GRAUMAN

HI-TECH DEV COURSES

Fullstack Course

HTML | CSS | JavaScript | React

SQL | Java | REST

Shahar Grauman

www.grauman.co.il | 054-8002004 | info@grauman.co.il

Relational Database

A relational database organizes data in tables.

- A table has rows and columns.
- Tables are related based on common columns.

Popular RDBMS (Relationship **D**atabase **M**anagement **S**ystem**):**

**Microsoft SQL Server and Access, Oracle,
IBM DB2, MySQL, SQLite and more**

Structured Query Language (SQL)

Created in 1970 - designed for interacting with the relational databases.

SQL defines a set of commands, such as SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE, etc.

The first standard was made in 1986 by the **ANSI** (American National Standard Institute)

Database Structure

We split entities into **tables**, such as Customers, Orders, Products…

A collection of tables called **Schema**.

Database contains 1 schema or more.

Each **table name is unique** in a schema, has a **PK** (primary key) and each column has a **value type**.

Regarding value types, there are many, but here are part of them:

BIT – 1 to 64 bits. BIT(3) can hold 3 bits like 101
TINYINT – 1 byte
INT – a whole number ranging from -2147483648 to 2147483647.
BIGINT – 8 bytes (twice the INT), -2^63 - 1 to 2^64 - 1
DOUBLE or **FLOAT** – numbers with floating point.
VARCHAR – string up to 65535 characters (usually limited: VARCHAR(50)).
DATETIME – YYYY-MM-DD HH:MM:SS (DATE is a type only for the date part).
TIMESTAMP – auto initialization and update.
YEAR – 1 byte with valid values from 1901 to 2155.
TIME – hhh:mm:ss, from '-838:59:59' to '838:59:59'.
ENUM – list of permitted values (colors ENUM('red', 'green', 'blue'), red is 1, green is 2… empty string is 0, can have NULL)

MySQL Shell

CLI – Command **L**ine **I**nterface tool**.**

Connecting to our MySQL server:

|connect --mysql root@localhost:3306

Show all databases/schemas:

show databases;

Selecting a specific db (for further usage):

|use college;

Showing college's tables:

show tables;

Querying:

*select * from student;*

Showing table structure:

describe student;

Showing the creation script:

show create table student;

Creating new db/schema:

create database grauman;

Selecting this newly created db:

use grauman;

Create movie table:

```
create table movie (id int not null auto_increment,
          name varchar(30),
          category enum('comedy', 'horror', 'family', 'kids', 'action'),
          length time,
          producer int,
          primary key (id));
```

*('Primary key' can be used in the same column declaring line)

Create producer table:

```
create table producer (id int not null auto_increment,
          name varchar(30),
          age tinyint,
          country varchar(50),
          primary key (id));
```

Adding values to producer table:

```
insert into producer (name, age, country) values ('Kaspulski', 67, 'Poland');
```

Or multiple rows:

```
insert into producer values(default, 'Lorem', 49, 'Italy'), (default, 'Tarra
Yanovsky', 81, 'Netherland'),(default, 'Yokomora', 56, 'Japan');
```

Adding FK (Foreign Key) to movie table:

```
alter table movie add constraint fk_movie_producer foreign key
(producer) references producer(id);
```

<u>Adding values to movie table:</u>

insert into movie values(default, 'Gone by the bit', 'horror', '7:11:48', 1);

Or multiple rows:

insert into movie values(default, 'The (semicolon) Terminator', 'action', '2:30:00',2),(default, 'Bugs Life', 'kids', '1:30:00',3),(default, 'Programming Heroes', 'action', '2:05:00',1),(default, 'Learning Curves', 'comedy', '1:47:00',3);

<u>Select multiple columns into 1:</u>

select concat(name, '-', category) 'movie-category' from movie;

<u>Select rows containing some value:</u>

*select * from movie where category like '%rr%'*

'%rr%' – 'rr' anywhere inside this column value

'ho% - starting with 'ho'

'%or' – ending with 'or'

'ho_r' – starting with 'ho' and ending with 'r'

*add *not* just before *like* to negate

<u>Range:</u>

between val1 and val2

<u>Order by (default to asc or desc):</u>

*select * from movies order by length desc;*

With sub-ordering in case of similarity:

*select * from producer order by name, age desc;*

<u>Select values from a list:</u>

*select * from movie where id in (1,3);*

Limit:

*select * from movies order by length desc limit 2;*

Rename output column names:

select id, name Movie from movie;

Aggregations:

Sum:

select time(sum(length)) from movie;

Select without duplicated rows - distinct:

select distinct producer from movie;

Max/Min:

select name, max(length) from movie;

Avg:

select avg(age) ‘avg age’ from producer;

select sec_to_time(avg(time_to_sec(length))) from movie;

Round, Floor, Ceil:

select round(avg(age)) ‘avg age’ from producer;

Group By:

select producer, count(*) from movie group by producer;

*Notice that group by uses aggregations (such as count) on some column to aggregate upon.

Joins:

Cartesian select:
*select * from movie,producer;*
Is equivalent to:
*select * from movie join producer;*

*select * from movie,producer where movie.producer=producer.id;*
Is equivalent to:
*select * from movie join producer on movie.producer=producer.id;*

left/right join – Nulls where no match:
*select * from movie right join producer on movie.producer=producer.id;*

Using – specify colum(s) to be compared:
*select * from movie left join producer using(id);*

*Notice that in this example I show comparison by id which isn't PK-FK relationship, meaning that if id values from the left side aren't present in the id values from the right side – Nulls are shown.

Delete:
Delete from producer where name = 'Yokomora';

Update:
Update producer set name='Yokomora!' where name = 'Yokomora';

Create new user:
Create user 'someuser'@'localhost' identified by '123456';
grant select, insert, update on grauman. to 'someuser'@'localhost';*

|*connect --mysql someuser@localhost:3306*
(…enter password)
use grauman;
*select * from movie;* (fine)
delete from movie; (denied)

<ins>More utility functions:</ins>

select log(2,128); Or *select log2 (128);*
select log(10,10000); Or *select log10(10000);*

select date_add(date_format(now(), '%Y-%m-%d %T'), interval 1 year);
select from_days(datediff('2018-10-10', '2010-11-11'));
select extract(year from now());
select lpad('lala', 10, '!');
select replace('shahar', 'ha', 'HA');

In MySQL we can create procedures for using later on

Northwind

1. Show FirstName, HireDate, Region and Country from Employees
2. From Products, show ProdId, ProdName, Price (from UnitPrice)
3. From Customers, show Customer Id and City + Address in 1 column alias as Full Address
4. Which countries are the employees coming from?
5. Show Product names, their price and their price without VAT
6. Which orders came to their destination in delay?
7. Which employees living in London or Seattle or Tacoma?
8. Which customers living in an unknown area?
9. Show the last 7 orders
10. Show product names, but only the first 10 characters.
11. For each employee show the name + last name and position (if ‘Sales’ exists, replace with ‘Marketing’)
12. From Products show:
    a. Product id + supplier id (with ‘AND’ in between)
    b. Price without VAT rounded bottom. Name it ‘No VAT’

Show only the products which their new price is over 30.

13. Show employees: first name in lowercase, last name reversed. Show only employees which have a manager.
14. Show all employee details for employees having longer last name than the first.
15. Show from orders: Order number, employee number, order date, required date and ship name:

   a. Employee number is 7
   b. Shipping name is of QUICK-Stop/Around the Horn/ Frankenversand
   c. Should be delivered in less than 2 weeks

16. From products, show product number, name and price
    for products which cost more than Chocolade.
17. Show order number, date and ship address and customer code, name and phone
    for orders made in 1996 and customer code starting with A or C.
18. Show employee name and hire date for employees hired later than employee #6
19. Show product number, name and unit price for products which cost more than the average
20. Show product name and quantity for products having less quantity than the minimum quantity of category #5
21. Show all product details which in the same category as Chai.
    Don't show Chai itself.
22. Show product name and price for products with the same price as in category #5
23. Show product name and price for products which cost more than at least 1 product in category #5
24. Show product name and price for products which cost more than all the products in category #5
25. Show order numbers and dates for all orders which their customers are from Franch, Germany or Sweden and order date was in 1997
26. Show product names and code.
    Include the products which cost more than the average price of those having more than 50 units in stock.

27. Show all product names in Beverages or Condiments categories and their supplier region is unknown
28. Create procedure – ‘Top Product Sales By Year’ which gets a year as an argument, and returns the name and the total of items of the most sold product in a given year