

Relational Databases

Peninah Waweru
Software Engineer

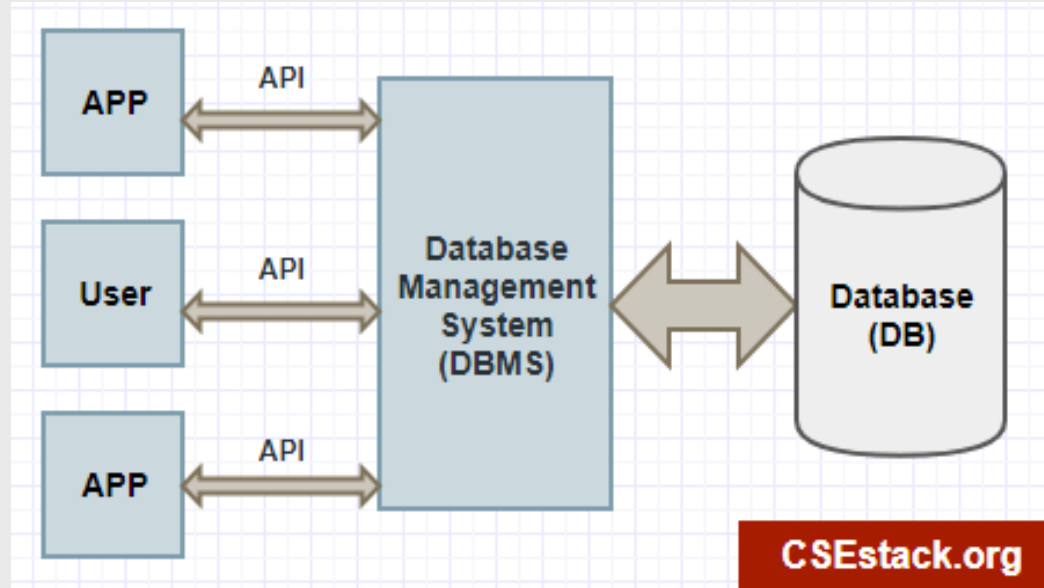
<https://www.youtube.com/watch?v=pXtN4y3O35M&list=PLQkxiw9o8gzjh8o9-e1PWjvFDuO3EPSxn&index=2>

Database

A database is a set of data stored on a computer.

- Relational Database
- Object Database
- Distributed Database
- Graph Database
- Cloud Database
- Operational Database
- Document Oriented Database
- Key-Value Database

Database Management System is systems for creating, manipulating, accessing a database.



Downloads

<https://www.postgresql.org/download/>

<https://www.jetbrains.com/datagrip/download/>

Relational Database

A Relational Database is a data store that allows us to identify and access data in relation to another piece of data in the database.

A Relational Database Management System (RDBMS) is a program that allows you administer a relational database.

Some advantages of RDBMS:

- Data structure
- Multi-user access
- Privileges
- Network access
- Speed
- Maintenance
- Language



Microsoft®
SQL Server®



ORACLE®

DATABASE

Assignment

Part 1

Identify at least 5 Relational Database Management Systems. For each, list all the (20 marks):

- a) Features
- b) Uses
- c) Advantages
- d) Disadvantages

Part 2

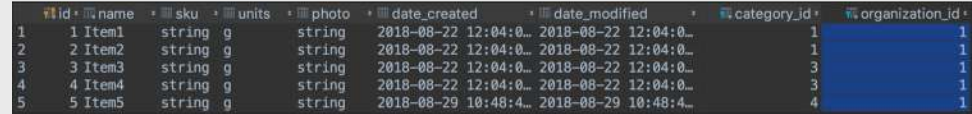
Recommend the one you think is most suitable for a cloud hosted application and provide reasons why. (10marks)

The Relational Model

A data model defines how data is connected to each other and how they are processed and stored.

A relational model is a method of structuring data using relations.

- All data are represented as relations (tables).
- Tables are comprised of rows and columns.
- Rows are unordered.
- A proper relational table contains no duplicate rows.
- Each table has a primary key.
- Most primary keys are single column.
- A table is linked to another by including the other table's primary key (foreign key).



	id	name	sku	units	photo	date_created	date_modified	category_id	organization_id
1	1	Item1	string	g	string	2018-08-22 12:04:0...	2018-08-22 12:04:0...	1	1
2	2	Item2	string	g	string	2018-08-22 12:04:0...	2018-08-22 12:04:0...	1	1
3	3	Item3	string	g	string	2018-08-22 12:04:0...	2018-08-22 12:04:0...	3	1
4	4	Item4	string	g	string	2018-08-22 12:04:0...	2018-08-22 12:04:0...	3	1
5	5	Item5	string	g	string	2018-08-29 10:48:4...	2018-08-29 10:48:4...	4	1

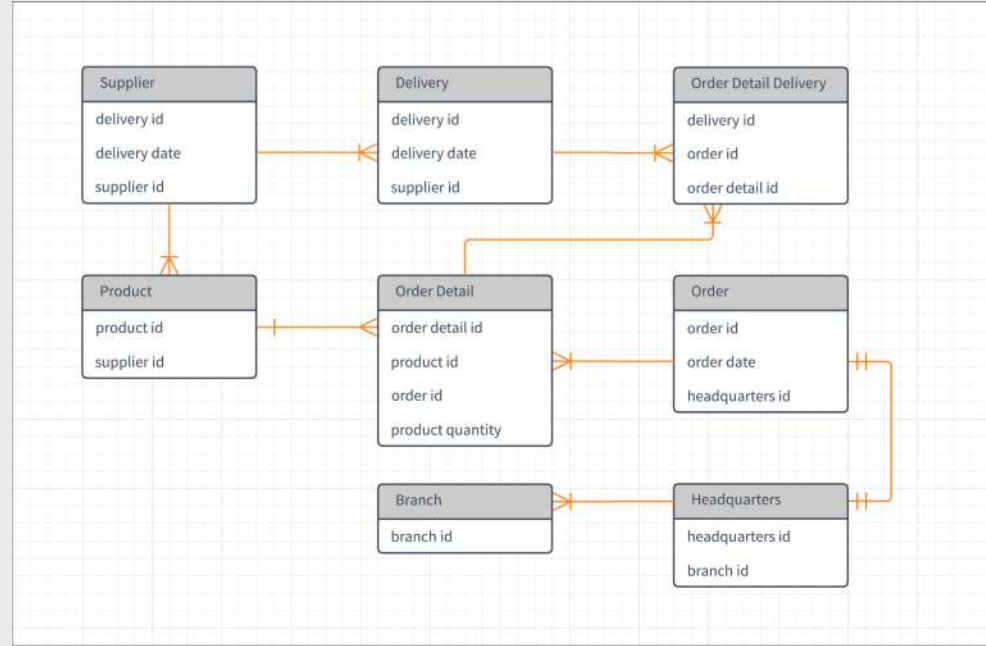
Entity Relationship Diagrams

An **entity** is a class of distinct identifiable objects or concepts.

In a relational model, entities have relationships with one another. These could be either:

- One-to-one
- One-to-many
- Many-to-many

A result of developing an ERD for your database is a normalized database that facilitates access and avoids duplicate data.



Qualities of Good Database Design

1. Reflects real-world structure of the problem.
2. Can represent all expected data over time.
3. Avoids redundant storage of data items.
4. Provides efficient access to data.
5. Supports the maintenance of data integrity over time.
6. Clean, consistent, and easy to understand.

Database Design Rules of Thumb

- Keep data items atomic (e.g., first and last names are separate). Concatenating columns together later on-the-fly is generally easy, but separating them is not. (First Normal Form)
- Define the primary key first. Use a descriptive name (PARCELID, not ID)
- In fact, use descriptive names that give a new user a decent chance of guessing what they mean for *all* your columns! (E.g., use PARCEL_COUNT rather than PACT)
- Use a single column for the primary key whenever possible; multi-column primary keys are appropriate for many-to-many relationships
- Use lookup tables rather than storing long values
- Use numeric keys whenever possible
- Avoid intelligent keys (exception: lookup tables)
- Avoid using multiple columns to represent a one-to-many relationship (e.g., columns such as CHILD1, CHILD2 in a table called PARENT rather than putting the children in a separate table. (First Normal Form)
- For readability, use the primary key name for foreign keys unless the same foreign key is used multiple times in the same table (e.g., state of work and state of residence for a person might both be foreign keys that reference a table of states)

Database Design Rules of Thumb

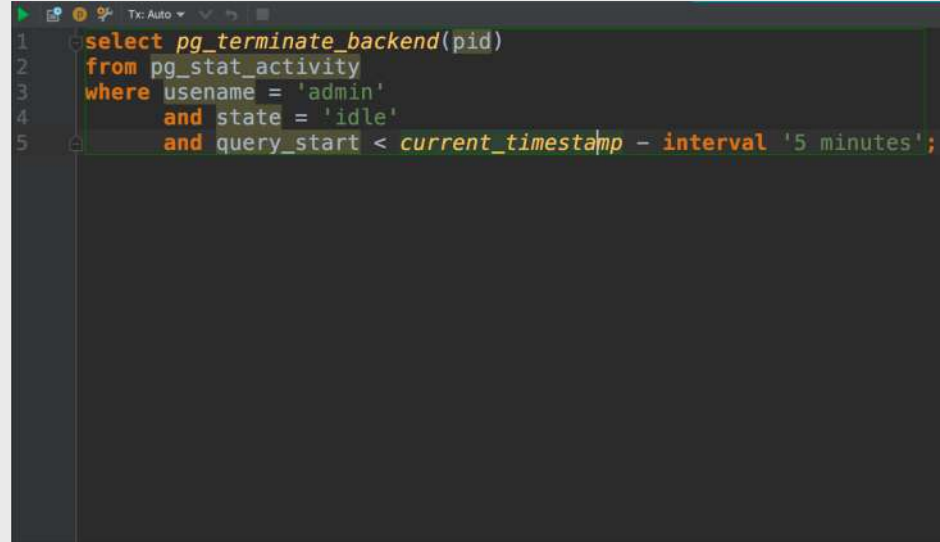
- Do not include two columns whose values are linked together (e.g., county name and county ID) unless one of the columns is the primary key of the table (Third Normal Form)
- Avoid allowing NULL values in columns that have a discrete range of possible values (e.g., integers between 1 and 10, inclusive)
- Avoid using multiple tables with similar structures that represent minor variants on the same entity (e.g., putting Boston parcels and Cambridge parcels in separate tables).
- Plan ahead for transferring data to a different database. For example, you may want to move data from Oracle to DBF, or Microsoft Access to Oracle.
 - Avoid column names with characters with other than UPPER CASE letters (A-Z), digits (0-9), and the underscore (_). Other characters may not be accepted by a database. Some database systems may be case sensitive with regard to column names, while others are not.
 - Keep your column names relatively short. Different databases support different numbers of characters in column names (e.g., 30 for Oracle, 64 for Microsoft Access, 10 for DBF). Try to make column names differ in the first few characters rather than at the end to avoid column name duplication if the names are truncated during the conversion process (e.g., use COL1 and COL2, not LONG_COLUMN_NAME_1 and LONG_COLUMN_NAME_2).
 - *Note that keeping column names short may be at odds with keeping your column names meaningful for neophytes. Be aware that you are making a tradeoff!*
- Remember that these are rules of thumb, *not* absolute laws! Bend the rules if you must but have a justification for your decision. The limitations of a GIS software package often provide a good reason.

Structured Query Language (SQL)

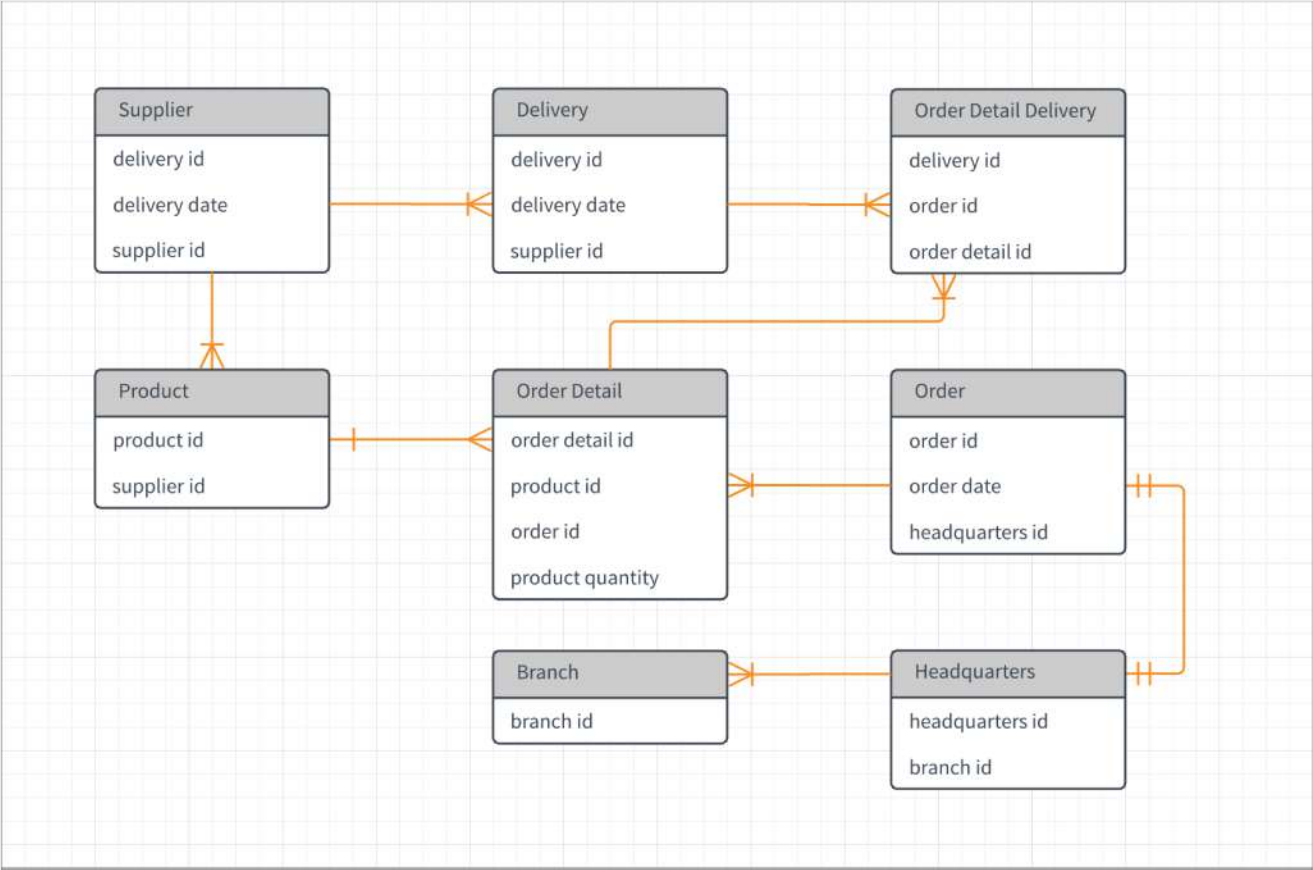
SQL is a database computer language designed for the retrieval and management of data in a relational database.

Popular because:

- Provides access to data in RDMS.
- Users can describe data.
- Users can define the data in a database and manipulate that data.
- Can be embedded in other languages using SQL modules, libraries and pre-compilers.
- Users can create, alter and drop databases and tables.
- Users can create, view, store procedures and functions in the database.
- Users can set permissions on tables, procedures and views.

A screenshot of a code editor window showing a SQL query. The query is written in a dark-themed editor with syntax highlighting. The query is:

```
1 select pg_terminate_backend(pid)
2 from pg_stat_activity
3 where username = 'admin'
4 and state = 'idle'
5 and query_start < current_timestamp - interval '5 minutes';
```



SQL Statements

SELECT column1, column2 **FROM** table_name;

SELECT * FROM table_name;

SELECT column1, column2 **FROM** table_name **FROM** table **WHERE** CONDITION;

SELECT column1, column2 **FROM** table_name **FROM** table **WHERE** CONDITION1 **OR** CONDITION2;

SELECT * FROM table_name **WHERE** column_name **BETWEEN** value1 **AND** value2;

SELECT column1, column2 **FROM** table_name **WHERE** CONDITION **ORDER BY** column_name **ASC**;

SELECT SUM(column_name) **FROM** table_name **WHERE** CONDITION **GROUP BY** column_name;

CREATE TABLE table_name (column1 int, column2 varchar, **PRIMARY KEY**(column1));

DROP TABLE table_name;

TRUNCATE TABLE table_name;

Joins

A JOIN is a means for combining fields from two tables by using values common to each.

Types:

- **INNER JOIN** – returns rows when there is a match in both tables.
- **LEFT JOIN** – returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN** – returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN** – returns rows when there is a match in one of the tables
- **SELF JOIN** – used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- **CARTESIAN JOIN** – returns the Cartesian product of the sets of records from the two or more joined tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

Thank you

Peninah Waweru
Software Engineer

—

peninahw@ke.ibm.com