4. Introduction to Relational Databases (RDBMS)

Semana 1

Review of data fundamentals

> Data is unorganized information, that is processed to make it meaningful.

- structured
  - RDBMS
- semi-structured
  - some organizational properties but not enought for table format
  - tags and metadata
- unstructured
  - does not follow any specific semantics
  - NoSQL

OLTP (Transactional or Online transaction processing System)

- desgined to store high volume day-to-day operational data
- typically, relational but can also be non-relational

OLAP (Analytical or Online analytical processing System)

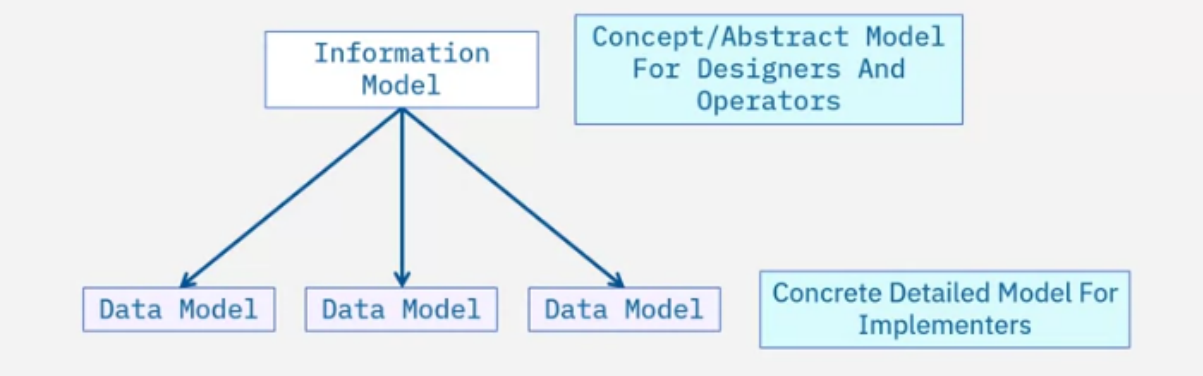
- optimized for complex data analytics
- incllude relational and non-relational databases, data warehouses, data lakes and big data stores

Relational databases examples

- ibm db2
- microsoft sql server

- oracle
- mysql

Information and Data Models



Entities

- can be from the real world

Information model

- conceptual
- relationships
- can be hierarchical (child nodes...)
  - a child node can have only one parent node
  - a parent node can have many children

Data model

- blueprint of a database system
- **relational model**
  - data independence
  - **data is stored in tables**
  - logical data independence
  - physical data independence
  - physical storage independence

ER data model

Entity relational data model (like genexus)

Entity relational diagram ERD

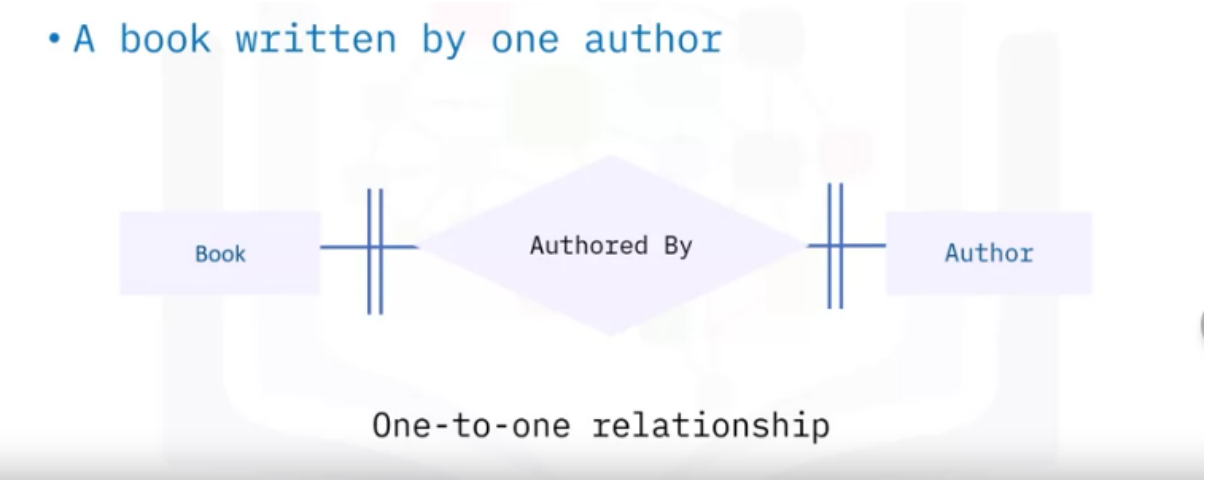
- entities and attributes
- attributes tell us more about the entity
- entity → rectangle
- attribute → oval

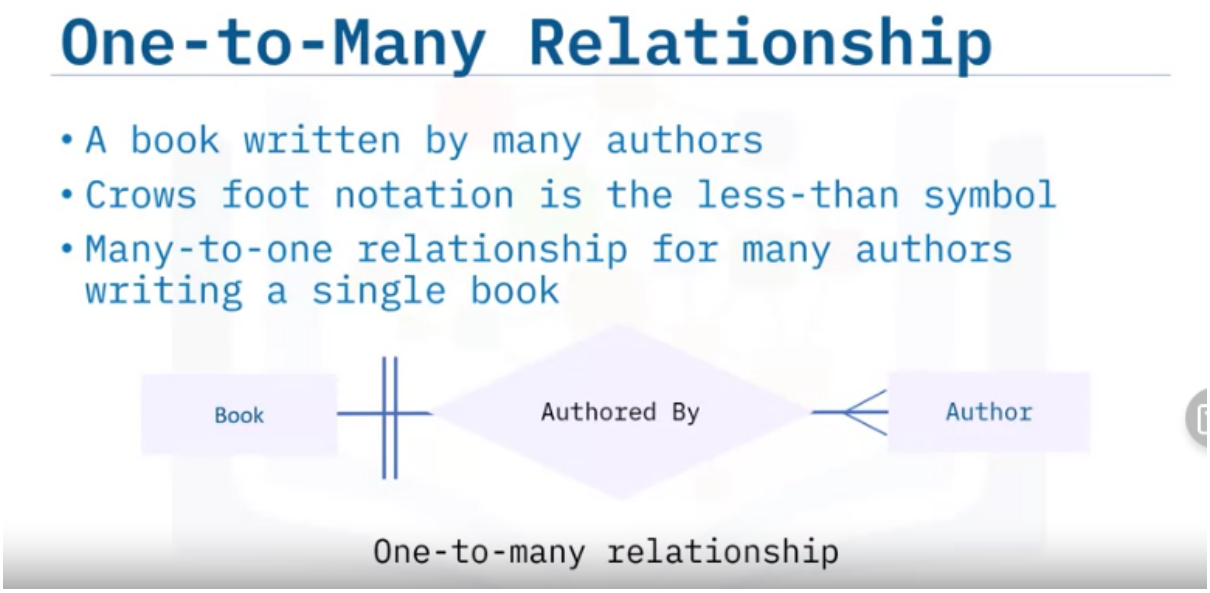
ERDs and types of relationship

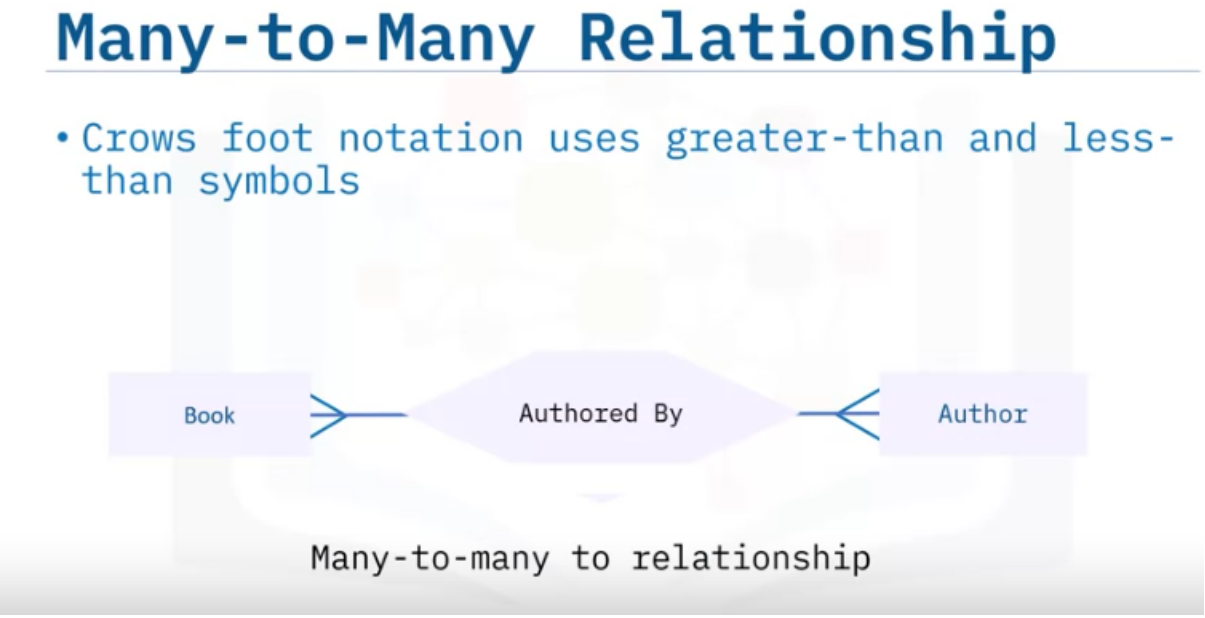
One book needs to be written by AT LEAST 1 author

One author can write many books

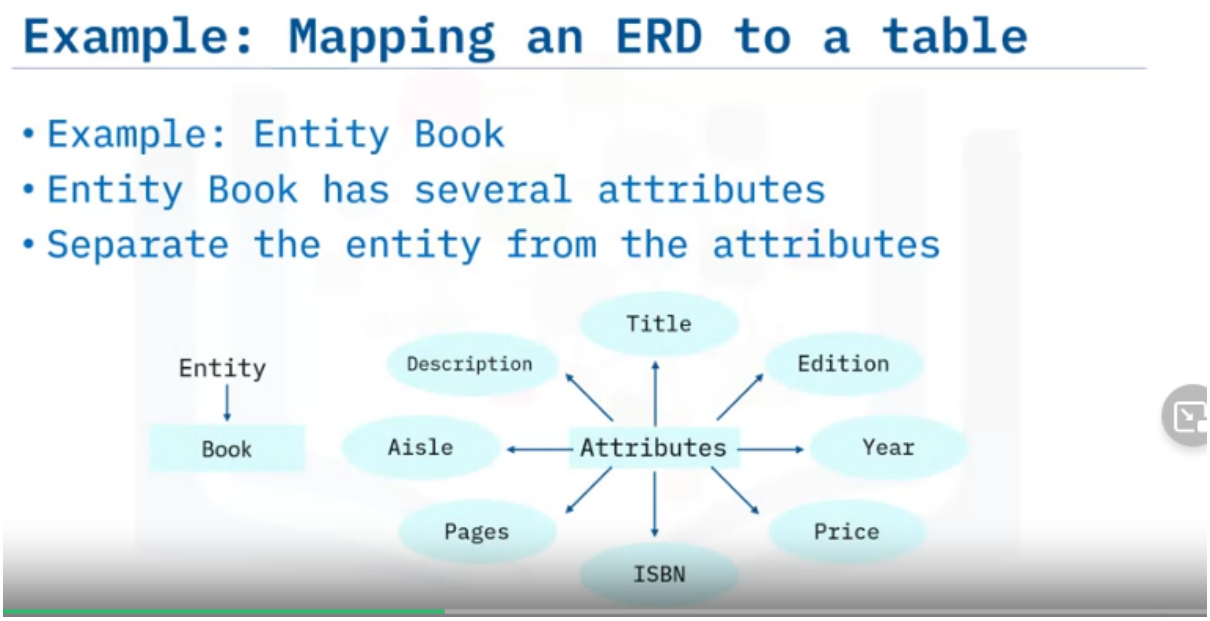
One book can be written by many authors







Mapping entities to tables



Book = table

Attributes = columns

The entity becomes the table

Data types

- character string
  - fixed lenght CHAR(10)
  - variable lenght VARCHAR(20), LONGCHAR
- integer
  - int
  - smallint
  - bigint
- decimal

Relational model concepts

- set of relations
- relation = table
- relation schema

    - name of a relation, name and type of each column (attributes)
- relation instance
    - table made up of rows and columns
    - rows are tuples
    - cardinality refers to number of tuples or rows
    - degree number of columns or attributes

Summary and highlights

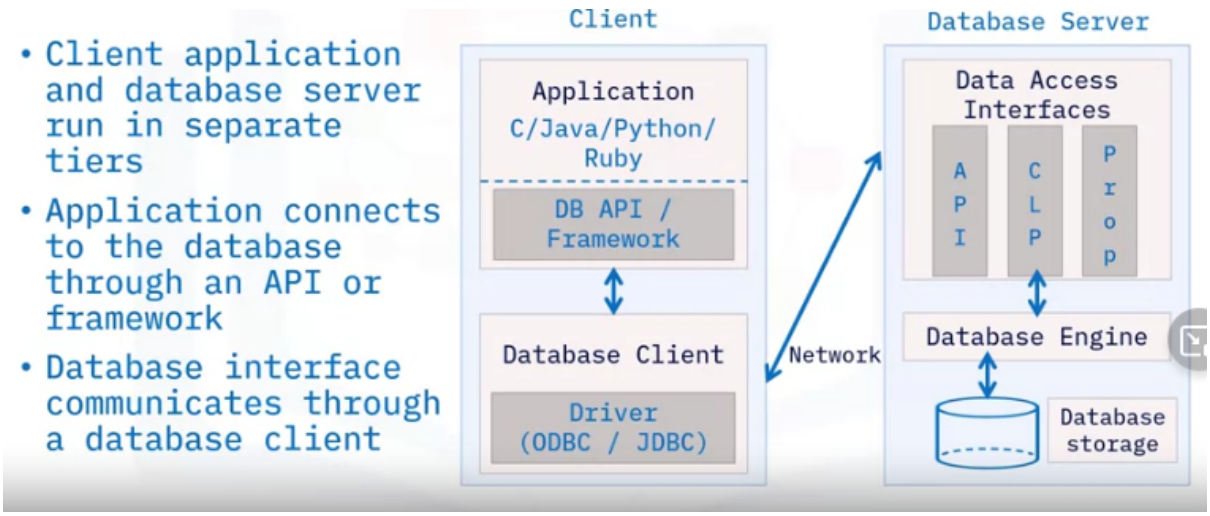
- The relational model is the most used data model for databases because this model allows for logical data independence, physical data independence, and physical storage independence.
- Entities are objects that exist independently of any other entities in the database, while attributes are the data elements that characterize the entity.
- The building blocks of a relationship are entities, relationship sets, and crows foot notations.
- Relationships can be one-to-one, one-to-many, or many-to-many.
- When translating an Entity-Relationship Diagram to a relational database table, the entity becomes the table and the attributes become columns in the table.
- Data types define the type of data that can be stored in a column and can include character strings, numeric values, dates/times, Boolean values and more.
- The advantages of using the correct data type for a column are data integrity, data sorting, range selection, data calculations, and the of standard functions.
- In a relational model, a relation is made up of two parts: A *relation schema* specifying the name of a relation and the attributes and a *relation instance*, which is a table made up of the attributes, or columns, and the tuples, or rows.
- Degree refers to the number of attributes, or columns, in a relation.
- Cardinality refers to the number of tuples, or rows in a relation.

Database architecture

- Local / Desktop

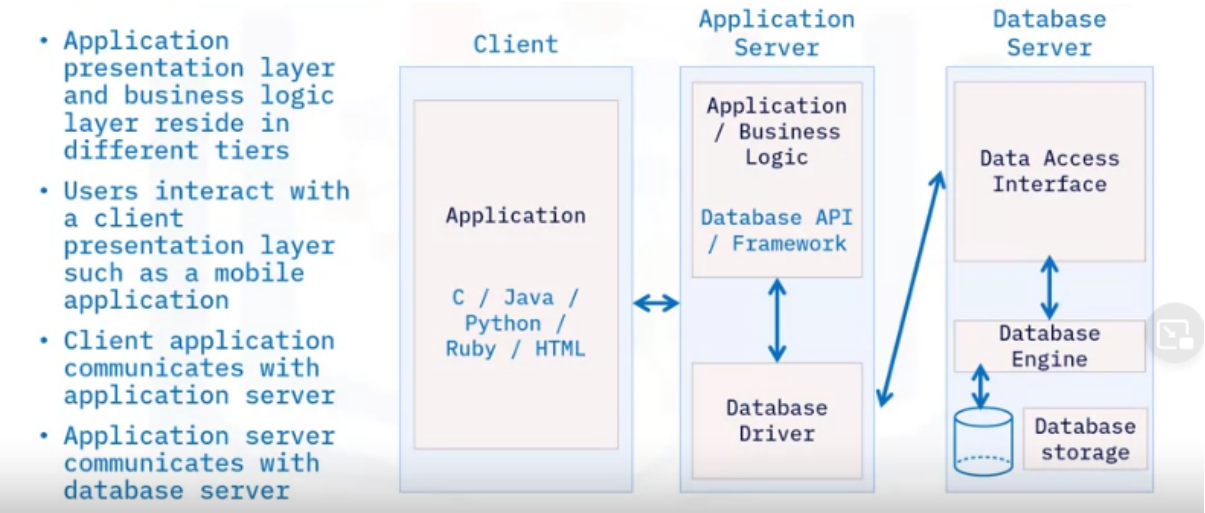
    - resides on the user's system
    - single user enviroment
    - single-tier architecture
- Client / Server
    - resides on a database server
    - users access database from client systems
    - middle-tier (application server layer)
    - typical for production enviroments
- Cloud
    - no need to download of install software
    - users can access the cloud easily
    - access through application server layer or interface in the cloud
    - development, testing and full production enviroments

2-Tier database architecture



The Data Access layer server includes interfaces for different types of clients which can include data industry standard APIs such as **JDBC** and ODBC, Command Line Processor (CLP) interfaces as well vendor specific or proprietary interfaces.

3-Tier database architecture

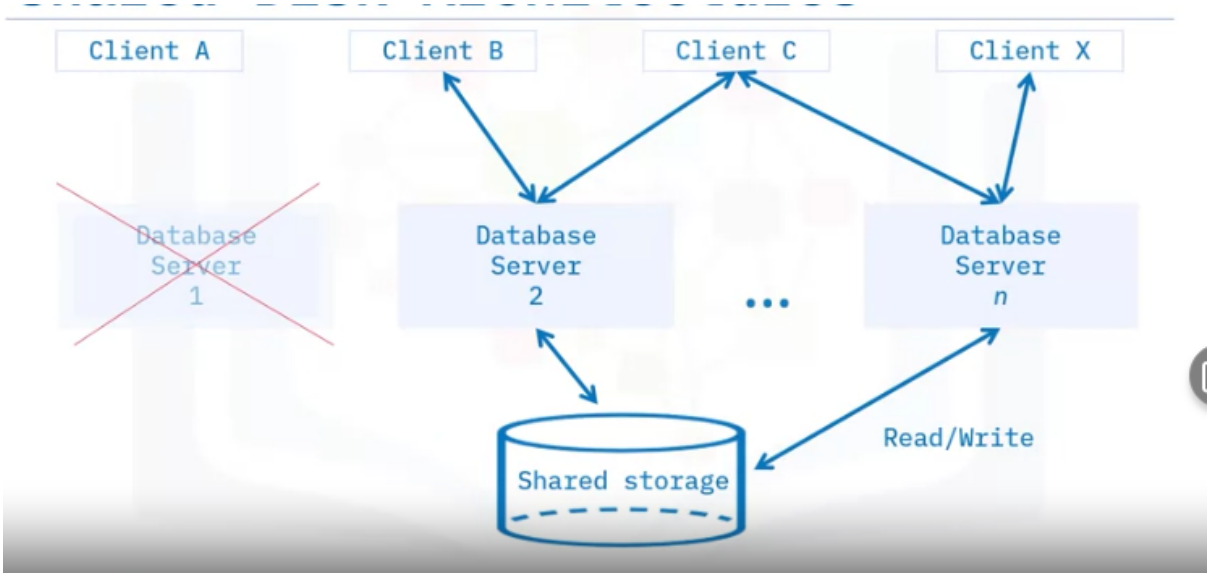


Distributed architecture and clustered databases

- mission critical / large scale workloads
- high availability / high scalability requirements
- databases distributed on a cluster of servers
- shared disk architecture
  - share common storage
- shared nothing architecture
  - replication
  - partitioning

Share disk architectures

- workload processes faster
- In case of a failure, the clients connected to that server can be re-routed to another server
- high availability



Replication

- if the replica is in the same location → high availability replica
- geographically distributed → disaster recovery replica

Partitioning and sharding

- multiple logical partition
- each partition has a subset of the original data
- partitioning
  - very large tables are split across multiple logical partitions
- sharding
  - each shard contains its comput resources

Database usage patterns

Data engineers and database administrators

- GUI or web-based tools
- command line
  - db2 create database sample
  - db2 > connect to sample
- sql scripts or batch files

- APIs

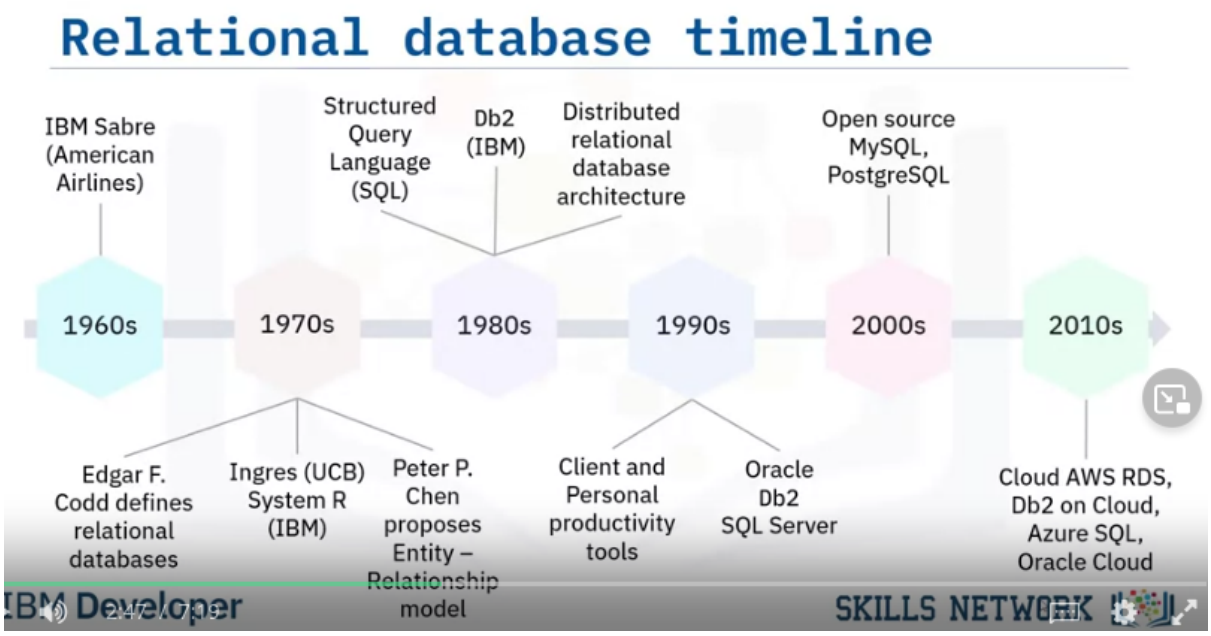
Data scientists and business analysts

- DS tools: jupyter, R studio, zeppelin, SAS, SPSS
- BI tools: excel, cognos, powebi, tableau, microStrategy
- sql query tools

Application developers and programmers

- programming languages: c++, c#, java, javascript, python, rubi, .NET, php
  - they talk to de DB
    - sql interfaces and APIs: JDBC, ODBC, REST APIs
- ORM frameworks
  - example: django

Introduction to Relational Database Offerings



Commercial databases:

- oracle
- microsoft sql server
- ibm db2

Opensource databases:

- mysql
- postgresql
- sqlite

IBM Db2

- db2 database
- db2 warehouse
- db2 on cloud
- db2 warehouse on cloud
- db2 big sql
- db2 event store for z/OS

Features:

- AI-powered funcionallity
  - machine learning algorithms
  - column store
  - data skipping
- Common SQL-Engine
- Support for all data types
- High availability and disaster recovery
- Scalability
- Table partitioning in db2 warehouse

Products

Db2 database:

- on premises
- supported on linuc and windows
- provides

    - performance
    - high availability
    - scalabilitiy
    - resilience

Db2 warehouse:

- on premises
- optimized for OLAP
- provides
    - advanced data analytics
    - massively parallel processing (MPP)
    - machine learning

Db2 on cloud:

- fully managed
- cloud-based
- provides:
    - performance
    - high availability
    - scalability
    - resilience

Db2 warehouse on cloud:

- fully managed
- cloud-based
- provides:
    - advanced data analytics
    - MPP
    - machine learning

Db2 big sql:

- sql-on-hadoop
  - mpp
  - advanced querying
- works with:
  - hadoop HDFS and webHDFS
  - RDBMS
  - NoSQL
- platforms:
  - cloudera data platform
  - ibm cloud park for data

Db2 event store:

- memory-optimized
- analyze streamed data for event-driven applications
- includes IBM watson studio

Db2 for z/OS:

- enterprise data server
- runs on IBM Z providing:
  - avaialbility
  - scalability
  - security
- mission critical

Cloud park for data

- fully intergated and AI platform
- runs on red hat openshift
- enables you to connect to data, organize data, analyze it and infuse AI

MySQL

- MySQL is an object-relational database management system.

- It is a popular low maintenance database and is available in various flavors and editions, including a clustered version for demanding workloads.
- You can run MySQL on many versions of UNIX, as well as Microsoft Windows and Linux and you can write client applications for it using most modern programming languages.
- MySQL uses standard SQL syntax, as well as its own extensions for additional functionality such as the LOAD DATA statement that very quickly reads rows from a text file into a database table
- replication

Storage engines

- InnoDB
  - transactions
  - row-level locking
  - clustered indexes
  - foreign keys
- MyISAM
  - for data warehouse and web applications (mainly read operations)
  - table-level locking
- NDB
  - cluster
  - high availability
  - high redundancy

Clustering options

- InnoDB storage engine
  - primary and secondary dbs
- MySQL cluster edition: NDB storage engine
  - server nodes access data nodes stored in memory
  - multiple nodes → prevents failure

PostgreSQL

- LAPP stack
- extensions PostGIS
- object-relational db (inherital, overloading)
- two node synchronous replication
- multi-master read/write replication → replicate changes with each other
- patitioning
- sharding (horizontal partitions)

Summary and highlights

There are four types of database topology:

- **Single tier.** The database is installed on a user's local desktop.
- **2-tier.** The database resides on a remote server and users access it from client systems.
- **3-tier.** The database resides on a remote server and users access it through an application server or a middle tier.
- **Cloud deployments.** The database resides in the cloud, and users access it through an application server layer or another interface that also resides in the cloud.

In shared disk distributed database architectures, multiple database servers process the workload in parallel, allowing the workload to be processed faster. There are three shared nothing distributed database architectures:

- **Replication.** Changes taking place on a database server are replicated to one or more database replicas. In a single location, database replication provides high availability. When database replica is stored in a separate location, it provides a copy of the data for disaster recovery.
- **Partitioning.** Very large tables are split across multiple logical partitions.
- **Sharding.** Each partition has its own compute resources.

There are different classes of database users, who use databases in different ways:

- Three main classes of users are Data Engineers, Data Scientists and Business Analysts, and Application Developers.

- Database users can access databases through Graphical and Web interfaces, command line tools and scripts, and APIs and ORMs.
- Major categories of database applications include Database Management tools, Data Science and BI tools, and purpose built or off the shelf business applications.
- Relational databases are available with commercial licenses or open source.
- MySQL is an object-relational database that supports many operating systems, a range of languages for client application development, relational and JSON data, multiple storage engines, and high availability and scalability options.
- PostgreSQL is an open source, object-relational database that supports a range of languages for client application development, relational, structured, and non-structured data, and replication and partitioning for high availability and scalability

Semana 2

Types of SQL statements (DDL vs DML)

DDL (data definition language) statements

- define, change or drop data
- define or change objects in tables
- types of sql statements
  - create
  - alter: create new column
  - truncate: detele data in a table but not the table
  - drop: delete table

DML (data manipulation language) statements

- read and modify data
- manipulate data in tables
- CRUD operations (Create, Read, Update and Delete rows)
- types of sql statements

    - insert: insert rows
    - select
    - update: edit rows
    - delete: remove rows

Creating tables

- visual or UI tools
    - db2 on cloud console
    - mysql phpMyAdmin
    - postgreSQL PGAdmin
- create table SQL statement
- administrative APIs

1. select schema to hold the tables
2. fully name of table = schemaName.tableName
3. add columns until you construct the entire table

CREATE TABLE statement

```
CREATE TABLE table_name
  (
  column_name1 datatype optional_parameters,
  column_name2 datatype optional_parameters,
  ...
  )

CREATE TABLE provinces(
  id char(2) PRIMARY KEY NOT NULL,
  name varchar(24)
)

CREATE TABLE author(
  author_id CHAR(2) PRIMARY KEY NOT NULL,
  lastname VARCHAR(15) NOT NULL,
  email VARCHAR(40),
  country CHAR(2)
)
```

ALTER, DROP and TRUNCATE tables

```
ALTER TABLE table_name
  ADD COLUMN column_name_1 datatype
  ...
  ADD COLUMN telephone_number BIGINT
  ...
  ALTER COLUMN email_address SET DATA TYPE VARCHAR(20);
  ...

DROP TABLE table_name;

TRUNCATE TABLE table_name
  IMMEDIATE; //CANNOT BE UNDONE
```

Data movement utilities

- 2 main categories
  - back up and restore
    - preserve all objects in a data base: schemas, tables, user defined types, security, data
  - import and export
    - command line, API, GUI, third party tools
    - files:
      - DEL → delimited ascii, ex, csv
      - ASC → non delimited ascii
      - PC/IXF → PC version of integration exchange format
      - JSON
    - db2 command line:
      - db2 import from *filename* of *fileformat* messages *messagesfile* into *table*
      - db2 export to *filename* of *fileformat* messages *messagesfile* select * from *table*
  - load
    - xml, large objects, user-defined types
    - faster than import, because import performs sql commands
    - doesnt perform as many checks

- preferred option for lading very large datasets
- initiate from command line / API / Visual tool

Loading data

INSERT is not practical for loading thousands of records:

```
INSERT INTO authors (author_id, lastname, firstname, email, city, country)
  VALUES
    (1001, 'Thomas, 'John', 'asd@asd', 'New York', 'USA'),
    ....;
```

- you can load data from different sources and in different formats
- db2 web console load utility supports
  - delimited text files on local computer
  - s3 object storage
  - cloud object storage

1. source → authenticate
2. target
3. define configurable aspects
4. finalize

LAB

Other Contributor(s)

Estimated time needed: 30 minutes In this lab, you will learn how to create tables and load data in Db2. In this lab, you will use IBM Db2 Database. Db2 is a Relational Database

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-SkillsNetwork/labs/Lab%20-%20Create%20Tables%20and%20Load%20Data%20in%20Db2/instructional-labs.md.html?origin=www.coursera.org

Summary and highlights

- DDL statements, including CREATE, ALTER, TRUNCATE, and DROP, are used for defining objects like tables in a database.

- DML statements, including INSERT, SELECT, UPDATE, and DELETE, are used for manipulating data in tables.
- Many Relational Database Management Systems (RDBMS) have schemas that contain tables, views, functions, and other database objects.
- Most RDBMS provide a GUI through which you can create and alter the structure of tables.

You can also create and alter tables by using DDL SQL statements:

- CREATE TABLE. Creates entities (tables) in a relational database and sets the attributes (columns) in a table, including the names of columns, the data types of columns, and constraints (for example, the Primary Key.)
- ALTER TABLE. Changes the structure of a table by adding or removing columns, modifying the data type of columns, and adding or removing keys and constraints.
- DROP TABLE. Deletes a table from a database.
- TRUNCATE TABLE. Removes all rows in a table.

There are utilities that help you to manage the movement of data:

- You use the BACKUP and RESTORE utilities to create and recover copies of entire databases, including all objects like tables, views, constraints, and data.
- You use the IMPORT utility to insert data into a specific table from different formats, such as DEL/CSV, ASC and IXF, and the EXPORT utility to save data from a specific table into various formats, such as CSV.
- You can use the LOAD utilities, instead of INSERT statements, to quickly insert large amounts of data a variety of different data sources into tables.
- The Load Data utility is a simple to use interface in the Db2 Web Console

Database Objects & Hierarchy (Including Schemas)

1. instance
    - logical boundary for databases, objects and configuration
    - unique database server enviroment
    - allows isolation between databases
2. database

- stores, manages and provides access to data
- containts objects like tables, views, indexes
- relational database → relations between tables

3. schema
    - organize database objects
    - naming context
    - database configuration information
    1. tables
    2. constraints
        a. example: an employee number must be unique
    3. indexes
    4. views

Primary and foreign keys

```
CREATE TABLE book(
  book_id INT NOT NULL,
  PRIMARY KEY(book_id);
)
```

A foreign key is a column that contains the same information as the primary key on another table

```
CREATE TABLE copy(
  copy_id INT NOT NULL,
  book_id INT NULL,
  ...
  CONSTRAINT fk_copy_book FOREIGN KEY(book_id)
    REFERENCES book(book_id)
    ON UPDATE NO ACTION
);

Note: ON UPDATE NO ACTION means if any existing row is updated in the foreign key colu
mn of the referencing table (the table containing the foreign key), the update will on
ly be allowed if the new value of the foreign key column exists in the referenced prim
ary key column of the referenced table (the table containing the primary key). Howeve
r, any update on a row of the referenced primary key column of the referenced table is
always rejected if there is the existence of a corresponding row in the referencing fo
reign key column of the referencing table.
```

```
    ON DELETE NO ACTION means if any row in the referenced table (the table containing the primary key) is deleted, that row in the referenced table and the corresponding ro w in the referencing table (the table containing the foreign key) are not deleted.
```

Indexes

```
SELECT title FROM book WHERE unique_book_id = 4;

CREATE UNIQUE INDEX unique_book_id
  ON book(book_id);
```

when you create a primary key, an index is created based on that key, but you can also create your own indexes

Indexes:

PROS:

improved performance of SELECT queries

reduce need to sort data

guaranteed uniqueness of rows

CONS:

each index uses disk space

decreased performance of INSERT, UPDATE, DELETE queries

Normalization

- data duplication leads to inconsistencies
- normalization reduces data duplication
- increases speed of transction
- improves the integrity of data
- most used
  - first normal form
  - second normal form
  - third normal form

First normal form

- each row must be unique
- each cell must contain only a single value
- also called 1NF
- example:

  Let’s look at how you would normalize a simple table.

  In this example, the Book table contains some basic information about books, including title,

  formats, and authors.

  To meet first normal form requirements, each cell must contain a single value, not a list.

  In this example, you can see that all formats of a book are listed in the same cell.

  To normalize this table, you can add an extra row, and split the two formats of Patterns

  of Software into their own row.

  So now you have a row for the paperback version, and a row for the hardback version.

  Each cell in the table now has only one entry, and .so the table is in first normal form.

  BEFORE:

Book_id	Title	Format	Author_name
101	Lean Software Development	Paperback	Mary Poppendieck
201	Facing the Intelligence Explosion	Paperback	David Robson
301	Scala in Action	Hardback	Yehuda Katz
401	Patterns of Software	Hardback, Paperback	Mary Poppendieck
501	Anatomy of LISP	Paperback	Eric Redmond

  AFTER:

Book_id	Title	Format	Author_name
101	Lean Software Development	Paperback	Mary Poppendieck
201	Facing the Intelligence Explosion	Paperback	David Robson
301	Scala in Action	Hardback	Yehuda Katz
401	Patterns of Software	Paperback	Mary Poppendieck
401	Patterns of Software	Hardback	Mary Poppendieck
501	Anatomy of LISP	Paperback	Eric Redmond

Second normal form

- database must be in fist normal form
- create separate tables for sets of values that apply to multiple rows
- also called 2NF
- example

  BEFORE:

Book_id	Title	Format	Author_name
101	Lean Software Development	Paperback	Mary Poppendieck
201	Facing the Intelligence Explosion	Paperback	David Robson
301	Scala in Action	Hardback	Yehuda Katz
401	Patterns of Software	Paperback	Mary Poppendieck
401	Patterns of Software	Hardback	Mary Poppendieck
501	Anatomy of LISP	Paperback	Eric Redmond

  AFTER:

Book_id (Primary Key)	Title	Author_name
101	Lean Software Development	Mary Poppendieck
201	Facing the Intelligence Explosion	David Robson
301	Scala in Action	Yehuda Katz
401	Patterns of Software	Mary Poppendieck
501	Anatomy of LISP	Eric Redmond

Book_id (Foreign Key)	Format
101	Paperback
201	Paperback
301	Hardback
401	Hardback
401	Paperback
501	Paperback

Third normal form

- database must be in first and second normal form
- eliminate columns that do not depend on the key
- also called 3NF
- example

BEFORE:

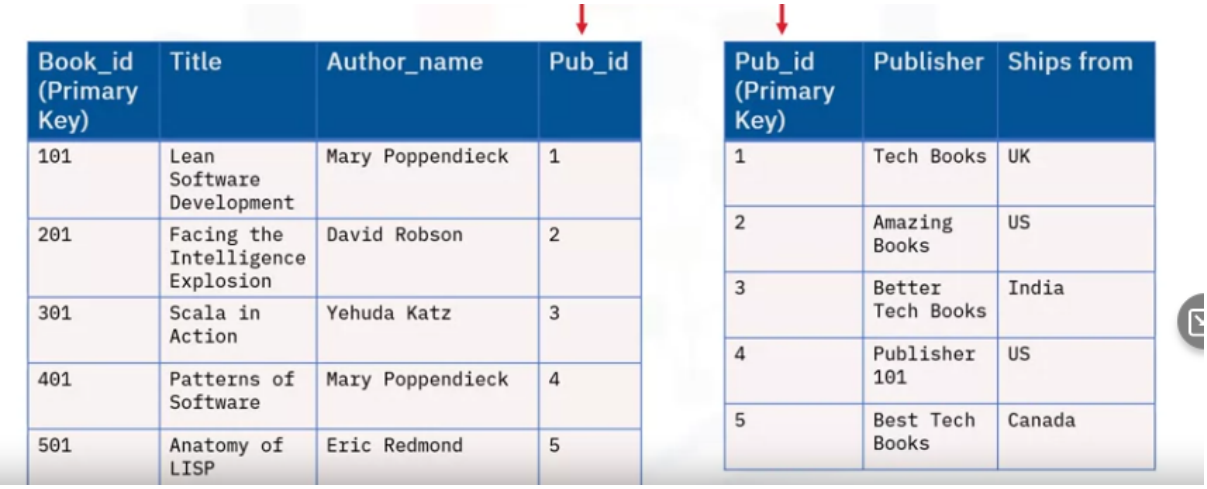
the Ships from data does not depend on the Primary Key.

To meet the requirements of 3NF, you must separate the Publisher and Ships from information

into a Publishers table.

Book_id (Primary Key)	Title	Author_name	Publisher	Ships from
101	Lean Software Development	Mary Poppendieck	Tech Books	UK
201	Facing the Intelligence Explosion	David Robson	Amazing Books	US
301	Scala in Action	Yehuda Katz	Better Tech Books	India
401	Patterns of Software	Mary Poppendieck	Publisher 101	US
501	Anatomy of LISP	Eric Redmond	Best Tech Books	Canada

AFTER:



Book_id (Primary Key)	Title	Author_name	Pub_id
101	Lean Software Development	Mary Poppendieck	1
201	Facing the Intelligence Explosion	David Robson	2
301	Scala in Action	Yehuda Katz	3
401	Patterns of Software	Mary Poppendieck	4
501	Anatomy of LISP	Eric Redmond	5

Pub_id (Primary Key)	Publisher	Ships from
1	Tech Books	UK
2	Amazing Books	US
3	Better Tech Books	India
4	Publisher 101	US
5	Best Tech Books	Canada

There are other higher normal forms.

OLTP → 3NF or BCNF

data is written and read frequently

OLAP → 3NF or 1NF

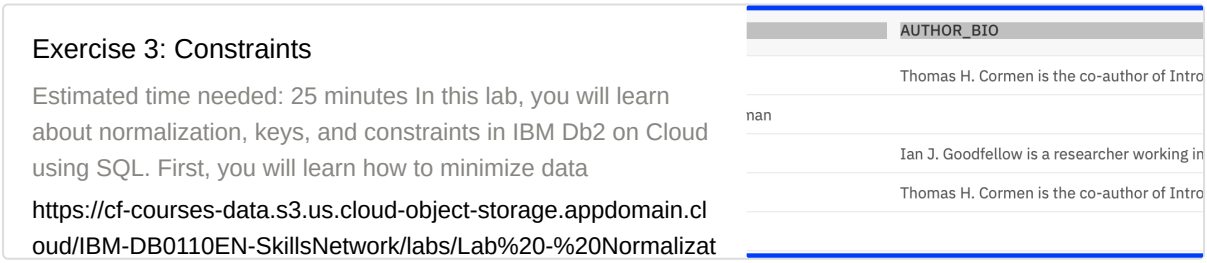
data is mostly read-only

Relational Model Constraints - Advanced

- constraints help implement the business rules
- constraints
  - entity integrity
    - primary key
    - prevent duplicate values
    - no null primary key
  - referential integrity
    - relationships between tables
  - semantic integrity
    - correctness of the meaning of the data

- domain
- null
- check
  - limiting values accepted by an attribute
  - enforce domain integrity

LAB




```
CREATE TABLE BookShop(
  BOOK_ID VARCHAR(4) NOT NULL,
  PRICE_USD DECIMAL(6,2) CHECK(Priice_USD>0) NOT NULL
);
```

Summary and highlights

The objects in a Relational Database Management System (RDBMS) object hierarchy include:

- **Instances.** This is a logical boundary for a database or set of databases where you organize and isolate database objects and set configuration parameters.
- **Relational databases.** This is a set of objects used to store, manage, and access data.
- **Schemas.** A user or system schema is a logical grouping of tables, views, nicknames, triggers, functions, packages, and other database objects. Schemas provide naming contexts so that you can distinguish between objects with the same name.
- **Database partitions.** You can split very large tables across multiple partitions to improve performance.
- **Database objects.** Database objects are the items that exist within the database, such as tables, constraints, indexes, views, and aliases.

Primary key and Foreign Keys have several uses:

- Primary keys enforce uniqueness of rows in a table, whereas Foreign keys are columns in a table that contain the same information as the primary key in another table.
- You can use primary and foreign keys to create relationships between tables. Relationships between tables reduce redundant data and improve data integrity.
- Indexes provide ordered pointers to rows in tables and can improve the performance of SELECT queries, but can decrease the performance of INSERT, UPDATE, and DELETE queries.

Normalization reduces redundancy and increases consistency of data. There are two forms of normalization:

- **First normal form (1NF).** In this form, the table contains only single values and has no repeating groups.
- **Second normal form (2NF).** This form splits data into multiple tables to reduce redundancy.

You can define six relational model constraints:

- **Entity integrity constraint.** Ensures that the primary key is a unique value that identifies each tuple (or row.)
- **Referential integrity constraint.** Defines relationships between tables.
- **Semantic integrity constraint.** Refers to the correctness of the meaning of the data.
- **Domain constraint.** Specifies the permissible values for a given attribute.
- **Null constraint.** Specifies that attribute values cannot be null.
- **Check constraint.** Limits the values that are accepted by an attribute.

Semana 3

Getting started with MySQL

- opensource RDBMS
- mariaDB is fork of mysql
- available in the cloud or for installation

- TOOLS
  - mysql command line
  - mysqladmin
  - mysql workbench desktop applicaion
  - phpMyAdmin web interface

Creating Databases and Tables in MySQL

```
CREATE DATABASE employees;

USE employees;

CREATE TABLE employee_details (firstname VARCHAR(20), lastname VARCHAR(20), startdate
 DATE, salary DECIMAL);

DESCRIBE employee_details; #show in command line the table
```

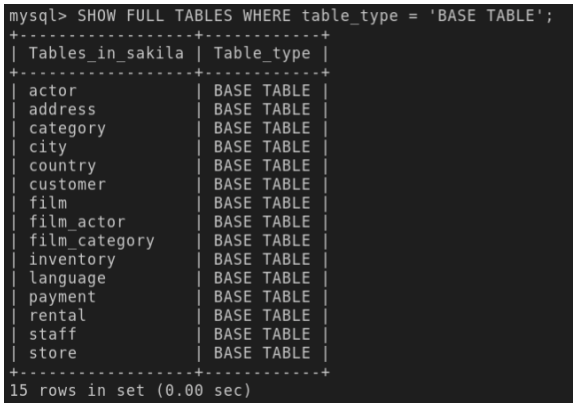
Loading Data in MySQL

```
mysqldump -u root employees > employeesbackup.sql #CREATE BACKUP

mysql -u root resstored_employees < employeesbackup.sql #RESTORE BACKUP

mysql > source employeesbackup.sql #RESTORE BACKUP already in command line

load data infile 'employeesdata.csv' into table employees_details #importing data usin
g load data infile

mysqlimport employees employees_details.csv #importing using mysqlimport. Table name c
omes from the file automatically, they must be exactly the same!!
```

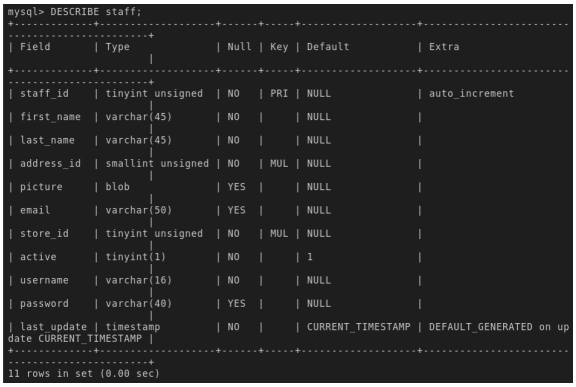
LAB on the command line

```
#to fetch the database from the command line
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-S
killsNetwork/datasets/sakila/sakila_mysql_dump.sql

#start mysql service
start_mysql

#access that will ask for password = MTIxMzMtbWFqb2Nh -> this password was generated w
hen i run the start_mysql command and it was shown on the terminal
mysql --host=127.0.0.1 --port=3306 --user=root --password
```

```
mysql> create database sakila;
mysql> use sakila;
mysql> source sakila_mysql_dump.sql; #restore the mysql dump file
mysql> SHOW FULL TABLES WHERE table_type = 'BASE TABLE'; #show tables
mysql> DESCRIBE staff; #explore the structure of staff
mysql> \q #quits the mysql command prompt session

#dump(backup the staff table
mysqldump --host=127.0.0.1 --port=3306 --user=root --password sakila staff > sakila_st
aff_mysql_dump.sql

#view contents of the dump file from terminal
cat sakila_staff_mysql_dump.sql
```



```
mysql> SHOW FULL TABLES WHERE table_type = 'BASE TABLE';
+-----------------+------------+
| Tables_in_sakila | Table_type |
+-----------------+------------+
| actor           | BASE TABLE |
| address         | BASE TABLE |
| category        | BASE TABLE |
| city            | BASE TABLE |
| country         | BASE TABLE |
| customer        | BASE TABLE |
| film            | BASE TABLE |
| film_actor      | BASE TABLE |
| film_category   | BASE TABLE |
| inventory       | BASE TABLE |
| language        | BASE TABLE |
| payment         | BASE TABLE |
| rental          | BASE TABLE |
| staff           | BASE TABLE |
| store           | BASE TABLE |
+-----------------+------------+
15 rows in set (0.00 sec)
```



```
mysql> DESCRIBE staff;
+-------------+------------------+------+-----+-------------------+-----------------------------
----------------------+
| Field       | Type             | Null | Key | Default           | Extra
                      |
+-------------+------------------+------+-----+-------------------+-----------------------------
----------------------+
| staff_id    | tinyint unsigned | NO   | PRI | NULL              | auto_increment
                      |
| first_name  | varchar(45)      | NO   |     | NULL              |
                      |
| last_name   | varchar(45)      | NO   |     | NULL              |
                      |
| address_id  | smallint unsigned | NO   | MUL | NULL              |
                      |
| picture     | blob             | YES  |     | NULL              |
                      |
| email       | varchar(50)      | YES  |     | NULL              |
                      |
| store_id    | tinyint unsigned | NO   | MUL | NULL              |
                      |
| active      | tinyint(1)       | NO   |     | 1                 |
                      |
| username    | varchar(16)      | NO   |     | NULL              |
                      |
| password    | varchar(40)      | YES  |     | NULL              |
                      |
| last_update | timestamp        | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on up
date CURRENT_TIMESTAMP |
+-------------+------------------+------+-----+-------------------+-----------------------------
----------------------+
11 rows in set (0.00 sec)
```

LAB on phpMyAdmin

```
start_mysql #generates pass = MTIxMzMtbWFqb2Nh, php link = https://majocarbajal-8080.t
heiadocker-5-labs-prod-theiak8s-3-tor01.proxy.cognitiveclass.ai

#that link should be pasted into the address bar in another tab
```

Using Keys and Constraints in MySQL

- primary keys
  - can be created on one clumn or a combination of columns
  - not null
  - unique
  - indexed
- auto increment (A_I)

- foreign keys
- null contraints

Summary & Highlights

MySQL is a free, open-source RDMS that you can download and install on your own systems or access on the Cloud. You can either self-manage a Cloud instance of MySQL or use a managed services provider, including IBM Cloud, Amazon RDS for MySQL, Azure Database for MySQL, or Google Cloud SQL for MySQL.

MySQL includes several options for creating databases and tables, loading and querying data, and importing and exporting data relational databases:

- mysql and mysqladmin command line interfaces. You use these CLIs to run SQL statements.
- MySQL Workbench. A desktop application for designing, developing, and administering MySQL databases.
- phpMyAdmin. An easy to use, third-party web interface for working with MySQL databases.
- API calls.

Using phpMyAdmin, you can:

- Add and modify columns after you create a table.
- Use backup and restore functionality to populate databases.
- Use import and export functionality to populate tables and save their data to files.
- Create primary keys by defining a primary index on one or more columns.
- Use autoincrement to automatically generate sequential numeric data in a column.

When creating foreign keys, you can define ON DELETE and ON UPDATE actions.

MySQL columns are NOT NULL by default.

You can configure a column to only accept unique values.

Getting Started with PostgreSQL

- object-relational database management system

- reliable and flexible
- relational and non-relational
- includes
  - OLTP
  - data analytics
  - geographic information systems
- download and install or cloud
- TOOLS
  - psql command line
  - pgAdmin
    - ERD tool (entity relational diagram)
  - Navicat, DBeaver
  - cloud vendor tools and APIs
- when you first connect to a postgresql database server, it connects automatically to the database DEFAULT

Creating Databases and Loading Data in PostgreSQL

```
CREATE DATABASE employees;

\connect employees;

CREATE TABLE employee_details (firstname VARCHAR(20), lastname VARCHAR(20), startdate
 DATE, salary DECIMAL);

\d employee_details #command line

psql restored_employees < employeesbackup.sql #restore backup

pg_dump employees > employeesbackup.sql #create backup of entire schema and data
```

LAB from command line

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-S
killsNetwork/datasets/sakila/sakila_pgsql_dump.sql
```

```
start_postgres

psql --username=postgres --host=localhost --password

create database sakila;

\connect sakila;

\include sakila_pgsql_dump.sql;

#to restore the database dump file OUTSIDE of the psql commmand prompt:

  wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN
-SkillsNetwork/datasets/sakila/sakila_pgsql_dump.tar

  pg_restore --username=postgres --host=localhost --password --dbname=sakila < sakila_
pgsql_dump.tar

#

\dt; #list all tables names from sakila database

\d store; #explore the structure of the store table

SELECT * FROM store; #retrieve all the records from the store table

\q #quit postgres


#dump/backup the store table from the database
pg_dump --username=postgres --host=localhost --password --dbname=sakila --table=store
 --format=plain > sakila_store_pgsql_dump.sql

#To only dump/backup the table store from the database in non-text format .tar,
pg_dump --username=postgres --host=localhost --password --dbname=sakila --table=store
 --format=tar > sakila_store_pgsql_dump.tar

#view dump file within the termminal
cat sakila_store_pgsql_dump.sql
```

Views

- A view is an alternative way of representing data from one or more tables or other views.
- You can interact with views in the same way as you interact with tables, inserting, updating, and deleting data as required.
- Views are a useful way of limiting access to sensitive data, simplifying data retrieval, and reducing access to underlying tables.
- For example, you could create a view to include the name and email columns from these two tables.

- One view can have records from different tables
- materialized views
  - needs to be refreshed to be updated with the data in the underline tables
  - result set is materialized or saved, for future use
  - cannot insert, update or delete rows
  - can imporove performance
  - limit access to sensitive data

Code to define a view in pgadmin

```
SELECT firstname, lastname, email

FROM employee_details

INNER JOIN employee_contact_info

ON employee_details.empid = employee_contact_info.empid
```

Summary & Highlights

PostgreSQL is an open-source object-relational database management system that you can download and install on your own systems or access on the Cloud.

You can either self-manage a Cloud instance of PostgreSQL or use a managed services provider, including IBM Cloud Databases for PostgreSQL, Amazon RDS, Google Cloud SQL for PostgreSQL, EnterpriseDB cloud, or Microsoft Azure for PostgreSQL.

PostgreSQL includes several options for creating databases and tables, loading and querying data, and importing and exporting data relational databases:

- The psql command line interface. You use this CLI to run SQL statements.
- pgAdmin. A graphical interface to the database server, which is available as a desktop application or as a web application that you can install on your web servers.
- Navicat and Dbeaver. Commercial graphical interface options that you can use to access PostgresSQL, MySQL, and other types of databases.

- Cloud vendor tools and APIs.

Using pgAdmin, you can:

- Use pg_dump to back up databases and psql to restore them.
- Use the Import/Export tool to load data into and export data from tables.

Using views:

- You can use views to limit access to sensitive data and simplify data retrieval.
- Views can be materialized, which means that the view store the result set for quicker subsequent access.
- Materialized views enhance performance because the view is saved and often stored in memory. However, you cannot insert, update, or delete rows in a materialized view, and they must be refreshed before you can see updated data.

Semana 4

Approach to Database Design (Including ERD)

Database design process

- Requirements analysis
  - identify main objects and their relationships
  - attributes
  - interview users and potential users
- Logical design
  - how to organize your data
  - should not specify technical requirements
  - characteristics of the objects become attributes
  - objects become entitites
  - identify many-to-many relationships and resolve them creating many 1-many relationships
  - normalization
- Physical design

- how to implement your logical desgin in yout database
- how your database will look
- data types, naming rules, indexes, constraints

Project

Scenario

In this scenario, you have recently been hired as a Data Engineer by a New York based coffee shop chain that is looking to expand nationally by opening a number of franchise locations. As part of their expansion process, they want to streamline operations and revamp their data infrastructure.

Your job is to design their relational database systems for improved operational efficiencies and to make it easier for their executives to make data driven decisions.

Currently their data resides in several different systems: accounting software, suppliers' databases, point of sales (POS) systems, and even spreadsheets. You will review the data in all of these systems and design a central database to house all of the data. you will then create the database objects and load them with source data. Finally, you will create subsets of data that your business partners require, export them, and then load them into staging databases that use different RDBMS.

In your scenario, you will be working with data from the following sources:

- Staff information held in a spreadsheet at HQ
- Sales outlet information held in a spreadsheet at HQ
- Sales data output as a CSV file from the POS system in the sales outlets
- Customer data output as a CSV file from a bespoke customer relationship management system
- Product information maintained in a spreadsheet exported from your supplier's database

THIS PROJECT WAS DONE IN PGADMIN

staff

staff_id	first_name	last_name	position	start_date	location
1	Sue	Tindale	CFO	08/03/2001	HQ
2	Ian	Tindale	CEO	3/8/2001	HQ
3	Marny	Hermione	Roaster	10/24/2007	WH
4	Chelsea	Claudia	Roaster	3/7/2003	WH
5	Alec	Isadora	Roaster	2/4/2008	WH
6	Xena	Rahim	Store Manager	7/24/2016	3
7	Kelsey	Cameron	Coffee Wrangler	10/18/2003	3
8	Hamilton	Emi	Coffee Wrangler	9/2/2005	3
9	Caldwell	Veda	Coffee Wrangler	9/9/2013	3
10	Ima	Winifred	Coffee Wrangler	10/12/2016	3

sales_outlet

sales_outlet_id	sales_outlet_type	address	city	telephone	postal_code	manager
2	warehouse	164-14 Jamaica Ave	Jamaica	972-871-0402	11432	
3	retail	32-20 Broadway	Long Island City	777-718-3190	11106	6
4	retail	604 Union Street	Brooklyn	619-347-5193	11215	11
5	retail	100 Church Street	New York	343-212-5151	10007	16

sales_transaction

transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	product_id	quantity	price
1	27/04/2019	09:53:55	8	42	0	38	2	3.75
1	27/04/2019	09:53:55	8	42	0	84	1	0.8
2	27/04/2019	08:00:34	8	42	0	51	2	3
3	27/04/2019	09:04:58	8	42	0	33	1	3.5
4	27/04/2019	08:48:32	8	42	8232	27	1	3.5
5	27/04/2019	09:21:40	8	45	8223	24	1	3

customer

customer_id	customer_name	customer_email	customer_since	customer_card_number	birthdate	gender
3001	Kelly Key	Venus@adipiscing.edu	04/01/2017	908-424-2890	29/05/1950	M
3002	Clark Schroeder	Nora@fames.gov	07/01/2017	032-732-6308	30/07/1950	M
3003	Elvis Cardenas	Brianna@tellus.edu	10/01/2017	459-375-9187	30/09/1950	M
3004	Rafael Estes	Ina@non.gov	13/01/2017	576-640-9226	01/12/1950	M
3005	Colin Lynn	Dale@Integer.com	15/01/2017	344-674-6569	01/02/1951	M

product

product_id	product_category	product_type	product_name	description	price
1	Coffee beans	Organic Beans	Brazilian - Organic	It's like Carnival in a cup. Clean and smooth.	18
2	Coffee beans	House blend Beans	Our Old Time Diner Blend	Our packed blend of beans that is reminiscent of the cup of coffee you used to get at a diner.	18
3	Coffee beans	Espresso Beans	Espresso Roast	Our house blend for a good espresso shot.	14.75
4	Coffee beans	Espresso Beans	Primo Espresso Roast	Our premium single source of hand roasted beans.	20.45
5	Coffee beans	Gourmet Beans	Columbian Medium Roast	A smooth cup of coffee any time of day.	15
6	Coffee beans	Gourmet Beans	Ethiopia	From the home of coffee.	21

Entities to create:

- staff
- sales_outlet
- sales_transaction
  - Attributes:
    - transaction_id
    - transaction_date
    - transaction_time

    - sales_outlet_id
    - staff_id
    - customer_id
    - product_id
    - quantity
    - price
- customer
- product

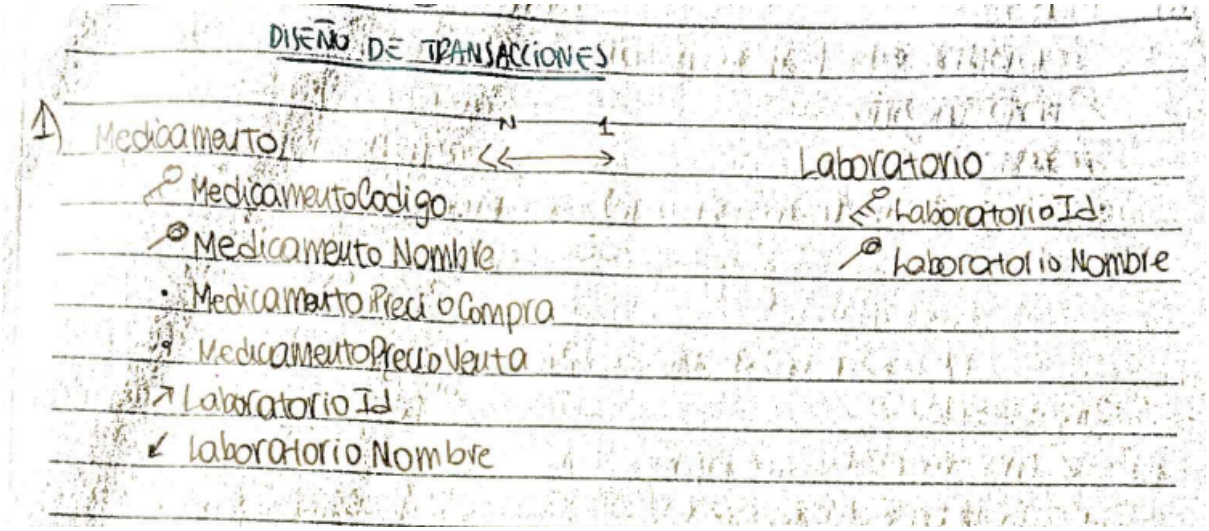
Normalize tables to second normal form:

Review the data in the sales transaction table. Note that the transaction id column does not contain unique values because some transactions include multiple products.

→ to put the transaction table in second normal form, i create sales_detail table to put productid and price

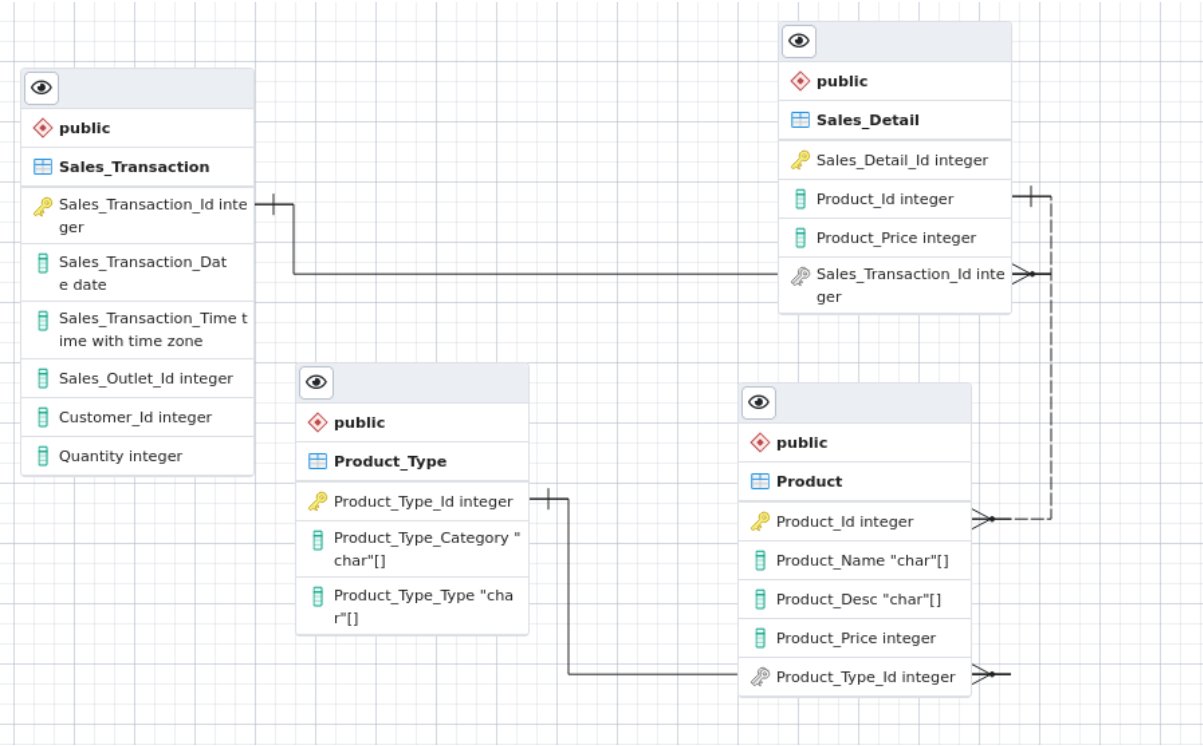
Review the data in the product table. Note that the product category and product type columns contain redundant data.

One to many relationship example:



ERROR unique constrain

This EDR:



Generates this SQL:

```
-- This script was generated by a beta version of the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/new
 if you find any bugs, including reproduction steps.
BEGIN;


CREATE TABLE public."Sales_Transaction"
(
    "Sales_Transaction_Id" integer NOT NULL,
    "Sales_Transaction_Date" date,
    "Sales_Transaction_Time" time with time zone,
    "Sales_Outlet_Id" integer,
    "Customer_Id" integer,
    "Quantity" integer,
    PRIMARY KEY ("Sales_Transaction_Id")
);

CREATE TABLE public."Product"
(
    "Product_Id" integer NOT NULL,
    "Product_Name" "char"[],
    "Product_Desc" "char"[],
    "Product_Price" integer,
    "Product_Type_Id" integer,
```

```
    PRIMARY KEY ("Product_Id")
);

CREATE TABLE public."Sales_Detail"
(
    "Sales_Detail_Id" integer NOT NULL,
    "Product_Id" integer,
    "Product_Price" integer,
    "Sales_Transaction_Id" integer,
    PRIMARY KEY ("Sales_Detail_Id")
);

CREATE TABLE public."Product_Type"
(
    "Product_Type_Id" integer NOT NULL,
    "Product_Type_Category" "char"[],
    "Product_Type_Type" "char"[],
    PRIMARY KEY ("Product_Type_Id")
);

ALTER TABLE public."Sales_Detail"
    ADD FOREIGN KEY ("Sales_Transaction_Id")
    REFERENCES public."Sales_Transaction" ("Sales_Transaction_Id")
    NOT VALID;


ALTER TABLE public."Product"
    ADD FOREIGN KEY ("Product_Type_Id")
    REFERENCES public."Product_Type" ("Product_Type_Id")
    NOT VALID;


ALTER TABLE public."Product"
    ADD FOREIGN KEY ("Product_Id")
    REFERENCES public."Sales_Detail" ("Product_Id")
    NOT VALID;

END;
```

When I run the SQL, I get:

> there is no unique constraint matching given keys for referenced table "Sales_Detail"

Solution:

In PostgreSQL, it’s vitally important that a foreign key references columns that either are a primary key or form a unique constraint. If your query references a column that does not have the UNIQUE constraint, you’ll get the “There is no unique constraint matching given keys for referenced table” error.

I created the tables separatley and then went to the Sales_Detail table and manually created the UNIQUE contraint for Sales_Detail_Id and another one for Product_id

Upload and run this SQL code, it creates all the other tables:

```
-- This script was generated by a beta version of the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/new
 if you find any bugs, including reproduction steps.
BEGIN;


-- The DROP statements have been added to ensure that no errors occur if the script is
mistakenly run more than once
DROP TABLE IF EXISTS public.staff CASCADE;
DROP TABLE IF EXISTS public.sales_outlet CASCADE;
DROP TABLE IF EXISTS public.customer CASCADE;
DROP TABLE IF EXISTS public.sales_detail CASCADE;
DROP TABLE IF EXISTS public.product CASCADE;
DROP TABLE IF EXISTS public.product_type CASCADE;
DROP TABLE IF EXISTS public.sales_transaction CASCADE;
--

CREATE TABLE public.staff
(
    staff_id integer,
    first_name character varying(50),
    last_name character varying(50),
    "position" character varying(50),
    start_date date,
    location character varying(5),
    PRIMARY KEY (staff_id)
);

CREATE TABLE public.sales_outlet
(
    sales_outlet_id integer,
    sales_outlet_type character varying(20),
    address character varying(50),
    city character varying(40),
    telephone character varying(15),
    postal_code integer,
    manager integer,
    PRIMARY KEY (sales_outlet_id)
);

CREATE TABLE public.customer
(
    customer_id integer,
    customer_name character varying(50),
    email character varying(50),
    reg_date date,
    card_number character varying(15),
```

```
    date_of_birth date,
    gender character(1),
    PRIMARY KEY (customer_id)
);

CREATE TABLE public.sales_detail
(
    sales_detail_id integer,
    transaction_id integer,
    product_id integer,
    quantity integer,
    price double precision,
    PRIMARY KEY (sales_detail_id)
);


CREATE TABLE public.product
(
    product_id integer,
    product_name character varying(100),
    description character varying(250),
    product_price double precision,
    product_type_id integer,
    PRIMARY KEY (product_id)
);

CREATE TABLE public.product_type
(
    product_type_id integer,
    product_type character varying(50),
    product_category character varying(50),
    PRIMARY KEY (product_type_id)
);

CREATE TABLE public.sales_transaction
(
    transaction_id integer,
    transaction_date date,
    transaction_time time without time zone,
    sales_outlet_id integer,
    staff_id integer,
    customer_id integer,
    PRIMARY KEY (transaction_id)
);

ALTER TABLE public.sales_transaction
    ADD FOREIGN KEY (staff_id)
    REFERENCES public.staff (staff_id)
    NOT VALID;


ALTER TABLE public.sales_transaction
    ADD FOREIGN KEY (sales_outlet_id)
    REFERENCES public.sales_outlet (sales_outlet_id)
    NOT VALID;


ALTER TABLE public.sales_transaction
```

```
    ADD FOREIGN KEY (customer_id)
    REFERENCES public.customer (customer_id)
    NOT VALID;


ALTER TABLE public.sales_detail
    ADD FOREIGN KEY (transaction_id)
    REFERENCES public.sales_transaction (transaction_id)
    NOT VALID;


ALTER TABLE public.sales_detail
    ADD FOREIGN KEY (product_id)
    REFERENCES public.product (product_id)
    NOT VALID;


ALTER TABLE public.product
    ADD FOREIGN KEY (product_type_id)
    REFERENCES public.product_type (product_type_id)
    NOT VALID;

END;
```

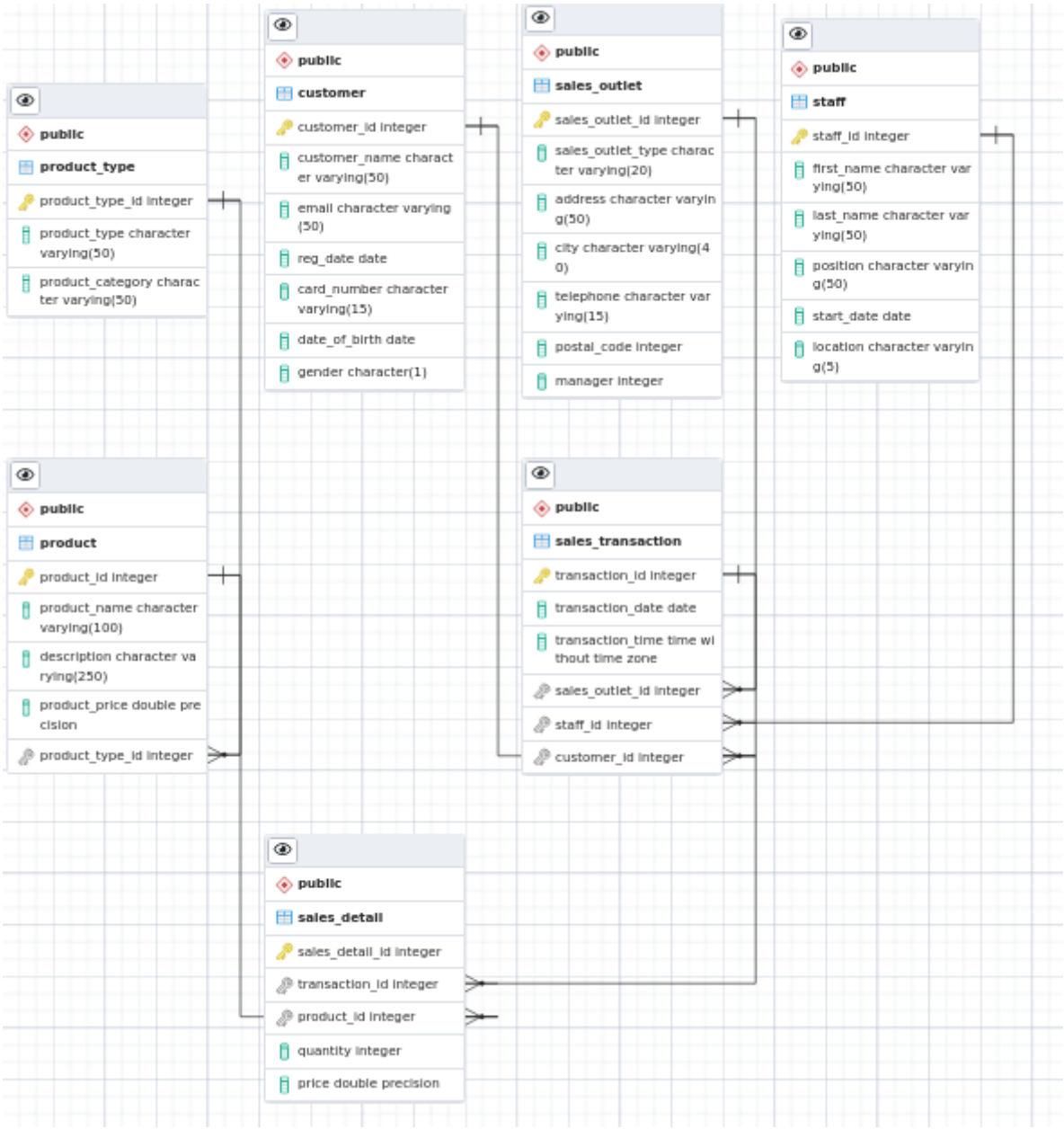
Run the other sql script to populate tables

```
1  SELECT * FROM public.product_type
2  ORDER BY product_type_id ASC LIMIT 100
3
```

Data Output | Explain | Messages | Notifications

	product_type_id [PK] integer	product_type character varying (50)	product_category character varying (50)
1	1	Organic Beans	Coffee beans
2	2	House blend Beans	Coffee beans
3	3	Espresso Beans	Coffee beans
4	4	Gourmet Beans	Coffee beans
5	5	Premium Beans	Coffee beans
6	6	Organic Beans	Coffee beans
7	7	Green beans	Coffee beans
8	8	Herbal tea	Loose Tea
9	9	Black tea	Loose Tea
10	10	Green tea	Loose Tea
11	11	Chai tea	Loose Tea
12	12	Drinking Chocolate	Packaged Chocolate
13	13	Organic Chocolate	Packaged Chocolate
14	14	Drip coffee	Coffee
15	15	Organic brewed coffee	Coffee
16	16	Gourmet brewed coffee	Coffee
17	17	Premium brewed coffee	Coffee

This is the correct EDR, generated from the tables created with the create tables script:



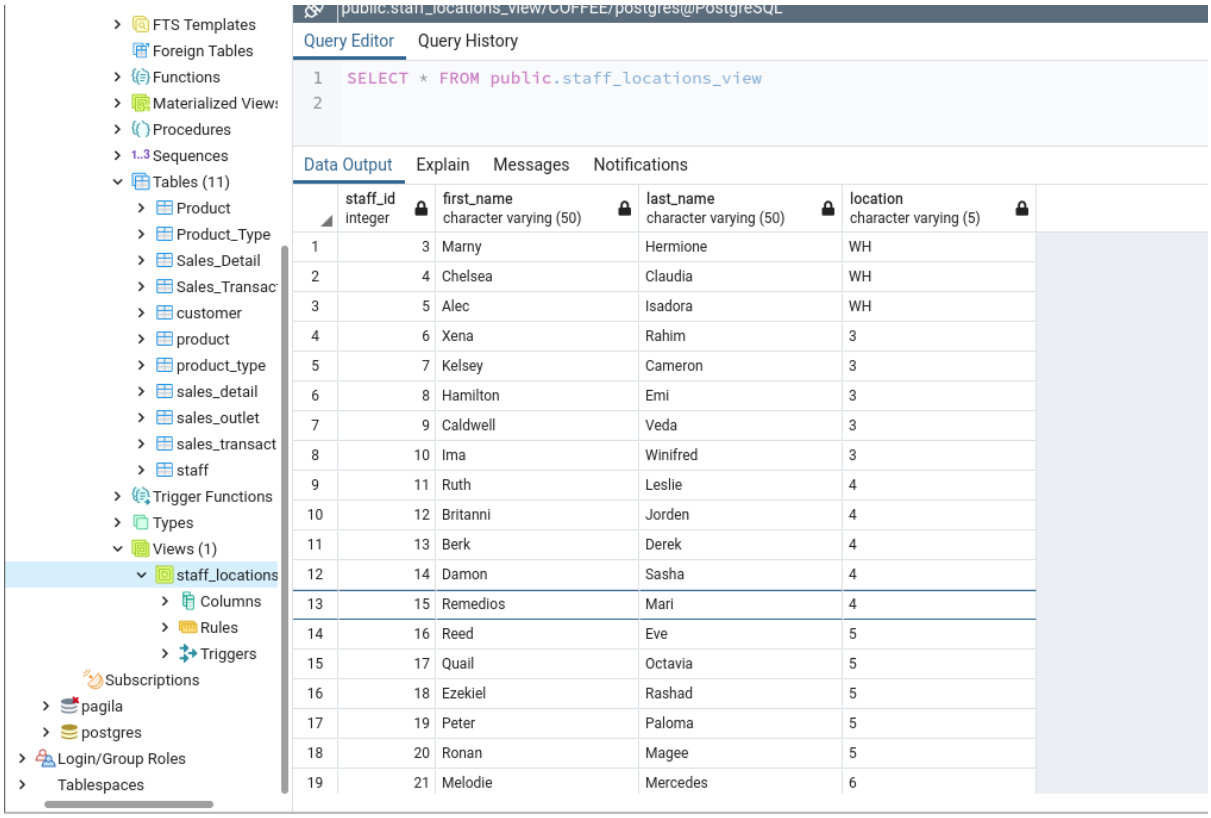
Generate view:

The external payroll company have requested a list of employees and the locations at which they work. This should not include the CEO or CFO who own the company. In this task, you will create a view in your PostgreSQL database that returns this information and export the results to a CSV file.

Create the view and put this in the code tab:

```
SELECT staff.staff_id,
staff.first_name,
staff.last_name,
staff.location
```

```
FROM staff
WHERE "position" NOT IN ('CEO', 'CFO');
```



Create a materialized view and export the data:

A marketing consultant requires access to your product data in their MySQL database for a marketing campaign. You will create a materialized view in your PostgreSQL database that returns this information and export the results to a CSV file.

Go to Materialized-View on the tree menu and use this code:

```
SELECT product.product_name, product.description, product_type.product_category
FROM product
JOIN product_type
ON product.product_type_id = product_type.product_type_id;
```

After that **refresh the materialized view with data**

Import data into a IBM Db2 database

1. create database
2. create columns
3. import staff_locations.csv

Import data into a MySQL database

For this part we are using **phpMyAdmin**

Create coffee_shop_products database and import product_info_m-view.csv into a new table

When I imported the csv, column names were taken as first row so I run this to delete only that row DELETE FROM `product-info` LIMIT 1 , case I didint find an option to take first row as column name.

product_name	description	product_category
Brazilian - Organic	It's like Carnival in a cup. Clean and smooth.	Coffee beans
Espresso Roast	Our house blend for a good espresso shot.	Coffee beans
Primo Espresso Roast	Our premium single source of hand roasted beans.	Coffee beans
Columbian Medium Roast	A smooth cup of coffee any time of day.	Coffee beans
Ethiopia	From the home of coffee.	Coffee beans
Jamacian Coffee River	Ya man, it will start your day off right.	Coffee beans
Guatemalan Sustainably Grown	Green beans you can roast yourself.	Coffee beans
Peppermint	Cool and refreshing to help calm your nerves.	Loose Tea
English Breakfast	The traditional cup to start your day.	Loose Tea
Serenity Green Tea	Mountain grown and harvested at the optimal time.	Loose Tea
Traditional Blend Chai	A traditional blend.	Loose Tea
Morning Sunrise Chai	Fair trade and organic and has a warm finish.	Loose Tea
Spicy Eye Opener Chai	A spicier blend to awaken your taste buds.	Loose Tea
Dark chocolate	This drinking chocolate is smooth and creamy.	Packaged Chocolate
Our Old Time Diner Blend Sm	An honest cup a coffee.	Coffee
Our Old Time Diner Blend Rg	An honest cup a coffee.	Coffee
Our Old Time Diner Blend Lg	An honest cup a coffee.	Coffee
Brazilian Sm	It's like Carnival in a cup. Clean and smooth.	Coffee
Brazilian Rg	It's like Carnival in a cup. Clean and smooth.	Coffee
Brazilian Lg	It's like Carnival in a cup. Clean and smooth.	Coffee

Final quiz

1. Question 1: Data types define the type of data that can be stored in which part of a database table?
   Columns

2. Question 2: Popularity of cloud databases has more than doubled in the past decade. Which cloud service model is driving this?
   SaaS model
3. Question 3: The DBMS on the server in a 2-tier environment includes multiple layers. Which layer is also referred to as the persistence layer?
   Database Storage Layer
4. Question 4: Which NoSQL functionality does PostgreSQL use to store non-hierarchical data?
   HSTORE
5. Question 5: How can the Entity Relationship Diagram (ERD) model be used?
   The ERD model helps you to define entities and their attributes, and map them to tables, and identify the relationships between the tables
6. Question 6: Which of the following sources is NOT supported by the Db2 Web Console for loading data?
   Python Code
7. Question 7: When creating a foreign key, how can you define an action to take if a parent table row is updated?
   NOT create table command and NOT alter table command
8. Question 8: Which of the following is a disadvantage of using Indexes?
   Uses disk space
9. Question 9: Which of the following constraints limits the values accepted by an attribute?
   Check constraint
10. Question 10: What is the difference between system schemas and user schemas?
    System schemas store configuration information and metadata.
11. Question 11: Which MySQL tool can you use to visually design a MySQL database?
    MySQL workbench
12. Question 12: When creating a MySQL database using phpMyAdmin, at which point in the process do you define the length of the data in a column?
    When you define the columns in the table
13. Question 13: What is the maximum size of a data file you can import with phpMyAdmin?

2MB

14. Question 14: On which operating systems can you install PostgreSQL?
    macOs, windows, linux
15. Question 15: Why would you refresh a materialized view in a PostgreSQL database before you use it?
    The materialized view stores the data, so if you need to use the most current data you should refresh the view first.