Overview

Welcome to this comprehensive overview of PostgreSQL and how to use it with pgAdmin. Discover the amazing features and learn how to set up and manage databases with ease. we will explore the importance of PostgreSQL, dive into its key features, and learn how to perform basic SQL operations using pgAdmin.

What is PostgreSQL?

PostgreSQL is an advanced open-source relational database management system known for its stability, scalability, and extensive features. It provides a robust foundation for data-driven applications.

Features of PostgreSQL

Relational Database

PostgreSQL offers full support for tables, relationships, and SQL queries, making it ideal for managing structured data.

Extensibility

With support for custom data types, functions, and extensions, PostgreSQL allows developers to tailor the database to their specific needs.

Data Integrity

PostgreSQL ensures data integrity with features such as constraints, triggers, and transactions, ensuring reliable and accurate data.

How to install & set up?

1. Download the PostgreSQL installer from the official website.
2. Run the installer and follow the on-screen instructions to complete the installation.
3. Configure the installation settings, including the database superuser password and port number.
4. Start the PostgreSQL service and verify the successful installation.

Ref. Link

https://www.postgresql.org/download/

https://www.postgresqltutorial.com/postgresql-getting-started/install-postgresql/

Introduction to pgAdmin

pgAdmin is a popular administration and development platform for Postgresql. It provides a user-friendly interface to manage databases, execute queries, and perform various administrative tasks.

Getting Started with pgAdmin

1

Installation and Setup

Install pgAdmin on your local machine and configure the necessary connection details to connect to your Postgresql database.

2

User Interface Overview

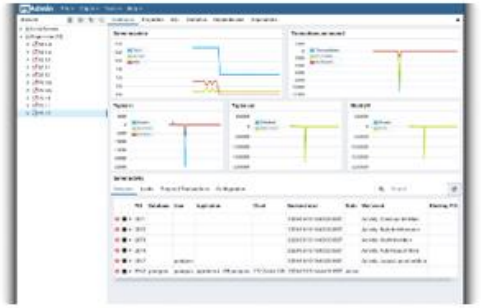
Familiarize yourself with the pgAdmin user interface, including the navigation pane, object browser, query tool, and dashboard.

3

Connecting to a Database

Learn how to establish a connection to a Postgresql database using pgAdmin and explore its contents.

Navigating pgAdmin



Dashboard

Get an overview of the database server status and recent activities.

Object Browser

Browse and manage database objects like tables, views, and functions.

Query Tool

Execute SQL queries, view results, and analyze data.

Working with Postgresql and pgAdmin

1

Creating and Managing Databases

Discover how to create and manage databases in Postgresql using pgAdmin. Learn about database options, schemas, and permissions.

2

Performing Basic SQL Operations

Master essential SQL operations such as querying tables, inserting data, updating records, and deleting data in Postgresql.

3

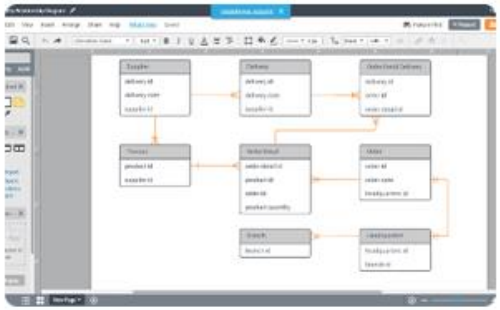
Managing Database Objects (Tables, Views, Indexes)

Explore the creation, alteration, and deletion of database objects like tables, views, and indexes using pgAdmin.

Hands-on demo with pgAdmin

Step-by-step walkthrough

Follow along as we demonstrate how to perform basic SQL operations using pgAdmin, giving you the confidence to navigate and interact with your databases.



Schema design best practices

Learn about effective schema design principles and how to organize your database tables, relationships, and constraints efficiently.

Data analysis and visualization

See how Postgresql, in conjunction with pgAdmin's data analysis and visualization features, can help reveal valuable insights within your datasets.

How to Create Database

How to use the PostgreSQL CREATE DATABASE statement to create new databases in the PostgreSQL database server.

Using Query

```
CREATE DATABASE database_name
```

Using pgAdmin

The pgAdmin tool provides you with an intuitive interface for creating a new database. First, log in to the PostgreSQL database server using pgAdmin.

Second, right-click the Databases node and select Create > Database… menu item

PostgreSQL Data Types

PostgreSQL data types including Boolean, character, numeric, array, json, uuid, and special types.

- **Boolean** for storing true and false value
- **Character** types such as char, varchar, and text.
- **Numeric** types such as integer and floating-point number.
- **UUID** for storing Universally Unique Identifiers
- **Array** for storing array strings, numbers, etc.
- **JSON** stores JSON data

TatvaSoft

Understanding PostgreSQL Constraints

- **Primary key** – How to define a primary key when creating a table or adding a primary key to an existing table.
- **Foreign key** – show you how to define foreign key constraints when creating a new table or add foreign key constraints for existing tables.
- **UNIQUE constraint** – make sure that values in a column or a group of columns are unique across the table.
- **NOT NULL constraint** – ensure values in a column are not NULL.
- **CHECK constraint** – add logic to check value based on a Boolean expression.

How to Create Table

A relational database consists of multiple related tables. A table consists of rows and columns. Tables allow you to store structured data like customers, products, employees, etc. To create a new table, you use the CREATE TABLE statement. The following illustrates the basic syntax of the CREATE TABLE statement:

```
CREATE TABLE [IF NOT EXISTS] table_name (
    column1 datatype(length) column_contraint,
    column2 datatype(length) column_contraint,
    column3 datatype(length) column_contraint,
    table_constraints
);
```

Performing Basic Operations with PostgreSQL

1

INSERT statement

Learn how to insert data into Postgresql tables, exploring different scenarios and best practices for data insertion.

2

UPDATE statement

Discover the power of updating data in Postgresql, exploring various techniques and considerations for modifying existing records.

3

DELETE statement

Unlock the ability to remove unwanted data from your Postgresql database while understanding the impact and potential pitfalls of deletion.

4

SELECT statement

Discover the art of querying data in Postgresql, leveraging advanced filtering, sorting, and aggregation techniques to extract meaningful insights.

Querying Data

1. **Select**
2. **Column aliases** – Assign temporary names to columns or expressions in a query. (select lastname **AS** surname)
3. **Order By** – Sort the result set returned from a query.
4. **Select Distinct** – Removes duplicate rows in the result set.

Filtering Data

1. **Where** – filter rows based on a specified condition.
2. **Limit** – get a subset of rows generated by a query.
3. **Fetch** – limit the number of rows returned by a query.
4. **In** – select data that matches any value in a list of values.
5. **Between** – select data that is a range of values.
6. **Like** – filter data based on pattern matching.
7. **Is Null** – check if a value is null or not.

Joins

SELECT * FROM a
INNER JOIN b ON a.key = b.key

SELECT * FROM a
LEFT JOIN b ON a.key = b.key

SELECT * FROM a
RIGHT JOIN b ON a.key = b.key

POSTGRESQL
JOINS

SELECT * FROM a
LEFT JOIN b ON a.key = b.key
WHERE b.key IS NULL

SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
WHERE a.key IS NULL

SELECT * FROM a
FULL JOIN b ON a.key = b.key

SELECT * FROM a
FULL JOIN b ON a.key = b.key
WHERE a.key IS NULL OR b.key IS NULL

Grouping Data

1. **Group By** – divide rows into groups and apply an aggregate function on each.

2. **Having** – apply conditions to groups.

```
SELECT
    column_1,
    column_2,
    aggregate_function(column_3)
FROM
    table_name
GROUP BY
    column_1,
    column_2,
```

```
SELECT
    column1,
    aggregate_function(column2)
FROM
    table_name
GROUP BY
    column1
HAVING
    condition;
```

Sub Query

1. **Subquery** – write a query nested inside another query.
2. **ANY** – retrieve data by comparing a value with a set of values returned by a subquery.
3. **ALL** – query data by comparing a value with a list of values returned by a subquery.
4. **EXISTS** – check for the existence of rows returned by a subquery

THANK YOU

ANY QUESTIONS ?