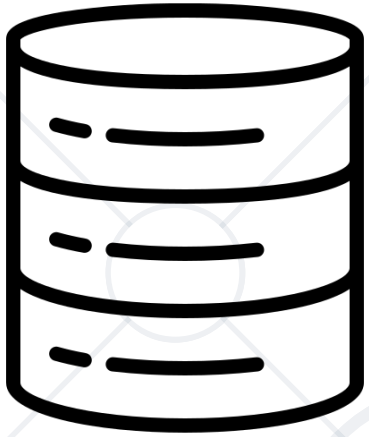
PostgreSQL Introduction. Data Types. Table Basics.



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions

Software University

sli.do

#python-db

Table of Contents

Data Management

When Do We Need a Database?

Storage vs. Management

SALES RECEIPT

Date: 07/16/2016
Order#:[00315]

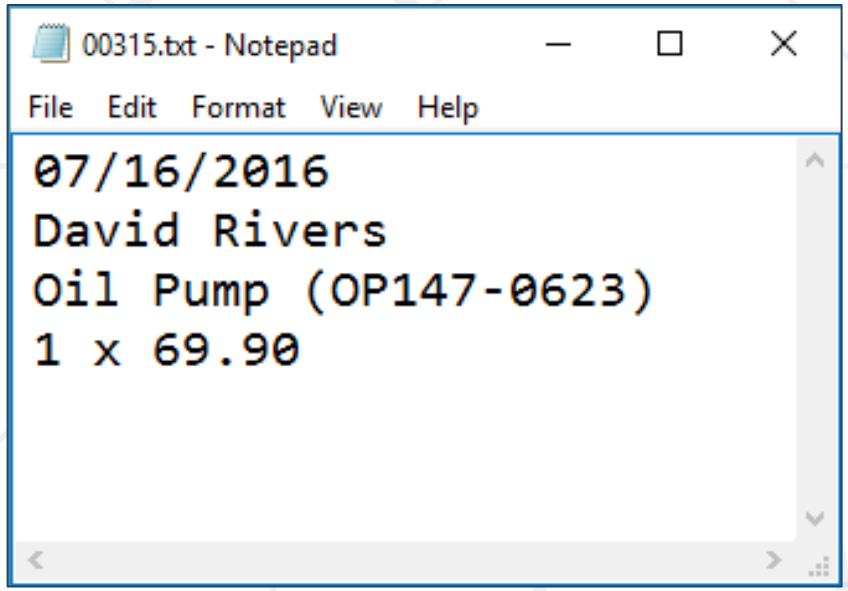
Customer: David Rivers
Product: Oil Pump
S/N: OP147-0623

Unit Price: 69.90
Qty: 1

Total: 69.90

00315 – 07/16/2016
David Rivers
Oil Pump (OP147-0623)
1 x 69.90

Storage vs. Management



Order#	Date	Customer	Product	S/N	Qty
00315	07/16/2016	David Rivers	Oil Pump	OP147-063	1

Database

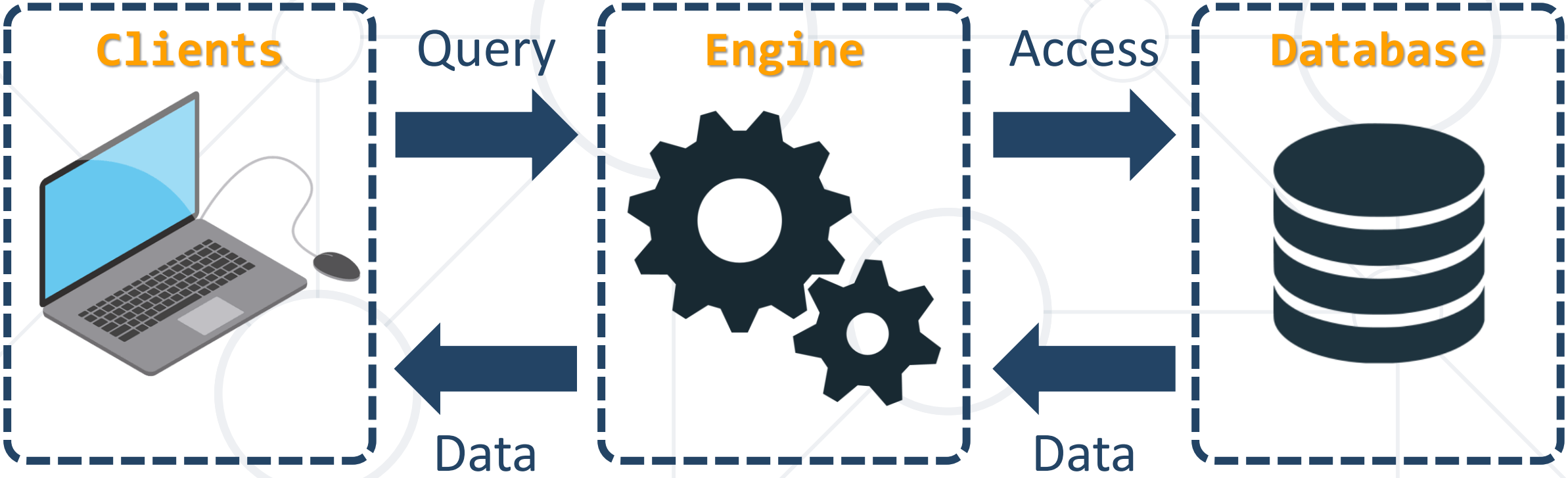
- A database is an **organized** collection of **related** information
  - The user doesn't have **direct access** to the stored data
  - Access to data is usually **provided by a DBMS**

Database Management System

- **D**ata**B**ase **M**anagement **S**ystem
  - **Provides tools** to define, manipulate, retrieve and manage data in a database
  - **Parses requests** from the user and takes the appropriate action

Database Engine Flow

- PostgreSQL uses the Client-Server Model



Database Table Elements

- The table is the main **building block** of any database

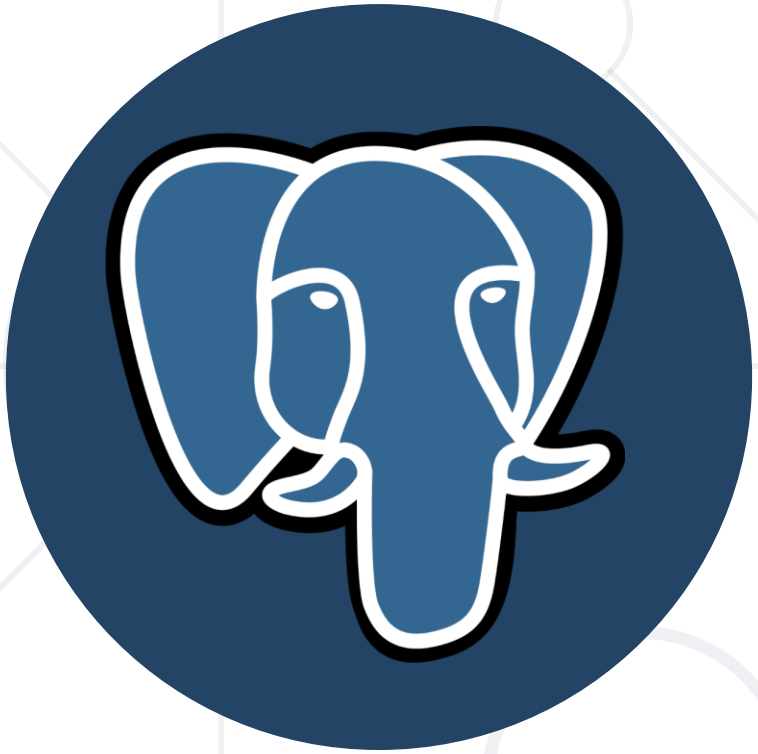
Column

CustomerID	FirstName	Birthdate	CityID
1	Brigitte	03/12/1975	101
2	August	27/05/1968	102
3	Benjamin	15/10/1988	103
4	Denis	07/01/1993	104

Row

Cell

- Each **row** is called a **record** or **entity**
- Columns (**fields**) define the **type** of data they contain



PostgreSQL

What is PostgreSQL?

- **O**bject–**R**elational **D**atabase **M**anagement **S**ystem (**ORDBMS**)
- Widely used **open-source cross-platform** system

Rank			DBMS	Database Model
Jan 2023	Dec 2022	Jan 2022		
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	6.	Redis	Key-value, Multi-model
7.	7.	7.	IBM Db2	Relational, Multi-model
8.	8.	8.	Elasticsearch	Search engine, Multi-model
9.	9.	9.	Microsoft Access	Relational
10.	10.	10.	SQLite	Relational

What Makes PostgreSQL Stand Out?

- First DBMS that implements **multi-version concurrency control feature**
  - Data can be safely read and updated at the same time
- Able to add **custom functions**
- Designed to be **extensible**
- Defining custom data types, plugins, etc.
- Very active community

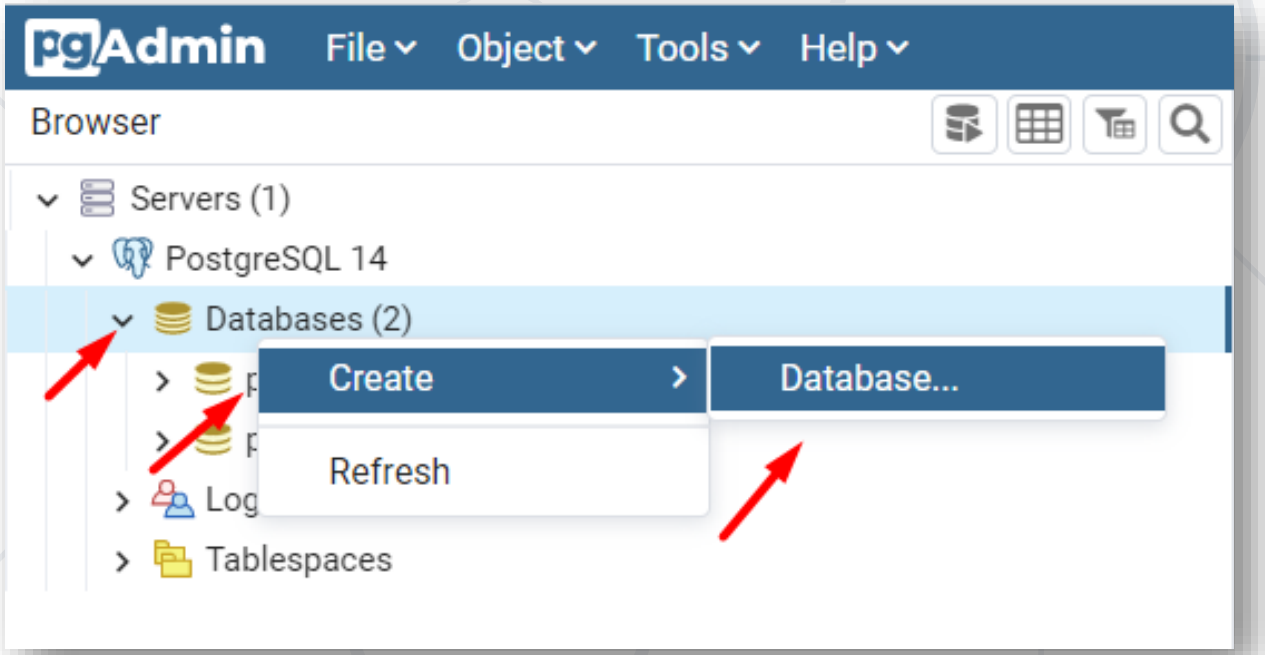
What is pgAdmin?

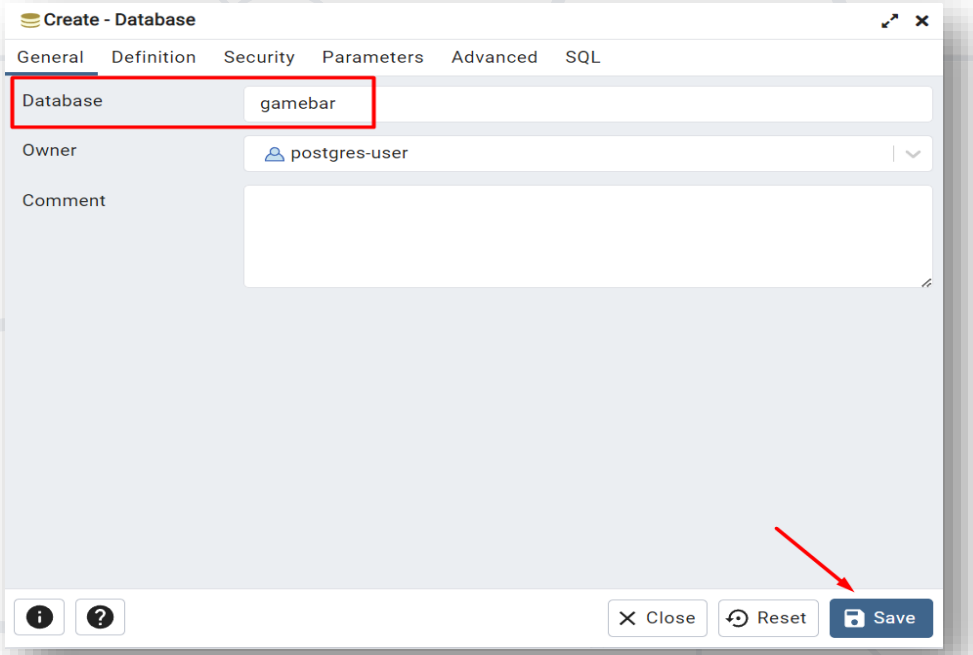
- Open-source **administration** and **development platform** for PostgreSQL
  - **Graphical** user interface for using PostgreSQL

Installation

- Install PostgreSQL and pgAdmin locally
  - PostgreSQL
  - pgAdmin
- You have the option to utilize PostgreSQL and pgAdmin using Docker (recommended for Docker enthusiasts but is not mandatory)
  - Docker Installation Guide
  - PostgreSQL and pgAdmin with Docker

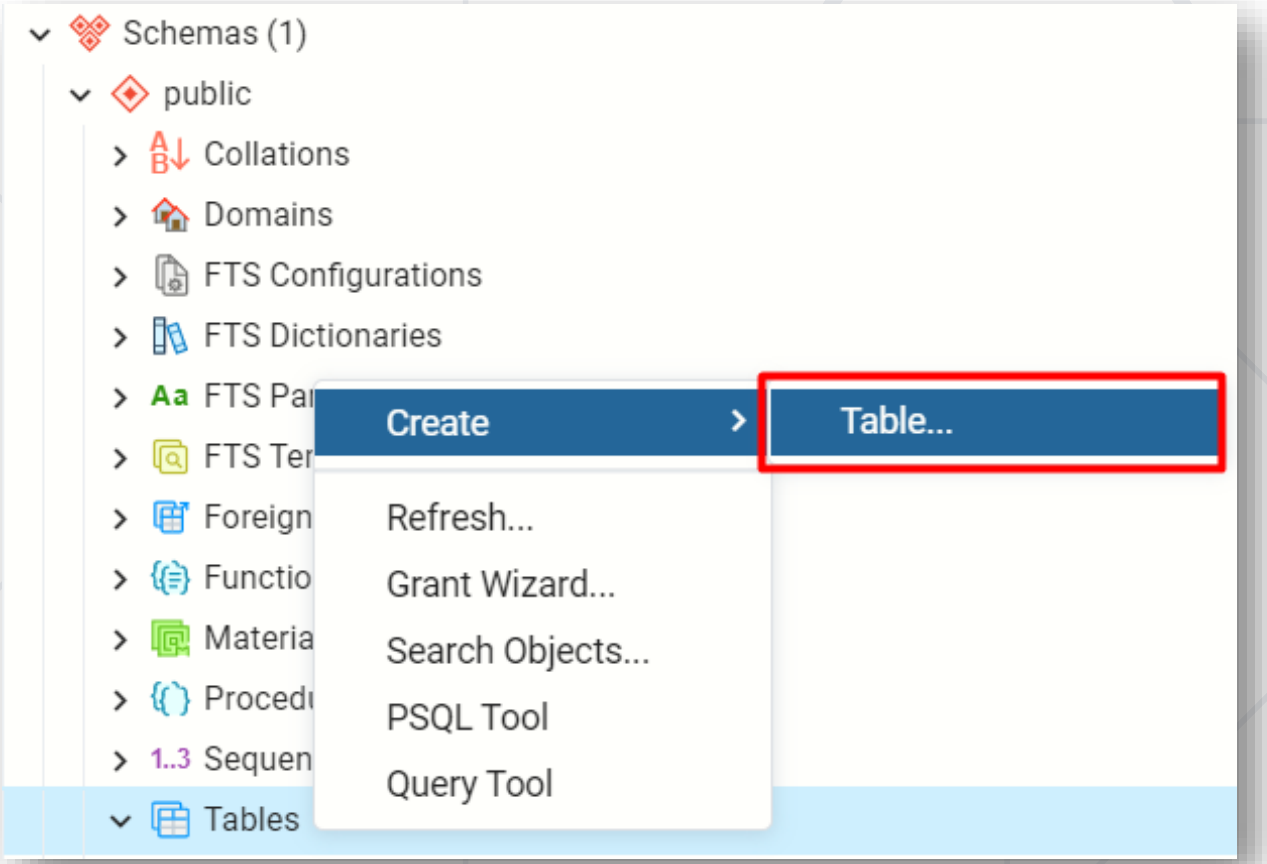
Create a New Database in pgAdmin 4

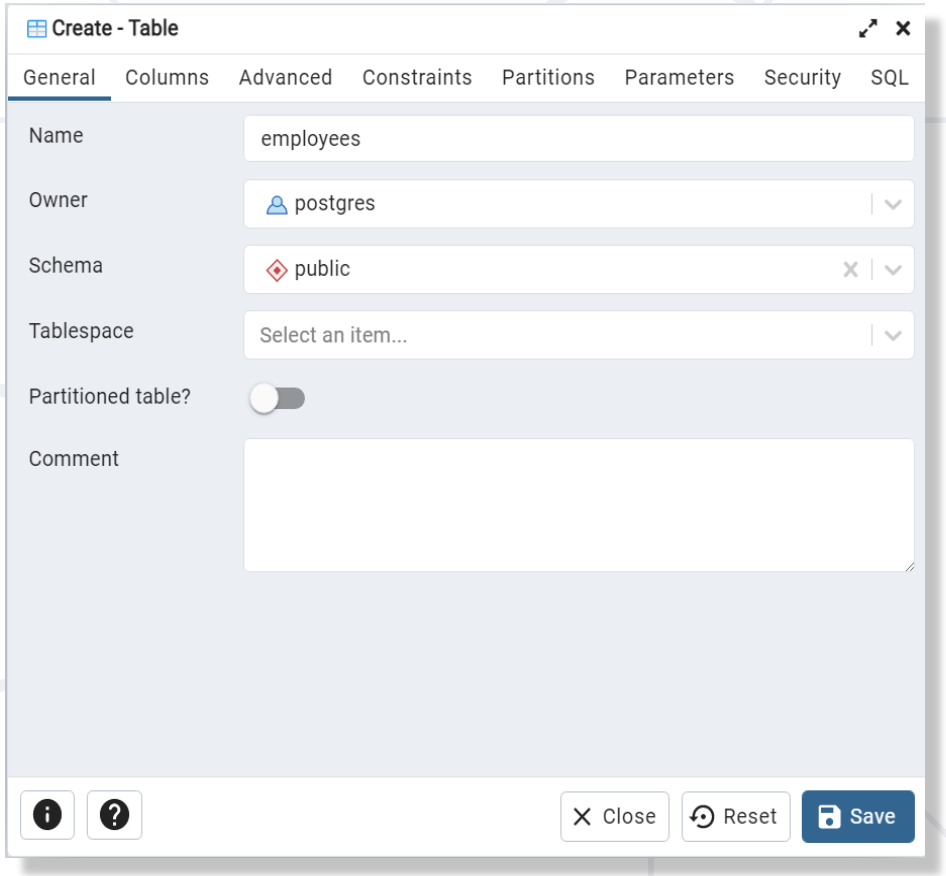




Create a New Table

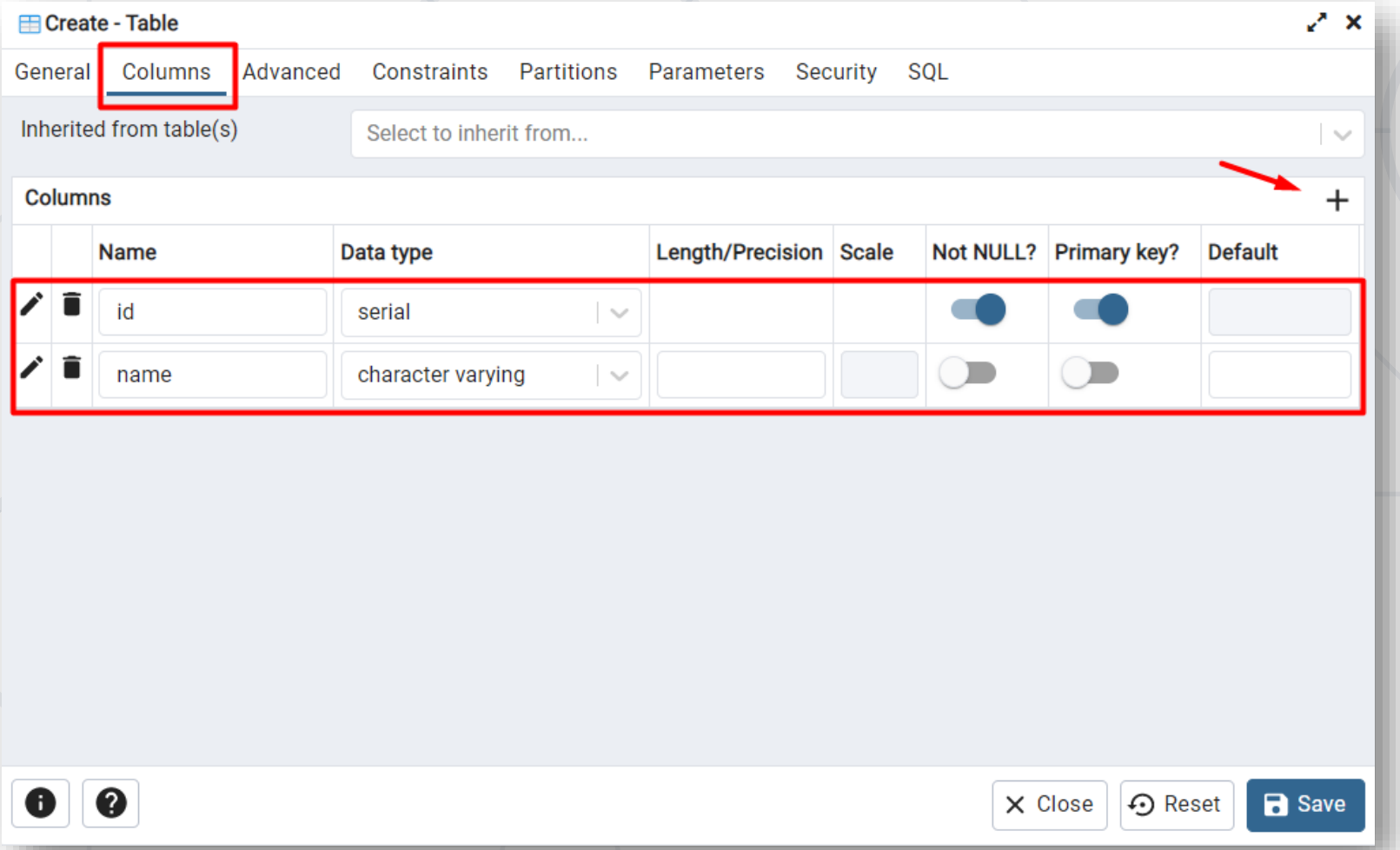
- Right click on the Database/ Schemas/ Tables





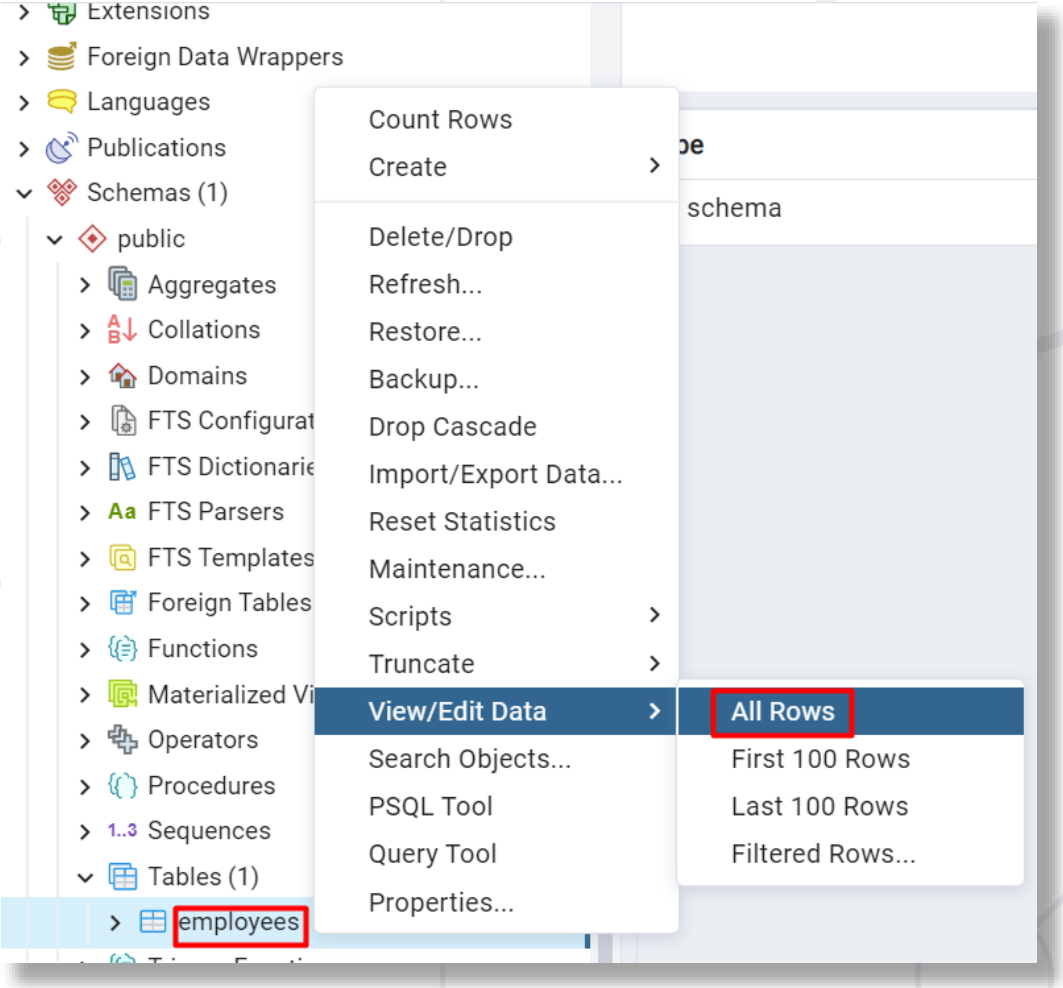
Create a New Table

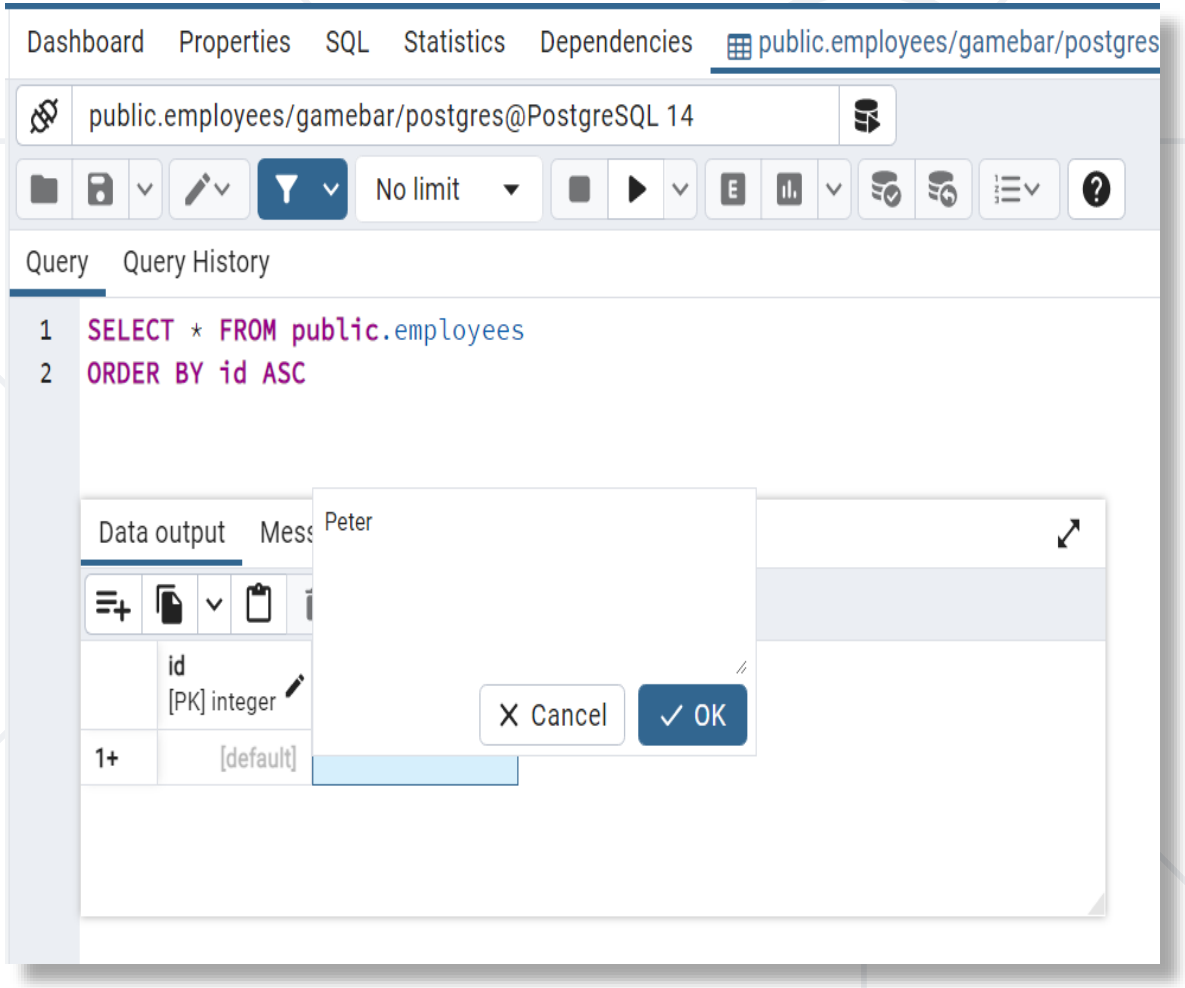
- Create columns in the table



View / Edit Data

- Right-click on the created table **`employees`**





Structured Query Language

What is SQL?

- **Programming** language
- Designed for **managing** data in a **relational** database
  - Access **many records** with **one single command**
  - **Eliminates** the need to specify **how** to reach a record
- Developed at **IBM** in the early 1970s

SQL Queries

- We communicate with the database engine using SQL
- Queries provide greater **control** and **flexibility**
- To create a database using SQL:

Database name

```
CREATE DATABASE gamebar;
```

- SQL keywords are conventionally **capitalized**

SQL Elements

- Subdivided into several language elements
  - Queries
  - Clauses
  - Expressions
  - Predicates
  - Statements

Update clause

Expression

Statement

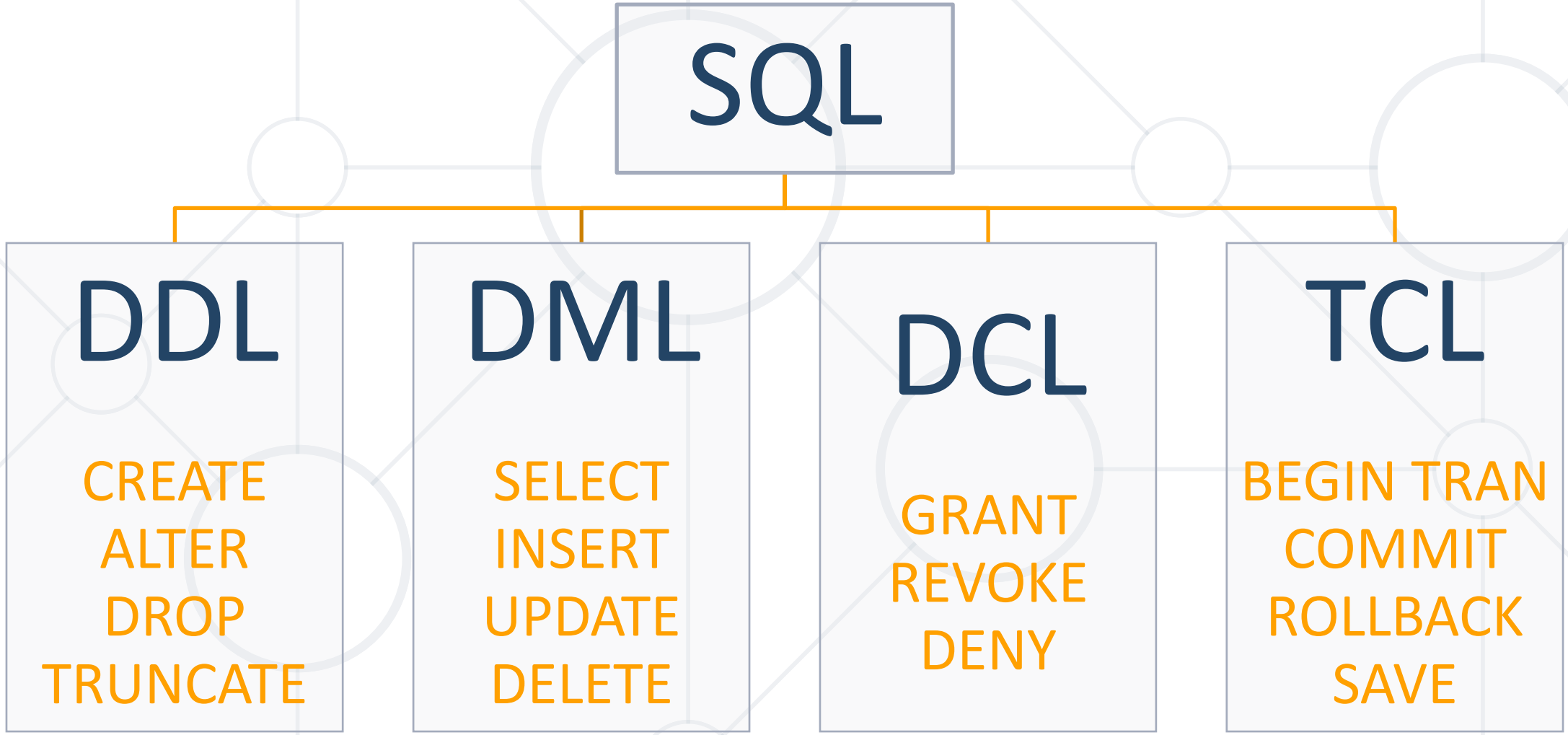
```
UPDATE employees
SET   salary = salary * 1.1
WHERE job_title = "Cashier";
```

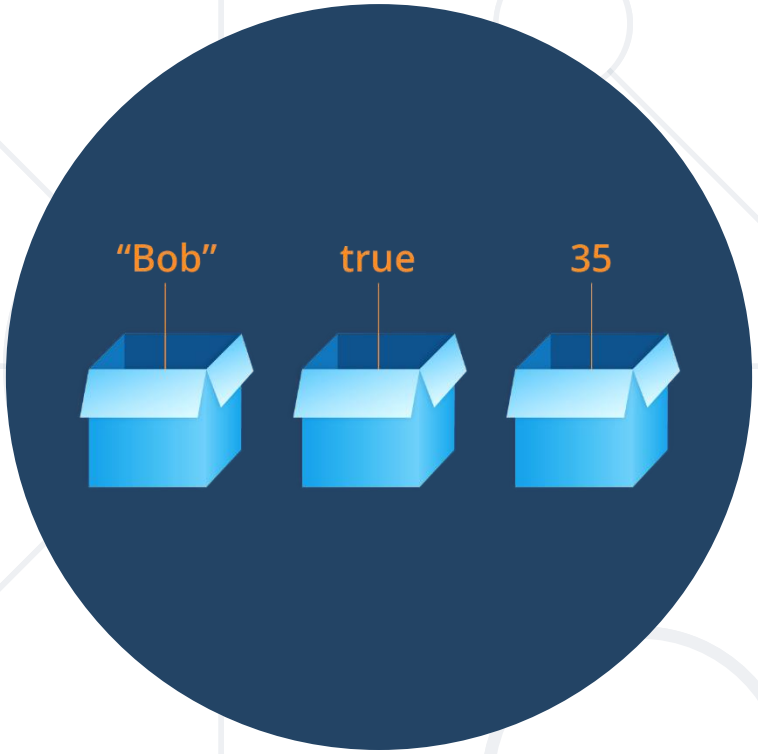
Predicate

SQL vs NoSQL Databases

- **SQL Database**:
  - **Relational** database management system
  - **Predefined** Schema
  - Suited for **complex** queries
  - **Vertically** scalable
- **NoSQL database**:
  - **Non-relational** database system
  - **Dynamic** Schema
  - Suited for **hierarchical** data storage
  - **Horizontally** scalable

SQL Logical Division





Data Types in SQL

Numeric Data Types

- **Integer types**
  - SMALLINT, INTEGER/INT, BIGINT
- **Arbitrary Precision Numbers**
  - DECIMAL, NUMERIC
- **Floating-Point Types**
  - REAL, DOUBLE PRECISION
- **Serial Types**
  - SMALLSERIAL, SERIAL, BIGSERIAL

The type INTEGER/INT is the common choice

Recommended for storing quantities where exactness is required

Storing and retrieving a value might show a slight difference

Used for creating unique identifier columns

Problem: Employees Lab Part I

- Using table "**`employees`**" add columns or modify existing ones:
  - The **`id`** of an employee, unique and automatically incremented
    - It is a **`"PRIMARY KEY"`**
  - The **`salary`**, specified to the second decimal place and has 10 digits
  - The **`devices_number`** given to an employee

Solution: Employees Lab Part I

Columns

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
id	serial			✓	✓	
salary	numeric	10	2	✗	✗	
devices_number	integer			✗	✗	

id [PK] integer	salary numeric (10,2)	devices_number integer
1	1580.33	5
2	2450.00	3
3	950.00	0

String Data Types

- **CHARACTER/CHAR**[(M)]
  - Fixed-length e.g., CHAR(30)
  - CHAR without the length specifier (m) is the same as CHAR(1)
- **CHARACTER VARYING/VARCHAR**[(N)]
  - Variable-length with limit e.g., VARCHAR(30)
  - VARCHAR without (n) can store a string with unlimited length
- **TEXT**
  - Stores strings of any length

CHAR vs VARCHAR

- Storing data in CHAR and VARCHAR examples

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 bytes
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Problem: Departments Lab Part I

- Create a table "**`departments`**" containing:
  - The **`id`** of a department, unique and automatically incremented
    - It is a **`"PRIMARY KEY"`**
  - The **`name`** with a max length of 50 characters
  - The department **`code`**, always containing 3 characters
  - The **`description`** of a department that can be of any length

Solution: Departments Lab Part I

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		id	serial			✓	✓	
		name	character varying	50				
		code	character	3				
		description	text					

	id [PK] integer	name character varying (50)	code character (3)	description text
1	1	Human Recourses	HRS	The Human...

Date Types

- **DATE** - for values with a date part but **no time part**
  - 2016-06-23
- **TIME** - for values with time but **no date part**
  - 14:01:10
- **TIMESTAMP** - both date **and** time parts
  - 2020-10-05 14:01:10
- **TIMESTAMPTZ** - both date **and** time parts **with time zone**
  - 2020-10-05 14:01:10+02:00

Problem: Issues Lab Part I

- Create a table "**`issues`**" containing:
  - The **`id`** of an issue, unique and automatically incremented
    - It is a **`"PRIMARY KEY"`**
  - The **`description`** with a max length of 150 characters
  - The **`date`** it was created
  - The **`start`**, the date and time when it was started

Solution: Issues Lab Part I

Columns

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
id	serial			✓	✓	
description	character varying	150				
date	date					
start	timestamp without time...					

	id [PK] integer	description character varying (150)	date date	start timestamp without time zone
1	1	Set up a GH actions pi...	2023-01-30	2023-01-30 12:34:10

Column Constraints

- Set **no repeating values** in the entire table

```
email VARCHAR (50) UNIQUE
```

- If a value is **not specified**, use the default one

```
balance DECIMAL (10,2) DEFAULT 0
```

- Set a column that must **not assume a null value**

```
name VARCHAR (100) NOT NULL
```

Column Constraints

- Set a **primary key** to uniquely define a record

```
id INT NOT NULL PRIMARY KEY
```

- To **automatically increment** the primary key, use SERIAL

```
id SERIAL PRIMARY KEY
```

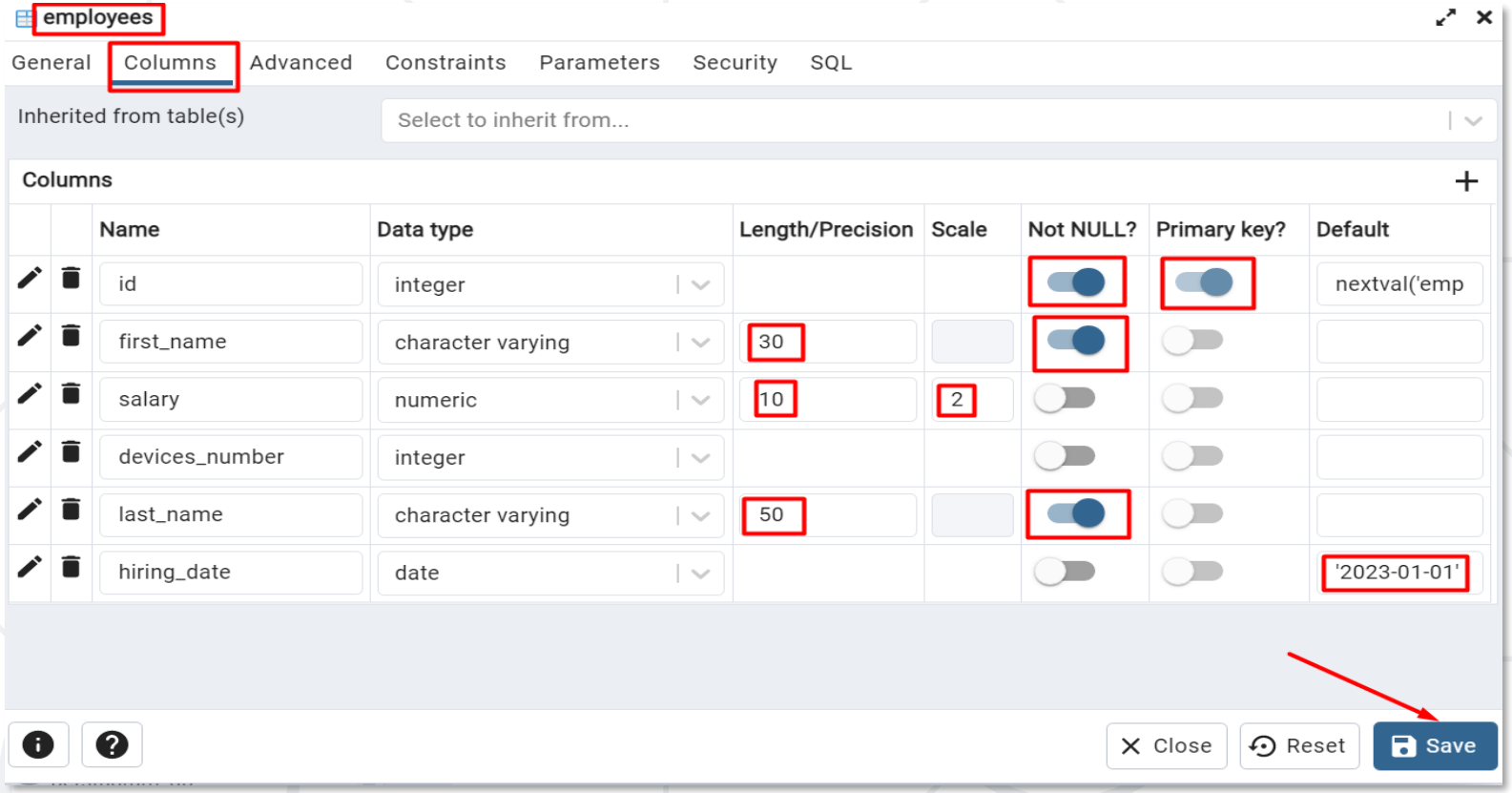
- To **check** the value being entered into a record
  - If false, the value is NOT entered into the table

```
salary DECIMAL(10, 2) CHECK(salary > 0)
```

Problem: Employees Lab Part I

- Add/modify columns and constraints into table **"employees"** :
  - The **id** should be unique and automatically incremented
    - It is a **"PRIMARY KEY ","NOT NULL"**
  - The **first_name** with a max length of 30 characters, not null
  - The **last_name** with a max length of 50 characters, not null
  - The **hiring_date**, default "2023-01-01"
  - The **salary**, specified to the second decimal place
  - The **devices_number** given to the employee

Solution: Employees Lab Part I



```sql
CREATE TABLE people (
  id INT NOT NULL,
  email VARCHAR NOT NULL,
  first_name VARCHAR(50),
  last_name VARCHAR(50)
);
```

Data Definition

Creating and Modifying Database Objects

Create Table

```
CREATE TABLE example_table(
    column1 datatype,
    ...
);
```

Table name

Column name

Type of the column data

- Inside the parenthesis, add the information for creating the columns for the table
- Without parenthesis, we will get an error message

Create Table Using SQL

Table name

```
CREATE TABLE employees (

  id SERIAL PRIMARY KEY,
  email VARCHAR (50) NOT NULL,
  first_name VARCHAR (50),
  last_name VARCHAR (50)
);
```

Column constraint

Column name

Data type

Altering Tables Using SQL

- A table can be changed using the keywords **ALTER TABLE**

```
ALTER TABLE employees;
```

Table name

- Add a new column

```
ALTER TABLE employees
ADD COLUMN salary DECIMAL;
```

Column name

Data type

Altering Tables Using SQL

- Delete existing column

```
ALTER TABLE employees
DROP COLUMN email;
```

Column name

- Modify the data type of the existing column

```
ALTER TABLE employees
ALTER COLUMN last_name TYPE VARCHAR(100);
```

Column name

New data type

Dropping and Truncating

- To delete all the entries in a table

```
TRUNCATE TABLE employees;
```

Table name

- To drop a table - delete data and structure

```
DROP TABLE employees;
```

Table name

- To drop the entire database

```
DROP DATABASE gamebar;
```

Database name

Problem: Lab Part II

- Using simple SQL queries:
  - Create a database **"gamebar"**. Open its **Query Tool**
  - Create tables **"employees"**, **"departments"** and **"issues"**
  - Alter tables
  - Modify columns and add constraints
  - Truncate table
  - Drop table

Solution: Lab Part II

```sql
CREATE DATABASE gamebar;
CREATE TABLE employees (
  id SERIAL PRIMARY KEY NOT NULL,
  first_name VARCHAR (30),
  last_name VARCHAR (50),
  hiring_date DATE DEFAULT '2023-01-01',
  salary NUMERIC(10, 2),
  devices_number INT
);
CREATE TABLE departments (-- TODO);
CREATE TABLE issues (-- TODO);
```

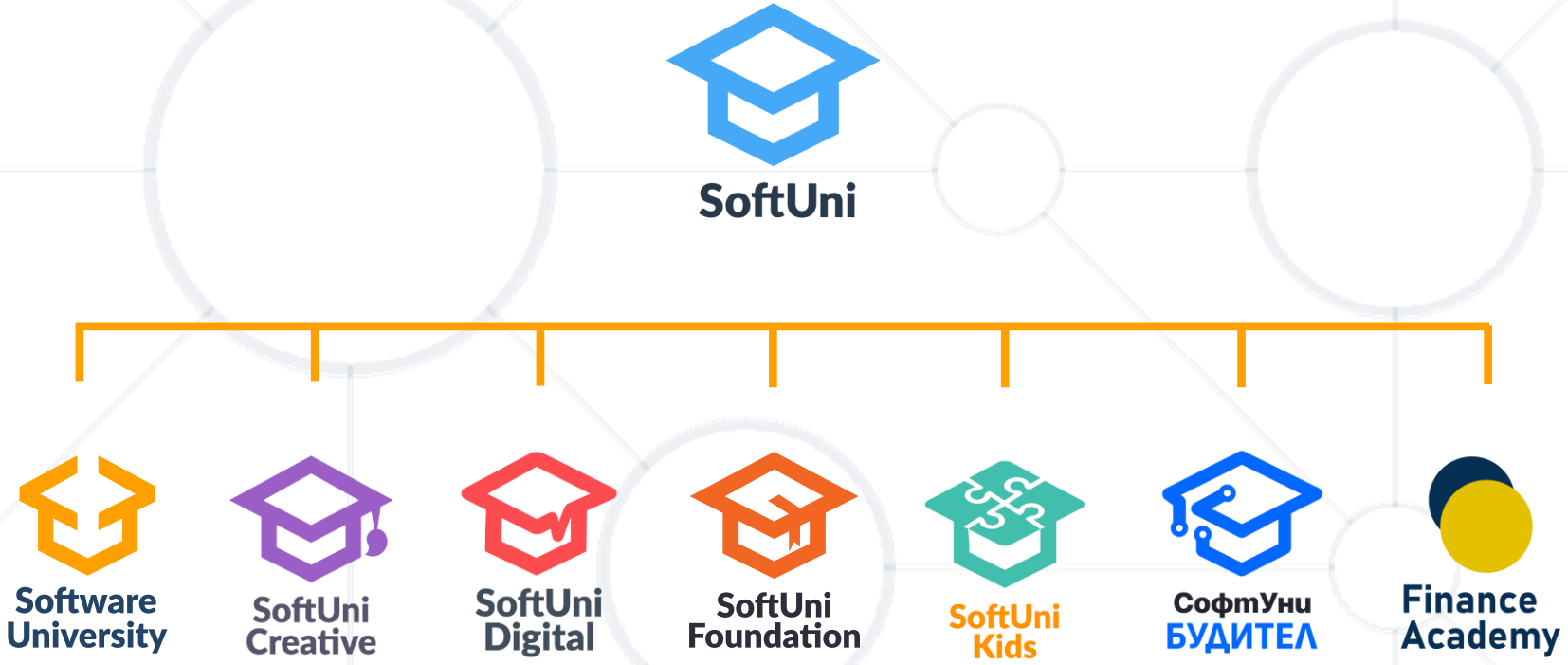
Solution: Lab Part II

```sql
ALTER TABLE employees
ADD COLUMN middle_name VARCHAR(50);

ALTER TABLE employees
ALTER COLUMN salary SET NOT NULL,
ALTER COLUMN salary SET DEFAULT 0,
ALTER COLUMN hiring_date SET NOT NULL;

ALTER TABLE employees
ALTER COLUMN middle_name TYPE VARCHAR(100);

TRUNCATE TABLE issues;
DROP TABLE departments;
```

Summary

- Data **Management**
- **PostgreSQL**
- Structured **Query Language**
- Data **Types**
- **Table** Basics

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction, or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg