



Università di Roma



# Progettazione Fisica e SQL

*Ing. Alessandro Pellegrini, PhD*  
*pellegrini@diag.uniroma1.it*

# Da qui in poi...

- ▶ ...la documentazione ufficiale è vostra amica!
- ▶ Per SQL (DBMS e Query):
  - <https://dev.mysql.com/doc/refman/8.0/en/>
- ▶ Per la programmazione dei client in C:
  - <https://dev.mysql.com/doc/refman/8.0/en/c-api.html>
- ▶ Per MySQL Workbench:
  - <https://dev.mysql.com/doc/workbench/en/>

# Query SQL

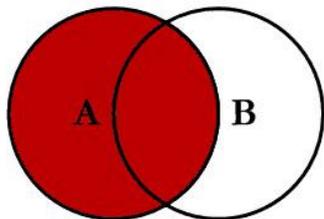
# Select

## SELECT

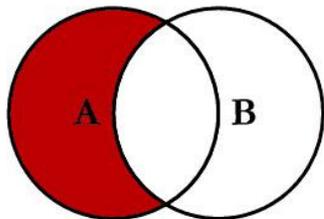
```
[ALL | DISTINCT | DISTINCTROW]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[ FROM table_references
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]
```

# Le operazioni di Join

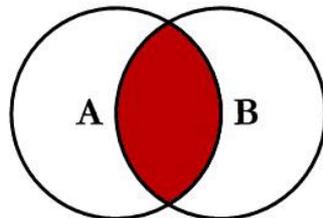
## SQL JOINS



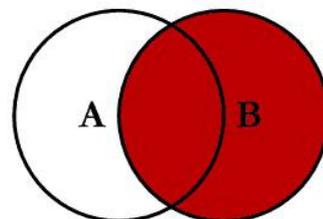
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



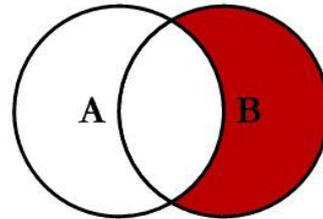
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



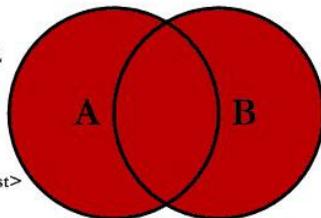
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



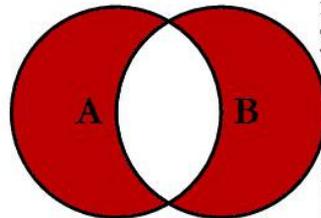
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Città, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittàPart, OraPart, CittàArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città con un aeroporto di cui non è noto il numero di piste.

```
select Città  
from AEROPORTO  
where NumPist is NULL
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le nazioni da cui parte e arriva il volo AZ274.

```
select A1.Nazione, A2.Nazione  
from AEROPORTO as A1 join VOLO on A1.Citta = CittaArr  
join AEROPORTO as A2 on CittaPart = A2.Citta  
where IdVolo='AZ274'
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - I tipi di aereo usati nei voli che partono da Torino.

```
select TipoAereo  
from VOLO  
where CittaPart = 'Torino'
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - I tipi di aereo e il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile, visualizzare solamente il tipo.

```
select VOLO.TipoAereo, NumPasseggeri
from VOLO left join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where CittaPart = 'Torino'
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Città, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittàPart, OraPart, CittàArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città da cui partono voli internazionali

```
select CittàPart
from AEROPORT as A1 join VOLO on CittàPart=A1.Città
join AEROPORT as A2 on CittàArr=A2.Città
where A1.Nazione <> A2.Nazione
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente

```
select CittaPart
from VOLO
where CittàArr = 'Bologna'
order by CittaPart
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Il numero di voli internazionali che partono il giovedì da Napoli

```
select count(*)  
from VOLO join AEROPORTO on CittaArr=Citta  
where Nazione<>'Italia' and CittaPart='Napoli' and  
DAYOFWEEK(GiornoSett)=5 -- Domenica è 1
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Il numero di voli internazionali che partono ogni settimana da città italiane

```
select count(*)  
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart  
join AEROPORTO as A2 on A2.Citta=CittaArr  
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia'  
group by YEARWEEK(GiornoSett)
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città francesi da cui partono più di venti voli alla settimana diretti in Italia

```
select A1.Citta, YEARWEEK(GiornoSett) as Settimana,  
       count(A1.Citta)  
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart  
join AEROPORTO as A2 on A2.Città=CittàArr  
where A1.Nazione = 'Francia' and A2.Nazione = 'Italia'  
group by YEARWEEK(GiornoSett), A1.Citta  
having count(A1.Citta) > 20
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando operatori insiemistici

```
select distinct CittaPart
from VOLO, AEROPORTO where CittaPart=Citta and Nazione='Italia'
except
select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia'
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando un'interrogazione nidificata con l'operatore not in

```
select distinct CittaPart
from VOLO, AEROPORTO where CittaPart=Citta
and CittaPart not in (select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia')
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando un'interrogazione nidificata con l'operatore not exists

```
select distinct CittaPart  
from VOLO join AEROPORTO as A1 on CittaPart=A1.Citta  
where Nazione='Italia' and not exists (select * from VOLO join  
AEROPORTO as A2 on A2.Citta=CittaArr  
where A1.Citta=CittaPart and A2.Nazione <> 'Italia' )
```

## Esercizio 4.14

- Dato il seguente modello relazionale:  
AEROPORTO(Citta, Nazione, NumPiste)  
VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr, TipoAereo)  
AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando una join ed un operatore di conteggio

```
select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
left join AEROPORTO as A2 on CittaArr = A2.Citta
where A1.Nazione = 'Italia'
group by CittaPart
having count(case (A2.Nazione <> 'Italia') when true then 1
else NULL end) = 0
```

# Stored Procedures e Transazioni

# Creazione Stored Procedure

CREATE

[OR REPLACE]

[DEFINER = { user | CURRENT\_USER | role | CURRENT\_ROLE }]

PROCEDURE sp\_name ([proc\_parameter[,...]])

[characteristic ...] routine\_body

proc\_parameter:

[ IN | OUT | INOUT ] param\_name type

type:

Any valid MariaDB data type

characteristic:

LANGUAGE SQL

| [NOT] DETERMINISTIC

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

| COMMENT 'string'

routine\_body:

Valid SQL procedure statement

# Un esempio

```
set autocommit = 0;
```

```
create procedure `assign`(in var_matricola varchar(45), out var_token varchar(128))  
begin  
  declare exit handler for sqlexception  
  begin  
    rollback; -- rollback any changes made in the transaction  
    resignal; -- raise again the sql exception to the caller  
  end;  
end;
```

```
set transaction isolation level repeatable read;  
start transaction;  
if ... then  
  signal sqlstate '45001' set message_text = "Si è verificata una condizione di errore";  
end if;  
commit;  
end
```

# SQL States

- ▶ Alcune procedure SQL hanno la necessità di fornire al DBMS (che eventualmente lo propagherà al client chiamate) un codice che fornisca delle informazioni sul successo o sul fallimento dell'esecuzione—una sorta di valore di ritorno.
- ▶ Gli SQL States sono codici di 5 byte (i primi due per la classe)
- ▶ Alcuni esempi:
  - 3F000: invalid schema name
  - 30000: invalid SQL statement identifier
  - 23000: integrity constraint violation
  - 22012: data exception: division by zero
  - 45000: unhandled user-defined exception

# Livelli di Isolamento

## ▶ **READ UNCOMMITTED**

- ▶ Gli statement SELECT sono eseguiti in modalità non bloccante, ma è possibile che venga utilizzata una versione precedente di una riga. Pertanto, utilizzando questo livello di isolamento, le letture possono non essere coerenti (dirty read).
- ▶ Questo fenomeno si verifica poiché una transazione può leggere dati da una riga aggiornata da un'altra transazione che non ha ancora eseguito l'operazione di commit.

# Dirty Reads

T1

SELECT age FROM users WHERE id = 1;

UPDATE users SET age = 21 WHERE id = 1;

SELECT age FROM users WHERE id = 1;

T2

ROLLBACK;

# Livelli di Isolamento

- ▶ **READ COMMITTED**
- ▶ Il DBMS acquisisce un lock per ogni dato che viene letto o scritto. I lock associati ai dati che sono stati aggiornati in scrittura sono mantenuti fino alla fine della transazione, mentre i lock associati ai dati acceduti in lettura sono rilasciati alla fine della singola lettura.
- ▶ Possono verificarsi anomalie di tipo *unrepeatable reads*.

# Unrepeatable Reads

T1

```
SELECT * FROM users WHERE id = 1;
```

```
SELECT * FROM users WHERE id = 1;  
COMMIT;
```

T2

```
UPDATE users SET age = 21 WHERE id = 1;
```

```
COMMIT;
```

# Livelli di Isolamento

## ▶ REPEATABLE READ

- ▶ Con questo livello di isolamento, vengono mantenuti i lock sia dei dati acceduti in lettura sia in scrittura fino alla fine della transazione. Non vengono però gestiti i *range lock*, pertanto possono verificarsi anomalie di tipo *phantom read*.
- ▶ Le *phantom read* sono associate ad inserimenti che avvengono in concorrenza.
- ▶ Nelle versioni più nuove di InnoDB, è il livello di isolamento predefinito.

# Phantom Reads

T1

```
SELECT * FROM users WHERE age BETWEEN 10 AND 30;
```

```
INSERT INTO users(id,name,age) VALUES ( 3, 'Bob', 27 );
```

```
COMMIT;
```

```
SELECT * FROM users WHERE age BETWEEN 10 AND 30;
```

```
COMMIT;
```

T2

# Livelli di Isolamento

- ▶ **SERIALIZABLE**
- ▶ Tutti i lock vengono mantenuti fino alla fine della transazione e ogni volta che una SELECT utilizza uno specificatore di tipo WHERE, viene acquisito anche il *range lock*.
- ▶ In sistemi non basati su lock, questo livello di isolamento può essere implementato mediante il concetto di *read/write set* o in generale di *multiversion concurrency control*.

# Quale scegliere?

1. Per molte applicazioni, la maggior parte delle transazioni possono essere costruite in maniera tale da non richiedere l'utilizzo di livelli di isolamento molto alti (ad esempio SERIALIZABLE), riducendo l'overhead dovuto ai lock
2. Lo sviluppatore deve accertarsi con cautela che le modalità di accesso delle transazioni non causino bug software dovuti alla concorrenza e al rilassamento dei livelli di isolamento.
3. Se si utilizzano unicamente alti livelli di isolamento, la probabilità di *deadlock* cresce notevolmente.

# Modalità di accesso transazionali

- ▶ READ ONLY: la transazione si limita alla lettura di dati
- ▶ READ WRITE: la transazione legge e scrive dati
- ▶ Dare queste informazioni al DBMS consente di generare dei piani di esecuzione ottimizzati dal punto di vista dell'acquisizione dei lock
- ▶ Sono informazioni ortogonali al livello di isolamento richiesto

# Come marcare le transazioni

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [,transaction_characteristic] ...
```

```
transaction_characteristic: {
    ISOLATION LEVEL level
    | access_mode
}
level: {
    REPEATABLE READ          access_mode: {
    | READ COMMITTED          | READ WRITE
    | READ UNCOMMITTED       | READ ONLY
    | SERIALIZABLE           }
}
```

Nel caso in cui la marcatura non sia globale, le caratteristiche si riferiscono unicamente alla successiva transazione nella sessione. In seguito, verranno ripristinati i valori globali anche per la sessione in corso.

# SQL Avanzato

# Indici

```
CREATE [OR REPLACE] [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX
  [IF NOT EXISTS] index_name
  [index_type]
  ON tbl_name (index_col_name,...)
  [WAIT n | NOWAIT]
  [index_option]
  [algorithm_option | lock_option] ...
```

```
index_col_name:
  col_name [(length)] [ASC | DESC]
```

```
index_type:
  USING {BTREE | HASH | RTREE}
```

```
index_option:
  KEY BLOCK SIZE [=] value
| index_type
| WITH PARSER parser_name
| COMMENT 'string'
```

```
algorithm_option:
  ALGORITHM [=] {DEFAULT|INPLACE|COPY}
```

```
lock_option:
  LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

# Utilizzo di Trigger per emulare le Asserzioni

- ▶ MySQL non supporta le asserzioni (in particolare il comando `STOP ACTION` non è supportato)
- ▶ Si possono utilizzare trigger “before update” per emulare il funzionamento delle asserzioni

```
create assertion AT_MOST_ONE_MANAGER as CHECK
((select count(*) from `employees` E
  where E.ruolo = 'MANAGER') <= 1
)
```

# Utilizzo di Trigger per emulare le Asserzioni

```
create trigger AT_MOST_ONE_MAGANGER
before insert on `employees` for each row
begin
    declare counter INT;
    select count(*) from `employees` E
    where E.ruolo = 'MANAGER' into counter;
    if counter > 1 then
        signal sqlstate '45000';
    end if;
end
```

# Utilizzo di Trigger per emulare le Asserzioni

- ▶ Spesso è di interesse effettuare dei controlli sui valori che si stanno per inserire
  - In questo caso, la keyword `NEW` permette di accedere agli attributi della tupla che si sta inserendo
- Si può anche effettuare (nel caso di trigger di tipo `BEFORE UPDATE`) effettuare dei controlli sulla tupla che si sta per aggiornare
  - I vecchi valori sono accessibili mediante la keyword `OLD`
  - Questo può essere ad esempio utile per implementare meccanismi di storicizzazione

# Eventi temporizzati

```
set global event_scheduler = on;
```

```
create event if not exists `cleanup`
```

```
on schedule
```

```
every 2 day
```

```
on completion preserve
```

```
comment 'Remove registrations which have expired'
```

```
do
```

```
delete from `assegnazione` where `confermato_il` is null and  
`richiesto_il` < (NOW() - interval 2 day)
```

# Funzioni

delimiter !

```
create function `valid_email`(email varchar(45))
```

```
returns bool
```

```
deterministic
```

```
begin
```

```
    if email regexp '^[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9._-]  
@'[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9]\\.?[a-zA-Z]{2,63}$' then
```

```
        return true;
```

```
    end if;
```

```
    return false;
```

```
end!
```

```
delimiter ;
```

# Cursori

```
create procedure `itera_su_tabella`()  
begin  
  declare done int default false;  
  declare varchar(45) var_nome;  
  declare cur cursor for select nome from tabella;  
  declare continue handler for not found set done = true;  
  open cur;  
  read_loop: loop  
    fetch cur into var_nome;  
    if done then  
      leave read_loop;  
    end if;  
  end loop;  
  close cur;  
end
```

# MySQL Workbench

# Creazione di un progetto SQL

The screenshot displays the MySQL Workbench interface for a new project named 'mydb'. The main window is titled 'Untitled - MySQL Workbench'. The menu bar includes File, Edit, View, Arrange, Model, Database, Tools, Scripting, and Help. The toolbar contains icons for file operations and navigation. The left sidebar shows 'No selection' and 'EER Diagrams'. The main workspace is divided into several sections:

- Physical Schemas:** A tree view showing the 'mydb' MySQL Schema. Below it are sections for Tables (0 items), Views (0 items), Routines (0 items), and Routine Groups (0 items), each with an 'Add' button.
- Schema Privileges:** A section for 'Users (0 items)' with an 'Add User' button. Below it is a 'Roles (5 items)' section with a table of roles and their privileges:

Role	Privileges
Add Role	owner, table.readonly, table.insert
table.modify	routine.execute

- SQL Scripts:** A section for 'Add Script'.
- Model Notes:** A section for notes.

At the bottom, there are tabs for 'User Types' and 'History', and a status bar indicating 'New document.'.

# Editor dello schema

The screenshot shows the MySQL Workbench EER Diagram editor. The main workspace displays three tables: **assegnazione**, **esami**, and **progetti**. The **assegnazione** table has attributes: id INT, matricola VARCHAR(45), cognome VARCHAR(45), nome VARCHAR(45), email VARCHAR(45), richiesto\_# TIMESTAMP, confermato\_# TIMESTAMP, token VARCHAR(128), esame INT, and progetto INT. The **esami** table has attributes: id INT, nome VARCHAR(45), crediti INT, and abilitata TINYINT(1). The **progetti** table has attributes: id INT, esame INT, titolo VARCHAR(45), specifica LONGTEXT, and contatore BIGINT. Relationships are shown with dashed lines: a one-to-many relationship between **assegnazione** and **esami** (foreign key on **assegnazione** to **esami**), and a one-to-many relationship between **assegnazione** and **progetti** (foreign key on **assegnazione** to **progetti**).

The left sidebar shows the Navigator with a search for '100' and a tree view containing 'progetti', 'Tables', 'Views', and 'Routine Groups'. The bottom panel shows the 'assegnazione - Table' properties:

Property	Value	Description
Name:	assegnazione	The name of the table. It is recommended to use alpha-numeric characters. Spaces should be avoided and be replaced by ...
Charset/Collation:	Default Charset / Default Collation	The charset/collation specifies which language specific characters can be stored in the table and their sort order. Common choices are Latin1 or UTF8
Engine:	InnoDB	The database engine that is used for the table. This option affects performance, data consistency and more
Comment:		

# Editor dello schema

The screenshot shows the MySQL Workbench EER Diagram editor. The main workspace displays three tables: **assegnazione**, **esami**, and **progetti**. The **assegnazione** table has columns: id (INT), matricola (VARCHAR(45)), cognome (VARCHAR(45)), nome (VARCHAR(45)), email (VARCHAR(45)), richiesto\_# (TIMESTAMP), confermato\_# (TIMESTAMP), token (VARCHAR(128)), esame (INT), and progetto (INT). The **esami** table has columns: id (INT), nome (VARCHAR(45)), crediti (INT), and abilitata (TINYINT(1)). The **progetti** table has columns: id (INT), esame (INT), titolo (VARCHAR(45)), specifica (LONGTEXT), and contatore (BIGINT). Relationships are shown with dashed lines: a one-to-many relationship between **assegnazione** and **esami**, a one-to-many relationship between **assegnazione** and **progetti**, and a one-to-many relationship between **esami** and **progetti**.

The bottom panel shows the details for the **assegnazione** table. The columns and their datatypes are:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
matricola	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
cognome	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					

Column Details: Charset/Collation: Default Charset, Default Collation. Generated Column Storage Type:  VIRTUAL  STORED.

# Editor dello schema

The screenshot shows the MySQL Workbench EER Diagram editor. The main workspace displays three tables: **assegnazione**, **esami**, and **progetti**. The **assegnazione** table has columns: id (INT, primary key), matricola (VARCHAR(45)), cognome (VARCHAR(45)), nome (VARCHAR(45)), email (VARCHAR(45)), richiesto\_il (TIMESTAMP), confermato\_il (TIMESTAMP), token (VARCHAR(128)), esame (INT), and progetto (INT). The **esami** table has columns: id (INT, primary key), nome (VARCHAR(45)), crediti (INT), and abilitata (TINYINT(1)). The **progetti** table has columns: id (INT, primary key), esame (INT), titolo (VARCHAR(45)), specifica (LONGTEXT), and contatore (BIGINT). Relationships are shown with dashed lines: 'matricola' in 'assegnazione' is linked to 'nome' in 'esami'; 'progetto' in 'assegnazione' is linked to 'id' in 'progetti'; 'esame' in 'progetti' is linked to 'esame' in 'assegnazione'; and 'progetto' in 'esami' is linked to 'id' in 'progetti'.

The bottom panel shows the 'assegnazione - Table' properties. The 'Indexes' tab is active, displaying the following table:

Index Name	Type
PRIMARY	PRIMARY
studente-corso	UNIQUE
fk_assegnazione_esami1_idx	INDEX
fk_assegnazione_progetti1_idx	INDEX

The 'Index Columns' tab is also active, showing the following table:

Column	#	Order	Length
<input type="checkbox"/> id		ASC	0
<input checked="" type="checkbox"/> matricola	1	ASC	0
<input type="checkbox"/> cognome		ASC	0
<input type="checkbox"/> nome		ASC	0
<input type="checkbox"/> email		ASC	0
<input type="checkbox"/> richiesto_il		ASC	0
<input type="checkbox"/> confermato_il		ASC	0
<input type="checkbox"/> token		ASC	0

The 'Index Options' panel shows: Storage Type: BTREE, Key Block Size: 0, and Visible: checked.

# Editor dello schema

The screenshot shows the MySQL Workbench EER Diagram editor. The main workspace displays three tables: 'assegnazione', 'esami', and 'progetti'. 'assegnazione' is connected to 'esami' and 'progetti'. The 'assegnazione' table has columns: id INT, matricola VARCHAR(45), cognome VARCHAR(45), nome VARCHAR(45), email VARCHAR(45), richiesto\_il TIMESTAMP, confermato\_il TIMESTAMP, token VARCHAR(128), esame INT, and progetto INT. 'esami' has columns: id INT, nome VARCHAR(45), crediti INT, and abilitata TINYINT(1). 'progetti' has columns: id INT, esame INT, titolo VARCHAR(45), specifica LONGTEXT, and contatore BIGINT. The 'assegnazione' table has a foreign key 'fk\_assegnazione\_esami1' to 'esami' and another 'fk\_assegnazione\_progetti1' to 'progetti'. The 'progetti' table has a foreign key 'fk\_progetti\_esame1' to 'assegnazione'.

Below the diagram, the 'Foreign Keys' tab is active for the 'assegnazione' table. It shows the following foreign keys:

Foreign Key Name	Referenced Table
fk_assegnazione_esami1	'progetti', 'esami'
fk_assegnazione_progetti1	'progetti', 'progetti'

The 'Foreign Key Columns' tab is also active, showing the columns for the selected foreign key:

Column	Referenced Column
<input type="checkbox"/> cognome	
<input type="checkbox"/> nome	
<input type="checkbox"/> email	
<input type="checkbox"/> richiesto_il	
<input type="checkbox"/> confermato_il	
<input type="checkbox"/> token	
<input checked="" type="checkbox"/> esame	id
<input type="checkbox"/> progetto	

The 'Foreign Key Options' tab shows the following options:

- On Update: SET NULL
- On Delete: CASCADE
- Foreign Key Comment: (empty)
- Skip in SQL generation

# Editor dello schema

The screenshot displays the MySQL Workbench EER Diagram editor for a database named 'applicazione-1.mwb\*'. The interface includes a menu bar (File, Edit, View, Arrange, Model, Database, Tools, Scripting, Help), a toolbar, and a Navigator pane on the left showing a tree view of the database structure (progetti, Tables, Views, Routine Groups). The main workspace shows three tables:

- assegnazione**: id INT (PK), matricola VARCHAR(45), cognome VARCHAR(45), nome VARCHAR(45), email VARCHAR(45), richiesto\_ii TIMESTAMP, confermato\_ii TIMESTAMP, token VARCHAR(128), esame INT, progetto INT.
- esami**: id INT (PK), nome VARCHAR(45), crediti INT, abilitata TINYINT(1).
- progetti**: id INT (PK), esame INT, titolo VARCHAR(45), specifica LONGTEXT, contatore BIGINT.

Relationships are shown with dashed lines and crow's foot notation:

- Assegnazione (id) to Esami (id): 1:M relationship.
- Assegnazione (progetto) to Progetti (id): 1:M relationship.
- Esami (progetto) to Progetti (id): 1:M relationship.
- Esami (esame) to Progetti (esame): 1:M relationship.

The bottom pane shows the 'esami - Table' editor with the 'Triggers' tab selected. A trigger named 'esami\_BEFORE\_INSERT' is defined as follows:

```
1 • CREATE DEFINER = CURRENT_USER TRIGGER `progetti`.`esami_BEFORE_INSERT` BEFORE INSERT ON `esami` FOR EACH ROW
2 | BEGIN
3 |
4 | END
5 |
```

# Editor dello schema

The screenshot shows the MySQL Workbench EER Diagram editor. The main workspace displays three tables: 'assegnazione', 'esami', and 'progetti'. 'assegnazione' has fields: id INT, matricola VARCHAR(45), cognome VARCHAR(45), nome VARCHAR(45), email VARCHAR(45), richiesto\_il TIMESTAMP, confermato\_il TIMESTAMP, token VARCHAR(128), esame INT, and progetto INT. 'esami' has fields: id INT, nome VARCHAR(45), crediti INT, and abilitata TINYINT(1). 'progetti' has fields: id INT, esame INT, titolo VARCHAR(45), specifica LONGTEXT, and contatore BIGINT. Relationships are shown with dashed lines: 'assegnazione' to 'esami' (one-to-many), 'assegnazione' to 'progetti' (one-to-many), and 'esami' to 'progetti' (one-to-many).

Below the diagram, the 'esami - Table' window is open, showing the table structure and data:

#	id	nome	crediti	abilitata
1	1	Concurrent and Parallel Programming	3	false
2	2	Data Centers and High Performance Computing	6	false
3	3	Advanced Operating Systems and Virtualization	9	false
4	4	Basi di Dati e Conoscenza	9	false
5	5	Basi di Dati e Conoscenza	12	false
*				

# Gestione dei privilegi

MySQL Model x EER Diagram x pellegrini.tk - Warning - not supported x

File Edit View Arrange Model Database Tools Scripting Help

studente: Role

▼ EER Diagrams

Add Diagram EER Diagram

▼ Physical Schemas

progetti MySQL Schema

Tables (3 items)

Add Table	assegnazione	esami	progetti
-----------	--------------	-------	----------

Views (1 items)

Add View	assegnazioni
----------	--------------

Routines (2 items)

Add Routine	assegnazione	conferma
-------------	--------------	----------

Routine Groups (0 items)

Add Group
-----------

▼ Schema Privileges

Users (2 items)

Add User	docente	studente
----------	---------	----------

studente - Role x

Objects	Privileges for Selected Object
assegnazione	<input type="checkbox"/> GRANT OPTION
progetti	<input type="checkbox"/> REFERENCES
esami	<input type="checkbox"/> ALTER
assegnazione	<input type="checkbox"/> DELETE
conferma	<input type="checkbox"/> INDEX
	<input checked="" type="checkbox"/> INSERT
	<input checked="" type="checkbox"/> SELECT
	<input checked="" type="checkbox"/> UPDATE
	<input type="checkbox"/> TRIGGER

Drag objects from Physical Schemas to the Objects list to give rights on it for this role. Once you select an Object, you may select the privileges granted to the role for that object.

Check All Uncheck All

User Types History Role Privileges

# Gestione dei privilegi

The screenshot shows the MySQL Workbench interface with the following components:

- Menu Bar:** File, Edit, View, Arrange, Model, Database, Tools, Scripting, Help.
- Toolbar:** Includes icons for Add Diagram, EER Diagram, and other database actions.
- Physical Schemas:** A tree view showing the 'progetti' MySQL Schema.
- Tables (3 items):** Includes 'assegnazione', 'esami', and 'progetti'.
- Views (1 item):** Includes 'assegnazioni'.
- Routines (2 items):** Includes 'assegnazione' and 'conferma'.
- Routine Groups (0 items):** Empty list.
- Schema Privileges:** A list of users: 'docente' and 'studente'.
- studente - Role:** A detailed view of the 'studente' role's privileges.

**studente - Role**

Objects	Privileges for Selected Object
<ul style="list-style-type: none"><li>assegnazione</li><li>progetti</li><li>esami</li><li><b>assegnazione</b> (selected)</li><li>conferma</li></ul>	<ul style="list-style-type: none"><li><input type="checkbox"/> ALL</li><li><input type="checkbox"/> ALTER ROUTINE</li><li><input type="checkbox"/> CREATE ROUTINE</li><li><input checked="" type="checkbox"/> EXECUTE</li></ul>

**Privileges for Selected Object**

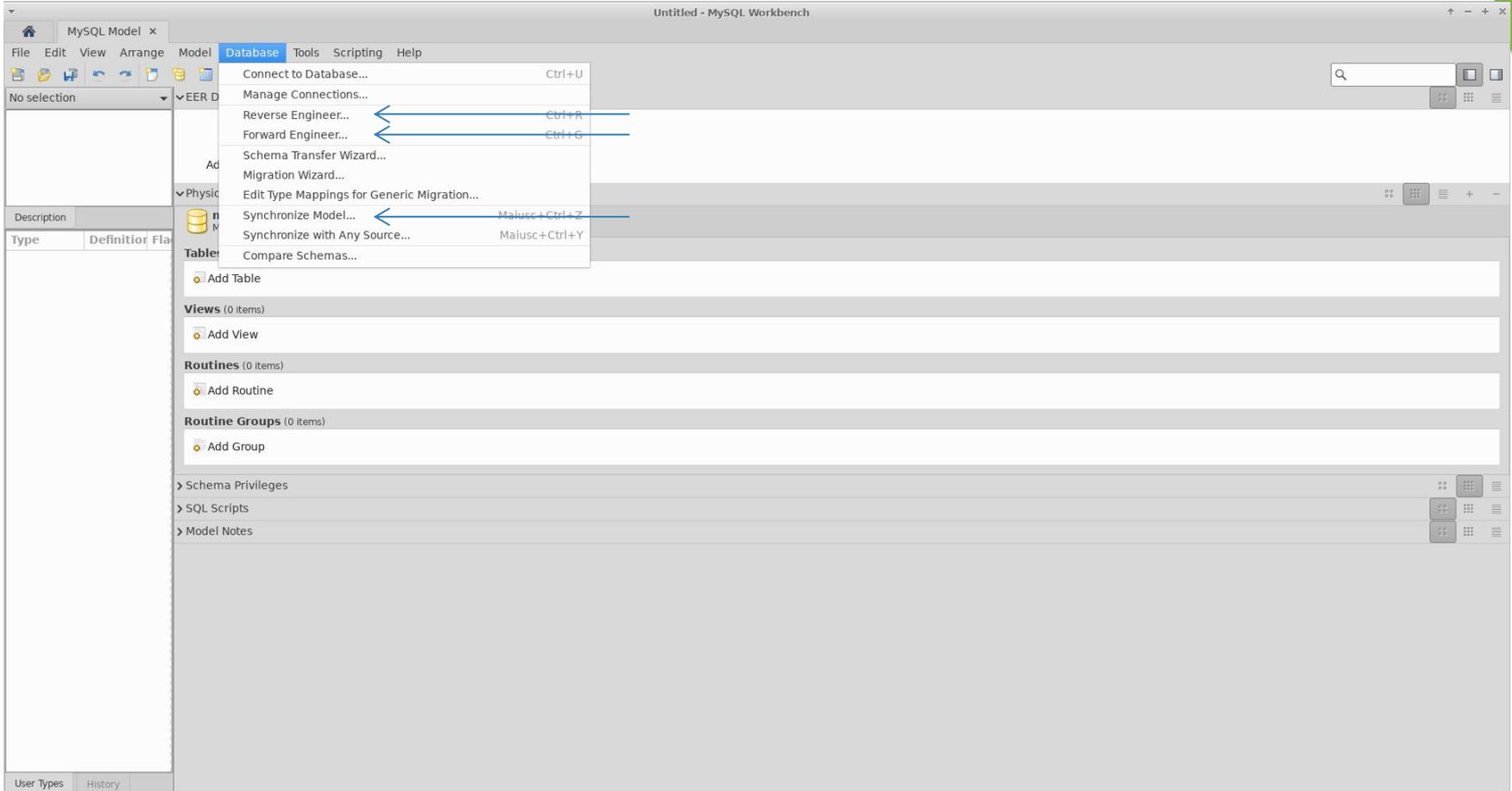
Drag objects from Physical Schemata to the Objects list to give rights on it for this role. Once you select an Object, you may select the privileges granted to the role for that object.

# Editor delle procedure

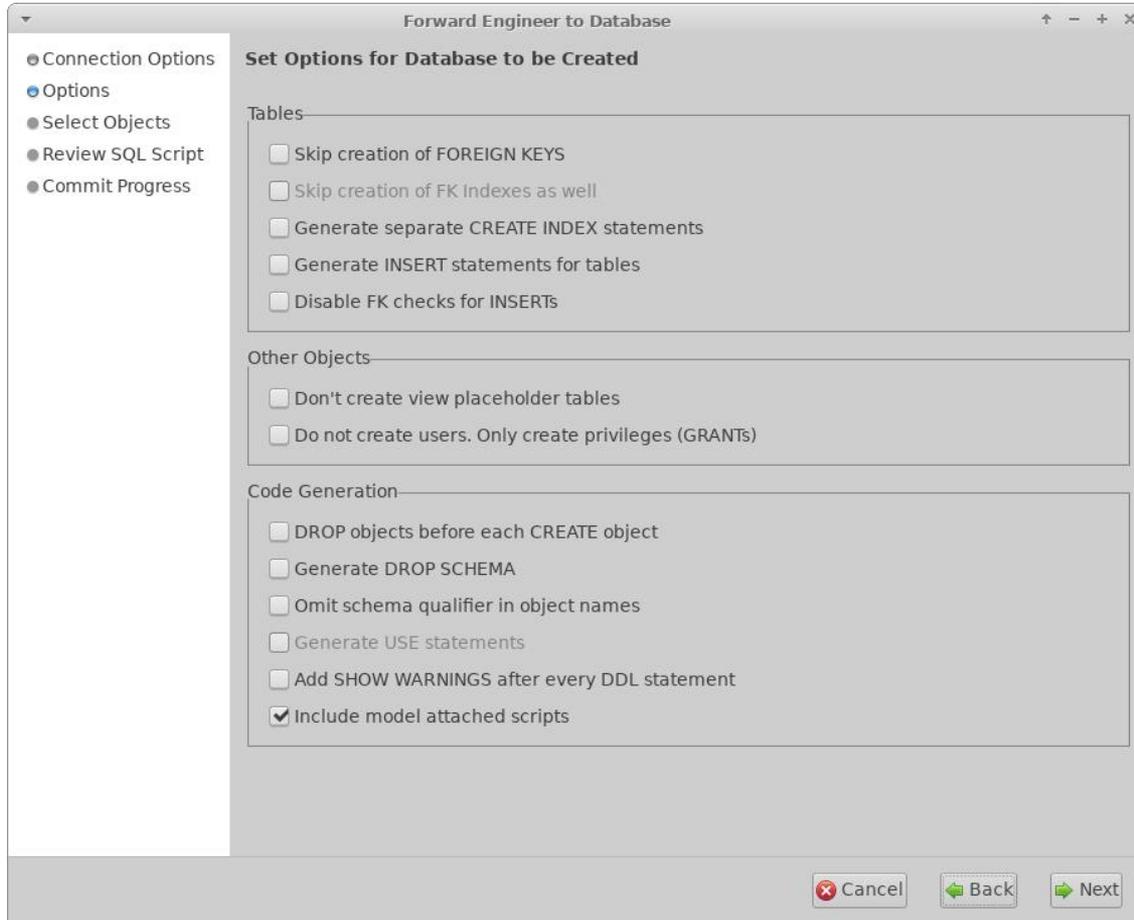
The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Arrange, Model, Database, Tools, Scripting, and Help. The toolbar contains icons for file operations and database actions. The left sidebar shows the 'Physical Schemas' tree with a 'progetti' schema containing tables 'assegnazione', 'esami', and 'progetti', and a view 'assegnazioni'. The main editor window is titled 'assegnazione - Routine' and shows the following SQL code:

```
DDL:
1 • create procedure `assegnazione` (in var_matricola varchar(45), in var_nome varchar(45), in var_cognome varchar(45), in var_email varchar(45), in var_id_esame int, out
2   begin
3     declare stud_id int;
4     declare conferma int;
5     declare progetto int;
6
7     # We don't want to leave doomed transactions around
8     declare exit handler for sqlexception
9     begin
10      rollback; -- rollback any changes made in the transaction
11      resignal; -- raise again the sql exception to the caller
12    end;
13
14    # Avoid concurrency problems embedding everything in a transaction
15    start transaction;
16
17    # Check if the student has already requested or has been assigned a project
18    select id, conferma, id_taken from `assegnazioni` where
```

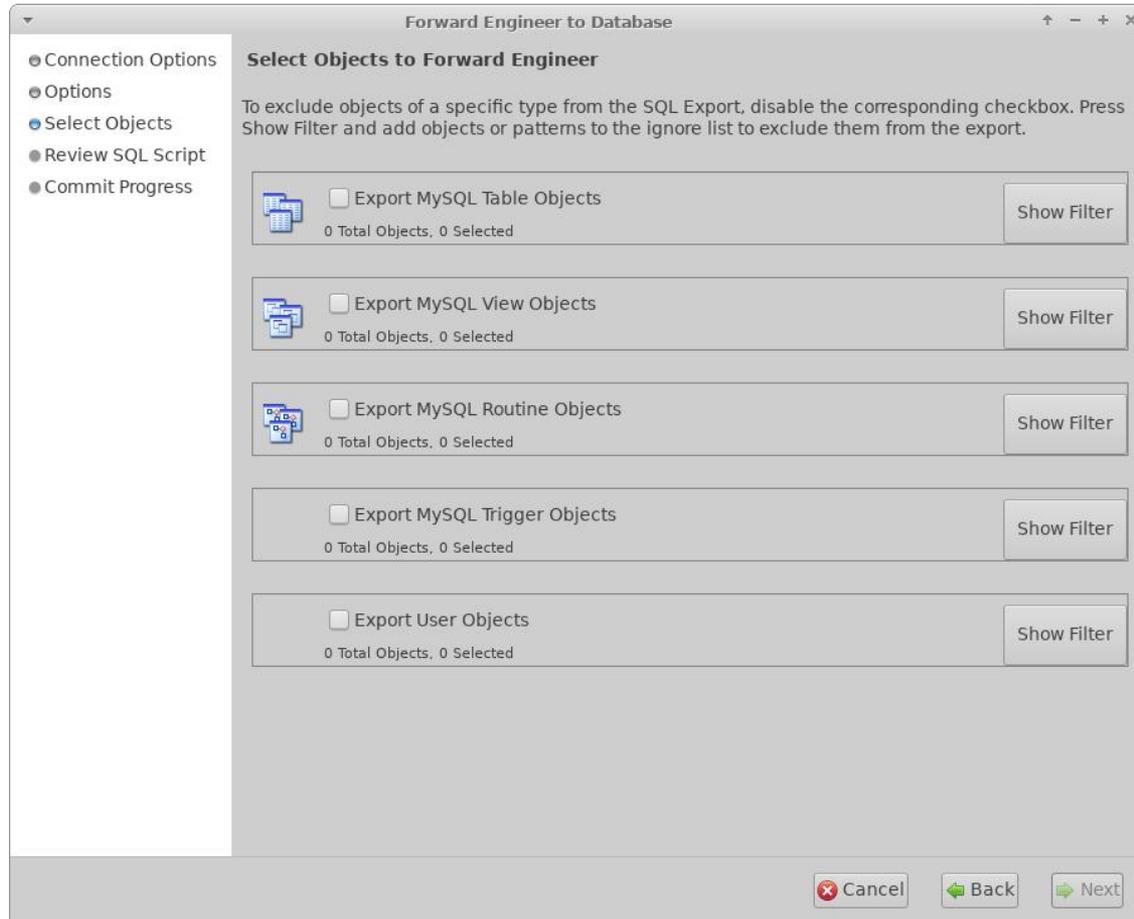
# Deploy



# Deploy



# Deploy



# DBA Operations

The screenshot displays the MySQL Workbench interface. The title bar reads "applicazione-1.mwb\* - MySQL Workbench". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The toolbar contains icons for file operations, query execution, and server management. The left sidebar is divided into three sections: MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), and PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup). The main workspace shows the "Administration - Server Status" window with a toolbar that includes a "Limit to 1000 rows" dropdown. The content area is mostly blank, with a small "1" visible at the top left. At the bottom left, the "Object Info" tab shows "No object selected". The "Action Output" tab is active, displaying a table with the following structure:

#	Time	Action	Message	Duration / Fetch
---	------	--------	---------	------------------

Management support for target host enabled successfully.

# DBA Operations

The screenshot shows the MySQL Workbench interface for the 'Administration - Server Status' page. The window title is 'applicazione-1.mwb\* - MySQL Workbench'. The main content area displays the following information:

- Connection Name:** pellegriini.tk
- Host:** pellegriini.tk
- Socket:** /var/run/mysql/mysql.sock
- Port:** 3306
- Version:** 10.3.17-MariaDB-0+deb10u1 (Debian 10)
- Compiled For:** debian-linux-gnu (x86\_64)
- Configuration File:** unknown
- Running Since:** Tue Sep 17 13:21:42 2019 (74 days 11:14)

A 'Refresh' button is located below the 'Running Since' information.

**Available Server Features:**

Performance Schema:	<input type="radio"/> Off	PAM Authentication:	<input type="radio"/> Off
Thread Pool:	<input type="radio"/> n/a	Password Validation:	<input type="radio"/> n/a
Memcached Plugin:	<input type="radio"/> n/a	Audit Log:	<input type="radio"/> n/a
Semisync Replication Plugin:	<input type="radio"/> Off	Firewall:	<input type="radio"/> n/a
SSL Availability:	<input type="radio"/> Off	Firewall Trace:	<input type="radio"/> n/a

**Server Directories:**

- Base Directory: /usr
- Data Directory: /var/lib/mysql/
- Disk Space in Data Dir: unable to retrieve
- Plugins Directory: /usr/lib/x86\_64-linux-gnu/mariadb19/plugin/
- Tmp Directory: /tmp

**Performance Metrics (Right Panel):**

- Server Status: Running
- JLoad: ---
- Connections: 7
- Traffic: 7.24 KB/s
- Key Efficiency: 99.1%
- Selects per Second: 0
- InnoDB Buffer Usage: 7.6%
- InnoDB Reads per Second: 0
- InnoDB Writes per Second: 0

**Bottom Panel:**

Object Info: Session | No object selected

Action Output:

#	Time	Action	Message	Duration / Fetch
---	------	--------	---------	------------------

# DBA Operations

The screenshot shows the MySQL Workbench Administration - Client Connections window for a server named 'pellegrini.tk'. The window displays a table of active connections with the following summary statistics:

- Threads Connected: 7
- Threads Running: 7
- Threads Created: 2886
- Threads Cached: 0
- Rejected (over limit): 0
- Total Connections: 26365
- Connection Limit: 30
- Aborted Clients: 6
- Aborted Connections: 25764
- Errors: 0

The table below lists the details of the connections:

Id	User	Host	DB	Command	Time	State	Info
1	system user		None	Daemon	0	InnoDB pu	NULL
2	system user		None	Daemon	0	InnoDB pu	NULL
3	system user		None	Daemon	0	InnoDB pu	NULL
4	system user		None	Daemon	0	InnoDB pu	NULL
5	system user		None	Daemon	0	InnoDB shi	NULL
6	event_scheduler	localhost	None	Daemon	643414	Waiting for	NULL
263	remote-admin	host157-239-dyr	progetti	Sleep	868	NULL	
263	remote-admin	host157-239-dyr	progetti	Sleep	867	NULL	
263	remote-admin	host157-239-dyr	None	Sleep	843	NULL	
263	remote-admin	host157-239-dyr	progetti	Sleep	95	NULL	
263	remote-admin	host157-239-dyr	progetti	Sleep	93	NULL	
263	remote-admin	host157-239-dyr	None	Query	0	init	SHOW FULL PROCESSLIST
263	remote-admin	host157-239-dyr	None	Sleep	2	NULL	

At the bottom of the window, there are controls for refreshing the data and buttons for 'Kill Query(s)', 'Kill Connection(s)', and 'Refresh'. The 'Refresh Rate' is set to 'Don't Refresh' and the 'Hide sleeping connections' checkbox is unchecked.

# DBA Operations

The screenshot displays the MySQL Workbench Administration - Dashboard for a server named 'applicazione-1.mwb\*'. The interface is divided into several sections:

- MANAGEMENT:** Includes links for Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, and Data Import/Restore.
- INSTANCE:** Includes links for Startup / Shutdown, Server Logs, and Options File.
- PERFORMANCE:** Includes links for Dashboard, Performance Reports, and Performance Schema Setup.

The main dashboard area contains several performance and status widgets:

- Network Status:** Shows Incoming Network Traffic (7.00 B/s receiving) and Outgoing Network Traffic (4.97 KB/s sending).
- MySQL Status:** Shows Table Open Cache Efficiency at 91%.
- InnoDB Status:** Shows InnoDB Buffer Pool Usage at 7%, Redo Log data written (0 B/s), InnoDB Disk Writes (0.00 B/s), and InnoDB Disk Reads (0.00 B/s).
- Client Connections (Total):** Shows 7 connections out of a limit of 30.
- SQL Statements Executed (#):** Lists executed statements: SELECT (0/s), INSERT (0/s), CREATE (0/s), UPDATE (0/s), ALTER (0/s), DELETE (0/s), and DROP (0/s).

The bottom of the window features an 'Action Output' table with columns for #, Time, Action, Message, and Duration / Fetch.

#	Time	Action	Message	Duration / Fetch