Laurea in Ingegneria Informatica – SAPIENZA Università di Roma

Insegnamento di Basi di Dati

Esercitazione: Il DBMS MySQL

Domenico Fabio Savo

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Il DBMS MySQL

- *MySQL è un DBMS open-source disponibile gratuitamente su http://www.mysql.it/downloads/mysql/*

- In questa esercitazione si farà riferimento alla versione MySQL 5 per Windows.
  (è possibile utilizzare MySQL anche su sistemi Linux e MacOS)

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Installazione di MySQL

Per l'installazione procedere come segue:

1. Scaricare l'ultima versione di **MySQL Community Server** dal sito (è necessario registrarsi).
2. Eseguire il file `.msi` appena scaricato e seguire la procedura di installazione.
3. Dopo il termine dell'installazione è possibile lanciare la **Configuration Wizard** per configurare immediatamente il nostro server MySQL.
4. Selezionare la configurazione **"Standard"**.
5. Selezionare le check box per:
    a) eseguire MySQL come servizio;
    b) lanciare automaticamente MySQL all'avvio;
    c) includere la directory 'bin' nel path di Windows.
6. L'ultima schermata ci consente di impostare la password di **root**, tale password ci consentirà di amministrare il server.
7. Al termine, l'installazione e la configurazione sono completate.

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Il “client mysql”

- **Client mysql** è il programma client a riga di comando che consente di collegarsi al server MySQL per sfruttarne le funzionalità.
  (viene installato insieme al server MySQL)

- Dalla pagina web http://www.mysql.it/downloads/workbench/ è possibile scaricare ed installare dei client grafici, chiamati **MySQL GUI Tools**, che forniscono una interfaccia grafica intuitiva per la gestione e l'interrogazione delle basi di dati gestite dal DBMS MySQL.

- In questa esercitazione utilizzeremo il client a riga di comando. Si lascia allo studente il compito di familiarizzare con i client grafici su indicati.

Il “client mysql” (1/2)

- Per lanciare il client mysql è sufficiente richiamarlo dal prompt indicandogli utenza e password (nel nostro caso, useremo sempre l’utente **root**):

```
shell> mysql --user=root --password=xxx
```

oppure:

```
shell> mysql -uroot -p
```

In questo caso sarà il programma a chiedervi di introdurre la password senza visualizzarla.

Il “client mysql” (2/2)

- Una volta connessi al DBMS, appare il prompt di mysql:

```
mysql>
```

- A questo punto si posso digitare i comandi SQL per interagire con il DBMS. Per indicare al client il termine di un comando dobbiamo utilizzare il carattere «;».

- Per chiudere il client digitiamo:

```
mysql> quit
```

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

I permessi in MySQL

- Una volta connessi al server, l'utente deve possedere i permessi necessari per lavorare sulle varie basi di dati.
- Per chiedere quali basi di dati gestite dal DBMS sono accessibili dall'utente (nel nostro caso, **root**) utilizziamo il comando:

```
mysql> show databases;

        +--------------------+
        | Database           |
        +--------------------+
        | information_schema |
        | mysql              |
        | test               |
        +--------------------+
        3 rows in set (0.05 sec)
```

Creazione di una base di dati

- Per poter creare una nuova base di dati su cui lavorare utilizziamo il comando:

  CREATE DATABASE [IF NOT EXISTS] nome_db

- Con l'opzione IF NOT EXISTS possiamo evitare la segnalazione di errore nel caso esista già una base di dati con lo stesso nome.

- Per eliminare una base di dati si utilizza l'istruzione:

  DROP DATABASE [IF EXISTS] nome_db

- Con l'opzione IF EXISTS possiamo evitare la segnalazione di errore nel caso non esista una base di dati chiamata nome_db.

ES: Creazione di un database

Creiamo il DB "esempio" utilizzando il "client mysql".

Le istruzioni da utilizzare sono:

```
mysql> CREATE DATABASE esempio;

Query OK, 1 row effected (0.06 sec)
```

Ora le basi di dati gestiste dall'utente *root* sono:

```
mysql> show databases;

        +--------------------+
        | Database           |
        +--------------------+
        | information_schema |
        | mysql              |
        | test               |
        | esempio            |
        +--------------------+
        4 rows in set (0.05 sec)
```

Importare i comandi

Anziché eseguire comandi SQL digitandoli su terminale è spesso più conveniente scriverli in un file di testo e poi richiamarli dall'interprete dei comandi MySQL.

Supponiamo di aver scritto alcuni comandi SQL in un file **miaquery.sql** nella directory corrente. Possiamo eseguire il file da MySQL con il comando:

```
mysql> source miaquery.sql
```

Ovviamente è possibile anche specificare il path completo del file.

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Creazione delle tabelle (1/4)

- Per selezionare la base di dati su cui lavorare usare il comando

  USE nome_database

- Una volta selezionata la base di dati, l'istruzione per definire uno schema di relazione (specificando attributi e vincoli) in MySQL è

```
CREATE TABLE [IF NOT EXISTS] nome_tabella
 [(
        [definizione attributi]
        [opzioni di tabella]
 )]
```

- La tabella viene creata nella base di dati in uso. In generale, è possibile indicare espressamente in quale base di dati creare la tabella usando *nome_db.nome_tabella*.
- *IF NOT EXISTS si usa per evitare messaggi di errore nel caso la tabella esista già nella base di dati in uso.*

Creazione delle tabelle (2/4)

```
CREATE TABLE [IF NOT EXISTS] nome_tabella
[(
    [definizione attributi]
    [opzioni di tabella]
)]
```

- Le *definizioni attributi* si riferiscono agli attributi della tabella, la loro sintassi è:

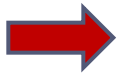
nome_colonna TIPO

[NOT NULL | NULL] (di default può contenere valori NULL)

[DEFAULT valore] (usato per impostare un valore di default)

[AUTO_INCREMENT] (per attributi di tipo intero per avere un valore sequenziale generato automaticamente)

[UNIQUE | [PRIMARY] KEY] (UNIQUE rappresenta una superchiave, PIMARY KEY indica la chiave primaria, oltre a non ammettere duplicati non può contenere valori NULL)

[reference_definition] ()

Creazione delle tabelle (3/4)

```
CREATE TABLE [IF NOT EXISTS] nome_tabella
[(
    [definizione attributi]
    [opzioni di tabella]
)]
```

- ***Reference_definition***

  Tramite le ***reference_definition*** è possibile definire vincoli di integrità referenziale, ovvero l'attributo su cui è definito può assumere solo valori specificati nell'attributo di un'altra tabella.

  REFERENCES nome_tabella [(colonna_indice,...)]

Creazione delle tabelle (4/4)

```
CREATE TABLE [IF NOT EXISTS] nome_tabella
[(
    [definizione attributi]
    [opzioni di tabella]
)]
```

- Le opzioni tabella si riferiscono all'intera tabella e permettono di definire diverse proprietà di questa.
- Le più importanti sono:

PRIMARY KEY (nome_attributo1, nome_attributo2,…)

Permette di definire come chiave primaria della tabella un insieme di attributi di questa.

INDEX (nome_attributo1, nome attributo2,…)

Permette di definire degli indici su uno o più attributi della tabella

FOREING KEY (nome_att1,nome,att2,…)

REFERENCE nome_tab(nome_att1,nome,att2,…)

Permette di definire vincoli di integrità referenziale su più attributi

ES: Creazione di una tabella (1/2)

Vogliamo creare le seguenti tabelle:

- **individui(nome, reddito, eta, sesso)**
  - nome è una stringa di 20 caratteri (chiave primaria)
  - reddito è un intero di 10 cifre
  - eta è un intero di 3 cifre
  - sesso è un carattere

- **genitori(figlio,genitore)**
  - figlio (stringa di 20 caratteri, chiave esterna su INDIVIDUI)
  - genitore (stringa di 20 caratteri, chiave esterna su INDIVIDUI)
  - chiave primaria formata da "figlio" e "genitore"

ES: Creazione di una tabella (2/2)

- Creazione tabella Individui:

```
mysql>    CREATE TABLE Individui(
             Nome       CHARACTER(20)   PRIMARY KEY,
             Reddito    NUMERIC(10),
             Eta        NUMERIC(3),
             Sesso      CHARACTER,
            );
```

- Creazione tabella Genitori:

```
mysql>    CREATE TABLE Genitori(
             Figlio      CHARACTER(20) REFERENCES Individui(Nome),
             Genitore    CHARACTER(20) REFERENCES Individui(Nome),
             PRIMARY KEY (Figlio,Genitore)
            );
```

Visualizzare le tabelle di un database

Per visualizzare le tabelle di una base di dati usare il comando:

```
mysql> show tables;
```

Dopo la creazione delle tabelle "Individui" e "Genitori" il risultato sarà

```
mysql> show tables;
+--------------------+
| Tables_in_esempio  |
+--------------------+
| genitori           |
| individui          |
+--------------------+
2 rows in set (0.01 sec)
```

Visualizzare lo schema di una tabella

Per visualizzare lo schema della tabella nome_tabella si utilizza l'istruzione

SHOW COLUMNS FROM nome_tabella

oppure

DESCRIBE nome_tabella

```
mysql> Describe genitori;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Figlio   | char(20) | NO   | PRI |         |       |
| Genitore | char(20) | NO   | PRI |         |       |
+----------+----------+------+-----+---------+-------+
2 rows in set (0.07 sec)
```

Modificare una tabella

È possibile modificare una tabella attraverso il comando ALTER TABLE:

```
ALTER TABLE nome_tabella
    ADD  nome_attributo TIPO |
    ADD INDEX (nome attributo,...) |
    ADD [CONSTRAINT [nome_vincolo]] PRIMARY KEY (nome_attributo,...) |
    ADD [CONSTRAINT [nome_vincolo]] UNIQUE (nome_attributo,...)
    ADD [CONSTRAINT [nome_vincolo]]
                        FOREIGN KEY (colonna_indice,...) [reference_definition] |
    CHANGE  vecchio_attributo  nuovo_attributo TIPO |
    DROP  nome_attributo
    DROP PRIMARY KEY
    DROP INDEX nome_attributo
    …….
    …….
```

Ridenominazione di una tabella

Per ridenominare una tabella usare il comando

ALTER TABLE *Nome_Tabella* RENAME *Nuovo_Nome*;

ES: Vogliamo cambiare il nome della tabella "Individui" con "Persone"

```
mysql>   ALTER TABLE Individui RENAME Persone;
```

Ridenominazione di una colonna

Per ridenominare una colonna di una tabella utilizzare il comando:

ALTER TABLE *Nome_Tabella* CHANGE *Nome_Colonna_da_cambiare Nuovo_Nome_Colonna Proprietà_della_Nuova_Colonna*;

ES: Vogliamo cambiare il nome del campo "Eta" con "Anni"

```
mysql>    ALTER TABLE Persone
          CHANGE Eta Anni NUMERIC(3);
```

NOTA: Per modificare il tipo di colonna utilizzare il medesimo comando cambiando solo il tipo della colonna.

Aggiungere una nuova colonna

Per aggiungere una nuova colonna ad una tabella utilizzare il comando:

ALTER TABLE *Nome_Tabella* ADD *Nome_della_Nuova_Colonna Proprietà_Colonna*;

ES: Aggiungiamo la colonna "n_telefono" alla tabella "Persone"

```
mysql>    ALTER TABLE Persone
          ADD n_telefono NUMERIC(20);
```

Per eliminare una colonna utilizzare il comando:

ALTER TABLE *Nome_Tabella* DROP *Nome_Colonna_da_canc*

Aggiungere un vincolo di chiave esterna (1)

Per aggiungere un vincolo di chiave esterna utilizzare il comando:

ALTER TABLE *Nome_Tabella*
ADD CONSTRAINT [nome_vincolo]
FOREIGN KEY (nome_col_che_referenzia)
REFERENCE Nome_Tabella_Referenziata(nome_colonna_refe);

Aggiungere un vincolo di chiave esterna (2)

ES: Date le tabelle:

- Aziende(Nome,Sede,Capitale)
- GruppoAziendale(Nome,Capogruppo)

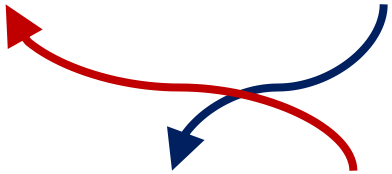
Vogliamo aggiungere alla tabella GruppoAziendale un vincolo di chiave esterna sull'attributo Capogruppo su Aziende

```
mysql>   ALTER TABLE GruppoAziendale
         ADD CONSTRAINT fk_capogruppo
         FOREIGN KEY (Capogruppo)
            REFERENCES Aziende(Nome);
```

Aggiungere un vincolo di chiave esterna (3)

La possibilità di aggiungere un vincolo di integrità referenziale permette di realizzare vincoli ciclici:

ES: Persona(Cod-Fiscale, Luogo-Nascita)



Città(Nome, Sindaco)

Dov'è il problema?

Aggiungere un vincolo di chiave esterna (4)

```
mysql>    CREATE TABLE Persona(
            Cod-Fiscale         CHARACTER(20) primary key,
            Luogo-Nascita       CHARACTER(1),
          FOREIGN KEY (Luogo-Nascita) REFERENCES Città(Nome));
```

Facciamo riferimento alla tabella "Città" che ancora NON ESISTE!!

Lo stesso accade se proviamo a creare prima la tabella "Città"

```
mysql>    CREATE TABLE Città(
            Nome                CHARACTER(20) primary key,
            Sindaco             CHARACTER(20),
          FOREIGN KEY (Sindaco) REFERENCES Persona(Cod-Fiscale));
```

La tabella "Persona" ancora non esiste

Aggiungere un vincolo di chiave esterna (5)

Soluzione:

Eseguo le seguenti istruzione in quest'ordine:

1- Creo la tabella "Persona" SENZA vincoli di foreign key;

2- Creo la tabella "Città" CON i vincoli di foreing key verso la tabella "Persona" (che ora esiste);

3- Aggiungo il vincolo di foreign key alla tabella "Persona" verso la tabella "Città" (che ora esiste).

Eliminare una tabella

È possibile eliminare una o più tabelle utilizzando il comando:

DROP TABLE [IF EXISTS] nome_tabella [, nome_tabella]

Con l'opzione IF EXISTS possiamo evitare la segnalazione di errore nel caso non esista una tabella chiamata nome_tabella.

ES:

```
mysql>   DROP TABLE Genitori;
```

Inserimento dei dati nelle tabelle

Per inserire dei dati in una tabella si utilizza l'istruzione:

INSERT INTO *nome_tabella* [(*nome_attributo1*,nome_attributo2,...)]
VALUES (valore1,valore2,...);

Attenzione:

- L'ordinamento degli attributi (se presente) e dei valori è significativo.
- Le due liste di attributi e di valori devono avere lo stesso numero di elementi.
- Se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione secondo l'ordine con cui sono stati definiti.
- Se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un valore nullo (che deve essere permesso) o un valore di default.

ES: Inserimento dati

Inseriamo alcune tuple nella tabella

Persone(Nome, Reddito,Eta,Sesso)

```
mysql> INSERT INTO PERSONE (Nome,Reddito,Eta,Sesso)
       VALUES ('Aldo',25,15,'M');
mysql> INSERT INTO PERSONE (Nome,Reddito,Eta,Sesso)
       VALUES ('Andrea',27,21,'M');
mysql> INSERT INTO PERSONE (Nome,Reddito,Eta,Sesso)
       VALUES ('Luisa',75,87,'F');
mysql> INSERT INTO PERSONE (Nome,Reddito,Eta,Sesso)
       VALUES ('Maria',55,42,'F');
```

Eliminazione di dati dalle tabelle

Per eliminare una ennupla utilizzare il comando:

DELETE FROM *nome_tabella* [WHERE *condizione*]

ES: Eliminiamo tutte le persone con meno di 18 anni dalla tabella "Persone"

```
mysql>  DELETE FROM persone WHERE eta<18;
```

Interrogare un database

Per effettuare un'interrogazione in SQL si utilizza l'istruzione SELECT

```
SELECT nome_attributo,…,nome_attributo
FROM nome_tabella, …,nome_tabella
[WHERE condizione]
```

Le tre parti sono solitamente chiamate:

- target list
- clausola from
- clausola where

Le ridenominazioni

SQL permette di specificare un "alias" degli attributi (nella target list usando il comando AS) e delle tabelle (nella clausola FROM).

La ridenominazione è usata per:

1. Ottenere segnature più esplicative nei risultati;
2. Creare abbreviazione ed evitare ambiguità;

```
mysql> SELECT p.nome as donne
       FROM persone p
       WHERE p.sesso = "F";
```

Cosa vedremo

1. Presentazione del DBMS MySQL
2. Come scaricare ed installare MySQL
3. Il "client mysql"
4. Creazione e gestione di una base di dati
5. Creazione e gestione delle tabelle
6. Esercizi di esame su SQL

Esercizio 1

La relazione **Dirige(azienda,direttore)** memorizza, per ogni azienda, il direttore.
La relazione **Hobby(persona,hobby)** specifica quali hobby hanno le varie persone.
Si chiede di esprimere in SQL le seguenti interrogazioni:

1. Mostrare i direttori che non hanno alcun hobby.
2. Mostrare le coppie di direttori che hanno almeno un hobby in comune.
3. Mostrare i direttori che hanno meno hobby del numero di aziende che dirigono.

Soluzione 1.1

Un direttore non ha alcun hobby se non appare nella tabella "Hobby".

```
SELECT direttore
FROM Dirige
WHERE direttore NOT IN (
      SELECT persona
      FROM Hobby)
```

Soluzione 1.2

Si calcola il join tra le relazioni "Dirige" ed "Hobby" ed ancora tra "Dirige" ed "Hobby" con una selezione per individuare le coppie di direttori tra loro diversi e che condividono il medesimo hobby.

```
SELECT d1.direttore, d2.direttore
FROM Dirige d1, Dirige d2, Hobby h1, Hobby h2
WHERE d1.direttore = h1.persona AND
      d2.direttore = h2.persona AND
      d1.direttore <> d2.direttore AND
      h1.hobby = h2.hobby
```

Soluzione 1.3

È sufficiente aggregare il join tra “Dirige” e “Hobby” su “direttore”, ed usare la clausola “having” per filtrare solo i direttori che hanno meno hobby delle aziende che dirigono

```
SELECT direttore
FROM Dirige, Hobby
WHERE direttore = persona
GROUP BY direttore
HAVING
count(distinct hobby) < count(distinct azienda)
```

Esercizio 2

La relazione **Member(codp,codg,anno)** memorizza a quale gruppo è affiliata ogni persona, insieme all'anno dal quale inizia tale affiliazione. In particolare, una tupla **<p,g,a>** nella relazione **Member** indica che la persona **p** è affiliata al gruppo **g** a partire dall'anno **a**.
La relazione **Pratica(codp,sport)** specifica quali sport praticano le varie persone.
Si chiede di esprimere in SQL le seguenti interrogazioni:

1. Mostrare il codice dei gruppi che hanno almeno due membri e tali che almeno uno di questi membri pratica almeno uno sport.
2. Per ogni gruppo, indicare la persona (o le persone, se sono più d'una) che sono affiliate a tale gruppo da meno tempo.
3. Chiamiamo "omogeneo" un gruppo G se esiste almeno uno sport praticato da tutti i membri del gruppo G. Mostrare tutti i gruppi omogenei.
4. Mostrare i gruppi formati solo da persone che non praticano alcuno sport.

Soluzione 2.1

Per mostrare il codice dei gruppi con almeno due membri e tali che almeno uno di questi pratichi uno sport eseguiamo il join tra: le relazioni "Pratica" e "Member" (in modo da ottenere i membri di un gruppo che praticano almeno uno sport) ed ancora con la relazione "Member" sul campo "codg" con una selezione per individuare coppie di persone tra loro diverse (per ottenere gruppi con almeno due membri).

```
SELECT distinct m1.codg
FROM Member m1, Member m2, Pratica p
WHERE m1.codg = m2.codg AND
      m1.codp <> m2.codp AND
      m1.codp = p.codp
```

Soluzione 2.2 (1/2)

Per individuare la persona che, per ogni gruppo, è iscritta da meno tempo calcoliamo il join sui campi "codg" ed "anno" tra la relazione "Member" e la relazione ottenuta eseguendo la query sulla relazione "Member" che, per ogni gruppo, restituisce la data di iscrizione più recente ed il relativo gruppo.

```
SELECT m1.codp
FROM   Member m1,
       (SELECT codg, max(anno) AS maxanno
        FROM Member
        GROUP BY codg) m2
WHERE m1.codg = m2.codg AND
      m1.anno = m2.maxanno
```

Soluzione 2.2 (2/2)

Il medesimo risultato si può ottenere creando una vista:

```
CREATE VIEW grupmaxanno(codg, maxanno) AS
SELECT codg, max(anno)
        FROM Member
        GROUP BY codg

SELECT m1.codp
FROM   Member m1,
       grupmaxanno m2
WHERE m1.codg = m2.codg AND
       m1.anno = m2.maxanno
```

Soluzione 2.3

Usando delle query nella clausola "FROM" calcoliamo due nuove relazioni: la prima ,"temp1", contiene il numero di persone appartenenti ad ogni gruppo, la seconda, "temp2", contiene il numero di persone che appartengono allo stesso gruppo e praticano il medesimo sport. Calcolando poi il join tra "temp1" e "temp2" sui campi "codg" ed il numero di persone otteniamo i gruppi "omogenei".

```
SELECT temp1.codg
FROM (SELECT codg, count(distinct codp) AS tot1
      FROM Member
      GROUP BY codg) temp1,
      (SELECT m.codg, p.sport,
              count(distinct m.codp) AS tot2
        FROM Member m, Pratica p
        WHERE m.codp = p.codp
        GROUP BY m.codg, p.sport) temp2
WHERE temp1.codg = temp2.codg  AND
      temp1.tot1 = temp2.tot2
```

Soluzione 2.3

Usando delle query nella clausola "FROM" calcoliamo due nuove relazioni: la prima ,"temp1", contiene il numero di persone appartenenti ad ogni gruppo, la seconda, "temp2", contiene il numero di persone che appartengono allo stesso gruppo [illegible] tra "temp1" e "ter[illegible]amo i gruppi "omoge[illegible]

Anche in questo caso la soluzione può essere ottenuta definendo due viste

```
SELECT temp
FROM (SELEC
      FROM M
      GROUP
      (SELEC

        FROM
        WHER
        GROUP BY m.codg, p.sport) temp2
WHERE temp1.codg = temp2.codg  AND
      temp1.tot1 = temp2.tot2
```

Soluzione 2.4

Si calcola il complemento dell'insieme dei gruppi che hanno almeno un membro che pratica almeno uno sport.

```
SELECT distinct codg
FROM Member
WHERE codg NOT IN (
            SELECT distinct m.codg
            FROM Member m, Pratica p
            WHERE m.codp = p.codp)
```

Esercizio 3

La relazione **Opera(codo,anno,museo)** memorizza in quali musei sono state custodite le varie opere d'arte nei diversi anni (la tupla **<c,a,m>** indica che nell'anno **a** l'opera con codice **c** è stata custodita nel museo con codice **m**).
La relazione **Museo(codm,nazione)** specifica in quali nazioni hanno sede i musei.
Si chiede di esprimere in SQL le seguenti interrogazioni:

1. Mostrare le opere custodite almeno una volta per due anni consecutivi in musei italiani (non necessariamente lo stesso museo nei due anni).
2. Per ogni museo francese e per ogni anno mostrare il numero di opere custodite in quell'anno in quel museo, ma solo se tale numero supera 10.
3. Mostrare le opere che non sono mai state custodite in musei italiani.

Soluzione 3.1

Calcoliamo il join tra le relazioni Opera, Museo ed ancora Opera e Museo imponendo la condizione che la nazione dei musei deve essere "Italia" . Si selezionano poi le coppie della stessa opera con la condizione che l'anno di esposizione della prima sia pari all'anno di esposizione della seconda più 1.

```
SELECT o1.codo
FROM opera o1, opera o2, "Museo" m1, "Museo" m2
WHERE o1.museo = m1.codm AND
      m1.nazione = 'italia' AND
      o2.museo = m2.codm AND
      m2.nazione = 'italia' AND
      o1.codo = o2.codo AND
      o1.anno = o2.anno + 1
```

Soluzione 3.2

Calcoliamo il join tra la relazione “Opera” e “Museo” con la condizione che la nazione del museo sia la ‘Francia’. Calcoliamo poi una aggregazione sulla coppia di attributi “museo” ed “anno” ed usiamo la clausola “having” per filtrare solo i musei che hanno esposto più di 10 opere in quell’anno.

```
SELECT      anno, museo, count(distinct codo)
FROM        Opera o, Museo m
WHERE       o.museo = m.codm AND
            m.nazione = 'francia'
GROUP BY    o.museo, o.anno
HAVING      count(distinct o.codo) > 10
```

Soluzione 3.3

Calcoliamo il complemento dell'insieme delle opere che sono state esposte almeno una volta in 'Italia'.

```
SELECT codo
FROM Opera
WHERE codo <> ALL
             (SELECT codo
             FROM Opera, Museo
             WHERE museo = codm AND
             nazione = 'italia')
```

Esercizio 4 (per casa)

La relazione **Volo(cod,partenza,arrivo,numpass,comp)** memorizza, per ogni volo aereo effettuato, il codice, l'aeroporto di partenza, quello di arrivo, il numero di passeggeri e la compagnia aerea, mentre la relazione **Compagnia(codice,nazione)** specifica in quali nazioni hanno sede le compagnie aeree.
Si chiede di esprimere in SQL le seguenti interrogazioni:

1. Mostrare i dati dei voli di compagnie italiane.
2. Mostrare gli aeroporti dai quali non partono voli di compagnie francesi.
3. Per ogni aeroporto a, mostrare il numero complessivo di passeggeri di voli partiti da a, ma solo se il numero di voli partiti da a è maggiore di 4.

Nella sezione "materiale didattico" della pagina
http://www.dis.uniroma1.it/~savo/bd1213.html
È possibile scaricare lo script SQL (esercizio-4.sql) per creare e popolare un DB su MySQL per provare le query di questo esercizio.