Database Development and Design

Qinshift Academy

Database Development and Design

Developing and Design of databases using PostgreSQL - Powerful, open-source object-relational database

Agenda

- Session 2
  - Quiz
  - Homework discussion
  - Filtering and Sorting data
    - Workshop
  - Combining sets (UNION, UNION ALL, INTERSECT)
    - Workshop
  - Table constraints (Default, Check, Unique)
    - Workshop
  - Referential integrity (Foreign Keys)
    - Workshop
  - Join Types (Left, Right, Inner, Cross Join)
    - Workshop
  - Knowledge check (Discussion, Homework)

Fill database with example data

Filtering data - Workshop

- Filtering expressions are used to reduce number of rows returned based on some criteria

- WHERE statement

```
SELECT *
FROM TableName
WHERE ColumnName = value
```

- Example

```
SELECT *
FROM movies
WHERE title = 'Titanic'
```

Filtering data - Workshop

- Find all actors with last name 'DiCaprio'
- Find all directors whose first name starts with 'M'
- Find all actors from 'Australian' nationality
- Find all American directors who have directed movies with rating 'R'
- Find all actors with birth date after '1980-01-01'
- Find all movies with 'R' rating
- Find all movies where plot_summary contains the word "deadly"

Sorting data

- Ordering the results based on specific order
- ORDER BY statement

```
SELECT *
FROM TableName
ORDER BY ColumnName ASC\DESC
```

- Example:

```
SELECT *
FROM Employee
ORDER BY FirstName ASC
```

Sorting data - Workshop

- Find all genres sorted by name A-Z
- Find all albums who’s name starts with “A” sorted by rating
- Find all songs ending on “O” sorted by duration
- Find all married artists sorted by spouse name
- Find all artists from ‘Macedonia’ sorted by city name

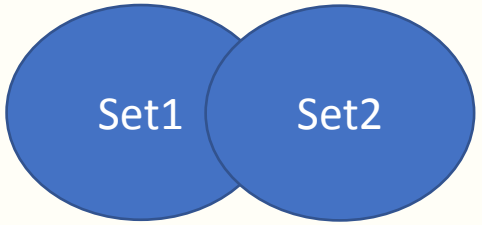
Combining sets

Union, Union ALL, Intersect

Combining sets - UNION

- UNION set operator unifies the results of the two input queries. As a set operator, UNION has an implied DISTINCT property, meaning that it does not return duplicate rows.
- Duplicates are eliminated
- Pre-requisites
  - All sets should have same number and order of columns
  - Columns should be of same or compatible data type
- Example:

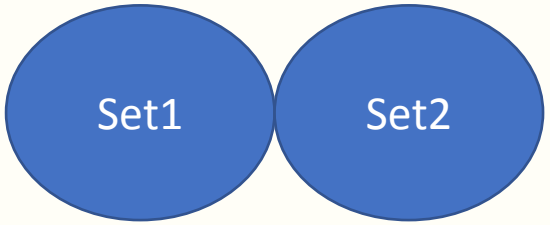
```
SELECT Region
FROM BusinessEntity
UNION
SELECT RegionName
FROM Customer
```



Combining sets – UNION ALL

- UNION ALL operator unifies the results of the two input queries. As a set operator, UNION ALL doesn't have an implied DISTINCT property, meaning that it can return duplicate rows.
- Duplicates remain
- Pre-requisites
  - All sets should have same number of columns
  - Columns should be of same or compatible data type
- Example:

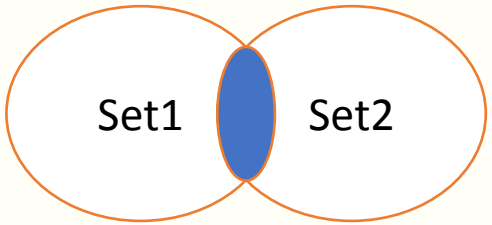
```
SELECT Region
FROM BusinessEntity
UNION ALL
SELECT RegionName
FROM Customer
```



Combining sets – INTERSECT

- INTERSECT operator returns only distinct rows that are common to both sets. In other words, if a row appears at least once in the first set and at least once in the second set, it will appear once in the result of the INTERSECT operator.
- Pre-requisites
  - All sets should have same number of columns
  - Columns should be of same or compatible data type
- Example:

```
SELECT FirstName, LastName
FROM Employee
WHERE FirstName = 'Aleksandar'
INTERSECT
SELECT FirstName, LastName
FROM Employee
WHERE LastName = 'Nikolovski'
```



Combining sets - Workshop

- Combine product names from both sales tables for 2023 and 2024 WITHOUT duplicates
- Combine product names from both sales tables for 2023 and 2024 WITH duplicates
- Get products names that were sold in both 2023 and 2024
- Get products names that were sold in 2023 but were not sold in 2024
- Combine product names from both sales tables for 2023 and 2024 WITHOUT duplicates whose amount was above 100$

Table constraints

NOT NULL, UNIQUE, CHECK, PRIMARY KEY, FOREIGN KEY

Table constraints – Not Null

- By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column. A NOT NULL constraint is always written as a column constraint.
- Example:

```
CREATE TABLE Customer (
Name TEXT NOT NULL,
Address TEXT NULL);
```

Table constraints – Unique

- The UNIQUE Constraint prevents two records from having identical values in a particular column.
- Example:

```
CREATE TABLE Customer (
Name TEXT,
Address TEXT,
PhoneNumber Text UNIQUE);
```

Table constraints – Check

- The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and is not entered into the table.
- Example:

```
CREATE TABLE Customer (
Name TEXT,
Address TEXT,
Age INTEGER CHECK(Age > 18));
```

Table constraints – Primary Key

- The PRIMARY KEY constraint uniquely identifies each record in a database table. There can be more UNIQUE columns, but only one primary key in a table. Primary keys are important when designing the database tables. Primary keys are unique ids.
- We use them to refer to table rows. Primary keys become foreign keys in other tables, when creating relations among tables. Due to a 'longstanding coding oversight', primary keys can be NULL in SQLite. This is not the case with other databases.
- A primary key is a field in a table, which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.
- A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a composite key.
- If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

```
CREATE TABLE Customer (
Id INTEGER PRIMARY KEY,
Name TEXT,
Address TEXT);
```

Table constraints – Foreign Key

- A foreign key constraint specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables. They are called foreign keys because the constraints are foreign; that is, outside the table. Foreign keys are sometimes called a referencing key.
- Example:

```
CREATE TABLE Company (
CompanyId INTEGER PRIMARY KEY,
Name TEXT,
Address TEXT);
```

```
CREATE TABLE Employee (
Name TEXT,
Address TEXT,
Age INTEGER,
CompanyId INTEGER references Company(CompanyId));
```

Table constraints - Workshop

- Product price must be a positive number
- Product must be in stock when added
- Discounted price must be lower than regular price
- The weight of the product must be more than 0 and less than 1 ton

Relations

Types of relations

- Types of relations between tables
  - One-to-Many Relationship (1:M) – Most common
    - A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A. For example, one Customer can have many Orders, but one order is only for 1 Customer.
  - Many-to-Many Relationships (M:M)
    - In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table - C, called a junction table, whose primary key consists of the foreign keys from both tables A and table B.

      For example the BusinessEntity table and the Customer table both have 1:M relation with Orders table which makes the M:M relation.
  - One-to-One Relationships (1:1)
    - In a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa. Example is the Customer table we have PhoneNumber column which can be placed in different table - CustomerPhone.

      Relations are also known as Foreign Key (FK) relations.

Relations - workshop

Add relations between:

- Student and Student details
- Departments and employees
- Courses and Students

Join types

Inner, Outer, Left, Right, Cross

Join types

- Often, data that you need to query is spread across multiple tables. The more normalized the environment is, the more tables you usually have.
- The tables are usually related through keys, such as a foreign key in one side and a primary key in the other. Then you can use joins to query the data from the different tables and match the rows that need to be related.
- The different types of joins that T-SQL supports:
  - Cross
  - Inner
  - Outer

Join types – Cross join

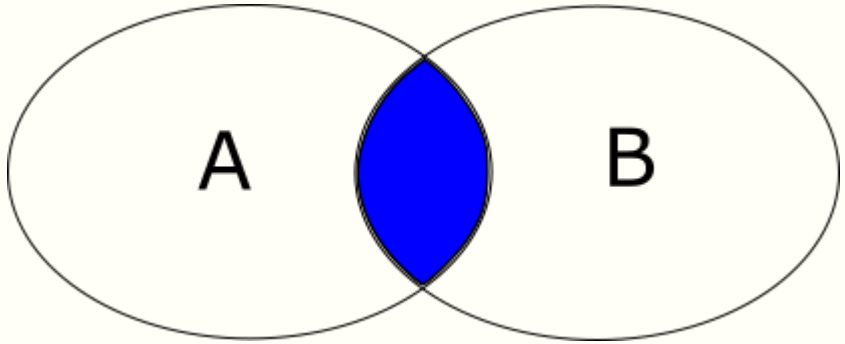
- A CROSS JOIN matches every row of the first table with every row of the second table. If the input tables have x and y columns, respectively, the resulting table will have x+y columns. Because CROSS JOINs have the potential to generate extremely large tables, care must be taken to use them only when appropriate.

```
SELECT * FROM Table1 CROSS JOIN Table2
```

Join types – Inner join

- A INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows, which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of table1 and table2 are combined into a result row.
- An INNER JOIN is the most common type of join and is the default type of join. You can use INNER keyword optionally.

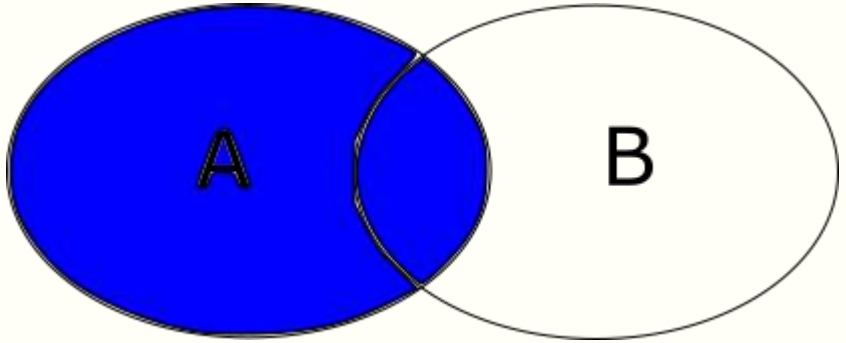
```
SELECT table1.column1, table2.column2 ...
FROM table1
INNER JOIN table2
ON table1.commonField = table2.commonField;
```



Join types – Outer join

- The OUTER JOIN is an extension of the INNER JOIN. SQL standard defines three types of OUTER JOINs: LEFT, RIGHT, and FULL and PostgreSQL supports all of these.
- In case of LEFT OUTER JOIN, an inner join is performed first. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. Thus, the joined table always has at least one row for each row in T1.
- Example:

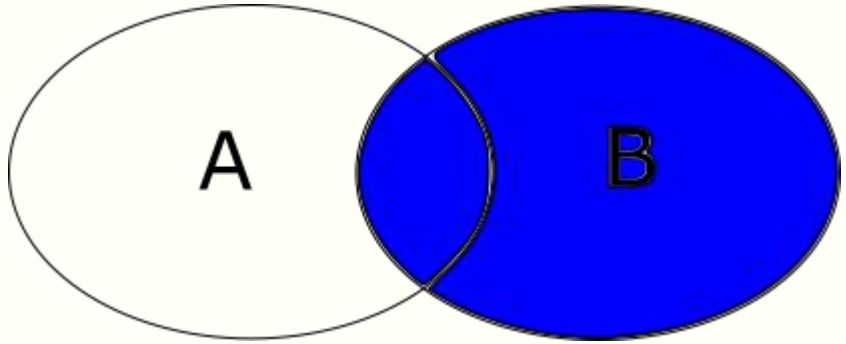
```
SELECT ...
FROM table1
LEFT OUTER JOIN table2
ON conditional_expression …
```



Join types – Right Outer Join

- First, an inner join is performed. Then, for each row in table T2 that does not satisfy the join condition with any row in table T1, a joined row is added with null values in columns of T1. This is the converse of a left join; the result table will always have a row for each row in T2.
- Example:

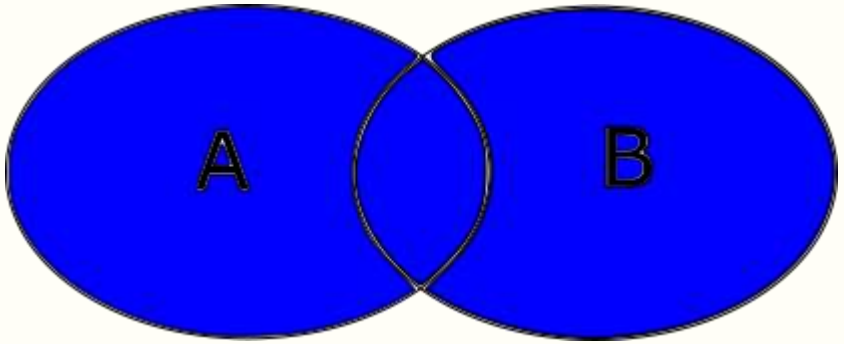
```
SELECT ...
FROM table1
RIGHT OUTER JOIN table2
ON conditional_expression ...
```



Join types – Full Outer Join

- First, an inner join is performed. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. In addition, for each row of T2 that does not satisfy the join condition with any row in T1, a joined row with null values in the columns of T1 is added.
- Example:

```
SELECT ...
FROM table1
FULL OUTER JOIN table2
ON conditional_expression ...
```



Join types – Workshop (dependent on script being executed)

Let's analyze the movies database and the table structure. Main focus should the relations. Database diagrams can help us visualize the database structure.

Then, lets create the following queries:

- Show all movies and their directors
- Show all movies and their genres
- Show all movies and their actors
- Show all movies and their reviews
- Show all movies and their production companies
- Show all movies and their filming locations
- Show all actors and their awards
- Show all movies and their revenues
- Show all users and their reviews
- Show all movies and their user watchlist counts

Homework 2

Homework – pre-requisite

A script is provided for inserting dummy data in already created music database

Homework requirement 1/3

- Find all movies released in 2019
- Find all actors from 'British' nationality
- Find all movies with 'PG-13' rating
- Find all directors from 'American' nationality
- Find all movies with duration more than 150 minutes
- Find all actors with last name 'Pitt'
- Find all movies with budget greater than 100 million
- Find all reviews with rating 5
- Find all movies in English language
- Find all production companies from 'California'

Homework requirement 2/3

- Show movies and their directors
- Show actors and their movies
- Show movies and their genres
- Show users and their reviews
- Show movies and their locations
- Show movies and their production companies
- Show actors and their awards
- Show movies and their awards
- Show users and their watchlist movies
- Show movies and their revenues

Homework requirement 3/3

- Show all R-rated movies and their directors
- Show all movies from 2019 and their genres
- Show all American actors and their movies
- Show all movies with budget over 100M and their production companies
- Show all movies filmed in 'London' and their directors
- Show all horror movies and their actors
- Show all movies with reviews rated 5 and their reviewers
- Show all British directors and their movies
- Show all movies longer than 180 minutes and their genres
- Show all Oscar-winning movies and their directors

Questions?

Trainer Name

Trainer mail

Assistant Name

Assistant mail

Qinshift Academy