Sistemi di Elaborazione delle Informazioni

DB ed SQL

Prof. Silvio Vassallo

1

Modello Relazionale

Il **modello relazionale** si basa sul concetto di RELAZIONE tra insiemi di oggetti.

Dati n insiemi $A_1, A_2, \ldots A_n$ si dice relazione un sottoinsieme di tutte le n-uple $a_1, a_2, \ldots . a_n$ che si possono costruire prendendo nell'ordine un elemento a_1 di A_1, a_2 di A_2 e così via.

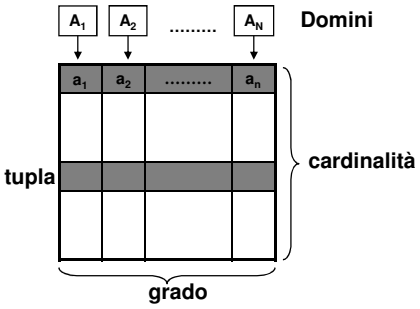
- A_i ⟹ domini
- n ⟹ grado
- n° tuple ⟹ cardinalità

2

Modello relazionale: concetti di base

La relazione è rappresentata con una tabella, avente tante colonne quanti sono i domini (**grado**) e tante righe quante sono le n-uple (**cardinalità**).

I nomi dei domini sono i nomi delle colonne, i valori che compaiono in una colonna sono **omogenei** tra loro (appartengono allo stesso dominio).

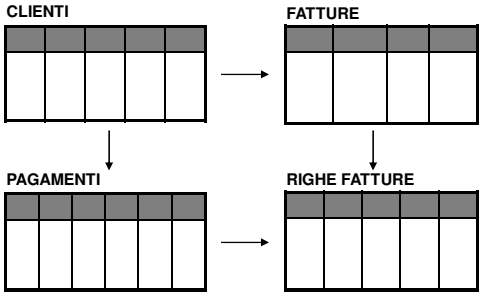


La relazione (*collezione di tuple*) è un'**entità**, ogni tupla è un'**istanza** dell'entità, le colonne sono gli **attributi** dell'entità, il dominio è l'insieme dei **possibili valori** di un attributo.

3

Modello relazionale: concetti di base

La **chiave** della relazione è un attributo o una combinazione di attributi che identificano univocamente le tuple: ogni riga della tabella possiede valori diversi per l'attributo (o gli attributi) chiave.



Il modello relazionale di un database è un **insieme di tabelle**, sulle quali si possono effettuare operazioni e tra le quali possono essere stabilite delle associazioni.

4

Requisiti fondamentali del modello relazionale

1. Tutte le righe contengono **lo stesso numero di colonne.**
2. Gli attributi rappresentano **informazioni elementari**
3. I valori assunti da un campo appartengono allo **stesso dominio**
4. Ogni riga è diversa da tutte le altre (**chiave primaria**)
5. **Ordine** righe **non prefissato**

5

Integrità sull'entità

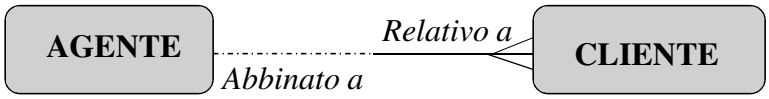
Ogni dato elementare contenuto nel modello relazionale deve essere accessibile attraverso la combinazione di:
- nome della tabella,
- nome e valore della chiave,
- nome della colonna contenente il dato.

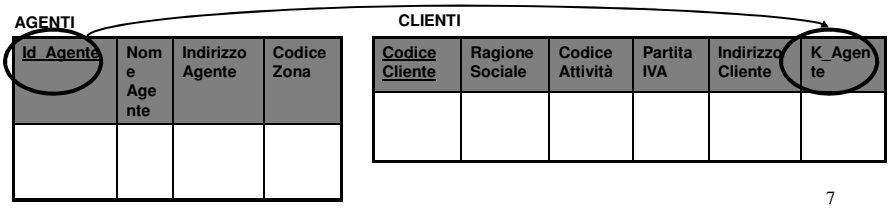
Nessuna componente della chiave primaria di una tabella può avere valore nullo.

6

Operazioni relazionali

Agiscono su una relazione per ottenere una nuova relazione.
Effettuano le **interrogazioni** alle basi di dati per ottenere le informazioni desiderate, estraendo da una tabella una sottotabella o combinando tra loro due o più tabelle.

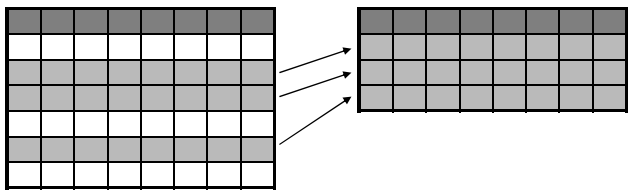




7

Selezione

La **selezione** genera una nuova relazione costituita solo dalle n-uple della relazione di partenza che soddisfano una determinata condizione: vengono selezionate le righe con i valori degli attributi corrispondenti alla condizione prefissata.



La relazione ottenuta possiede tutte le colonne della relazione di partenza e quindi ha lo **stesso grado**; la **cardinalità** della nuova relazione può essere **minore o uguale** alla tabella di partenza (solitamente è minore).

8

Selezione: esempio

Se si vuole l'elenco dei clienti della provincia di Milano, si effettua sulla relazione *Clienti* una selezione per

Provincia = "MI"

estraendo dalla tabella tutte le righe che hanno quel valore per l'attributo provincia, ottenendo così una nuova tabella.

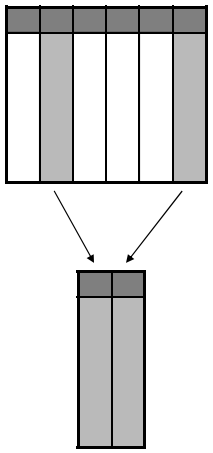
9

Proiezione

La **proiezione** genera una nuova relazione estraendo dalla tabella iniziale due o più colonne corrispondenti agli attributi prefissati.

La tabella ottenuta potrebbe avere ***più righe uguali***: in questo caso occorre richiedere che ne venga conservata solo una.

La relazione risultante ha **grado minore o uguale** al grado della relazione di partenza; la **cardinalità** è **uguale** a quella di partenza.



10

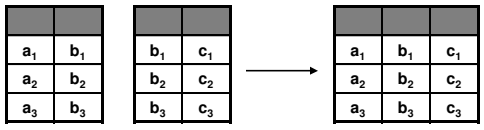
Proiezione: esempio

Se si vuole l'elenco dei codici di attività dei clienti con i relativi codici degli agenti, occorre applicare alla relazione *Clienti* l'operazione di proiezione secondo gli attributi *Id_Attività* e *Id_Agente*.

11

Congiunzione

La **congiunzione** (*join*) serve a combinare due relazioni aventi uno o più attributi in comune, generando una nuova relazione contenente le righe della prima e della seconda tabella, che possono essere combinate secondo i valori uguali dell'attributo comune.



Il **grado** della relazione generata è uguale a $\mathbf{N_1+N_2-1}$, dove N_1 e N_2 sono i gradi delle relazioni di partenza; la **cardinalità non è prevedibile a priori**.

12

Combinazioni di operazioni

Gli operatori possono essere applicati alle tabelle anche in successione, combinandoli tra loro in vario modo.

Vengono così effettuate interrogazioni sulle relazioni ottenute come risultato di un'interrogazione precedente.

Esempio

Ottenere l'elenco delle ragioni sociali e degli agenti per i clienti che hanno il codice di attività pari a 3109.

Occorre applicare la seguente sequenza di operazioni:
- Selezione di *Clienti* per Id*_Attività* = 3109
- Congiunzione della relazione ottenuta su *K_Agente* e di *Agenti* su *Id_Agente*
- Proiezione della relazione ottenuta su *RagioneSociale*, *NomeAgente*

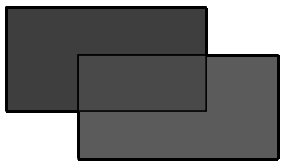
13

Operazioni tra tabelle con struttura omogenea

Due tabelle hanno **struttura omogenea** se hanno colonne con lo stesso numero di attributi, dello stesso tipo e nello stesso ordine.

In questo caso si possono applicare le usuali operazioni sugli insiemi.

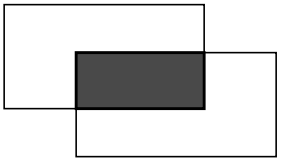
Unione
Genera una nuova tabella che contiene le righe della prima e della seconda tabella con riduzione a una di quelle ripetute.



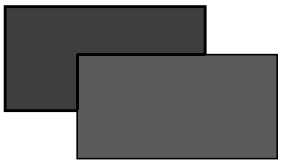
14

Operazioni tra tabelle con struttura omogenea

Intersezione
Genera, a partire da due tabelle omogenee, una nuova tabella che contiene soltanto le righe comuni.



Differenza
Genera una nuova tabella che contiene soltanto le righe della prima tabella che non sono contenute nella seconda.



15

La normalizzazione

Obiettivi della normalizzazione:
evitare la ripetizione e la ridondanza dei dati, durante la fase di definizione della struttura di una tabella, al fine di evitare futuri problemi nelle successive fasi di trattamento della tabella stessa, tramite operazioni di modifica o cancellazione di record in essa contenuti.

In pratica:
Si tratta di un insieme di operazioni, tramite le quali, a partire da una data tabella (non normalizzata), vengono create nuove tabelle, seguendo opportune regole, che trasformano la tabella originaria, in altre tabelle.

In ogni caso deve essere garantito che la trasformazione di una tabella in altre tabelle di forma normale superiore non provochi la perdita di informazioni.

16

Concetti chiave:

- ***chiave*** o ***chiave primaria*** : è un insieme di uno o più attributi che identificano in modo univoco un record della tabella.
- ***Chiave candidata*** : è un in insieme di uno o più attributi che possono svolgere la funzione di chiave (ci possono essere diverse chiavi candidate, ma una sola chiave primaria).
- ***Attributo non chiave*** : è un campo della tabella che non fa parte della chiave primaria.

17

Prima forma normale

Una tabella è in **1FN** se rispetta i requisiti fondamentali del modello relazionale cioè:

- Tutte le righe della tabella contengono lo **stesso numero di colonne**
- Gli attributi rappresentano **informazioni elementari**
- I valori che compaiono in una colonna sono dello stesso tipo, cioè appartengono allo **stesso dominio**
- **Non ci devono essere due righe con gli stessi valori nelle colonne** (chiave primaria)

18

Esempio di entità non normalizzata

Dipendenti

Matricola	Nome	Indirizzo	Familiari a carico

L'attributo *Familiari a carico* non è elementare in quanto è costituito da un gruppo di attributi ripetuti dello stesso tipo (i nomi dei familiari dell' i- esimo dipendente).

19

Tabelle normalizzate in 1FN

Soluzione: la tabella precedente viene scissa nella seguenti due tabelle

Dipendenti

Matricola	Nome	Indirizzo

Familiari

CodiceFam	NomeFam	K_Matricola

Vantaggi:
* Abbiamo 2 elementi distinti che rappresentano meglio la realtà. Inoltre è più facile aggiungere nuovi attributi.
* Risultano semplificate le operazioni di inserimento, cancellazione e modifica.

20

Seconda forma normale

Una tabella è in **2FN** quando è in 1FN e **tutti i suoi attributi non chiave dipendono unicamente dall' intera chiave**, cioè non possiede attributi che dipendono soltanto da una parte della chiave.

La seconda forma normale

elimina

la **dipendenza parziale degli attributi dalla chiave** e riguarda il caso di tabelle con chiavi composte, cioè formate da più attributi.

21

Es: si abbia un inventario di merci, le quali si trovano in alcuni magazzini dislocati in località diverse. Le informazioni essenziali possono essere rappresentate con la seguente tabella:

Merci

Id_merce	Magazzino	Quantità	LocalitàMagazzino

Osservazioni:

- la chiave è composta, in quanto il solo codice non basta per identificare la merce, la quale può essere presente in magazzini diversi.
- l'indirizzo del magazzino riguarda solo l'attributo Magazzino, quindi l'attributo"LocalitàMagazzino" dipende solo da una parte della chiave.

La tabella non è quindi in 2FN

22

La tabella vista non è in seconda forma normale
e ciò può provocare problemi di questo genere:

- La località del magazzino è ripetuta per tutte le righe dei prodotti presenti in quel magazzino;
- se la località del magazzino cambia, le righe delle merci presenti in quel magazzino devono essere aggiornate;
- La ridondanza può provocare l'inconsistenza delle informazioni (località scritta in modo differente in righe diverse / alcuni record vengano aggiornati ed altri no);
- Se in un certo periodo non ci fossero merci presenti in un magazzino,si perde l'informazione sulla località . La cancellazione di righe potrebbe quindi determinare una perdita complessiva di informazioni nella base di dati.

23

Risoluzione del problema:

La soluzione consiste nel costruire nuove tabelle, a partire dalla tabella non normalizzata, togliendo dalla tabella di partenza gli attributi che dipendono solo parzialmente dalla chiave primaria.

Es: nel caso precedente otterremo:

Merci

Id_merce	Magazzino	Quantità

Depositi

Magazzino	LocalitàMagazzino

Queste due tabelle sono in **2FN**

24

Terza forma normale

Una relazione è in terza forma normale (**3FN**) quando è in seconda forma normale e **tutti gli attributi non-chiave dipendono direttamente dalla chiave**, cioè non possiede attributi non-chiave che dipendono da altri attributi non-chiave.

La terza forma normale

elimina

quindi **la dipendenza *transitiva* degli attributi dalla chiave**.

25

Es: si consideri la gestione anagrafica di un'associazione di studenti di scuole diverse. Le informazioni più importanti siano rappresentate con la seguente tabella:

Studenti

Nome	Scuola	TelefonoScuola

Osservazioni:
Il nome è l'attributo chiave, e il telefono della scuola, pur essendo un'informazione che riguarda lo studente, dipende però dalla scuola cui lo studente è iscritto.
Nella tabella è quindi presente un attributo non-chiave (TelefonoScuola) che dipende da un altro attributo non-chiave (Scuola). La tabella non è in terza forma normale in quanto l'attributo TelefonoScuola dipende solo transitivamente dalla chiave (Nome).

26

Possibili anomalie

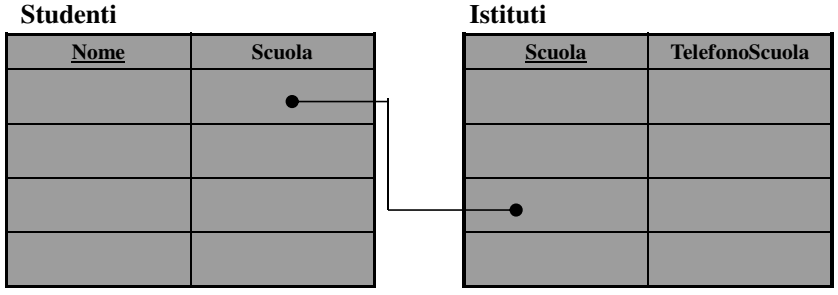
- il telefono di una scuola sarà **ripetuto** per ogni studente appartenente a quella scuola;
- se il telefono di una scuola cambia, occorrerà **modificare tutte le righe** contenenti studenti di quella scuola;
- la **ridondanza** può provocare **inconsistenza**, in quanto ci potrebbero essere numeri di telefono differenti, in righe diverse, per la stessa scuola, nel caso in cui questi siano stati scritti in modo diverso, oppure l'aggiornamento non sia stato fatto su tutte le righe;
- se una scuola non ha nessuno studente appartenente all'associazione, oppure gli studenti iscritti di una scuola escono tutti dall'associazione, non potremmo avere alcuna informazione sul telefono della scuola, con una conseguente perdita di informazioni.

27

Risoluzione del problema:

La normalizzazione in **3FN** si ottiene scomponendo la tabella di partenza in nuove tabelle, nelle quali tutti gli attributi dipendono unicamente e direttamente dalla chiave, togliendo gli attributi non-chiave che dipendono da un altro attributo non-chiave.

Es: nel caso precedente otterremo:



Queste due tabelle sono in **3FN**

28

L'integrità referenziale

Nella definizione dei concetti fondamentali del modello relazionale, oltre alla regola di "integrità sull'entità", che non consente valori nulli per la chiave, esiste una seconda regola di integrità, detta di "**integrità referenziale**".

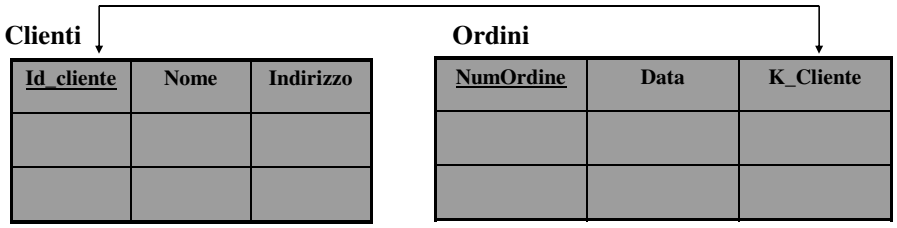
L'integrità referenziale è un insieme di regole del modello relazionale che garantiscono l'**integrità dei dati** quando si hanno relazioni associate tra loro attraverso la chiave esterna.
Queste regole servono per:

- rendere valide le associazioni tra le tabelle
- per eliminare gli errori di inserimento, cancellazione o modifica di dati collegati tra loro.

L'integrità referenziale viene rispettata **quando** per ogni valore non nullo della chiave esterna, esiste un valore corrispondente della chiave primaria nella tabella associata.

29

Es: si consideri un database relazionale che contiene una tabella dei clienti e una tabella degli ordini, in cui il codice del cliente della tabella Ordini è associato alla chiave della tabella Clienti:



Id_cliente	Nome	Indirizzo

NumOrdine	Data	K_Cliente

Applicare l'integrità referenziale al database significa garantire che un valore presente nella tabella Ordini per la chiave esterna K_Cliente abbia un corrispondente valore di Id_cliente in una delle righe della tabella Clienti.
Inoltre **non si deve consentire la cancellazione** di un cliente dalla tabella Clienti se ci sono righe nella tabella Ordini che si riferiscono ad esso.

30

Quando viene applicata l'integrità referenziale, è necessario osservare le seguenti regole pratiche:

- non è possibile immettere un valore nella chiave esterna della tabella associata, se tale valore non esiste tra le chiavi della tabella primaria. È possibile, comunque, immettere un valore nullo nella chiave esterna, per rappresentare il fatto che le righe non sono correlate. Per esempio un ordine non può essere assegnato ad un cliente che non esiste nella tabella Clienti.

- non è possibile eliminare una riga dalla tabella primaria, se esistono righe legate ad essa attraverso la chiave esterna nella tabella correlata. Per esempio non è possibile eliminare un cliente dalla tabella Clienti se ci sono ordini assegnati a quel cliente nella tabella Ordini.

- inoltre non si può modificare, come è ovvio, il valore alla chiave nella tabella primaria, se ad essa corrispondono righe nella tabella correlata. Per esempio non è possibile modificare il valore alla chiave di un cliente se ci sono ordini per quel cliente già registrati nella tabella Ordini.

31

I linguaggi per database

Le funzionalità del DBMS vengono attivate dall'utente usando appositi comandi, che costituiscono a tutti gli effetti un linguaggio attraverso il quale l'utente può comunicare con il sistema di elaborazione che gestisce il database.
I comandi che il DBMS mette a disposizione possono essere classificati nelle seguenti categorie di linguaggi:

- ❖ linguaggio per la descrizione dei dati, delle tabelle e delle viste, detto **DDL (Data Definition Language).**
- ❖ linguaggio detto **DMCL (Device Media Control Language)**, cioè il linguaggio di controllo dei supporti di memorizzazione dei dati.
- ❖ linguaggio per il trattamento (o manipolazione) dei dati contenuti nel database. detto **DML (Data Manipulation Language)**, che consente le usuali operazioni di accesso per inserimenti. modifiche o cancellazioni.
- ❖ linguaggio per fissare i vincoli di integrità, per stabilire le autorizzazioni agli accessi e i tipi di permessi consentiti agli utenti (inserimento di nuovi dati. sola lettura, modifica dei dati). **detto DCL (Data Control Language).**
- ❖ linguaggio per le interrogazioni alla base di dati, detto **Query Language**, che consente il ritrovamento dei dati che interessano, sulla base dei criteri di ricerca richiesti dall'utente

32

I linguaggi per database

La diffusione del modello relazionale ha poi favorito l'uso prevalente di linguaggi non procedurali: in questo modo l'utente non ha la necessità di conoscere ne le modalità con le quali le informazioni sono state fisicamente registrate, ne i cammini per ritrovare le informazioni contenute nella base di dati. Le informazioni vengono ritrovate effettuando interrogazioni e controllando il valore di verità di determinate condizioni, senza indicare le operazioni necessarie per arrivare alle informazioni richieste.

Si parla allora di *linguaggio per basi di dati*, intendendo un insieme completo di comandi che consente e facilita le operazioni di definizione del database, di manipolazione dei dati, e di interrogazione da parte degli utenti: vengono cioè unificate in un unico linguaggio le funzioni dei linguaggi DDL, DMCL, DML, DCL e Query Language.

I linguaggi per database relazionali si basano sulla visione tabellare dei dati, che facilita l'utente in quanto non contiene nessuna informazione sul percorso per l'accesso fisico ai dati.

33

Gli utenti

Un database viene utilizzato da persone diverse, per funzioni e per applicazioni diverse:

- **L'Amministratore della Base di Dati (DBA, Database Administrator)**, con i seguenti compiti:

1. Implementazione del modello logico del database nel sistema di elaborazione sui supporti fisici delle memorie di massa;
2. Gestione e trattamento dei dati (controllo di inserimenti, modifiche, cancellazioni);
3. Autorizzazione degli accessi;
4. Definizione delle viste per accessi parziali di utenti alla base di dati;
5. Controllo dei programmi applicativi che richiedono l'uso del database;
6. Manutenzione del database nel tempo, in termini di efficienza e di ottimizzazione delle risorse;
7. Controllo sugli interventi di recupero, nel caso di cattivi funzionamenti, e sulle copie di salvataggio periodiche;
8. Controllo della disponibilità degli spazi su memoria di massa.

34

Gli utenti

❖ **I programmatori**, che intendono utilizzare per le loro applicazioni i dati organizzati in un database, utilizzano un linguaggio DML, oppure comandi che sono un'estensione dei tradizionali linguaggi di programmazione, oppure un linguaggio specifico per basi di dati.

❖ **Gli utenti finali** possono accedere alla base di dati attraverso i comandi di un linguaggio di interrogazione (query language), oppure, per utenti finali ancora meno esperti,attraverso interfacce software che presentano sul video un menù o icone.

35

Storia SQL

- Structured Query Language
  (solitamente pronunciato "sequel").
- La "vita" di SQL inizia nel 1970 presso i laboratori di ricerca IBM di San Jose, dove E. F. Codd e altri svilupparono il modello di database relazionale che diede origine al sistema noto come DB2.
- Quando i database relazionali proliferarono nel corso degli anni Ottanta, SQL fu codificato per l'utilizzo nell'ambito dell'Information Technology commerciale. Nel 1986, l'American National Standards Institute (ANSI) e l'International Standards Organization (ISO) stabilirono il primo standard del linguaggio.

36

Storia

- Le revisioni successive dello standard, nel 1989 e nel 1992, aggiunsero il controllo di base dell'integrità dei dati e le funzioni per la loro definizione e manipolazione.

- La specifica di SQL del 1992 è la versione più attuale.
  Il nuovo aggiornamento SQL3, conosciuto anche come SQL-99, ha funzionalità rivolte all'integrazione di questo linguaggio nelle basi di dati orientati agli oggetti e il supporto per sistemi basati sulla conoscenza (I.A.).

37

Caratteristiche

- Linguaggio ad alto livello

- Paradigma di programmazione logico
  Sta al sistema di gestione del database analizzare la richiesta in rapporto alla propria struttura e stabilire quali operazioni sia necessario eseguire per recuperare l'informazione richiesta

- Si presta alle architetture client/server

38

Datatype standard

Character(n)	
Date	
Time	
Integer	
Smallint	
Real	
Float	

39

Istruzioni

- **DDL:** CREATE, DROP e ALTER TABLE
- **DML:** INSERT, DELETE e UPDATE
- **TRANSAZIONI:** COMMIT, ROLLBACK e SAVEPOINT
- **SQL:** SELECT
- **SICUREZZA:** GRANT e REVOKE

40

CREATE

- CREATE TABLE …

  Crea una nuova tabella nella base di dati

- CREATE VIEW …

  Crea una nuova vista nella base di dati

- CREATE INDEX …

  Crea un nuovo indice su uno o più attributi di una tabella

41

CREATE TABLE

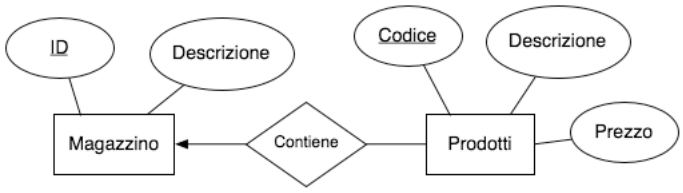
Per creare una nuova tabella nella base di dati:

CREATE TABLE *table_name*
(
 field$_1$ type$_1$ [default] [constraint$_1$],
 field$_2$ type$_2$ [default] [constraint$_2$],
 …
 field$_n$ type$_n$ [default] [constraint$_n$]
);

42

Esempio - CREATE TABLE

Diagramma Entità-Relazione



Modello Relazionale

Magazzino(ID_Magazzino, Descrizione);
Prodotti(ID_Prodotto, Descrizione, Prezzo);
Contiene(K_Magazzino, K_Prodotto);

43

Esempio - CREATE TABLE

```
create table Magazzino
(
   id _m integer primary key,
   descrizione char[100]
);;
```

Master

```
create table Prodotti
(
   Id_prod integer primary key,
   descrizione char[100],
   prezzo integer,
   K_magazzino integer,
      foreign key (K_magazzino) references Magazzino(id_m)
);;
```

Child

44

DROP

- DROP TABLE *table_name*;

  Rimuove la tabella dalla base di dati e i relativi indici

- DROP VIEW *view_name*;

  Rimuove la vista dalla base di dati

- DROP INDEX *index_name*;

  Rimuove l'indice dalla rispettiva tabella

45

ALTER TABLE

- ALTER TABLE *table_name*
  ADD (*col_name* type [constraint]);

  Aggiunge una o più colonne alla tabella

- ALTER TABLE *table_name*
  DROP (*col_name*);

  Rimuove una o più colonne della tabella

46

INSERT

INSERT INTO *table_name*
VALUES (V_1, V_2, …, V_N);
Inserisce la n-upla (v_1, …, v_n) nella tabella.

47

INSERT

INSERT INTO Personale

($campo_1$,….,$campo_n$)

VALUES
(‘Bianchi’,….,’Impiegato’);

48

DELETE

Per cancellare uno o più record da una tabella:

DELETE FROM *table_name*
WHERE *condizione*;

DELETE FROM Personale
WHERE Matricola='A124'

49

SELECT

SELECT …
FROM…
WHERE …

Seleziona tutti i dipendenti che abitano in provincia di Milano

SELECT *
FROM Personale
WHERE Prov = 'MI';

Seleziona tutte le province del personale:

SELECT DISTINCT Prov
FROM Personale;

50

Funzioni di Aggregazione

• COUNT conta il numero di righe
•SUM somma i valori contenuti in un campo
•AVG calcola la media dei valori di un campo
•MIN / MAX

•

SELECT SUM(StipBase)
FROM Personale
WHERE livello=5;

SELECT AVG(StipBase)
FROM Personale
WHERE funzione='Impiegato'

51

Condizioni di ricerca

Seleziona l'elenco dei dipendenti che sono stati assunti tra il 3/3/97 e il 3/3/98

```
SELECT Cognome, Nome, Funzione
  FROM Personale
  WHERE assunzione BETWEEN  03/03/97 AND
                    03/03/98;
```

Seleziona i dipendenti della provincia di Milano e di Como

```
SELECT Cognome, Nome, Funzione
  FROM  Personale
  WHERE Prov IN ('MI', 'CO');
```

52

Condizioni di ricerca

Selezionare gli impiegati il cui cognome inizia per 'Ros'

```
SELECT  Cognome, Nome
  FROM  Personale
  WHERE Cognome LIKE  'Ros%';
```

53

Interrogazioni nidificate

Seleziona l'elenco dei dipendenti che hanno uno stipendio inferiore alla media:

```
SELECT  Cognome, Nome
  FROM  Personale
  WHERE  Stipendio < (SELECT  AVG(Stipendio)
                FROM  Personale);
```

54

Ordinamento

Seleziona l'elenco dei dipendenti ordinandolo per cognome:

```
SELECT Cognome, Nome, Funzione
  FROM  Personale
  ORDERT BY  Cognome (ASC);
```

55

Raggruppamenti

Seleziona la lista delle funzioni dei dipendenti con la somma degli stipendi e il numero di dipendenti appartenenti alle varie funzioni:

```
SELECT  Funzione, SUM(Stipendio), count(*)
  FROM  Personale
  GROUP BY  Funzione;
```

raggruppa un insieme di righe con lo stesso valore HAVING

56

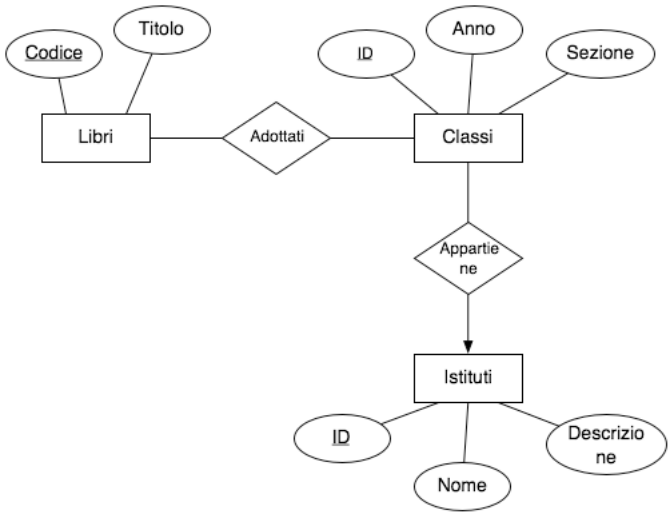
Esercizio

All'inizio dell'anno scolastico un rappresentante di libri vuole avere una quadro riassuntivo dell'anno scolastico precedente riguardo i libri adottati nelle varie classi dei vari istituti.

In particolare, si vuole avere a disposizione un report dove per ogni libro sia associato l'istituto ed il numero di classi (dell'istituto) che ha adottato il libro.

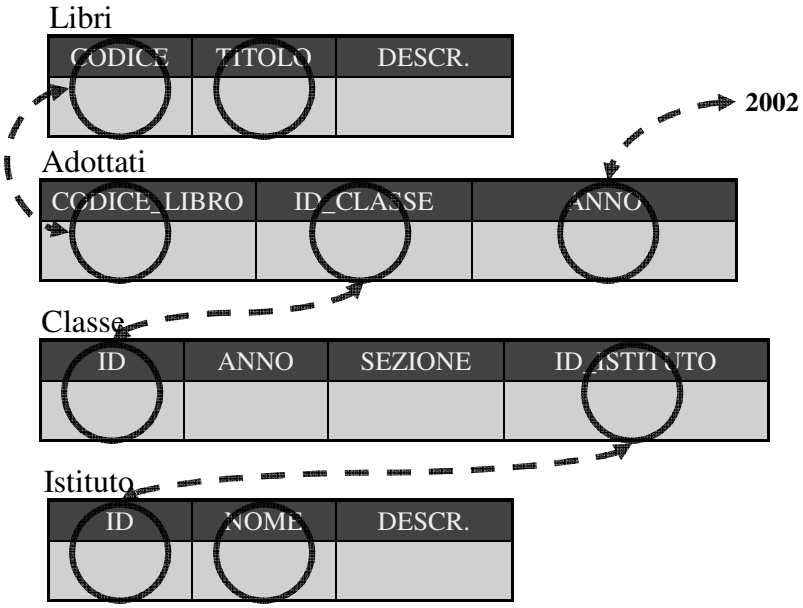
57

Esercizio



58

Esercizio



59

Esercizio

```
SELECT  titolo, istituto, numero
  FROM  (SELECT libri.titolo as titolo,
              istituto.nome as istituto,
              COUNT (*) as numero
         FROM libri, classi, adottati, istituto
          WHERE libri.codice = adottati.codice_libro and
                adottati.id_classe = classi.id and
                classi.id_istituto = istituto.id and
                adottati.anno = 2002
           GROUP BY  titolo, istituto)
  ORDER BY  titolo, istituto;
```

60