MySQL

di Sanarico Andrea

MySQL è un DBMS (DataBase Management System) server, nel senso che è possibile accedere alle sue funzioni e ai suoi servizi da un host client remoto.

Per aprire il programma:

Destinazione: d:\programmi\xampp\mysql\bin\mysql -h localhost -u root -p (-p solo se avete la password)

Alcune funzioni

select version(); Indica la versione di MySQL

show databases; Visualizza tutti i database salvati

use nome_database; Permette di usare il database

show tables; visualizza le tabelle di un determinato database

describe nome_tabella; Visualizza gli attributi della tabella

select <insieme di colonne> from <nome tabella> where <condizione per selezionare le righe>; seleziona alcune colonne e alcune righe di una tabella

insert into nome_tabella values (valori); Inserisce nella tabella i valori inseriti. Per inserire stringhe bisogna inserire i valori dentro ad apostrofi es. insert into prova values ('cane',8,'prova');

insert into nome_tabella campo_della_tabella values (valore); Inserisce nella tabella solo i valori di un determinato campo

Creare un nuovo database

create database nome_database; Crea un nuovo database

use nome_database; Predispone le prossime operazioni sul database

Creazione di una tabella

Studenti

Matricola	Alunno	Data_di_nascita	Indirizzo	Telefono

```
create table Studenti    Istruzione per creare una tabella nel DB seguita da il nome della tabella
(Matricola character(4) not null primary key, not null obbliga a riempire il campo e primary key assegna la
Alunno varchar(30) not null,                  chiave principale
 Data_di_nascita date not null,
 Indirizzo varchar(40) not null,
 Telefono varchar(10));
```

-Creare tabelle con attributo chiave composto

```
create table Classe
(Anno varchar(7) not null,
 Sezione varchar(2) not nul,
 Indirizzo varchar(15),
 primary key (anno, sezione), attributo chiave composto
 Citta varchar(30) default 'Martina Franca');
```

Per ogni attributo possiamo assegnare un valore di default che viene assegnato automaticamente dal DBMS se l'utente non inserisce alcun valore.

-Altro esempio di tabella

```
create table Cittadino
(Codice_Fiscale character (16) not null primary key,
 Cognome varchar(30) not null,
 Nome varchar(40) not null,
 Data_di_nascita date not null,
 Sesso character(1) not null,
 Citta_nascita varchar (30) not null,
 unique (Cognome,Nome,Data_di_nascita,Sesso,Citta_nascita));
```

Con Primary Key è implicita l'assunzione not null, con unique no, inoltre se in una tabella manca primary key, viene assunta come chiave principale il primo attributo indicato con unique.

Chiavi esterne o secondarie

Calciatori

Tessera	Cognome	Nome	Ruolo	Nome_Squadra

Rappresentazione n-->1

Squadre

Nome	Citta	Presidente

```
create table Squadre
(Nome varchar(30) not null primary key,
 Citta varchar(40) not null,
 Presidente varchar (50));
```

```
create table Calciatori
  (Tessera varchar(10) not null primary key,
 Cognome varchar(20) not null,
 Nome varchar(20) not null,
 Ruolo Cognome varchar(20),
 Nome_Squadra varchar(30),
 foreign key (Nome_Squadra) references Squadre (Nome)

 on delete set null on update cascade);
```

Quando si definisce una chiave esterna occorre aggiungere un vincolo di integrita referenziale, cioè devo comunicare al DBMS di controllare che un valore inserito per la chiave esterna sia presente come valore nella chiave principale nella tabella dal lato 1 a cui essa fa riferimento. L'integrita referenziale è assicurata da MySQL solo se il DB è del tipo INNODB.

-Cosa succede se una chiave della tabella dal lato 1 è cancellata (delete) o aggiornata (update)?

È pregiudicata l'integrita referenziale.
Per cercare di ripristinarla possiamo adottare le seguenti strategie:

- Set null (i valori della chiave esterna rimasta orfana vengono impostati a null automaticamente dal DBMS)

- Set default (viene impostato un valore di default se presente)

- Restrict (Il DBMS non consente la cancellazione o l'aggiornamento della chiave principale)

-Cascade (Tutte le righe della tabella dal lato n vengono cancellate o aggiornate a cascata)

Posso indicare l'azione set default solo se ho definito il valore di default per la chiave esterna, e posso indicare set null solo se non ho definito le chiavi esterne.

Associazione n --> m

Docente (entita)

Matricola	Cognome	Nome	Materia

Insegna (associazione)

Prof	Nome_classe	N_ore

Classe (entita)

Nome	Indirizzo

```
Create table Docente
(Matricola character(5) primary key not null,
 Cognome varchar(30) not null,
 Nome varchar(30) not null,
 Materia varchar (20)) type=innodb;
```

```
Create table Classe
(Nome varchar(3) primary key not null,
 Indirizzo varchar(20) not null)type=innodb;
```

```
Create table Insegna
(Prof character(5) not null,
```

```
 Nome_classe varchar(3) not null,
 N_ore tinyint,
primary key (Prof,Nome_classe),
foreign key (Prof) references Docente (matricola) on delete cascade on update cascade,
foreign key (Nome_Classe) references Classe (Nome) on delete cascade on update cascade) type=innodb;
```

Inserimento valori nella tabella

Per inserire dei valori all'interno della tabella bisogna utilizzare la seguente espressione:

```
insert into nome_tabella values (valori);
```

Per inserire stringhe bisogna inserire i valori dentro ad apostrofi. Attraverso questa procedura bisogna inserire tutti i campi della tabella.

```
es. create table cani
      (nome varchar(20) primary key not null,
       eta tinyint not null,
       razza varchar(20) not null);

      insert into cani values ('Lucky',8,'bastarda');
```

Se vogliamo inserirne sollo alcuni campi lo facciamo in questo modo

```
insert into cani  (razza) values ('bassotto');
```

Salvare in un file (.txt) il codice del database

Questa procedura permette di salvare il nostro codice all'interno di un file di testo. Per far ciт, senza entrare nel programma, inseriamo la seguente espressione:

```
mysqldump -u root nomedatabase> destinazione\nomefile.txt
es. mysqldump -u root prova > c:\codiceprova.txt
```

Per estrarre direttamente dal file il testo facciamo nel seguente modo:

- Creamo un nuovo databases;
- Usciamo da mySQL;
- Digitiamo la seguente espressione

```
mysql -u root nomenuovodb <c:\destinazione\nomefile.txt
es.mysql -u root nuovodatabase<c:\codiceprova.txt
```

Tipi di dato

Prima di tutto vediamo da quali dati è composto un database di MySQL. I tipi di dato, o domini della tabella, possono essere dichiarati mediante query di creazione tabella, che vedremo nelle pagine che seguono, oppure mediante ambiente GUI (*Graphics User Interface).* L'ambiente GUI è previsto sia in Access sia in MySQL Front. Innanzitutto occorre ricordare che MySQL è un database relazionale: permette quindi di creare alcune relazioni tra tabelle diverse. Come la maggior parte dei DBMS è formato da database, e ciascun database è formato da tabelle. Le tabelle sono formate da campi (colonne) e record (righe), secondo lo schema di qualunque altro database. I campi possiedono un nome (attributo) e un tipo (dominio), che devono essere decisi durante la fase di creazione delle tabelle; li riassumiamo nella tabella seguente:

MySQL

Tipo	Descrizione		Dimensione massima/formato
Bigint	intero lunghissimo	INTERI	da -2^{63} a $2^{63}-1$
Integer	intero lungo	INTERI	da -2^{31} a $2^{31}-1$
Smallint	intero	INTERI	da -32768 a 32767
Tinyint	intero ridotto	INTERI	da-128 a+127
Double	reale a doppia precisione	REALI	da $\pm 2.225*10^{-308}$ a $\pm 1.798*10^{308}$
Float	reale a singola precisione	REALI	da $\pm 1.176*10^{-38}$ a $\pm 3.403*10^{38}$
Decimal	decimale memorizzato come stringa	REALI	da $\pm 2.225*1O^{-308}$ a $\pm 1.798*1O^{308}$
Date	data in formato US (aaaa-mm-gg)	DATA	dal '1000-01-01' a '9999-12-31
Time	orario	DATA	formato HH:MM:SS
Datetime	data con ora	DATA	aaaa-mm-gg hh:mm:ss
Year	anno	DATA	formato AAAA
Character	stringa a lunghezza fissa	STRINGA	da 0 a 255 caratteri
Varchar	stringa a lunghezza variabile	STRINGA	da 0 a 255 caratteri
Text	campo testo a lunghezza fissa	STRINGA	da 0 a 65535 caratteri
Mediumtext	campo testo (memo)	STRINGA	16 MB di caratteri -1
Blob MediumBlob Long blob	immagini jpeg, bmp, gif (Binary Large Object), oppure file binary in esadecimale o di testo	OGGETTI	fino 64 KB fino a 16 MB circa 4 GB

MS Access

Tipo	Descrizione		Dimensione massima/formato
Bit	un bit memorizzato in 1 Byte	INTERI	0 (false) OPPURE 1 (true)
Tinyint	intero ridotto	INTERI	da 0 a 255
Smallint	intero	INTERI	da-32.768 a 32.767
Money	intero lungo a 8 Byte	INTERI	±922.337.203.685.477,5808
Integer	intero lungo	INTERI	da - 2.147.483.648 a
Float	reale a doppia precisione a (8 Byte)	REALI	da -1.79^{308} a $-4,94^{-324}$
Real	reale a singola precisione a (4 Byte)	REALI	da $-3,4^{38}$ a $-1,4^{-45}$
Decimal	reale a precisione variabile a (17 Byte)	REALI	
Character	stringa a lunghezza fissa	STRINGA	da 0 a 255 caratteri
text	campo testo a lunghezza fissa	STRINGA	da 0 a 2,14 GB
datatime	data e ora	DATA	un valore di data o di ora compreso tra 100 e 9999
Image	oggetti (OLE - Object Linked and Embedded -Microsoft)	OGGETTI	da 0 a 2,14 GB

legenda il seguente colore ---- indica una potenza. Per problemi di SW non è possibile agire diversamente. Es. -3.438 indica -3.4 elevato a 38.

Il tipo per le stringhe a lunghezza fissa è opportuno usarlo solo quando l'attributo contiene valori di lunghezza fissa. ES. codice bancomat, codice fiscale, partita IVA, a non per esempio per i nome di persone ecc...
In questo caso viene usato VARCHAR. Per un attributo con valori numerici per il quale non farò mai calcoli, è opportuno definire un dominio di tipo stringa.

Simboli operazionali

= uguale
<> o != diverso
> Maggiore
< Minore
>= Maggiore o uguale
<= Minore o uguale
like
between
in

Interrogazioni di un DBMS

SQL

-DDL (Data definition language)

-QL (Query language) select

-DCL (Data control language)

-DML (Data manipulation language)

Esempi

select <insieme di colonne> from <nome tabella> where <condizione per selezionare le righe>

Attraverso l'istruzione select possiamo effettuare le seguenti operazioni:

- proiezione: seleziona solo alcune colonne di una tabella (fig.1,2);
- Selezione: seleziona alcune riche di una tabella (fig.3);
- Congiunzione: riguarda il collegamento tra 2 tabelle che avviene atttraverso le chiavi esterne. É necessaria quando occorre reperire dati presenti in tabelle differenti (fig.4,5,6);

Calciatori

Tessera	Cognome	Ruolo	Reti	Nome_Squadra
A001	Adriano	Punta	120	Inter
A002	Maldini	Difensore	3	Milan
A003	Inzaghi	Attaccante	30	Milan

fig.1 select Tessera, Cognome, Ruolo from Calciatori;
fig.2 select Cognome, Ruolo from Calciatori where Reti>15

Calciatori

Tessera	Cognome	Ruolo
A001	Adriano	Punta
A002	Maldini	Difensore
A003	Inzaghi	Attaccante

Calciatori

Cognome	Ruolo	Reti	Nome_Squadra
Adriano	Punta	120	Inter
Inzaghi	Attaccante	30	Milan

fig.3 select * from Calciatori where Nome_Squadra='Milan';

Calciatori

Tessera	Cognome	Ruolo	Reti	Nome_Squadra
A002	Maldini	Difensore	3	Milan
A003	Inzaghi	Attaccante	30	Milan

Il risultato di una select è un ulteriore tabella. Posso visualizzare l'intera tabella senza eseguire ne una proiezione ne una selezione con l'istruzione:

select * from Calciatori;

Es.

Squadra

Nome	Citta
Inter	Milano
Milan	Milano
Juventus	Torino

Visualizzare Cognome, Ruolo e città dei calciatori che hanno segnato più di 15 reti.

fig.4 select Cognome, Ruolo, Citta from Calciatori join Squadra on Nome_Squadra=Squadra.Nome where Reti>15;

oppure

fig.5 select Calciatore.Cognome, Calciatore.Ruolo, Squadra.Citta;

oppure

fig.6 select Cognome, Ruolo, Citta from Calciatori, Squadra where Nome_Squadra=Squadra.Nome and Reti>15;

Dopo from mettiamo in collegamento 2 tabelle (JOIN);

Altri esempi

select Cognome, Ruolo, Citta as Residenza from.....where.....;

"AS" cambia il nome alla colonna che vberrà visualizzata (da citta a residenza)

select a.cognome, a.ruolo, b.citta from calciatori a, squadra b where a.nome=b.nome and a.reti>15;

select calciatori.cognome, calciatori.ruolo, squadra.citta from calciatori,squadra where calciatori.nome_squadra=squadra.nome and reti>15;

Operazioni sulle colonne

Prodotti

Codice	Nome	Descrizione	Prezzo	Sconto	Quantità
A001	Penna	Bic	0.50	2%	100

select Nome, Prezzo, (Prezzo -(Prezzo*Sconto/100)) as Prezzo_scontato from Prodotti where Nome='Penna';

Nome	Prezzo	Prezzo_scontato
Penna	0.50	0.49

Operazione JOIN

L'operazione di join viene fatta dal DBMS facendo prima un prodotto cartesiano tra le 2 tabelle e selezionando fra tutte le righe ottenute quelle che soddisfano le condizioni della join. É quindi un operazione onerosa in termini di CPU e occupazione di memoria. La chiave esterna, pur rappresentando una ridondanza è necessaria per collegare 2 tabelle tramite l'operazione di join.

Simboli operazionali Avanzati

Prodotti

Codice	Nome	Descrizione	Prezzo	Sconto	Quantità
A001	Penna	Bic	0.50	2%	100

-Between

Selezionare gli attributi con costo che và da 10 a 20 euro:

select * from prodotto where costo>=10 and costo<=20;

oppure

select * from prodotto where costo between 10 and 20;

-Like

Selezionare tutti gli articoli in cui nome inizia cola la parola “penna”:

Select * from prodotti where nome like “penna%;

il simbolo % include 0,1 o più caratteri qualunque;
il simbolo _ (underscore) include 0,1 caratteri; più ce ne sono più caratteri ci possono essere.

-In

Selezionare tutti gli articoli il cui sconto sia o del 5, o del 10 o del 20%:

select * from prodotti where sconto in (5,10,20);

IN consente di verificare se il valore del campo è presente o meno in un insieme di valori. Vale sia per i numeri che per le stringhe.

-Distinct

Visualizzare tutti gli articoli praticati sui prodotti:

select distinct sconto from prodotto;

DISTINCT consente di eviter la ripetizione dei valori di un campo nella tabella risultante. Se oltre alla colonna a cui distinct fa riferimento volgio mostrare altre colonne, il risultato non mostrerà mai due righe uguali.

Operatori di Aggregazione

Gli operatori di aggregazione lavorano su un gruppo di righe.

auto

Marca	Modello	Prezzo	Tipo	Giorni_di_attesa
Fiat	Punto	15000	Null	30
Alfa Romeo	156	20000	Berlina	15
Alfa Romeo	155	25000	Sportiva	Null
Ford	Fodus	15000	Null	20
Audi	A3	25000	Sport	30
Ferrari	Testa Rossa	150000	Corsa	30

-Count (conta le righe)

select count (marca) from auto; **à** risultato uguale 6

particolarità:

select count (tipo) from auto; **à** risultato uguale 4

Count (nome colonna) conta quali righe ci sono corrispondenti a valori diversi da null per quella colonna.

select count (*) from auto; **à** risultato uguale 6 perché conta tutte le righe anche se ci sono righe con valore null

select count(distinct marca) from auto; **à** risultato uguale 5

-Sum (somma)

select sum(prezzo) as somma_prezzi from auto;

SUM cosidera i valori null come 0 ai fini del calcolo della somma.

-Avg (media)

select avg(prezzo) as media_prezzi from auto;

Nel calcolo della media AVG trascura i valori null, cioè il dato non viene considerato.
Sia la somma che la media si applicano solo su una colonna di tipo numerico. Gli operatori di aggregazione, se non si usano i raggruppamenti, non possono comparire insieme a singole colonne nell'istruzione select.

~~Select marca,avg(prezzo) from auto;~~

Invece è possibile:

select sum (prezzo) avg(prezzo) from auto;

-Min (Minimo)

Restituisce il minimo sia di campi numerici che alfanumerici. In quest'ultimo caso restituiscono la stringa più piccola.

-Max (Massimo)

Restituisce il massimo sia di campi numerici che alfanumerici. In quest'ultimo caso restituiscono la stringa più grande.

-Stddev (Scarto quadratico medio)

Restituisce lo SQM di una colonna numerica.

Raggruppamenti

Calciatori

Tessera	Cognome	Nome	Reti_fatte	Ruolo	Foto	Nome_Squadra

Select avg (Reti_fatte) as media_reti, Nome_Squadra from Calciatori group by Nome_Squadra;

Tutte le funzioni di raggruppamento si possono applicare a singoli gruppi che sono formati inserendo in ogni gruppo le righe che hanno lo stesso valore della colonna per cui si raggruppa. In questo caso, ogni funzione di raggruppamento opera sulle righe di ogni gruppo, ovvero gruppo per gruppo.

Scrivere la query che ci dice, squadra per squadra, il numero di calciatori:

select count (*) as numero_giocatori, Nome_Squadra from Calciatori group by Nome_Squadra;

Visualizzare, squadra per squadra, in numero di centrocampisti:

select count(*) as n_centrocampisti,Nome_Squadra from Calciatori chere Ruolo=centrocampista group by Nome_Squadra;

Se una select usa GROUP BY possono comparire come colonne solo il nome della colonna per cui raggruppiamo (deve comparire!) più altre funzioni di raggruppamento. Se uso where insieme a GROUP BY viene fatta prima una selezione delle righe in base alla condizione del where e poi delle righe selezionati vengono creati i raggruppamenti. Posso, infine, fare una selezione anche sui raggruppamenti con la clausula HAVING.

-Having (seleziona i gruppi)

Calcolare la somma delle reti fatte dai difensori, squadra per squadra, ma solo se tale somma supera 5 reti:

select sum (Reti_fatte) as reti_difensori,Nome_Squadra from Calciatori where ruolo=difensore group by Nome_Squadra having sum(Reti_fatte)>5;

Sotto Query

Calciatori

Tessera	Cognome	Nome	Reti_fatte	Ruolo	Foto	Nome_Squadra

Visualizzare la lista degli attaccanti che hanno segnato un numero di reti superiore alla media delle reti di tutta la serie A:

select avg(Reti_fatte) from Calciatori **à** risultato uguale 7

select * from calciatori where Reti_fatte>7 and ruolo=attaccant

```
 select *
from calciatori
where ruolo=attaccante andReti_fatte> select avg(Reti_fatte)
                                      from Calciatori;
```

Visualizzare cognome, nome, foto e squadra di appartenenza del capocannoniere di serie A

```
Select Cognome, Nome, Foto, Nome_Squadra
from Calciatori
where Reti_fatte=select max (Reti_fatte)
                 from Calciatori;
```

Join Esterno

Articoli

codice	nome	prezzo	Cod_fornitore
A001	Filtro olio	25	1724
A002	Filtro aria	15	1724
A003	Tubo x	11	

Associazione n a 1 **à**

Fornitori

p_iva	nome	citta
1724	Rossi	Torino
5555	Neri	Milano

Visualizzare tutti gli articoli con prezzo superiore a 10 euro con le città dei rispettivi fornitori, anche per gli articoli prodotti all'interno dell'azienda.

```
Select articoli.nome,prezzo,citta as citta_fornitore
from fornitori RIGHT JOIN articoli
on codice_fornitori=p_iva
where prezzo>10;
```

articoli.nome	Prezzo	Citta_fornitore
Filtro olio	25	Torino
Filtro aria	15	Torino
Tubo x	11	

il JOIN ESTERNO si comporta come un join normale (o inner join), in più mostra le righe della tabella di destra o di sinistra che non hanno una corrispondenza con le righe dell'altra tabella.

Mostrare nome,citta dei fornitori, con nomi dei rispettivi articoli forniti, anche per quei fornitori che attualmente non forniscono nulla.

```
Select fornitori.nome,citta,articoli.nome
From fornitori LEFTJOIN articoli
On codice.fornitore=p_iva;
```

Join multipli

Docente (entita)

Matricola	Cognome	Nome	Materia

Insegna (associazione)

Cod_docente	cod_classe	ore

Classe (entita)

Nome	Indirizzo

Visualizzare nome e materia del docente, clasi in cui insegna e le ore di insegnamento

```
Select docente.nome,materia,ore,classe.nome
From docenti JOIN insegna JOIN CLASSI
on matricola=cod_docente
on classi.nome=cod_classe;
```

Update

Per modificare un valore all'interno di una tabella si esegue in questo modo:

Articoli

codice	nome	prezzo	Cod_fornitore
A001	Filtro olio	25	1724
A002	Filtro aria	15	1724
A003	Tubo x	11	

```
Update Articoli set nome=tubo_acqua where codice='A003';
```

Risultato:

Articoli

codice	nome	prezzo	Cod_fornitore
A001	Filtro olio	25	1724
A002	Filtro aria	15	1724
A003	Tubo acqua	11	

Delete

Per cancellare un valore all'inteno di una tabella si esegue in questo modo:

Articoli

codice	nome	prezzo	Cod_fornitore
A001	Filtro olio	25	1724
A002	Filtro aria	15	1724
A003	Tubo x	11	

```
Delete from  where codice='A003';
```

Risultato:

Articoli

codice	nome	prezzo	Cod_fornitore
A001	Filtro olio	25	1724
A002	Filtro aria	15	1724

Per svuotare la tabella:

delete from Articoli;

Normalizzazione

La normalizzazione riguarda la progettazione di un database relazionale.

Cos'è?

È un processo di scomposizione delle relazioni in relazioni più piccole che consente di ridurre le ridondanze (generano inconsistenza) ed evita inconvenienti di aggiornamento, cancellazione o inserimento dei dati.

Quando si applica?

Dopo aver progettato il diagramma E/R e dopo aver definito le relazioni (tabelle).

-FASE1: Prima **F**orma **N**ormale (**1FN**)

Una relazione è in 1FN se ogni attributo delle relazione è atomico (non ulteriormente scomponibile).

Es.
Squadra (Nome, Citta, Presidente,elenco_giocatori(n_tessera,Cognome,Nome,Ruolo…))

à

Squadra (Nome, Citta, Presidente,elenco_giocatori)
Elenco_giocatori (n_tessera,Cognome,Nome,Ruolo…)

È obbligatorio che tutte le relazioni siano in 1FN, cioè che ogni attributo non sia ulteriormente scomponibile (atomico)

-FASE2: Seconda **F**orma **N**ormale (**2FN**)

Una relazione è in 2FN quando è in 1FN e in più ogni attributo non chiave dipende pienamente da tutta la chiave e non solo da una parte.

Cittadino (Cognome, Nome, GG,MM,AAAA, sesso,indirizzo, segno_zodiacale)

Dipende solo da mm e gg

Questa relazione non è in 2FN poiché l'attributo non chiave segno_zodiacale non dipende da tutta la chiave, ma solo da una sua parte, cioè da MM e GG.

--Passaggio in 2FN

Cittadino (Cognome, Nome, GG,MM,AAAA, sesso,indirizzo)
Zodiaco (gg,mm, segno_zodiacale)

Dall'esempio si evince che la relazione non normalizzata occupa più memoria e vi sono problemi per l'aggiornamento e la cancellazione.
Il "prezzo da pagare" per la normalizzazione è che: poiché le tabelle vengono suddivise in tabelle più piccole per ricercare un informazione potrebbe esserci la necessità di fare una JOIN.

FASE3: Terza **F**orma **N**ormale (**3FN**)

Una relazione è in 3FN quando è in 2FN e ogni attributo non chiave non dipende transitivamente dalla chiave, bensì direttamente.

Cittadino (Codice_fiscale, GG,MM,AAAA, sesso,indirizzo, segno_zodiacale)

Dipendenza transitiva

Se il progettista decide di inserire tutti gli attributi, che ha individuato in fase di analisi in un'unica relazione e procede poi alla normalizzazione in 3FN, ottiene le stesse relazioni che riceverebbe se traducesse in relazione il diagramma E/R progettato.

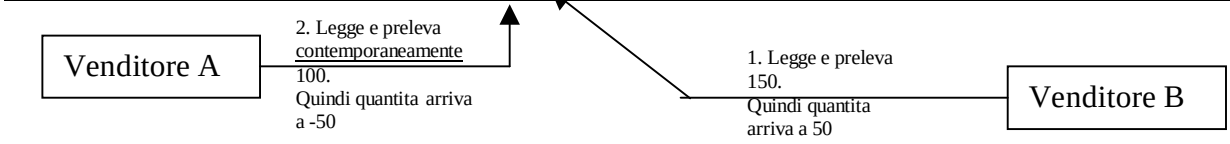
Transazioni

Risolve due problemi:

1. Accesso concorrente in scrittura da parte di più processi su una tabella
2. Integrità del database

Articoli

……	Quantità	……
	200	



Una transazione è l'insieme di più operazioni indivisibili su un db, ciò significa che ogni operazione della transazione viene eseguita nella RAM ma non su disco. Solo quando si conferma la transazione, tutte le operazioni di cui è composta vengono scritte a blocco sul disco. Se c'è una transazione in atto su una tabella, nessun altro processo può accedere alla tabella.
Con l'uso delle transazioni un solo processo per volta accede alle tabelle evitando problemi di accesso concorrente in scrittura….

Start Transaction (inizio transazione)

Commit è la fine ed è la conferma di una transazione
Rollback Si torna allo precedente all'inizio della transazione

Nella conferma di una transazione, o sono reindirizzate tutte le operazioni della transazione o non viene eseguita, o non viene eseguita nessuna (vale la regola o tutto o niente)
La transazione si può fare solo con tabelle innodb.

Grant e Revoke

L'amministrazione della base di dati, o comunque chi crea le relazioni del database, può stabilire anche il diritto di accesso per utenti specifici o per tutti gli utenti, nel caso di accessi multipli alle tabelle del database.
Il comando GRANT concede i permessi, specificando il tipo di accesso, le tabelle sulle queli è consentito l'accesso e l'elenco degli utenti ai quali è permesso di accedere.
Il tipo di accesso piò riguardare per esempio il diritto di modifica della struttura della tabella con l'aggiunta di nuove colonne, oppure di modifica dei dati contenuti nella tabella, oppure l'uso del comando Select.
Per concedere il diritto di modifica sulla tabella dei dipendenti agli utenti denominati user1 e user2, si deve usare il comando GRANT nella forma

```
Grant update on personale to user1,user2;
```

La revoca dei permessi con annullamento dei diritti di accesso viene effettuato con il comando REVOKE che ha una sintassi analoga a quella del comando GRANT

```
Revoke update on personale to user1,user2;
```

I permessi che possono essere concessi (o revocati) agli utenti sono indicati con le seguenti parole chiave che vanno specificate dopo GRANT e REVOKE:

ALTER per aggiungere o eliminare colonne, oppure per modificare i tipi di dati
DELETE per eliminare righe dalle tabelle
INDEX per creare indici
INSERT per inserire nuove righe alla tabella
SELECT per ritrovare i dati nelle tabelle
UPDATE per cambiare i valori contenuti nelle tabelle
ALL per tutti i permessi precedenti.

I permessi con le opzioni SELECT e UPDATE, nei comandi GRANT e REVOKE, diventano più restrittivi specificando, tra parentesi tonde e separati con la virgola, i nomi delle colonne che l'utente può vedere o modificare.
Per concedere il diritto di modifica del livello e dello stipendio base dei dipendenti all'utente denominato con User3, si deve usare il comando GRANT nella forma:

```
grant update (livello, StipBase) on personale to user3;
```

Indice: