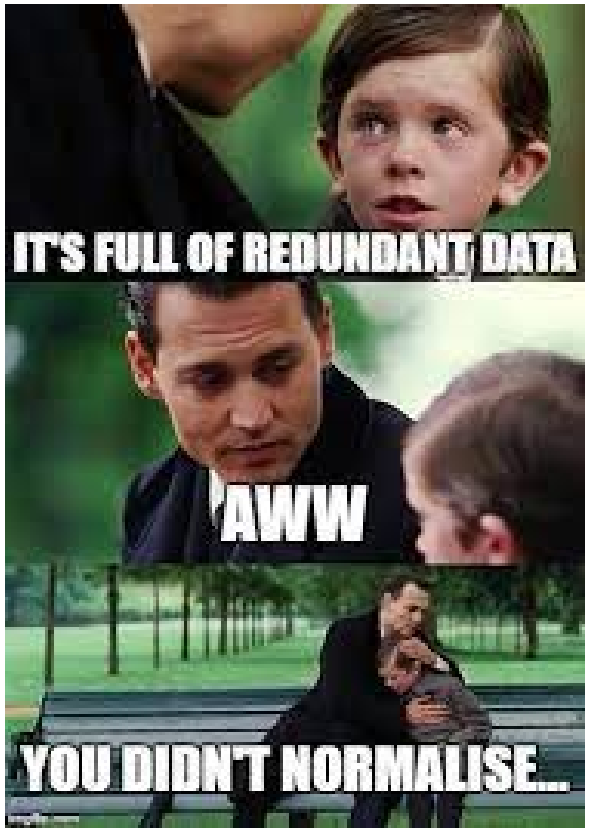
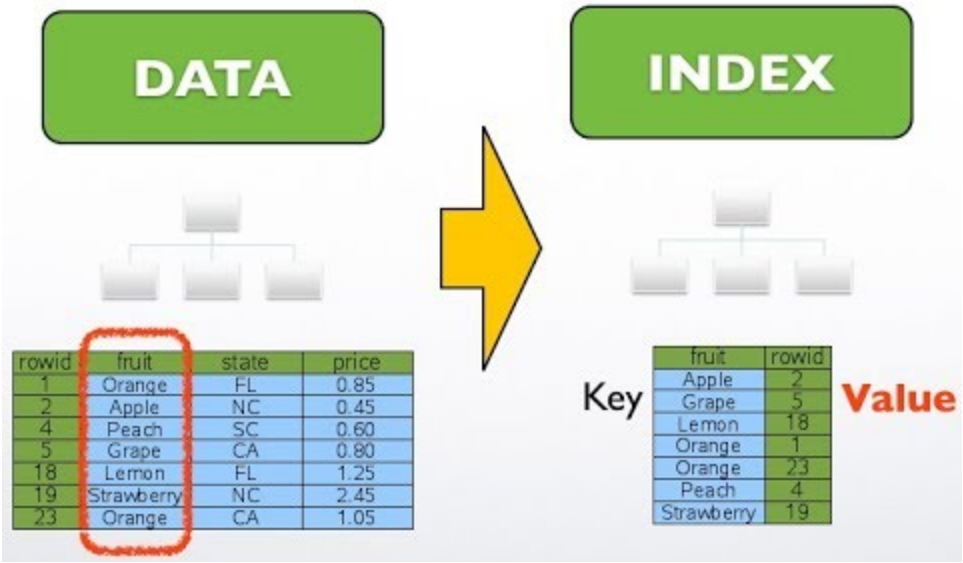
Lab5 Indexes, Constraints, Views



Lab5: Agenda 📋

- Indexes
- Constraints
- Views
- Εργαστηριακές Ασκήσεις
- Εξαμηνιαία Εργασία

Index


DATA
INDEX
rowid fruit state price
1 Orange FL 0.85
2 Apple NC 0.45
4 Peach SC 0.60
5 Grape CA 0.80
18 Lemon FL 1.25
19 Strawberry NC 2.45
23 Orange CA 1.05
Key
fruit rowid
Apple 2
Grape 5
Lemon 18
Orange 1
Orange 23
Peach 4
Strawberry 19
Value

Index

- when you query a table, the dbms will perform a table scan
  - will need to inspect every row of the table to answer the query
- At some number of rows it might take some time answer the query
- a dbms uses indexes to locate rows in a table
  - special tables kept in a specific order
  - facilitate the retrieval of a subset of a table's rows / columns
  - without the need to inspect every row in the table.

Index → Creation

- create index on the email column named idx_email

```
CREATE INDEX idx_email ON customer (email);
```

- view indexes

```
SHOW INDEX FROM customer;
```

- delete index

```
DROP INDEX idx_email ON customer;
```

Index → Unique indexes

- which columns are allowed to contain duplicate data and which are not
  - two customers with same name 🆗
  - two different customers to have the same email address ❓
- unique index
  - regular index
  - NO duplicate values in the indexed column

Index → Create Unique indexes

```
CREATE UNIQUE INDEX idx_email ON customer (email);
```

```
MariaDB [sakila]> INSERT INTO customer (store_id, first_name, last_name, email, address_id, active)
 VALUES (1,'ALAN','KAHN', 'ALAN.KAHN@sakilacustomer.org', 394, 1);
ERROR 1062 (23000): Duplicate entry 'ALAN.KAHN@sakilacustomer.org' for key 'idx_email'
```

- PK column(s) already have checks for uniqueness

Index → Multicolumn indexes

- indexes that span multiple columns
- search for customers by first and last names

```
CREATE INDEX idx_full_name ON customer (last_name, first_name);
```

- is it useful for queries that specify only the customer's first name ?

Index → Example index on Sakila

```
show index from customer;
```

```
MariaDB [sakila]>show index from customer;
+----------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name          | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| customer |          0 | PRIMARY           |            1 | customer_id | A         |         599 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| customer |          1 | idx_fk_store_id   |            1 | store_id    | A         |           2 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| customer |          1 | idx_fk_address_id |            1 | address_id  | A         |         599 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| customer |          1 | idx_last_name     |            1 | last_name   | A         |         599 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.01 sec)
```

Index → Effect of index

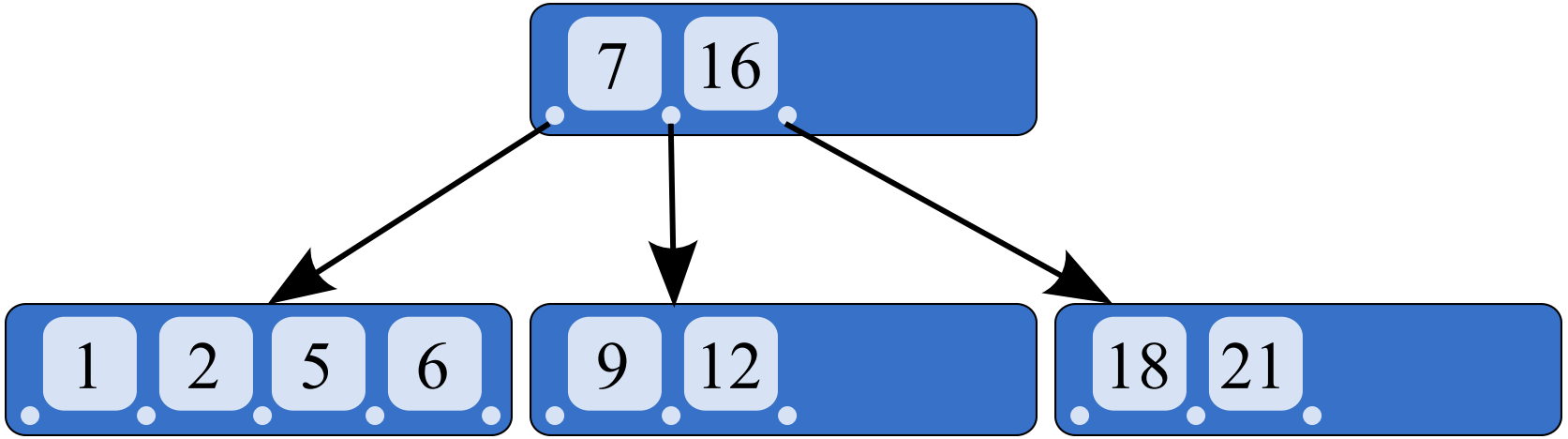
```
MariaDB [sakila]>explain select * from customer where first_name = 'CAROL';
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  599 |    10.00 | Using where |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
MariaDB [sakila]>explain select * from customer where last_name = 'GARCIA';
+----+-------------+----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key           | key_len | ref   | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | customer | NULL       | ref  | idx_last_name | idx_last_name | 137     | const |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

Index → Types of Indexes

B-tree indexes (default)

- balanced-tree indexes
- great at handling columns that contain many different values,



- with insert/ update/ delete the server will attempt to keep the tree balanced

Index → Types of Indexes

Bitmap indexes

- great for low-cardinality data
- breaks down if the number of values stored in the column climbs too high in relation to the number of rows (high-cardinality)

Text indexes

- specialized indexing and search mechanisms for documents

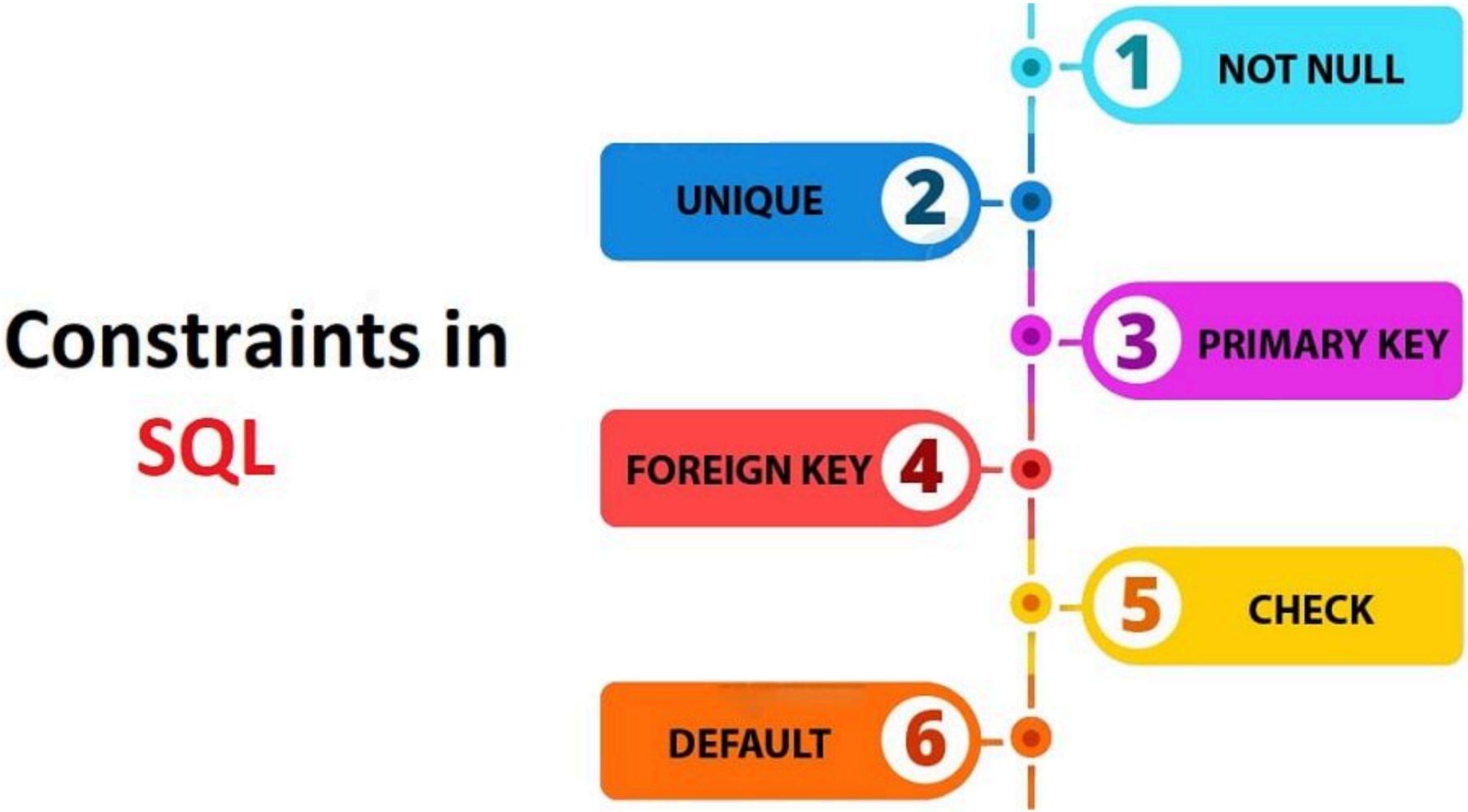
Index → The Downside of Indexes

- why not index everything?
  - every time a row is added to (removed/ updated,) from a table, all indexes on that table must be modified
    - more work for the server
  - disk space
  - database admin has to periodicaly check

Index → Strategy

- All primary key columns are indexed
- Build indexes on all columns that are referenced in foreign key constraints
- Index any columns that will frequently be used to retrieve data
- Look at the server's execution plan, and modify indexing strategy to fit the most common access paths

Constraints



SQL Constraints

> ℹ A restriction placed on one or more columns of a table or a table

- **NOT NULL**
  - Ensures that a column cannot have a NULL value
- **UNIQUE**
  - Ensures that all values in a column are different
- **DEFAULT**
  - Sets a default value for a column if no value is specified
- **Primary key** constraints
  - Identify column(-s) that guarantee uniqueness within a table (NOT NULL and UNIQUE)
- **Foreign key** constraints
  - Restrict column(-s) to contain only values found in another table's primary key columns
- **Check** constraints
  - Ensures that the values in a column satisfies a specific condition

Constraints → Creation

- generally created at the same time as the associated table
- alter table

```
CREATE TABLE customer (
customer_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
store_id TINYINT UNSIGNED NOT NULL,
first_name VARCHAR(45) NOT NULL,
address_id SMALLINT UNSIGNED NOT NULL,
active BOOLEAN NOT NULL DEFAULT TRUE,
create_date DATETIME NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
```

Constraints → Sakila Customer Constraints

```
PRIMARY KEY (customer_id),
KEY idx_fk_store_id (store_id),
KEY idx_fk_address_id (address_id),
KEY idx_last_name (last_name),
CONSTRAINT fk_customer_address FOREIGN KEY (address_id)
  REFERENCES address (address_id) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_customer_store FOREIGN KEY (store_id)
  REFERENCES store (store_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Sakila Customer Constraint Creation

- ALTER TABLE

```sql
ALTER TABLE customer
ADD CONSTRAINT fk_customer_address FOREIGN KEY (address_id)
REFERENCES address (address_id) ON DELETE RESTRICT ON UPDATE CASCADE
```

Constraints → Sakila Customer Constraint

- 3 Constraints
  1. PRIMARY KEY (customer_id)
  2. CONSTRAINT fk_customer_address FOREIGN KEY (address_id)
  3. CONSTRAINT fk_customer_store FOREIGN KEY (store_id)

Constraints → Foreign key Constraint

```
ON DELETE RESTRICT ON UPDATE CASCADE
```

- on delete restrict
  - raise an error if a row is deleted in the parent table that is referenced in the child table
  - protects against orphaned records

Constraints → Foreign key Constraint

```
ON DELETE RESTRICT ON UPDATE CASCADE
```

- on update cascade
  - on update the parent table update automatically data referenced in the child table
  - protects against orphaned records

Constraints → Foreign key Constraint different options

- ON DELETE
  - RESTRICT (default ✓)
  - CASCADE 😥
  - SET NULL 😥
- ON UPDATE
  - RESTRICT (default ✓)
  - CASCADE
  - SET NULL

Constraints → Check constraints 🤓

```sql
CREATE TABLE t2
     (
      CHECK (c1 <> c2),
      c1 INT CHECK (c1 < 100),
      c2 INT CHECK (c2 > 0),
      c3 INT CHECK (c3 < 10),
      CONSTRAINT c1_nonzero CHECK (c1 <> 0),
      CHECK (c1 > c3)
    );
```

Constraints → Check constraints 🤓

```
MariaDB [sakila]> insert into t2 values(0,1,3);
ERROR 4025 (23000): CONSTRAINT `c1_nonzero` failed for `sakila`.`t2`

MariaDB [sakila]> insert into t2 values(1,1,3);
ERROR 4025 (23000): CONSTRAINT `CONSTRAINT_1` failed for `sakila`.`t2`

MariaDB [sakila]> insert into t2 values(5,6,3);
Query OK, 1 row affected (0.016 sec)
```

Constraints → Tips 🥋

- Do not hesitate to put constraints on the database
  - helps in a consistent database
- Consider how your application should respond to various cases
- Cascades may or may not be firing triggers
- **ON DELETE RESTRICT ON UPDATE CASCADE** 🥰

Views

CustomerID	CustomerName	City
1	John	NYC
2	Mike	Rome
3	Sunny	Berlin

OrderID	Quantity	CustomerID
100	2	2
101	4	3
102	10	1

CREATE VIEW

CustomerID	OrderID	Quantity
1	102	10
2	100	2
3	101	4

Views → What is a View

- a mechanism for querying data
- a query that is stored in the data dictionary.
- It looks and acts like a table but without any data (~virtual table)
- users are unaware of querying a view

Views → Why Use Views?

- Data Security
- Data Aggregation
- Hiding Complexity
- Joining Partitioned Data

Views → Creation

```sql
CREATE VIEW customer_vw
(customer_id, first_name, last_name, email)
AS
SELECT
  customer_id, first_name, last_name,
  concat(substr(email,1,2), '*****', substr(email, -4)) email
FROM customer;
```

Views → Query View

```sql
SELECT first_name, last_name, email  FROM customer_vw;
```

```
+-------------+--------------+-------------+
| first_name  | last_name    | email       |
+-------------+--------------+-------------+
| MARY        | SMITH        | MA*****.org |
| PATRICIA    | JOHNSON      | PA*****.org |
| LINDA       | WILLIAMS     | LI*****.org |
                  ...
| WADE        | DELVALLE     | WA*****.org |
| AUSTIN      | CINTRON      | AU*****.org |
+-------------+--------------+-------------+
599 rows in set (0.004 sec)
```

Views → Query View

```sql
SELECT cv.first_name, cv.last_name, p.amount FROM customer_vw cv
INNER JOIN payment p ON cv.customer_id = p.customer_id WHERE p.amount >= 11;
```

```
+------------+-----------+--------+
| first_name | last_name | amount |
+------------+-----------+--------+
| KAREN      | JACKSON   |  11.99 |
| VICTORIA   | GIBSON    |  11.99 |
| VANESSA    | SIMS      |  11.99 |
            ...
| TERRANCE   | ROUSH     |  11.99 |
+------------+-----------+--------+
10 rows in set (0.062 sec)
```

Views → Updatable Views 🤯

- modify data through a view ❓
- a view is **updatable** if
  - No aggregate functions are used
  - No group by or having clauses
  - No union, union all, or distinct.
  - No subqueries exist in the select or from clause - any subqueries in the where clause do not refer to tables in the from clause
  - from clause
    - includes at least one table or updatable view
    - uses only inner joins if there is more than one table or view

Views → Simple update case

```sql
UPDATE customer_vw SET last_name = 'SMITH-ALLEN'  WHERE customer_id = 1;
```

```sql
SELECT first_name, last_name, email FROM customer  WHERE customer_id = 1;
```

```
+------------+-------------+-------------------------------+
| first_name | last_name   | email                         |
+------------+-------------+-------------------------------+
| MARY       | SMITH-ALLEN | MARY.SMITH@sakilacustomer.org |
+------------+-------------+-------------------------------+
```

Views → Updating Complex Views

```sql
CREATE VIEW customer_details AS
SELECT c.customer_id, c.store_id, c.first_name, c.last_name, c.address_id, c.active, c.create_date,
       a.address, ct.city, cn.country, a.postal_code
FROM customer c
INNER JOIN address a
ON c.address_id = a.address_id
INNER JOIN city ct
ON a.city_id = ct.city_id
INNER JOIN country cn
ON ct.country_id = cn.country_id;
```

Views → Updating Complex Views

- update on customer
  - ```
    UPDATE customer_details  SET last_name = 'SMITH-ALLEN', active = 0  WHERE customer_id = 1;
    ```
    - ```
      Query OK, 1 row affected (0.017 sec)
      Rows matched: 1  Changed: 1  Warnings: 0
      ```
- update on address
  - ```
    UPDATE customer_details  SET address = '999 Mockingbird Lane'  WHERE customer_id = 1
    ```
    - ```
      Query OK, 1 row affected (0.017 sec)
      Rows matched: 1  Changed: 1  Warnings: 0
      ```

Views → update columns from both tables

- ```
  UPDATE customer_details
  SET last_name = 'SMITH-ALLEN', active = 0, address = '999 Mockingbird Lane'
  WHERE customer_id = 1;
  ```
  - ```
    ERROR 1393 (HY000): Can not modify more than one base table through a join view
    ```

Views → Insert into a View 🙈🙉🙊

- insert customer
  - ```
    INSERT INTO customer_details (customer_id, store_id, first_name, last_name, address_id, active, create_date)
    VALUES (9998, 1, 'BRIAN', 'SALAZAR', 5, 1, now());
    ```
    - ```
      Query OK, 1 row affected (0.019 sec)
      ```
- insert customer + address
  - ```
    INSERT INTO customer_details  (customer_id, store_id, first_name, last_name,  address_id, active, create_date, address)
    VALUES (9999, 2, 'THOMAS', 'BISHOP', 7, 1, now(),  '999 Mockingbird Lane');
    ```
    - ```
      ERROR 1393 (HY000): Can not modify more than one base table through a join view 'sakila.customer_details'
      ```

Views → Insert into a View 🙈🙉🙊

```
INSERT INTO customer_vw (customer_id, first_name,  last_name) VALUES (99999,'ROBERT','SIMPSON');
```

```
ERROR 1471 (HY000): The target table customer_vw of the INSERT is not insertable-into
```

Εργαστηριακές Ασκήσεις 💻

1. Your manager wants a report that includes the name of every country, along with the total payments for all customers who live in each country. Create a view for the report.
2. Create a table 'suppliers' with 2 unique constraints. (One should be on name, address fields and the other on the phone field). Insert 2 suppliers with name and address as 'ABC Inc', '4000 North 1st Street' and 'XYZ Inc', '4000 North 1st Street'.
3. You are unhappy about the name, address constraint. Delete the constraint.
4. Your manager thinks a constrain is needed on the suppliers table name field. Write a query to achieve that.
5. Create a view to locate customers living in 'France'.
6. Create a view to list staff (id, name, phone city, country, store_id)

Εξαμηνιαία Εργασία 💻

- Database Schema Design
  - i. Start thinking about the entities you need
    - Identify entities, attributes and relationships from the problem description
    - identify cardinality ratios of the relationships found
  - ii. Design an E/R diagram for your database
    - Look for any issues that are apparent in the E/R diagram
  - iii. Convert the E/R diagram to a relational schema
    - Identify primary keys, foreign keys, and any other constraints
    - Normalize the schema if necessary

Εξαμηνιαία Εργασία 💻

- Materialize Schema: DDL statements
  - i. Create your tables
    - create a table for each entity
    - a table (representing an entity) should have:
      - a column for each attribute, with appropriate data type
      - a primary key and possibly some candidate keys
    - include a foreign key (one-to-many relationships)
    - add indexes & constraints to your tables
  - ii. Create views as needed

Εξαμηνιαία Εργασία 💻

- Add Information to the Database: DML script
  - Populate the database with data

> ℹ️ start running your SQL commands from a separate file. This makes it much easier to alter and change your SQL code.

Lab 5: Indexes, Constraints, Views - Wrap Up 🏁

1. [x] `Indexes`
2. [x] `Constraints`
3. [x] `Views`
4. [x] `Εργαστηριακές Ασκήσεις`
5. [x] `Εξαμηνιαία Εργασία`

> **Απορίες** https://discord.gg/9UvTeWNJzs🔗
> ⛔ 📧

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. Your manager wants a report that includes the name of every country, along with the total payments for all customers who live in each country. Create a view for the report.

```sql
CREATE VIEW country_payments AS
SELECT c.country, (SELECT sum(p.amount) FROM city ct
  INNER JOIN address a ON ct.city_id = a.city_id
  INNER JOIN customer cst ON a.address_id = cst.address_id
  INNER JOIN payment p ON cst.customer_id = p.customer_id
  WHERE ct.country_id = c.country_id
) tot_payments
FROM country c
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

2. Create a table 'suppliers' with 2 unique constraints. (One should be on name, address fields and the other on the phone field). Insert 2 suppliers with name and address as: ('ABC Inc', '4000 North 1st Street') and ('XYZ Inc', '4000 North 1st Street').

```sql
CREATE TABLE suppliers (
    supplier_id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL UNIQUE,
    address VARCHAR(255) NOT NULL,
    PRIMARY KEY (supplier_id),
    CONSTRAINT uc_name_address UNIQUE (name , address)
);
INSERT INTO suppliers(name, phone, address)
VALUES( 'ABC Inc',  '(408)-908-1111', '4000 North 1st Street');
INSERT INTO suppliers(name, phone, address)
VALUES( 'XYZ Inc',  '(805)-908-1111','4000 North 1st Street');
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

3. You are unhappy about the name, address constraint. Delete the constraint.

```
SHOW INDEX FROM suppliers;
DROP INDEX uc_name_address ON suppliers;
```

4. Your manager thinks a constraint is needed on the suppliers table name field. Write a query to achieve that.

```
ALTER TABLE suppliers
ADD CONSTRAINT uc_name
UNIQUE (name);
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

5. Create a view to locate the customers living in 'France'.

```
CREATE VIEW FRCustomers AS
SELECT c.customer_id, c.store_id, c.first_name,
c.last_name, c.address_id, c.active, c.create_date, ct.city
FROM customer c
INNER JOIN address a ON c.address_id = a.address_id
INNER JOIN city ct ON a.city_id = ct.city_id
INNER JOIN country cn ON ct.country_id = cn.country_id
where cn.country='France';
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

6. Create a view to list staff (id, name, phone city, country, store_id).

```
CREATE VIEW stafflist AS SELECT
        s.staff_id AS ID,
        CONCAT(s.first_name, ' ', s.last_name) AS name,
        a.phone AS phone,
        city.city AS city,
        country.country AS country,
        s.store_id AS stid
    FROM
        staff s
        JOIN address a ON s.address_id = a.address_id
        JOIN city ON a.city_id = city.city_id
        JOIN country ON city.country_id = country.country_id;
```