Introduction to SQL (Structured Query Language)

Database Design

Department of Computer Engineering

Sharif University of Technology

Maryam Ramezani maryam.ramezani@sharif.edu

Introduction to SQL commands

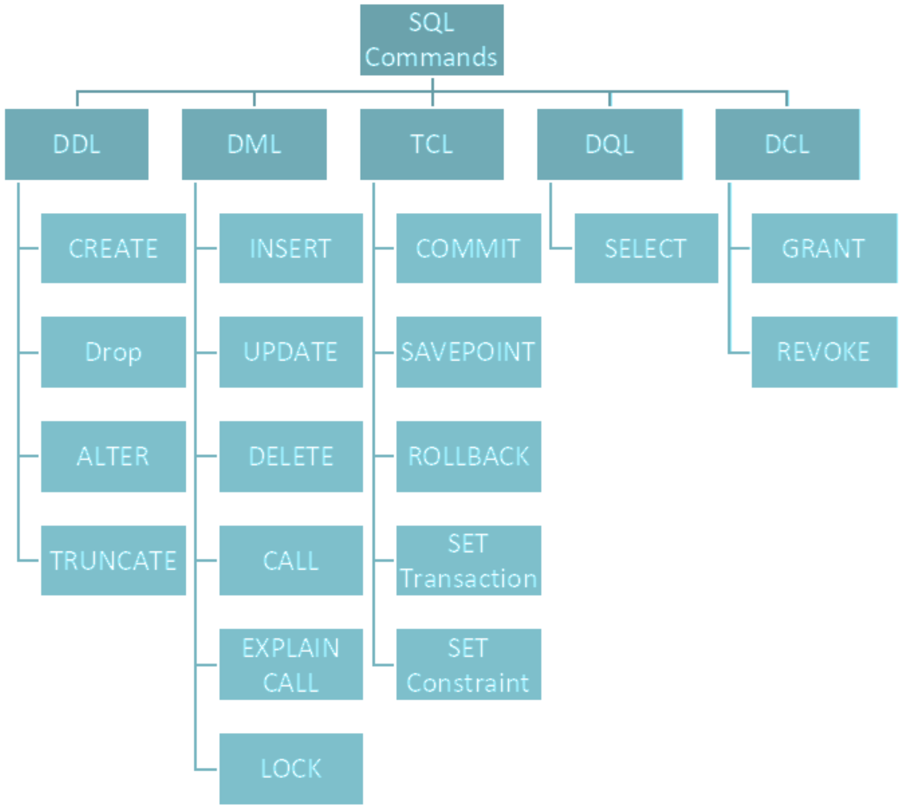
Database Languages

- Database Languages are known as data Sublanguages.
- DBMSs have a facility for embedding the sublanguage in a high-level programming language e.g. C, C++, Java Or VB. The high-level language is then known as a host Language.
- Most data sublanguages also provide interactive commands that can be input directly from a terminal.

SQL Commands

- A data sublanguage consists of five parts:
  - Data Definition Language (DDL)
    - Used to specify the database schema.
  - Data Manipulation Language (DML)
    - Used to read and update the database.
  - Data Query Language (DQL)
    - Used for performing queries on the data within schema objects.
  - Data Control Language (DCL)
    - DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.
  - Transaction Control Language (TCL)
    - Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group are successfully completed. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure.

SQL Commands



Data Definition Language (DDL)

- This is a language that allows the DBA or user to describe and name the entities, attributes and relationships required for the application, together with any associated integrity and security constraints (Begg & Connolly, 2002)

- DDL is not only used to specify new database schemas but also to modify exiting ones. **A database schema is a logical grouping of objects that belong to a user.**

- **All created objects / structures (such as tables, views, indexes) are stored in a database schema.**

DDL

- Relation DB schema objects are created and maintained by using SQL DDL statements (such as CREATE, ALTER, DROP).

- The result of compiling DDL statements is a set of tables stored in special files collectively called the system catalog. The system catalog may also be referred to as a data dictionary.

- Example of DDL SQL statement:
  - CREATE
  - ALTER
  - DROP
  - RENAME
  - TRUNCATE
  - COMMENT

Data Dictionary

- The data dictionary integrates the meta-data; definitive information about the structure is recorded in a data dictionary.
  - For example: definitions about the records, data items and other objects of interest to users or required by the DBMS.

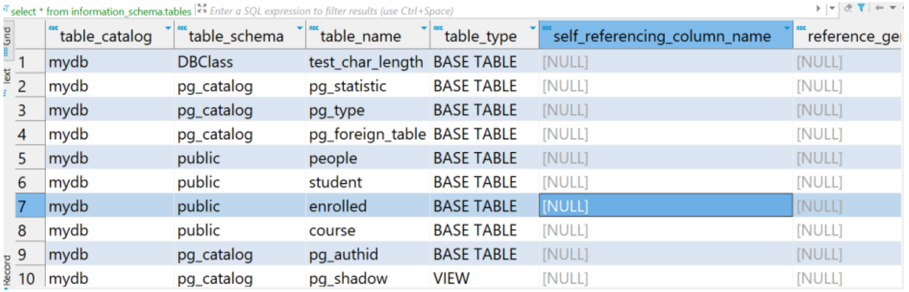
- The DBMS consults the data dictionary before accessing or manipulating the data.

- https://dataedo.com/kb/query/postgresql

Data Dictionary

- Table managing all tables.
  - `select * from information_schema.tables`

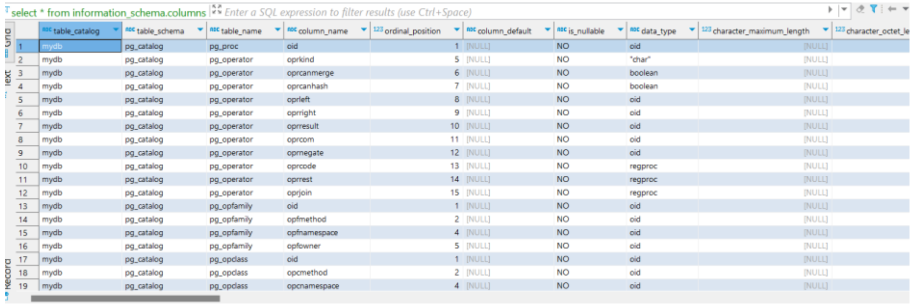
- Table managing all columns
  - `select * from information_schema.columns`



select * from information_schema.tables

Enter a SQL expression to filter results (use Ctrl+Space)

	table_catalog	table_schema	table_name	table_type	self_referencing_column_name	reference_ge
1	mydb	DBClass	test_char_length	BASE TABLE	[NULL]	[NULL]
2	mydb	pg_catalog	pg_statistic	BASE TABLE	[NULL]	[NULL]
3	mydb	pg_catalog	pg_type	BASE TABLE	[NULL]	[NULL]
4	mydb	pg_catalog	pg_foreign_table	BASE TABLE	[NULL]	[NULL]
5	mydb	public	people	BASE TABLE	[NULL]	[NULL]
6	mydb	public	student	BASE TABLE	[NULL]	[NULL]
7	mydb	public	enrolled	BASE TABLE	[NULL]	[NULL]
8	mydb	public	course	BASE TABLE	[NULL]	[NULL]
9	mydb	pg_catalog	pg_authid	BASE TABLE	[NULL]	[NULL]
10	mydb	pg_catalog	pg_shadow	VIEW	[NULL]	[NULL]



select * from information_schema.columns

Enter a SQL expression to filter results (use Ctrl+Space)

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	character_maximum_length	character_octet_le
1	mydb	pg_catalog	pg_proc	oid	1	[NULL]	NO	oid	[NULL]	
2	mydb	pg_catalog	pg_operator	oprkind	5	[NULL]	NO	"char"	[NULL]	
3	mydb	pg_catalog	pg_operator	oprcanmerge	6	[NULL]	NO	boolean	[NULL]	
4	mydb	pg_catalog	pg_operator	oprcanhash	7	[NULL]	NO	boolean	[NULL]	
5	mydb	pg_catalog	pg_operator	oprleft	8	[NULL]	NO	oid	[NULL]	
6	mydb	pg_catalog	pg_operator	oprright	9	[NULL]	NO	oid	[NULL]	
7	mydb	pg_catalog	pg_operator	oprresult	10	[NULL]	NO	oid	[NULL]	
8	mydb	pg_catalog	pg_operator	oprcom	11	[NULL]	NO	oid	[NULL]	
9	mydb	pg_catalog	pg_operator	oprnegate	12	[NULL]	NO	oid	[NULL]	
10	mydb	pg_catalog	pg_operator	oprcode	13	[NULL]	NO	regproc	[NULL]	
11	mydb	pg_catalog	pg_operator	oprrest	14	[NULL]	NO	regproc	[NULL]	
12	mydb	pg_catalog	pg_operator	oprjoin	15	[NULL]	NO	regproc	[NULL]	
13	mydb	pg_catalog	pg_opfamily	oid	1	[NULL]	NO	oid	[NULL]	
14	mydb	pg_catalog	pg_opfamily	opfmethod	2	[NULL]	NO	oid	[NULL]	
15	mydb	pg_catalog	pg_opfamily	opfnamespace	4	[NULL]	NO	oid	[NULL]	
16	mydb	pg_catalog	pg_opfamily	opfowner	5	[NULL]	NO	oid	[NULL]	
17	mydb	pg_catalog	pg_opclass	oid	1	[NULL]	NO	oid	[NULL]	
18	mydb	pg_catalog	pg_opclass	opcmethod	2	[NULL]	NO	oid	[NULL]	
19	mydb	pg_catalog	pg_opclass	opcnamespace	4	[NULL]	NO	oid	[NULL]	

- See the tables owned by the user.
  - `select * from pg_catalog.pg_stat_user_tables`

Data Manipulation Language (DML)

- This is a language that provides a set of operations to support the basic data manipulation operations on the data held in the database.

- Some of the operations include:
  - Insertion of new data into the database
  - Modification of data stored in the database
  - Retrieval of data contained in the database (Query language).
  - Deletion of data from the database.

- Examples of DML SQL statements;
  - INSERT
  - UPDATE
  - DELETE
  - MERGE

Basic SQL Summary

SQL Statements

Statement	**Description**
SELECT	Retrieves data from the database
CREATE , ALTER, DROP, RENAME, TRUNCATE	DDL: Sets up, changes and removes data structures from tables
INSERT, UPDATE, DELETE MERGE	DML: Enters new rows, changes existing rows and removes unwanted rows from tables in a database
COMMIT, ROLLBACK, SAVEPOINT	Transaction Control (TC): Manages changes made by DML. Changes to data may be grouped into logical transactions
GRANT, REVOKE	Data Control Language (DCL): Gives and removes access rights to both the database and the structures within it.

Database, Schema, Table

Database and Schema

- **CREATE DATABASE** DatabaseName
- **DROP DATABASE** DatabaseName
- **CREATE SCHEMA** SchemaName
- **DROP SCHEMA** SchemaName

- Example: dbcourse.student means the student table in dbcourse schema

- Tables belonging to other users are not in the user's schema.
- You should use the owner's name as a prefix to those tables.

Create table

- **CREATE TABLE** SQL syntax

```
CREATE TABLE tablename
(columnname1 data_type,
columnname2 data_type, …);
```

- Example:

```
CREATE TABLE student
(s_id CHAR(5),
 s_first VARCHAR2(20));
```

- Basic data types
  - Character
  - Number
  - Date/time
  - Large object

Naming Rules

Table names and column names:

- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle server reserved word

Data Type

ISO SQL Data Types

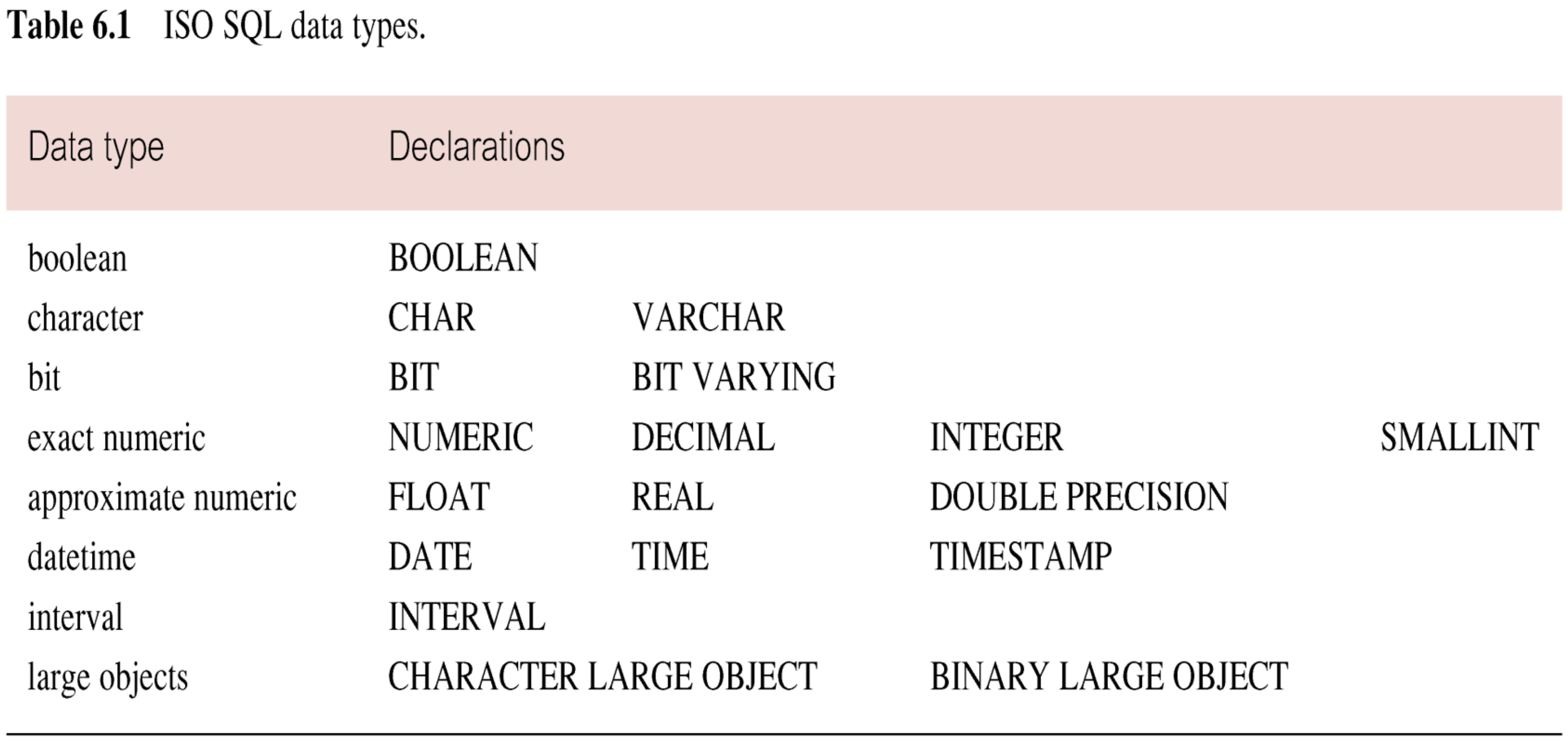


Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

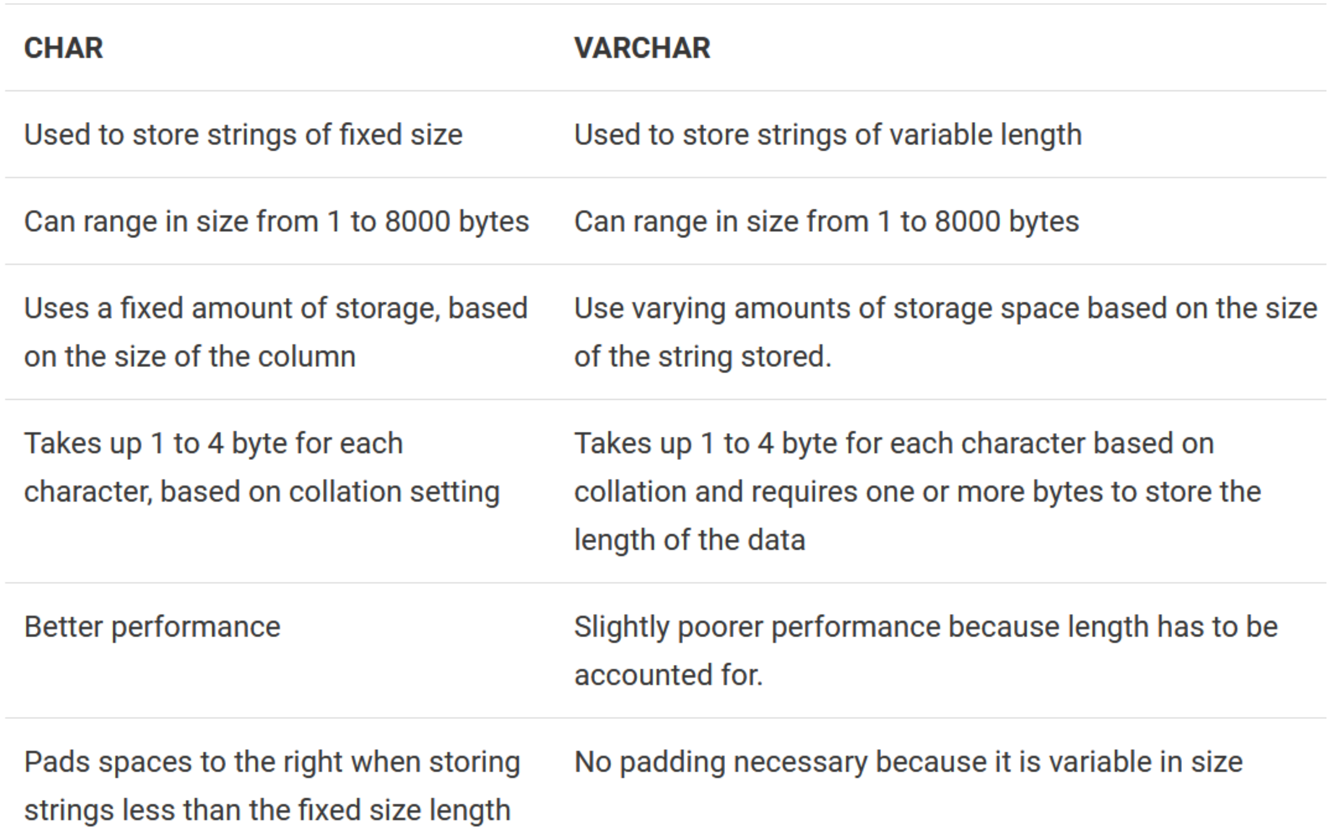
Character Data Types

- VARCHAR
  - Variable-length character data (up to 4000 characters)
  - Syntax: *columnname* VARCHAR(*maximum_size)*
  - If user enters data value less than maximum_size, DBMS only stores actual character values
- CHAR
  - Fixed-length character data (default = 2000)
  - Syntax: *columnname* CHAR(*maximum_size*)
  - If user enters data value less than maximum_size, DBMS adds trailing blank spaces to the end of entry
  - The CHAR data type uses the storage more efficiently and processes data faster than the VARCHAR2 type.

Character Example

```
create table Test_Char_Length (first char(4), second varchar(5))



insert into Test_Char_Length values ('12','123')
insert into Test_Char_Length values ('1234','1234')
insert into Test_Char_Length values ('1234','12345')
insert into Test_Char_Length values ('1234','123456')
insert into Test_Char_Length values ('12345','12345')


select * from test_char_length


SELECT CONCAT('(', first, ')'), CONCAT('(', second, ')') FROM
Test_Char_Length;
```

Character Data Types



CHAR	VARCHAR
Used to store strings of fixed size	Used to store strings of variable length
Can range in size from 1 to 8000 bytes	Can range in size from 1 to 8000 bytes
Uses a fixed amount of storage, based on the size of the column	Use varying amounts of storage space based on the size of the string stored.
Takes up 1 to 4 byte for each character, based on collation setting	Takes up 1 to 4 byte for each character based on collation and requires one or more bytes to store the length of the data
Better performance	Slightly poorer performance because length has to be accounted for.
Pads spaces to the right when storing strings less than the fixed size length	No padding necessary because it is variable in size

Text

VARCHAR vs. TEXT

- Indexing Ability: VARCHAR can be fully indexed, while TEXT columns can be indexed only up to a certain length.
- Sorting Possibility: VARCHAR can be sorted using the entire length of the String, but this is not possible for TEXT
- Storage usage: TEXT occupies 2 + length of string storage space, while VARCHAR occupies 1 + length of string, up to 255 characters, and 2 + length of string greater than 255 characters. So, up to 255 characters, VARCHAR even uses lesser storage than TEXT.
- Performance Optimization: Based on the database technology!!! For example: VARCHAR can be stored in MySQL's memory storage; however, TEXT is not supported by it. So, if a query involves a TEXT column, temporary tables are created on the disk storage. Using disk-based tables takes a toll on the resources, and query run completion takes longer. PostgreSQL does not differentiate between TEXT and VARCHAR in terms of storage or performance. Both types are variable-length strings that can store very large amounts of text.
- Length: VARCHAR can enforce a maximum length constraint, which TEXT does not

Number

- The NUMBER data type is used to store negative, positive, integer, fixed-decimal, and floating-point numbers.

- When a number type is used for a column, its **precision** and **scale** can be specified.

  - Precision is the total number of significant digits in the number, both to the left and to the right of the decimal point.
  - Scale is the total number of digits to the right of the decimal point.

Number -- integer

- An **integer** is a whole number without any decimal part.
- The data type for it would be defined as NUMBER(3), where 3 represents the maximum number of digits.

Number – fixed-point

- Decimal number has a specific number of digits to the right of the decimal point.
- The PRICE column has values in dollars and cents, which requires two decimal places – for example, values like 2.95, 3.99, 24.99, and so on.
- If it is defined as NUMBER(4,2), the first number specifies the **precision** and the second number the **scale**.

Number – floating-point

- A **floating-point** decimal number has a variable number of decimal places
- To define such a column, do not specify the scale or precision along with the NUMBER type.
- By defining a column as a floating-point number, a value can be stored in it with very high precision

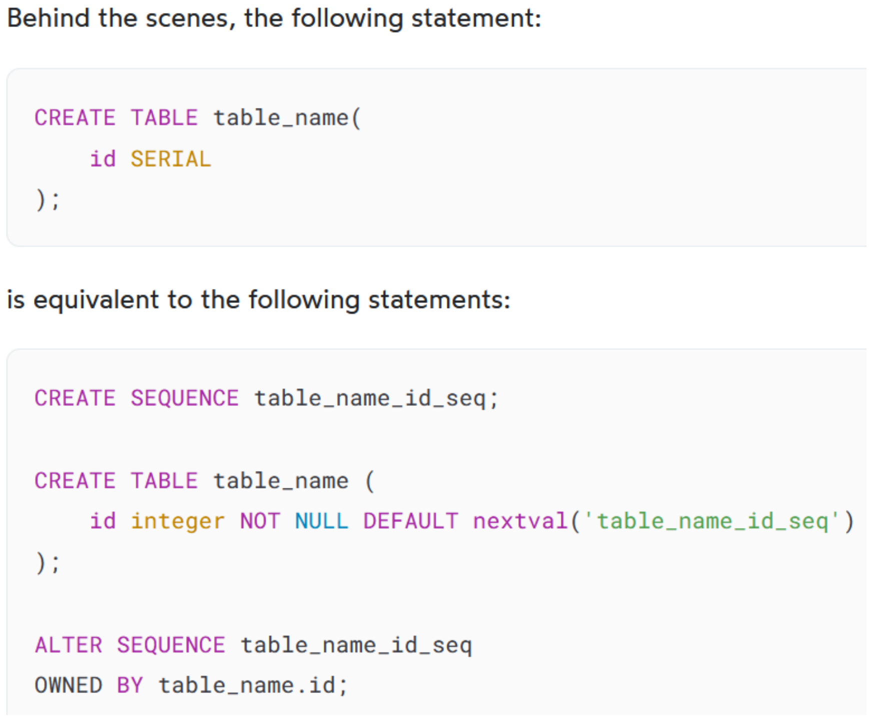
Number Example

In postgres number types are NUMERIC(p, q), DECIMAL(p, q), REAL, INTEGER, SMALLINT, FLOAT(p), DOUBLE PRECISION. The types decimal and numeric are equivalent.

```
create table Test_Number (f1 numeric, f2 numeric (2), f3 numeric (2,1))

insert into Test_Number values (232.34,24,3.1)

select * from Test_Number
```

Auto Increment

- Use the PostgreSQL pseudo-type SERIAL to create an auto-increment column for a table.



Behind the scenes, the following statement:

```
CREATE TABLE table_name(
    id SERIAL
);
```

is equivalent to the following statements:

```
CREATE SEQUENCE table_name_id_seq;

CREATE TABLE table_name (
    id integer NOT NULL DEFAULT nextval('table_name_id_seq')
);

ALTER SEQUENCE table_name_id_seq
OWNED BY table_name.id;
```

Auto Increment

- Mysql> Create table grocery_inventory (id int not null primary key auto_increment, item_name varchar (50) not null, item_desc text, item_price float not null, curr_qty int not null);

- Auto_Increment is a table modifier/constraint that will request MySQL to add the next available number to the ID field for you.

- Postgres Example:

```
CREATE TABLE CountNum (name char(5),regNo serial)
insert into CountNum values('DB'),('DS');
select * from CountNum
```

Date And Time Data Types

- Datetime data subtypes
  - Store actual date and time values
  - DATE
  - TIMESTAMP
- Interval data subtypes
  - Store elapsed time interval between two datetime values
  - INTERVAL YEAR TO MONTH
  - INTERVAL DAY TO SECOND

Date And Time Data Types

- DATE
  - Stores dates from Dec 31, 4712 BC to Dec 31, AD 4712
  - Default date format: DD-MON-YY
  - Default time format: HH:MI:SS AM
  - Syntax: `columnname DATE`

- TIMESTAMP
  - Stores date values similar to DATE data type . It stores the year, month, and day of the DATE data type, plus hour, minute, and second values as well as the fractional second value. Also stores fractional seconds.
  - Syntax: `columnname TIMESTAMP (fractional_seconds_precision)`

    If omitted, default is 6 decimal place
  - Example : `shipment_date TIMESTAMP(2)`

Date And Time Data Types

- INTERVAL YEAR TO MONTH
  - Stores time interval expressed in years and months using the following syntax:
  - Example:
    - `create table Interval_Time (time_enrolled INTERVAL YEAR TO MONTH)`
    - `insert into Interval_Time values (INTERVAL '13' MONTH),(INTERVAL '1' MONTH),(INTERVAL '18' MONTH);`
    - `select * from Interval_Time`

Date And Time Data Types

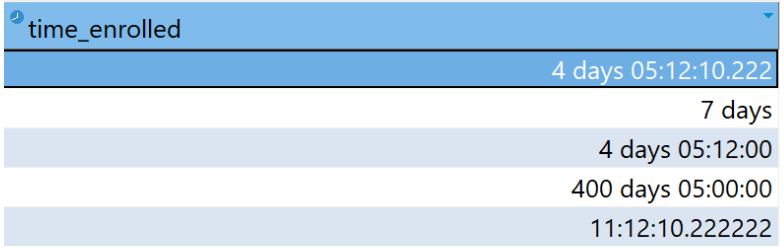
- INTERVAL YEAR TO MONTH
  - Example:

```
INTERVAL '123-2' YEAR TO MONTH
Indicates an interval of 123 years, 2 months.

INTERVAL '123' YEAR
Indicates an interval of 123 years 0 months.

INTERVAL '300' MONTH
Indicates an interval of 300 months.
```

Date And Time Data Types

- INTERVAL DAY TO SECOND
  - INTERVAL DAY TO SECOND stores a period of time in terms of days, hours, minutes, and seconds.
  - Example:
    - `create table Interval_Time_Day (time_enrolled INTERVAL DAY TO SECOND)`

    -
      ```
      insert into Interval_Time_Day values
        (INTERVAL '4 5:12:10.222' DAY TO SECOND),
        (INTERVAL '7' DAY),
        (INTERVAL '4 5:12' DAY TO minute),
        (INTERVAL '400 5' DAY TO hour),
        (INTERVAL '11:12:10.2222222' HOUR TO second)
      ```
    - `select * from Interval_Time_Day`



time_enrolled
4 days 05:12:10.222
7 days
4 days 05:12:00
400 days 05:00:00
11:12:10.222222

Large Object (LOB) Data Types

- Store binary data such as:
  - Digitized sounds or images
  - References to binary files from word processor or spreadsheet
- How? Additional topic for study.