	Gilli Luca	Via Oro
SI	734563	Tilli Nedo	Via Abete
GR	234322	Billi Aldo	Via Sole

Vincoli di integrità referenziale

Un vincolo di integrità referenziale (foreign key) fra un insieme di attributi X di una relazione R_1 e un'altra relazione R_2 è soddisfatto se i valori su X di ogni tupla in R_1 compaiono nell'istanza di R_2 come valori della chiave (primaria)

Codice	Data	Agente	Art	Prov	Numero
174655	25-12-1999	672	44	FI	G03635
156732	13-05-2000	223	12	SI	734563
456345	17-08-1999	345	44	GR	234322

INFRAZIONI

?

Matricola	Cognome	Nome
672	Rossi	Mario
223	Verdi	Francesco
532	Belli	Carlo

AGENTI

Occorre definire un vincolo di integrità referenziale fra l'attributo Agente della relazione INFRAZIONI e la relazione AGENTI (chiave primaria Matricola)

Integrità referenziale: un esempio

Targa	Proprietario	Indirizzo
BE370TZ	Fabio Rossi	Via Abete 13 Siena
FIG04546	Giovanni Neri	Via Moro 7 Firenze
AB234TV	Michele Bessi	Via Sila 4 Terni
TV443AZ	Maria Bianchi	Via Belli 4 Roma

AUTOVEICOLI

INCIDENTI

Codice	Targa1	Targa2	Luogo	Danni
5434	BE370TZ	AB234TV	Siena	Fari
5534	FIG04546	TV443AZ	Monteroni	Parauri

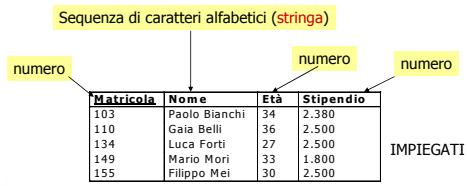
1. vincolo di integrità referenziale fra l'attributo Targa1 della relazione INCIDENTI e la relazione AUTOVEICOLI
2. vincolo di integrità referenziale fra l'attributo Targa2 della relazione INCIDENTI e la relazione AUTOVEICOLI

SQL

- SQL (Structured Query Language) è un linguaggio di interrogazione per basi di dati relazionali
- Contiene funzionalità anche di
  - Data Definition Language (definizione dello schema)
  - Data Manipulation Language (operazioni di modifica - aggiornamento, inserimento, cancellazione)
- E' stato standardizzato da organismi internazionali (ANSI,ISO) in varie versioni:
  - 1986 SQL; 1989 SQL-89; 1992 SQL-92 o SQL-2; ?? SQL-3
- Nessun sistema commerciale offre un'implementazione completa di SQL-2
  - La parte standard di SQL permette di utilizzare gli stessi comandi su DBMS diversi fra loro

Definizione dei dati

- Si possono usare 6 domini (tipi di dato) predefiniti
- I tipi di dato sono utilizzati per definire il tipo per gli attributi (colonne) delle relazioni



Matricola	Nome	Età	Stipendio
103	Paolo Bianchi	34	2.380
110	Gaia Belli	36	2.500
134	Luca Forti	27	2.500
149	Mario Mori	33	1.800
155	Filippo Mei	30	2.500

IMPIEGATI

Il tipo carattere

- Rappresenta singoli caratteri alfanumerici oppure stringhe di lunghezza fissa o variabile

```
char
char(lunghezza)
varchar(lunghezza)
```

Si definisce l'attributo `Nome` della relazione IMPIEGATI come sequenza di caratteri di lunghezza massima 20

`Nome char(20)` — `Paolo_Bianchi_______`

`Nome varchar(20)` — `Paolo_Bianchi`

Il tipo bit

Corrisponde ad attributi che possono assumere solo due valori (0,1)

- Attributi di questo tipo (flag) indicano se l'oggetto rappresentato possiede o meno una certa proprietà

```
bit
bit(lunghezza)
bit varying (lunghezza)
```

Si definisce l'attributo `Lavoratore` nella relazione STUDENTI per indicare se si lo studente è o meno lavoratore

`Lavoratore bit`

Tipi numerici esatti

- Rappresentano numeri interi o numeri decimali in virgola fissa (con un numero prefissato di decimali)

```
numeric   numeric(precisione)  numeric(precisione,scala)
decimal   decimal(precisione)  decimal(precisione,scala)
integer
smallint
```

Si definisce l'attributo `Eta` nella relazione IMPIEGATI

`Eta decimal(2)` Rappresenta tutti i numeri fra -99 e +99

Si definisce l'attributo `Cambio` nella relazione PAGAMENTO per indicare il valore del cambio del dollaro preciso al centesimo di lira

`Cambio numeric(8,2)` Rappresenta tutti i numeri fra -9999,99 e +9999,99

Tipi numerici approssimati

Sono utili per rappresentare ad esempio grandezze fisiche (rappresentazione in virgola mobile)

```
float              float(precisione)
double precision
real
```

Si definisce l'attributo `Massa` nella relazione ASTEROIDI

`Massa real`

Date

- Permettono di rappresentare istanti di tempo

```
date
time        time(precisione)
timestamp   timestamp(precisione)
```

- Ciascuno di questi domini è decomponibile in un insieme di campi
  - anno,mese,giorno,ora,minuti,secondi

```
DataDiNascita date        18/09/99
OraDiConsegna time        19.24.16
Arrivo timestamp          18/09/00 21.15.20

year(Arrivo) = 2000       minute(Arrivo) = 15
```

Intervalli temporali

- Permettono di rappresentare intervalli di tempo come durate di eventi

```
interval PrimaUnitàDiTempo
interval PrimaUnitàDiTempo to UltimaUnitàDitempo
```

Esempi

Anzianità di servizio in anni e mesi

```
AnzianitaServizio interval year to month
```

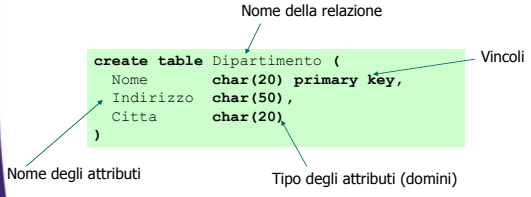
Tempo di consegna in giorni ed ore

```
TempoConsegna interval day to hour
```

Non è possibile costruire intervalli che comprendano mesi e giorni

Definizione di tabelle

Una tabella è costituita da un insieme ordinato di attributi e di vincoli



Create table

- La sintassi per la definizione di una tabella è

```
create table NomeTabella (
  NomeAttributo1 Dominio [Valore di default] [vincoli],
  ……
  NomeAttributoN Dominio [Valore di default] [vincoli],
  AltriVincoli
)
```

opzionali

- Una tabella è inizialmente vuota

La tabella STUDENTI

```
create table Studenti (
  Matricola      char(9) primary key,
  Cognome        varchar(50),
  Nome           varchar(50),
  DataDiNascita  date,
  Lavoratore     bit default null
)
```

Se non si specifica un valore si inserisce per default il valore NULL

Matricola	Cognome	Nome	Data di nascita	Lavoratore

Studenti

Vincoli intrarelazionali semplici

- Operano su un solo attributo della relazione

`not null`	L'attributo non può assumere il valore NULL
`unique`	Non possono esistere due righe che hanno gli stessi valori per l'attributo o insieme di attributi specificati
`primary key`	Identifica la chiave primaria. Può essere specificato una sola volta.

Esempi di vincoli

```
create table Impiegato (
  Cognome        varchar(50) not null unique,
  Nome           varchar(50) not null unique,
  Dipartimento   integer,
  Stipendio      integer default 0
)
```

Non sono la stessa cosa!

```
create table Impiegato (
  Cognome        varchar(50) not null,
  Nome           varchar(50) not null,
  Dipartimento   integer,
  Stipendio      integer default 0,
  unique(Cognome,Nome)
)
```

Vincoli di integrità referenziale

- Creano un legame fra i valori dell'attributo della tabella corrente e i valori di un attributo di un'altra tabella (tabella esterna)
- E' richiesto che il valore dell'attributo, se non è nullo, sia presente tra i valori dell'attributo di riferimento della tabella esterna
- L'attributo della tabella esterna a cui si fa riferimento deve essere dichiarato **`unique`** (ad esempio la chiave)
- Si possono utilizzare i costrutti
  - **`references`**
  - **`foreign key`**

references

```
create table Impiegato (
  Matricola      char(6) primary key,
  Cognome        varchar(50) not null,
  Nome           varchar(50) not null,
  Diparti        char(15)
                 references Dipartimento(NomeDip),
  Stipendio      integer default 0,
  unique(Cognome,Nome)
)
```

Il campo `Dipart` può assumere solo i valori che compaiono nel campo `NomeDip` della tabella `Dipartimento`

foreign key

```
create table Impiegato (
  Matricola      char(6) primary key,
  Cognome        varchar(50) not null,
  Nome           varchar(50) not null,
  Diparti        char(15)
                 references Dipartimento(NomeDip),
  Stipendio      integer default 0,
  unique(Cognome,Nome),
  foreign key (Nome,Cognome)
          references Anagrafica(Nome,Cognome)
)
```

La coppia di campi (`Nome,Cognome`) può assumere solo le coppie di valori che compaiono nei campi (`Nome,Cognome`) della tabella `Anagrafica`

Cosa accade quando si viola un vincolo

- In genere il comando viene rifiutato generando un errore
- Per modifiche o cancellazioni di righe nella tabella esterna si può decidere come e se la modifica si propaga alla tabella corrente
  - **`cascade,set null,set default, no action`**

Matricola	Cognome	Nome	Data di nascita
A80198760	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968

Cancellazione di una riga

Studente	Voto	Corso
A80198760	28	01
A80293450	30	04
A80198760	27	03
A80293450	25	03
A80293450	21	05

```
foreign key(Studente) references
    Studenti(Matricola) on delete cascade
                        on update cascade
```

Indici

- Servono per velocizzare le interrogazioni a spese di un incremento della complessità degli aggiornamenti
- Si introduce un ordinamento sugli attributi specificati
- Non è una caratteristica standard di SQL

```
create index NomeCognome on Studenti(Cognome,Nome)
```

Crea un indice per effettuare ricerche più veloci utilizzando gli attributi `Nome` e `Cognome` della tabella `Studenti`.

L'ordinamento è per `Cognome`

Interrogazioni semplici

- SQL è un linguaggio di interrogazione dichiarativo
  - Le interrogazioni esprimono cosa si cerca e non come si cerca
  - l'interprete SQL del DBMS traduce l'interrogazione in operazioni interne sui dati che producono la risposta
  - L'utente non deve quindi preoccuparsi di come si trova il risultato
- Le operazioni di interrogazione vengono specificate con l'istruzione **`select`**

select

Matricola	Cognome	Nome	Data di nascita
A80198760	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968
A80198330	Neri	Luca	04/08/1970
A80295640	Rossi	Lorenzo	25/02/1969
A80197456	Melli	Mara	17/10/1966

Studenti

Campi considerati

Tabelle da cui si selezionano

```
select Matricola as NumeroMatricola
from Studenti
where Cognome = 'Rossi'
```

Condizioni

Nuovo nome per l'attributo

NumeroMatricola
A80293450
A80295640

Risultato: una relazione!

select *

Matricola	Cognome	Nome	Data di nascita
A80198760	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968
A80198330	Neri	Luca	04/08/1970
A80295640	Rossi	Lorenzo	25/02/1969
A80197456	Melli	Mara	17/10/1966

Studenti

Tutti i campi sono considerati

```
select *
from Studenti
where Cognome = 'Rossi'
```

Matricola	Cognome	Nome	Data di nascita
A80293450	Rossi	Andrea	13/04/1968
A80295640	Rossi	Lorenzo	25/02/1969

select (espressioni)

Prodotto	Cliente	Quantita	PrezzoUnitario
Chiodo 35mm	Bianchi	100	15
Vite 45mm	Rossi	50	20
Dado 5mm	Neri	200	8
Vite 8mm	Rossi	20	25
Bulloni 5cm	Melli	40	50

```
select Prodotto, Quantita*PrezzoUnitario as PrezzoTotale
from Ordini
where Cliente = 'Rossi'
```

Espressione che combina più attributi

Prodotto	PrezzoTotale
Vite 45mm	1000
Vite 8mm	500

select ….. from

- Per accedere a righe appartenenti a più di una tabella, si usa come argomento della clausola **`from`** la lista della tabelle a cui si vuole accedere
- Sul prodotto cartesiano delle tabelle elencate vengono applicate le condizioni della clausola **`where`**
- Se si vuole effettuare un join occorre scrivere in modo esplicito le condizioni che esprimono il legame tra le diverse tabelle

Il join con select

Matricola	Cognome	Nome	Data di nascita
A80198760	Bianchi	Anna	22/03/1967
A80293450	Rossi	Andrea	13/04/1968
A80295640	Felici	Lorenzo	25/02/1969

Studenti

Studente	Voto	Corso
A80198760	28	Analisi I
A80293450	30	Basi di Dati
A80295640	27	Analisi I
A80198760	30L	Fisica I
A80293450	21	Chimica

Esami

Attributi di tabelle

```
select Studenti.Cognome,Studenti.Nome,Esami.Voto
from Studenti,Esami
where (Studenti.Matricola = Esami.Studente) AND
      (Esame.Corso = 'Analisi I')
```

prodotto cartesiano

Condizione di join

Cognome	Nome	Voto
Bianchi	Anna	28
Felici	Lorenzo	27

select ….. from… where

- La clausola **where** ammette come argomento una condizione costruita combinando condizioni semplici con gli operatori **and**, **or** e **not**
  - ad esempio, confronti con =, <> ,<,>,<=,>= fra costanti o valori degli attributi
- Si selezionano solo le righe per cui la condizione è vera
- Per il confronto di stringhe si può usare l'operatore like che supporta i caratteri speciali:
  - _ (qualsiasi carattere)
  - % (qualsiasi sequenza di caratteri)

```
Matricola like '_801%'
```

→ A801334323
→ A801234322
→ B801343232

join interno

Modo alternativo per indicare il join di due tabelle che distingue le condizioni di join da quelle di selezione delle righe

Non si specificano le tabelle (non c'è ambiguità)

```
select Cognome,Nome,Voto
from Studenti inner join Esami on Matricola = Studente
where (Corso = 'Analisi I')
```

Condizione di selezione

Condizione di join

join esterno

- Nell'operazione di join può capitare che alcune righe di una tabella non siano selezionate perché non c'è nessuna riga dell'altra tabella che soddisfa la condizione di join
- Il join esterno esegue il join interno mantenendo però tutte le righe che fanno parte di una o di entrambe le tabelle coinvolte
- Tipologie di join esterno
  - **outer left**
    sono aggiunte le righe della relazione a sinistra del join che non righe corrispondenti nella tabella di destra
  - **outer right**
    sono aggiunte le righe della relazione a destra del join che non hanno righe corrispondenti nella tabella di sinistra
  - **outer full**
    compaiono tutte le righe delle due tabelle

outer left join

Guidatore

Cognome	Nome	NroPatente
Bianchi	Anna	VR 2030020Y
Rossi	Andrea	PZ 1012436B
Felici	Lorenzo	AP 4544442R

Automobile

Targa	Marca	Modello	NroPatente
AB574WW	Fiat	Punto	VR2030020Y
AA652FF	Renault	Clio	VR2030020Y
BJ747XX	Ford	Focus	PZ1012436B
BB421JJ	Renault	Megane	MI2020030U

Associa un altro nome alla tabella (alias)

```
select *
from Guidatore G left join Automobile A
     on (G.NroPatente = A.NroPatente)
```

ci sarebbe stata ambiguità

Cognome	Nome	NroPatente	Targa	Marca	Modello
Bianchi	Anna	VR 2030020Y	AB574WW	Fiat	Punto
Bianchi	Anna	VR 2030020Y	AA652FF	Renault	Clio
Rossi	Andrea	PZ 1012436B	BJ747XX	Ford	Focus
Felici	Lorenzo	AP 4544442R	NULL	NULL	NULL

Uso di variabili

- Permette di far riferimento a più esemplari della stessa tabella

Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 1.700

```
select I1.Nome,I1.Stip
from Impiegati I1, Supervisione S, Impiegati I2
where I1.Matr = S.Capo and
      I2.Matr = S.Imp and
      I2.Stip > 1700
```

Copia I1: usata per ricavare le informazioni sul capo

Copia I2: usata per riferirsi agli impiegati

Operatori aggregati

- Permettono di costruire interrogazioni che richiedono il calcolo di misure su insiemi di tuple, ad esempio:
  - contare il numero di studenti che hanno superato un esame
  - trovare il massimo voto preso da uno studente ad un esame
  - calcolare la media di uno studente
- Si definiscono gli operatori aggregati che combinano fra loro più tuple

count	conteggio righe
sum	somma
max	massimo
min	minimo
avg	media

Conteggio, massimo,media...

Studente	Voto	Corso
A80198760	28	01
A80293450	30	04
A80198760	27	03
A80293450	25	03
A80293450	21	05

```
select count(*) from esami
where Studente = 'A80293450'
```

```
select max(Voto) from esami
where Studente = 'A80293450'
```

```
select avg(Voto) from esami
where Studente = 'A80293450'
```

Informazioni sui voti dello studente A80293450

```
select count(distinct Studente)
from esami
```

Conteggio di quanti studenti hanno fatto almeno un esame

Interrogazioni con raggruppamento

- Il raggruppamento permette di applicare gli operatori aggregati distintamente a sottoinsiemi di righe

Calcolare la media di ogni studente

Studente	Voto	Corso
A80198760	28	01
A80293450	30	04
A80198760	27	03
A80293450	25	03
A80293450	21	05

```
select Studente,avg(Voto) as Media
from Esami
group by Studente
```

Studente	Voto
A80198760	28
A80198760	27
A80293450	30
A80293450	25
A80293450	21

Studente	Media
A80198760	27.500
A80293450	25.333

Interrogazioni con raggruppamento e selezione

- Si possono selezionare solo alcuni raggruppamenti in base a condizioni di tipo aggregato

Trovare la media degli studenti che hanno sostenuto almeno 3 esami

Studente	Voto	Corso
A80198760	28	01
A80293450	30	04
A80198760	27	03
A80293450	25	03
A80293450	21	05

```
select Studente,avg(Voto) as Media
from Esami
group by Studente
having count(*)>=3
```

Studente	Voto
A80198760	28
A80198760	27
A80293450	30
A80293450	25
A80293450	21

Studente	Media
A80293450	25.333

Interrogazioni nidificate

- Nella clausola **where** si possono utilizzare valori prodotti da altre istruzioni select utilizzando **any** (qualsiasi) o **all** (tutti) insieme agli operatori di confronto

Trovare nome, cognome e matricola degli studenti che non hanno fatto esami

```
select Matricola,Nome,Cognome from studenti
where matricola <> all (select studente
                        from esami
                        group by studente)
```

"diversa da tutte"
(si può usare anche **not in**)

Matricole degli studenti che hanno fatto almeno un esame

Un'altra interrogazione

Trovare la matricola degli studenti che hanno fatto almeno 3 esami

```
select Matricola from studenti
where matricola = any (select studente
                       from esami
                       group by studente
                       having count(*)>=3)
```

"uguale ad almeno una"
(si può usare anche **in**)

Matricole degli studenti che hanno fatto almeno 3 esami

Inserimento,cancellazione e modifica

Inserimento

```
insert into studenti values
  ('A80198760','Bianchi','Anna','1967-03-22'),
  ('A80293450','Rossi','Andrea','1968-04-13')
```

Cancellazione

```
delete from studenti
where Matricola = 'A80198760'
```

Cancella lo studente con matricola A80198760

```
delete from studenti
where Matricola not in (select Matricola
                        from esami
                        group by Matricola)
```

Cancella gli studenti che non hanno esami

Modifica

```
update esami set Voto = 30
where Matricola = 'A80198760'
      and corso = 'Analisi I'
```

Vincoli di integrità generici

- E' possibile specificare vincoli di integrità generici con la clausola **check**(*condizione*)

Vincoli sul valore del voto

```
create table esami (
  studente char(9) not null
                  references studenti(matricola)
  voto     integer not null check(voto<=30 and voto>=0),
  lode     char(1) check(lode=' ' or lode='L'),
  corso    char(20) not null,
  unique(studente,corso),
  check(not((voto<>30) and (lode='L')))
)
```

Vincolo su lode solo con 30

Vincoli sul valore della lode

Viste

- Corrispondono a tabelle virtuali ovvero che non esistono fisicamente ma il cui contenuto è ottenuto da altre tabelle
- In SQL si ottengono assegnando una lista di attributi e un nome ad una interrogazione con **select**

```
create view StudentiMeritevoli (Matricola,Nome,Cognome)
as select Matricola,Nome,Cognome
   from Studenti
   where Matricola in (select studente
                       from esami
                       group by studente
                       having avg(Voto)>=28)
```