PostGresSQL Developers

Module 1

A Brief history of PostGresSQL

A brief history of PostGresSQL

- The INGRES project created in 1977
- Project was run by Professor Michael Stonebraker
- The POSTGRES project created in 1986 as an open source project

Key Features

- Runs on all major operating systems
- Completely ANSI SQL compliant
- PostGres 9 (current release) is a major upgrade
- Now supports streaming replication and hot standby

Module 2

PostgresSQL data types

PostGRES SQL data types

- Supports CHAR/VARCHAR
- INT/SMALLINT/BIGINT
- DECIMAL/NUMERIC
- REAL
- SERIAL/BIGSERIAL
- Many others

Which data type to use (INTEGER)

- Use SMALLINT when space is at a premium
- BIGINT has a memory and performance penalty
- USE INT for most things.

Which datatype to use (NUMERIC)

- Numericprovides scale and precision
- Scale is the number of digits to the right of the decimal point
- Precision is the total number of digits.
- You should make sure that the precision and scale are set to be large enough to handle future use cases.
- Use numeric for currency. Avoid the built-in ‘MONEY’ type.

Which datatype to use (FLOAT)

- Postgres uses IEEE 754 standard for representing floating point numbers
- Note that unexpected behavior (i.e. overflow,underflow, equality imprecision) is possible!
- Using numeric is usually the better choice.

Which datatype to use (BOOLEAN)

- Standard boolean data type. All of the following values are supported
- ‘True’, ‘False’
- ‘T’,’F’
- 0,1
- ‘Y’,’N’
- ‘Yes’,’No’
- ‘On’,’Off’

Which datatype to use (SERIAL)

- The SERIAL datatype is really a convenience.
- Used for auto-increment keys.

Which datatype to use (DATETIME)

- *Datetime includes the following*
  - *Date ('YYYY-MM-DD')*
  - *Timestamp without timezone ('YYYY-MM-DD HH:MM:SS')*
  - *Timestamp with timezone ('YYYY-MM-#DD HH:MM:SS:TZ')*
  - *Time ('HH:MM:SS')*
  - *Time without timezone ('HH:MM:SS')*
  - *Time with timezone ('HH:MM:SS:TZ')*

  - *Format can be adjusted using the following:*
  - *SET <datestyle>*
  - *Modify postgressql.conf : DateStyle parameter*
  - *Set an environment variable PGDateStyle*

Module 3

PostgresSQL operators

PostgresSQL operators

- Operators in PostgresSQL are reserved words and symbols used in a PostgresSQL statements WHERE clause to perform operations such as comparisons and arithmetic operations.
- Operators are used to specify conditions in a SQL statement and to serve as conjunctions for multiple conditions in a statement.
- The operators are grouped into the following categories
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Bitwise operators

Arithmetic Operators

- All standard operators are supported.
- +, -. *. / % ^
- Also many others.

Postgres comparison operators

- All standard comparison operators supported
- = (!= and <>)
- >, >=, <, <=

PostgresSQL logical operators

- All standard logical operators supported
- AND, OR, NOT

PostgresSQL bitwise operators

- PostgresSQL supports bitwise operations
- &. |, ~
- Bit shifting (>>, <<)

Module 4

Postgres Type Conversion

PostgresSQL type conversion

- Postgres allows us to convert from one data type to another on the fly.
- The CAST function can allow us to do this
- Note that using the CAST function on uncastable conversions (For example attempting to convert ‘abc’ to a number) will raise an error.
- Also raising an error will be an attempt to cast to an unknown type.
- Use the ‘::’ notation as a shorthand for CAST.

Module 5

PostgresSQL Indices

PostgresSQL indices

- Usually implemented as B+ trees.
- The advantages of a B+ tree are that they significantly speed up SELECT queries on tables
- They suffer a performance penalty when users request INSERT, UPDATE or DELETE operations on the table

PostgresSQL indices

- Many different types of indices.

- Single Index
- Multicolumn Index
- Unique Index
- Partial Index
- Implicit Index

PostgresSQL indices

- A single index sorts the table on a single column.
- A multicolumn index is defined on more than one column of a table.
- Whether to create a single-column index or a multicolumn index, take into consideration the column(s) that you may use very frequently in a query's WHERE clause as filter conditions.
- Should there be only one column used, a single-column index should be the choice. Should there be two or more columns that are frequently used in the WHERE clause as filters, the multicolumn index would be the best choice

Other types of indices

- Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table.
- A partial index is an index built over a subset of a table; the subset is defined by a conditional expression (called the predicate of the partial index). The index contains entries only for those table rows that satisfy the predicate.
- Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

Module 6

PostgresSQL Views

PostgresSQL Views

- Views are pseudo-tables

- Views allow us to:
  - Structure data in a way that users or classes of users find natural or intuitive.
  - Restrict access to the data such that a user can only see limited data instead of complete table.
  - Summarize data from various tables, which can be used to generate reports.

Module 7

PostgresSQL Constraints

PostgresSQL constraints

- Constraints are rules enforced on the database.
- Used, for example, to prevent invalid data from being entered.
- Constraints can be defined at the table or column level
- Many types of constraints available, including:
  - NOT NULL Constraint – Ensures that a column cannot have NULL value.
  - UNIQUE Constraint – Ensures that all values in a column are different.
  - PRIMARY Key – Uniquely identifies each row/record in a database table.
  - FOREIGN Key – Constrains data based on columns in other tables.
  - CHECK Constraint – The CHECK constraint ensures that all values in a column satisfy certain conditions.
  - EXCLUSION Constraint – The EXCLUDE constraint ensures that if any two rows are compared on the specified column(s) or expression(s) using the specified operator(s), not all of these comparisons will return TRUE

PostgresSQL NULL constraint

- 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.
- A NULL is not the same as no data; rather, it represents unknown data.

Other PostgresSQL Constraints

- The UNIQUE Constraint prevents two records from having identical values in a particular column
- The PRIMARY KEY constraint uniquely identifies each record in a database table. There only one primary key in a table. Primary keys are unique ids.
- 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. They are called foreign keys because the constraints are foreign; that is, outside the table.

Other PostgresSQL constraints

- 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.

Module 8

PostgresSQL functions

PostgresSQL functions

- PostgresSQL stored procedures are known as ‘functions’.
- Functions can be written in many languages, including Python, PHP, the Postgres native language (PLPGSQL) , and others.

PostgresSQL built in functions

- Postgres supplies many useful builtin functions

- Aggregate Functions
  - Count
  - Sum
  - Max
  - Min
  - Array_AGG
  - AVG (Average)

PostgresSQL Triggers

- Postgres triggers are special callback functions.
- Function is called after a defined event is executed.
- Triggers can be specified to fire before, or after an event.
- Also can be specified to fire instead of an operation.

Module 9

Accessing PostgresSQL from Python

Accessing PostgresSQL from Python

- We use the psycopg2 library to perform low-level access to Postgres
- We use SQLAlchemy to do higher level Object Relational Mapping (ORM).
- We connect to a database using a connection string
- We get a cursor variable (client or server-side)
- We create and execute SQL statements.
- We can do commits and rollbacks.