Lab7 Filtering, Sets, More Joins

Lab7 Agenda

- Filtering
- Sets
- More Joins
- Εργαστηριακές Ασκήσεις
- Εξαμηνιαία Εργασία

Filtering

- A where clause may contain one or more conditions
- Condition Types
  - Equality Conditions
    - ```
      SELECT c.email  FROM customer c
      INNER JOIN rental r  ON c.customer_id = r.customer_id
      WHERE date(r.rental_date) = '2005-06-14';
      ```
      ```

Condition Types

- Inequality conditions
  - ```
    SELECT c.email  FROM customer c
    INNER JOIN rental r  ON c.customer_id = r.customer_id
    WHERE date(r.rental_date) <> '2005-06-14';
    ```
- Range Conditions
  - ```
    SELECT c.email  FROM customer c
    INNER JOIN rental r  ON c.customer_id = r.customer_id
    WHERE date(r.rental_date) < '2005-06-14';
    ```

Condition Types

- Range Conditions - between operator
  - ```
    SELECT customer_id, payment_date, amount FROM payment
    WHERE amount BETWEEN 10.0 AND 11.99;
    ```
- with strings
  - ```
    SELECT last_name, first_name  FROM customer
    WHERE last_name BETWEEN 'FA' AND 'FR';
    ```

Condition Types

- Range Conditions - between operator
  - ```
    +------------+------------+
    | last_name  | first_name |
    +------------+------------+
    | FARNSWORTH | JOHN       |
    | FENNELL    | ALEXANDER  |
    | FERGUSON   | BERTHA     |
                ...
    ```
  - the order in which the characters within a character set are sorted is called a collation (e.g. case/ accent sensitivity)
    - ```
      SHOW COLLATION;
      ```

Condition Types

- Membership Conditions
  - ```
    SELECT title, rating FROM film
    WHERE rating IN ('G','PG');
    ```
  - ```
    +---------------------------+--------+
    | title                     | rating |
    +---------------------------+--------+
    | ACADEMY DINOSAUR          | PG     |
    | ACE GOLDFINGER            | G      |
    | AFFAIR PREJUDICE          | G      |
                ...
    ```

Condition Types

- Membership Conditions Using subqueries
  - ```
    SELECT title, rating FROM film
    WHERE rating IN (SELECT rating FROM film WHERE title LIKE '%PET%');
    ```
- Membership Conditions not in
  - ```
    SELECT title, rating FROM film
    WHERE rating NOT IN ('PG-13','R', 'NC-17');
    ```

Using wildcards

- Wildcard characters

Wildcard characters	Wildcard character Matches
_	Exactly one character
%	Zero or more characters

- Sample search expressions

Search expression	explanation
F%	Strings beginning with F
%t	Strings ending with t
%bas%	Strings containing the substring 'bas'
_ _ t_	Four-character strings with a t in the third position

Using wildcards

- find all customers whose last name begins with Q or Y

```
SELECT last_name, first_name FROM customer
WHERE last_name LIKE 'Q%' OR last_name LIKE 'Y%';
```

```
+-------------+------------+
| last_name   | first_name |
+-------------+------------+
| QUALLS      | STEPHEN    |
| QUIGLEY     | TROY       |
| YOUNG       | CYNTHIA    |
+-------------+------------+
```

Using regular expressions

- find all customers whose last name begins with Q or Y

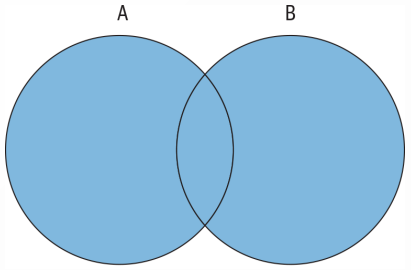
```
SELECT last_name, first_name FROM customer
WHERE last_name REGEXP '^[QY]';
```

```
+-------------+------------+
| last_name   | first_name |
+-------------+------------+
| QUALLS      | STEPHEN    |
| QUIGLEY     | TROY       |
| YOUNG       | CYNTHIA    |
+-------------+------------+
```

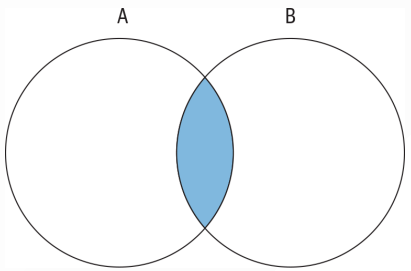
Sets

Sets Primer

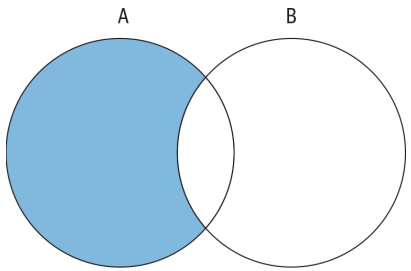
union $A \cup B$



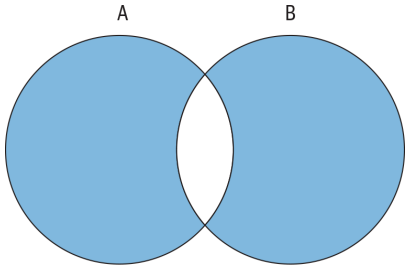
intersection $A \cap B$



Difference $A \setminus B$



Symmetric Difference $A \triangle B$



$$(A \cup B) \setminus (A \cap B)$$

union Operator

```
SELECT 'CUST' typ, c.first_name, c.last_name FROM customer c
UNION ALL
SELECT 'ACTR' typ, a.first_name, a.last_name FROM actor a;
```

```
+------+-------------+--------------+
| typ  | first_name  | last_name    |
+------+-------------+--------------+
| CUST | MARY        | SMITH        |
| CUST | PATRICIA    | JOHNSON      |
             ...
| ACTR | JULIA       | FAWCETT      |
| ACTR | THORA       | TEMPLE       |
+------+-------------+--------------+
```

union Operator

- union
  - sorts the combined set and removes duplicates
- union all
  - union all does not
- Select customers and actors whose name starts with 'J' and surname starts with 'D'

union Operator

```
SELECT c.first_name, c.last_name FROM customer c
WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
UNION
SELECT a.first_name, a.last_name FROM actor a
WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%';
```

```
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JENNIFER   | DAVIS     |
| JUDY       | DEAN      |
| JODIE      | DEGENERES |
| JULIANNE   | DENCH     |
+------------+-----------+
```

intersect Operator

```
SELECT c.first_name, c.last_name FROM customer c
WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
INTERSECT
SELECT a.first_name, a.last_name FROM actor a
WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%';
```

```
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JENNIFER   | DAVIS     |
+------------+-----------+
```

- Two sets are called **disjoint** if they have no elements in common
- intersect all operator: does not remove duplicates

except Operator

- returns the first result set minus any overlap with the second result set

```
SELECT a.first_name, a.last_name FROM actor a
WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%'
EXCEPT
SELECT c.first_name, c.last_name FROM customer c
WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%';
```

```
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JUDY       | DEAN      |
| JODIE      | DEGENERES |
| JULIANNE   | DENCH     |
+------------+-----------+
```

“ MySQL does not implement the except operator ”

except all operator

- except removes all occurrences of duplicate data from set A,
- except all removes only one occurrence of duplicate data from set A for every occurrence in set B

“ MySQL does not implement the except all operator ”

More Joins

Recall Inner Join

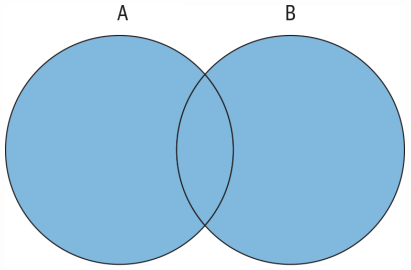
- show each customer's city

```
SELECT c.first_name, c.last_name, 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;
```

```
+-------------+--------------+----------------------------+
| first_name  | last_name    | city                       |
+-------------+--------------+----------------------------+
| MARY        | SMITH        | Sasebo                     |
| PATRICIA    | JOHNSON      | San Bernardino             |
```

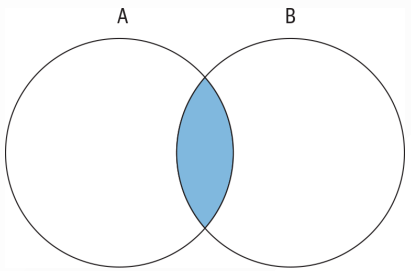
Recall: Sets

union $A \cup B$



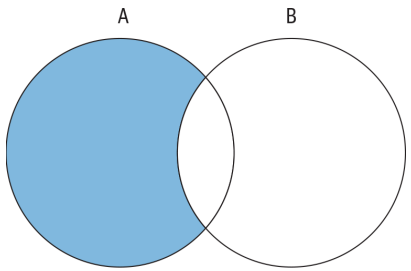
intersection

$A \cap B$



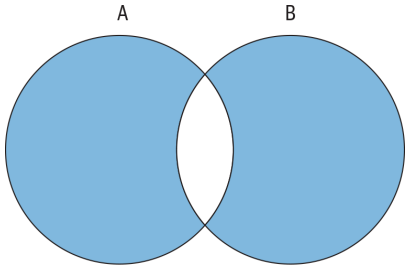
Difference

$A \setminus B$



Symmetric Difference

$A \triangle B$

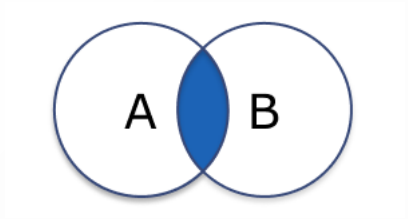


$(A \cup B) \setminus (A \cap B)$

Different Types of SQL JOINs

INNER JOIN

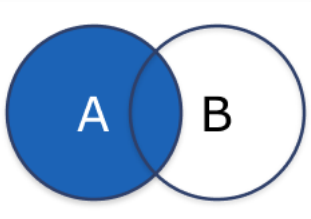
$A \cap B$




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

LEFT JOIN

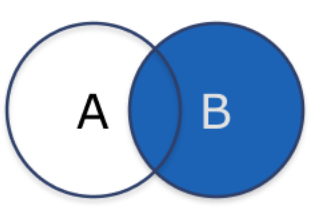
$A \cup (A \cap B)$




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

OUTER JOIN

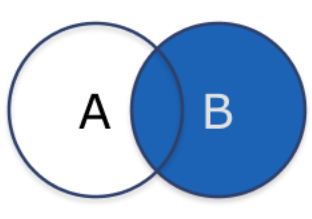
$B \cup (A \cap B)$




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

RIGHT JOIN

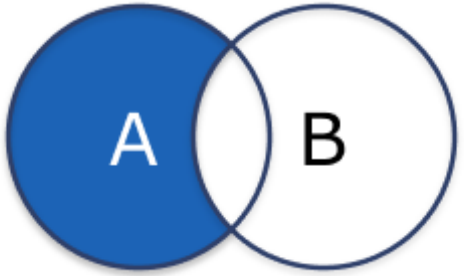
$B \cup (A \cap B)$




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

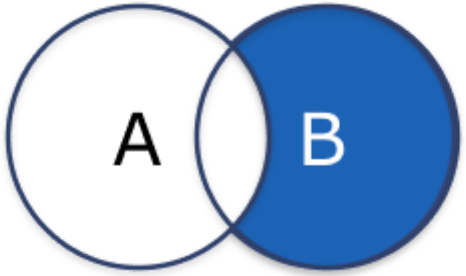
Different Types of SQL JOINs

Left Excluding JOIN $A \setminus B$



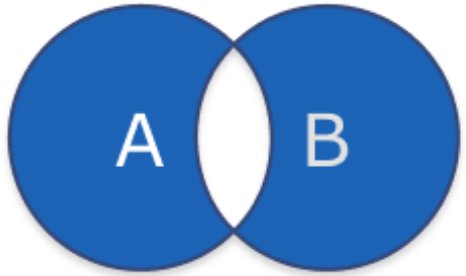

```
SELECT ...
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

Right Excluding JOIN $B \setminus A$



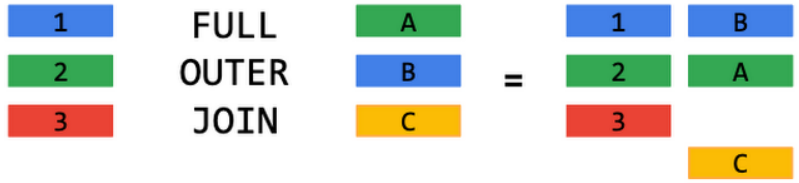

```
SELECT ...
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

Outer Excluding JOIN $(A \cup B) \setminus (A \cap B)$




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

join types

INNER JOIN		Only returns rows that meet the join condition
RIGHT OUTER JOIN		Returns all rows from the table on the right side of JOIN and matched rows from the left side of the JOIN
LEFT OUTER JOIN		Returns all rows from the table on the left side of JOIN and matched rows from the right side of the JOIN
FULL OUTER JOIN		Returns all rows from both sides even if join condition is not met
CROSS JOIN		Cartesian product between the two sides is a join but without a join condition. Returns all rows joined from both sides

Εργαστηριακές Ασκήσεις 💻

1. a. Select customer first_name, last_name and actor first_name, last_name columns from performing a **left join** between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

   b. Select customer first_name, last_name and actor first_name, last_name columns from performing a **right** join between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

   c. Select customer first_name, last_name and actor first_name, last_name columns from performing an **inner** join join between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

Εργαστηριακές Ασκήσεις 💻

2. Write a query that finds the first and last names of all actors and customers whose last name starts with 'L'.
3. List each film and the number of actors who are listed for that film.
4. How many copies of the film Hunchback Impossible exist in the inventory system?
5. List the total paid by each customer.
6. List all films categorized as family films.

Εξαμηνιαία Εργασία 💻

- Database Schema Design
  1. Start thinking about the entities you need
     - Identify entities, attributes and relationships from the problem description
     - identify cardinality ratios of the relationships found
  2. Design an E/R diagram for your database
     - Look for any issues that are apparent in the E/R diagram

Εξαμηνιαία Εργασία 💻

- Materialize Schema: DDL statements
  1. 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
  2. - Create views as needed
     - Create triggers for your tables
     - Create Stored Procedures & Functions for your application

Εξαμηνιαία Εργασία 💻

- Add Information to the Database: DML script
  - Populate the database with data
  - Write needed queries
  - Test and adapt offered functionality

“ hint: start running your SQL commands from a separate file. This makes it much easier to alter and change your SQL code ”

Wrap Up

1. [x] `Filtering`
2. [x] `Sets`
3. [x] `More Joins`
4. [x] `Εργαστηριακές Ασκήσεις`
5. [x] `Εξαμηνιαία Εργασία`

Wrap Up

😱 Απορίες https://discord.gg/g3fFxWVPfD

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. a. Select customer first_name, last_name and actor first_name, last_name columns from performing a **left join** between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

```
SELECT customer.first_name AS "customer_first_name",
        customer.last_name AS "customer_last_name",
        actor.first_name AS "actor_first_name",
        actor.last_name AS "actor_last_name"
FROM customer
LEFT JOIN actor ON
        customer.first_name = actor.first_name AND
        customer.last_name = actor.last_name;
```

```
| AUSTIN               | CINTRON              | NULL              | NULL             |
+----------------------+----------------------+-------------------+------------------+
599 rows in set (0.03 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. b. Select customer first_name, last_name and actor first_name, last_name columns from performing a **right** join between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

```
SELECT customer.first_name AS "customer_first_name",
        customer.last_name AS "customer_last_name",
        actor.first_name AS "actor_first_name",
        actor.last_name AS "actor_last_name"
FROM customer
RIGHT JOIN actor ON
        customer.first_name = actor.first_name AND
        customer.last_name = actor.last_name;
```

```
| NULL                 | NULL                 | THORA            | TEMPLE          |
+----------------------+----------------------+------------------+-----------------+
200 rows in set (0.00 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. c. Select customer first_name, last_name and actor first_name, last_name columns from performing an **inner** join join between the customer and actor column on the first_name/ last_name columns in each table. How may rows did you get ?

```
SELECT customer.first_name AS "customer_first_name",
        customer.last_name AS "customer_last_name",
        actor.first_name AS "actor_first_name",
        actor.last_name AS "actor_last_name"
FROM customer
INNER JOIN actor ON
        customer.first_name = actor.first_name AND
        customer.last_name = actor.last_name;
```

```
+---------------------+--------------------+------------------+-----------------+
| customer_first_name | customer_last_name | actor_first_name | actor_last_name |
+---------------------+--------------------+------------------+-----------------+
| JENNIFER            | DAVIS              | JENNIFER         | DAVIS           |
+---------------------+--------------------+------------------+-----------------+
1 row in set (0.00 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

2. Write a query that finds the first and last names of all actors and customers whose last name starts with 'L'.

```
SELECT first_name, last_name FROM actor  WHERE last_name LIKE 'L%'
UNION
SELECT first_name, last_name FROM customer WHERE last_name LIKE 'L%';
```

```
+------------+--------------+
| first_name | last_name    |
+------------+--------------+
| MATTHEW    | LEIGH        |
             ...
| LEWIS      | LYMAN        |
| JACKIE     | LYNCH        |
+------------+--------------+
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

3. List each film and the number of actors who are listed for that film.

```
select f.title as 'Film', count(fa.actor_id) as 'Number of Actors'
from film as f
join film_actor as fa
on f.film_id = fa.film_id
group by f.title;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

4. How many copies of the film Hunchback Impossible exist in the inventory system?

```
select f.title as Film, count(i.inventory_id) as 'Inventory Count'
from film as f join inventory as i
on f.film_id = i.film_id
where f.title = 'Hunchback Impossible'
group by f.film_id;
```

```
+----------------------+-----------------+
| Film                 | Inventory Count |
+----------------------+-----------------+
| HUNCHBACK IMPOSSIBLE |               6 |
+----------------------+-----------------+
1 row in set (0.00 sec)
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

5. List the total paid by each customer.

```
select concat(c.first_name,' ',c.last_name) as 'Customer Name',
sum(p.amount) as 'Total Paid'
from payment as p
join customer as c
on p.customer_id = c.customer_id
group by p.customer_id;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

6. List all films categorized as family films.

```
select f.title as 'Movie Title'
from film as f
join film_category as fc on fc.film_id = f.film_id
join category as c on c.category_id = fc.category_id
where c.name = 'Family';
```