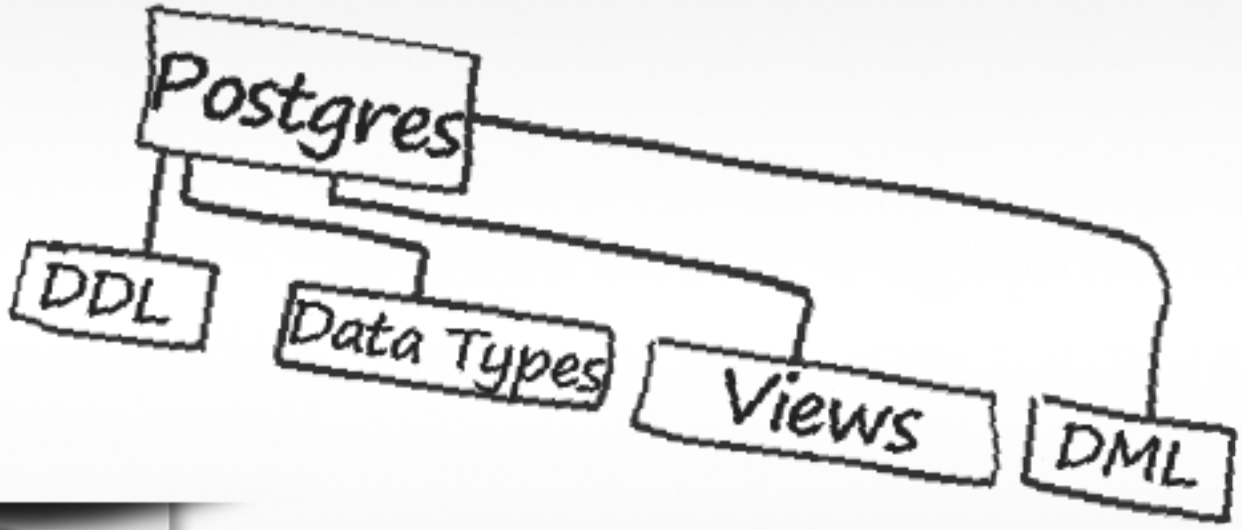
PostgreSQL

NOW



OPEN SOURCE DEPARTMENT

Information Technology Institute

Day 1 Contents

- Why PostgreSQL?
- History of PostgreSQL.
- ORDBMS concepts and terminology
- PostgreSQL application.
- Columns Data Types
- DML (Insert / update / delete /Truncate)

Why PostgreSQL?

- Designed for high volume environments.
- Cross platform
- Low / No Cost.
- Stability
- Open Source

https://en.wikipedia.org/wiki/PostgreSQL#Prominent_users

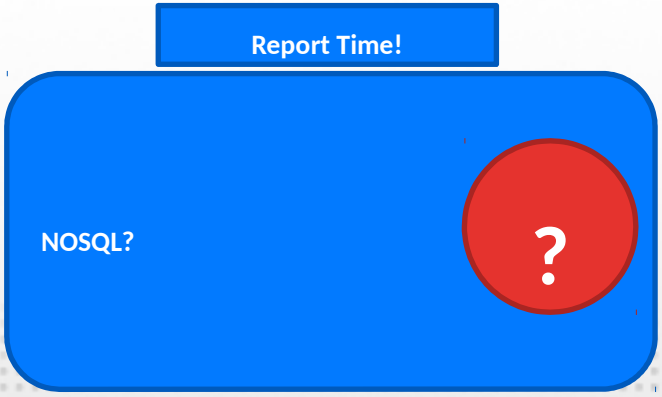
http://www.postgresql.org/about/users/

http://www.postgresql.org/about/sponsors/

https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

History of PostgreSQL

- PostgreSQL is derived from the POSTGRES package written at the University of California at Berkeley by a computer science professor named Michael Stonebraker
- POSTGRES used PostQUEL as query language



https://en.wikipedia.org/wiki/Michael_Stonebraker

History of PostgreSQL

- In 1994, Berkeley graduate students Andrew Yu and Jolly Chen replaced the PostQUEL query language interpreter with one for the SQL query language, creating Postgres95.
- We chose a new name, PostgreSQL, to reflect the relationship between the original POSTGRES and the more recent versions with SQL capability.

Relational database concepts

- Tables
  - A Collection of related data. The table has a name; a number of columns and a number of rows, a table in a database looks like a simple spreadsheet.
- Columns
  - Each column in the table has a unique name and contains different data. Additionally, each column has an **associated data type** as an integer, strings or Timestamp and so on . Columns are sometimes called fields or attributes.

Relational database concepts

- Rows
  - are a group of related data. Because of the tabular format, each row has the same attributes. Rows are also called records or tuples.
- Values
  - Each row consists of a set of individual values that correspond to columns. Each value must have the data type specified by its column.

Relational database concepts

- Primary Key
  - A primary key is unique. A key value can not occur twice in one table. With a key, you can find at most one row.
- Foreign Key
  - A foreign key is the linking pin between two tables.
- Referential Integrity
  - Referential Integrity makes sure that a foreign key value always points to an existing row.

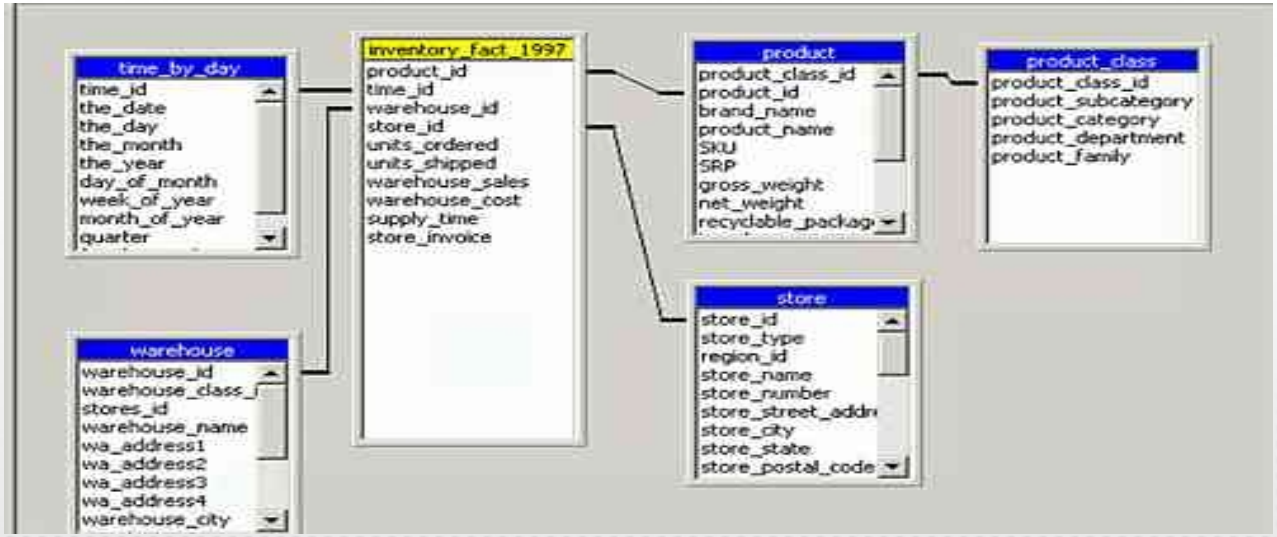
Relational database concepts

- Index
  - it is a data structure that improves the speed of data retrieval operations on a database table
  - Disadvantages: Storage Size & Insertion Time

Relational database concepts

- Schemas

— It is akin to a blueprint for the database. A schema should show the tables along with their columns, and the primary key of each table and any foreign keys. A schema does not include any data.



Relational database

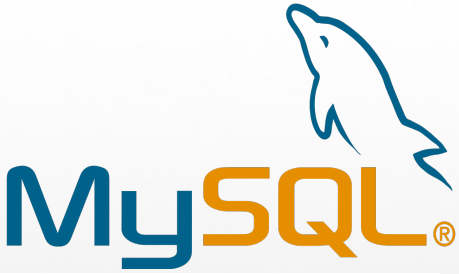
- RDBMS software:

—Enables you to implement a database with tables, columns and indexes.

—Guarantees the Referential Integrity between rows of various tables.

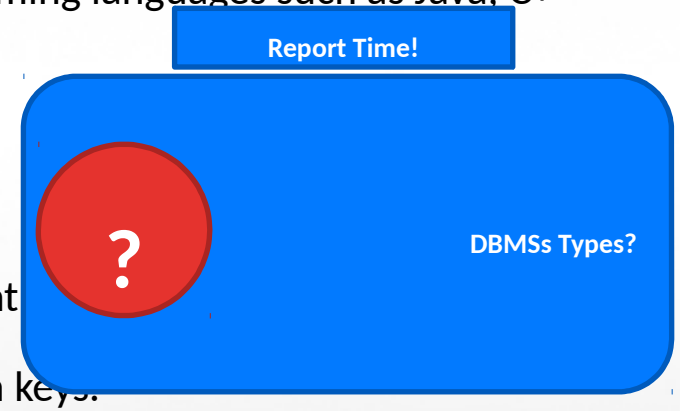
—Interprets an SQL query and combines information from various tables.

—C/C++, Java Interface



ORDBMS

- O for Object:
  - The basic goal for the Object-relational database is to bridge the gap between relational databases and the object-oriented modeling used in programming languages such as Java, C++.
- R for Relational:
  - It's called relational because all the data is stored into different established using primary keys or other keys known as foreign keys.



ERD

Entity Relationship Diagram

ERD Symbols and Notations

- Entity
  - An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using *singular* nouns.

ERD Symbols and Notations

- Attribute
  - An attribute is a property or characteristic of an entity, relationship. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item.
- Multivalued Attribute
  - If an attribute can have more than one value it is called an multivalued attribute. For example a person entity can have multiple hobbies values.

ERD Symbols and Notations

- Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams. Such as calculations

SUCH AS call duration, age.

- Relationship

A relationship describes how entities interact.

Example: Student, address, track, staff, courses, desk

Designing Your Database

- Simple tables

  —that describe a real-world object. They might also contain keys to other simple objects with which they have a one-to-one or one-to-many relation-ship.

- Linking tables

  —that describe a many-to-many relationship between two real objects.

Installing PostgreSQL

- For CentOS/Red Hat Distros:

```
# yum install postgresql postgresql-server pgadmin3
# service postgresql initdb
# chkconfig postgresql on
# service postgresql start
```

- For Ubuntu/Debian :

```
$ sudo apt-get install postgresql postgresql-client pgadmin3
```

Postgres Files

- All the data needed for a database is stored within the data directory, commonly referred to as PGDATA .A common location for PGDATA is /var/lib/pgsql/data.

Item	Description
PG_VERSION	A file containing the major version number of PostgreSQL
base	Subdirectory containing per-database subdirectories
global	Subdirectory containing cluster-wide tables, such as pg_database
pg_xlog	Subdirectory containing Log files
postmaster.opts	A file recording the command-line options the server was last started with
postmaster.pid	A lock file recording the current server PID and shared memory segment ID (not present after server shutdown)

Psql command

- You can list available databases using \l, as follows:

```
postgres-# \l
```

- to connect/select a desired database, use \c

```
postgres=# \c testdb;
```

- You are now logged into PostgreSQL testdb and ready to execute your commands inside testdb.
- To exit from the database, you can use \q.

Create Database

Using CREATE DATABASE using SQL statement

- The basic syntax of CREATE DATABASE statement is as follows:

```
CREATE DATABASE dbname;
```

- Following is a simple example,

```
postgres=# CREATE DATABASE testdb;
```

Create Database

- Using `createdb` Command

- The syntax for *createdb* is as shown below:

```
createdb [option...] [dbname [description]]
```

Parameter	Description
dbname	The name of a database to create.
description	Specifies a comment to be associated with the newly created database.
options	command-line arguments, which createdb accepts.

Create Database

Options

Option	Description
-e	Shows the commands being sent to the server.
-V	Print the app version and exit.
--help	Show help about dropdb command-line arguments, and exit.
-h host	Specifies the host name of the machine on which the server is running.
-p port	Specifies the TCP port on which the server is listening for connections.
-U username	User name to connect as.

Create Database

- As Example:

```
createdb -h localhost -p 5432 -U postgress testdb
password ******
```

- Above command will prompt you for password of the PostgreSQL admin user which is **postgres** by default so provide password and proceed to create your new dataabse.

Create Database

- CREATE DATABASE actually works by copying an existing database. By default, it copies the standard system database named **template1**.
- If you add objects to template1, these objects will be copied into subsequently created user databases.

Create Database

- There is a second standard system database named template0. This database contains the same data as the initial contents of template1, that is, only the standard objects predefined by your version of PostgreSQL.
- To create database using template0:

```
CREATE DATABASE dbname TEMPLATE template0;
```

Drop Database

Using Drop DATABASE using SQL statement

- The basic syntax of DROP DATABASE statement is as follows:

```
DROP DATABASE [ IF EXISTS ] dbname;
```

- Following is a simple example,

```
postgres=# DROP DATABASE testdb;
```

Drop Database

- We cannot drop a database that has any open connections, including our own connection from *psql* or *pgAdmin III*.
- We must switch to another database or *template1* if we want to delete the database we are currently connected to.

Drop Database

Using `dropdb` Command

The syntax for *dropdb* is as shown below:

- `dropdb [option...] dbname`

Parameter	Description
dbname	The name of a database to be deleted.
option	command-line arguments, which dropdb accepts.

Drop Database

Options

Option	Description
-e	Shows the commands being sent to the server.
-i	Issues a verification prompt before doing anything destructive.
-V	Print the dropdb version and exit.
--if-exists	Do not throw an error if the database does not exist.
--help	Show help about dropdb command-line arguments, and exit.
-h host	Specifies the host name of the machine on which the server is running.
-p port	Specifies the TCP port on which the server is listening for connections.
-U username	User name to connect as.

Drop Database

- As Example:

```
dropdb -h localhost -p 5432 -U postgress testdb
Password for user postgress: ****
```

- The above command drops database testdb.

Create Table

- Basic syntax of CREATE TABLE statement is as follows:

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

Create Table

- Following is an example:

```
CREATE TABLE COMPANY(
   ID              INT PRIMARY KEY,
   NAME            TEXT,
   AGE             INT,
   ADDRESS         CHAR(50),
   SALARY          INT
);
```

Column Data Type

- PostgreSQL has a rich set of native data types available to users.
- Users can add new types to PostgreSQL using the CREATE TYPE command.

Numeric data type

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
numeric(precision, scale)	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point, number 23.5141 has a precision of 6 and a scale of 4.
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Monetary data type

- The money type stores a currency amount with a fixed fractional precision.
- Input is accepted in a variety of formats, as integer and floating-point literals, as well as typical currency formatting, such as '$1,000.00'

Name	Storage Size	Description	Range
money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

Character data types

- An attempt to store a longer string will result in an error, unless the excess characters are all spaces, in this case string will be truncated to the maximum
- If the string is shorter than the declared length, values of type character will be space-padded; values of type character varying will simply store the shorter string.

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Date/Time data type

- Valid input for the time stamp types consists of the concatenation of a date and a time, followed by an optional time zone, followed by an optional AD or BC.

Name	Storage Size	Description	Examples
timestamp [without time zone]	8 bytes	both date and time (no time zone), From 4713 BC to 294276 AD	1999-01-08 04:05:06 January 8 04:05:06 99 BC
timestamp with time zone	8 bytes	both date and time, with time zone, From 4713 BC to 294276 AD	1999-01-08 04:05:06 -8:00 January 8 04:05:06 1999 PST

Date/Time data type

Name	Description
date	date (no time of day)
time [without time zone]	time of day (no date)
time with time zone	times of day only, with time zone
interval [fields]	time interval, field can be YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

Boolean data type

- Valid literal values for the "true" state are:

TRUE , 't‘ , 'true‘, 'y‘, 'yes‘, 'on‘ or '1‘

- For the "false" state, the following values can be used:

FALSE, 'f‘, 'false‘, 'n‘, 'no‘, 'off‘ or '0'

Name	Storage Size	Description
boolean	1 byte	state of true or false

Enumerated Types

- Enumerated (enum) types are data types that comprise a static, ordered set of values.
- Enum types are created using the CREATE TYPE command, for example:

```
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');

SELECT * FROM person WHERE current_mood > 'ok';
```