Welcome to the CoGrammar Databases

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: [Feedback on Lectures]()

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

- ***Timeframe:*** First 2 Weeks
- ***Guided Learning Hours (GLH):*** Minimum of 15 hours
- ***Task Completion:*** First four tasks

✅ Criterion 2: Mid-Course Progress

- ***Guided Learning Hours (GLH):*** **60**
- ***Task Completion:*** 13 tasks

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

- ***Completion:*** All mandatory tasks, including Build Your Brand and resubmissions by study period end
- ***Interview Invitation:*** Within 4 weeks post-course
- ***Guided Learning Hours:*** Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

- ***Final Job or Apprenticeship Outcome:*** Document within 12 weeks post-graduation
- ***Relevance:*** Progression to employment or related opportunity

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Databases

May 2024

Learning Objectives

- Explain and describe the concept of a **database** and its role in software development.
- Identify **different types of databases** and how they are used.
- Learn basic database **terminology and concepts** (tables, columns, rows, keys, etc.).
- Explain the importance of **database normalisation** and apply it to simple database designs.
- Gain **practical experience** using MariaDB and/or SQLite/MySQL to interact with databases.

Learning Objectives

- Introduce the concept of **Object-Relational Mappers** (ORMs).
- Describe the role and importance of **ACID** principles in database transactions.
- Recognise the role of **data lakes**, **data warehouses**, and **data marts** in modern cloud data management.
- Introduce the concept of **data modelling layers** in a cloud environment.
- Interact with SQL for **basic database** interaction

CoGrammar

Poll

How comfortable are you with the concepts of databases and Database Management Systems (DBMS)?

- Not comfortable at all
- Somewhat uncomfortable
- Neutral
- Somewhat comfortable
- Very comfortable

Poll

In the context of managing data, what's the primary responsibility of a Database Management System (DBMS)?

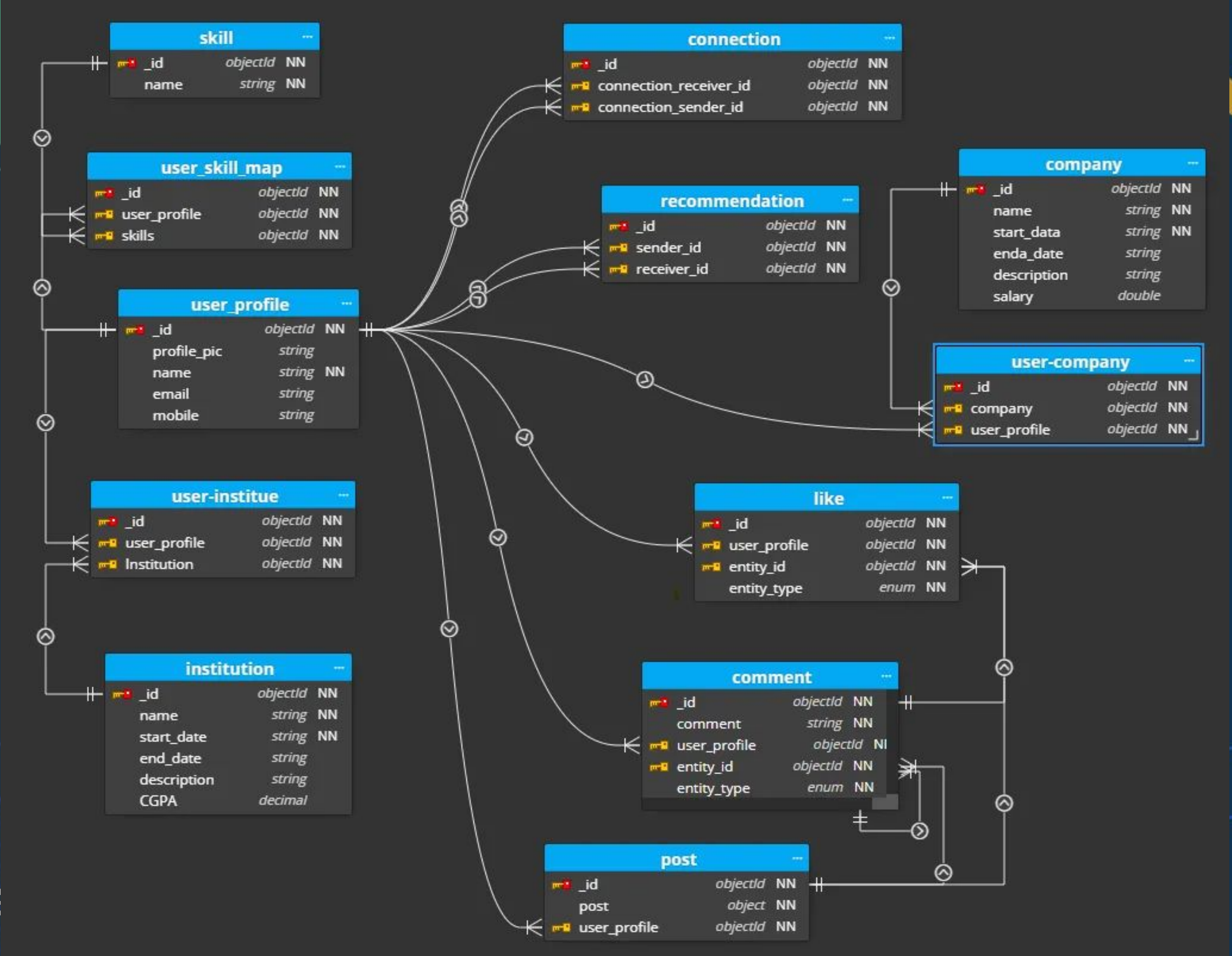
a. Organising, storing, and retrieving data in a structured and efficient manner.
b. Processing complex algorithms for data analysis.
c. Providing a user interface for interacting with various data sources.

Poll

Database Normalisation: When designing a database, what's the main objective of data normalisation?

a. To compress the data size and save storage space.
b. To eliminate data redundancy and improve data integrity
c. To define relationships between different data sets.

Introduction



skill

_id	objectId	NN
name	string	NN

user_skill_map

_id	objectId	NN
user_profile	objectId	NN
skills	objectId	NN

user_profile

_id	objectId	NN
profile_pic	string	
name	string	NN
email	string	
mobile	string	

user-institue

_id	objectId	NN
user_profile	objectId	NN
Institution	objectId	NN

institution

_id	objectId	NN
name	string	NN
start_date	string	NN
end_date	string	
description	string	
CGPA	decimal	

connection

_id	objectId	NN
connection_receiver_id	objectId	NN
connection_sender_id	objectId	NN

recommendation

_id	objectId	NN
sender_id	objectId	NN
receiver_id	objectId	NN

company

_id	objectId	NN
name	string	NN
start_data	string	NN
enda_date	string	
description	string	
salary	double	

user-company

_id	objectId	NN
company	objectId	NN
user_profile	objectId	NN

like

_id	objectId	NN
user_profile	objectId	NN
entity_id	objectId	NN
entity_type	enum	NN

comment

_id	objectId	NN
comment	string	NN
user_profile	objectId	NI
entity_id	objectId	NN
entity_type	enum	NN

post

_id	objectId	NN
post	object	NN
user_profile	objectId	NN

Databases

Picture a **clothing store** in a bustling mall. Keeping track of hundreds of different items, various sizes and colours, along with customer purchases and loyalty program information using manual methods can be a nightmare.

A **database** can streamline this process by storing detailed information about each clothing item (type, size, colour, price), managing inventory levels, and recording customer purchases with loyalty points.

This allows the store to analyse **trends**, identify popular items and sizes, target promotions effectively, and personalise the shopping experience for loyal customers – all contributing to increased sales and better customer satisfaction.

The Power of Data

- **Data** is the raw information that we collect and store.
- It can include **numbers**, **text**, **images**, **videos**, and more.
- Data is everywhere!
- From our online activity to scientific research, data plays a crucial role.

Introducing Databases

- A database is a structured **collection** of **data** organized for easy **access**, **retrieval**, and **management**.
- Picture a well-organized library, but with information you can search and access in seconds.

Introducing Databases

- Databases offer many benefits:
  - **Organization**: Keeps data organized and easy to find.
  - **Efficiency**: Saves time and effort compared to manual data management.
  - **Accuracy**: Reduces errors and inconsistencies in data.
  - **Sharing**: Allows multiple users to access and share data securely.
- There are different types of databases, but today we'll focus on **Relational Databases**, a popular and widely used type. Non Relationals include raw files like csv, or more complex systems like NoSQL

Database Fundamentals

CoGrammar

Unveiling the Database Toolbox

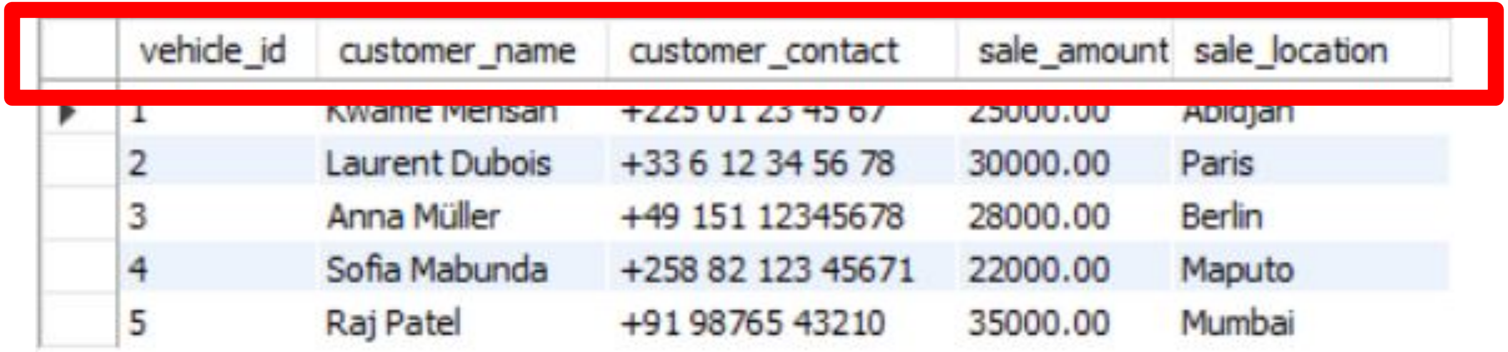
- **Databases** are like powerful digital toolboxes for **storing** and **managing information**.
- Let's explore the essential components that make them work:

- Data Types
- Keys
- Relationships
- Schema
- Tables
- Columns
- Rows
- CRUD
- Join
- Index
- View

Unveiling the Database Toolbox

Schema

The **structure** that defines the **organization** of data in a database, including **tables**, **columns**, **relationships**, **constraints**, and **other properties**, providing a **blueprint** for how data is stored and **accessed**.

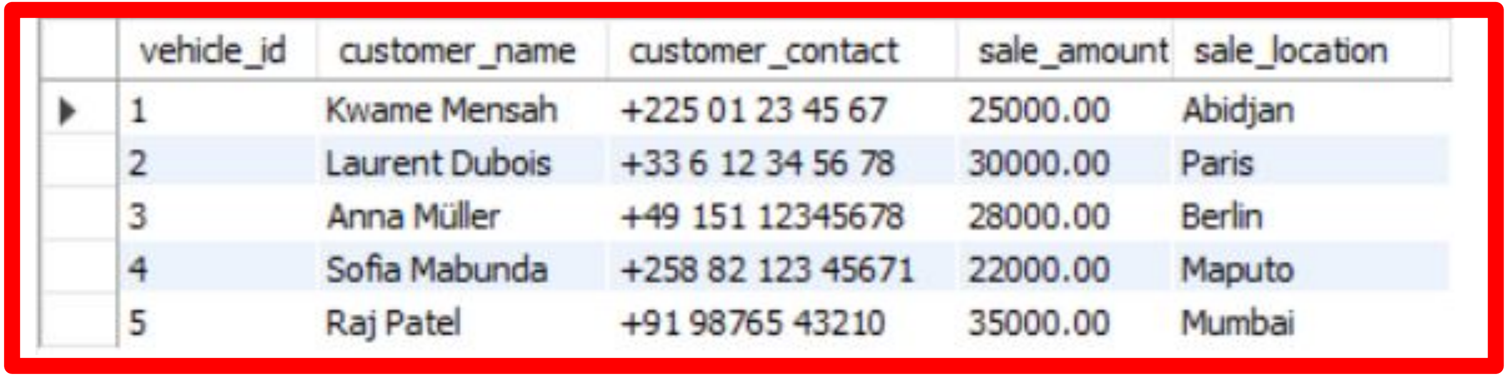


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Unveiling the Database Toolbox

Tables

Single **type** of data within a **table**, such as a person's name or age, and is **organized vertically** within the table structure. (e.g., "Customer Name," "Product Price").

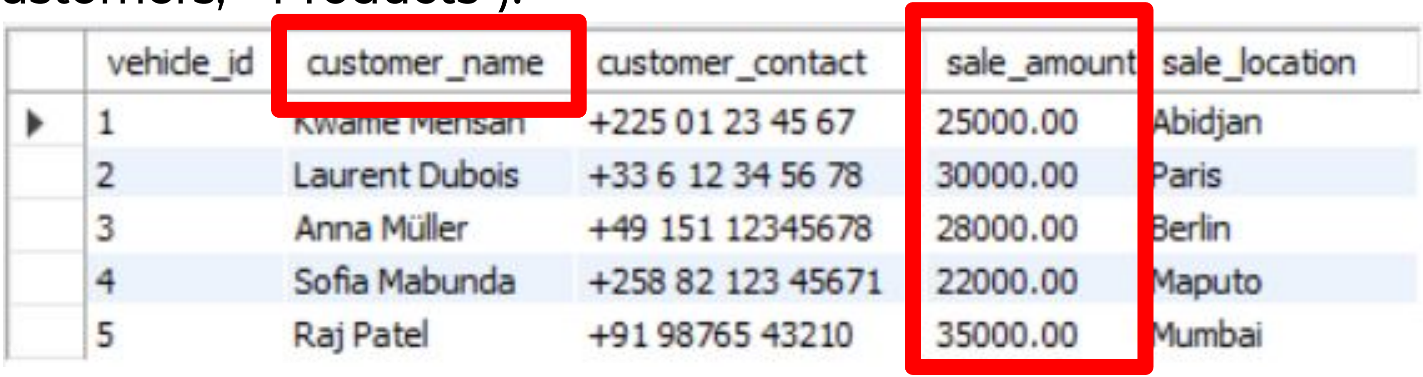


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Unveiling the Database Toolbox

Columns

Structured collection of data organized into **rows** and **columns**, where each row represents a unique record and each column represents a different **attribute** or piece of information. (e.g., "Customers," "Products").

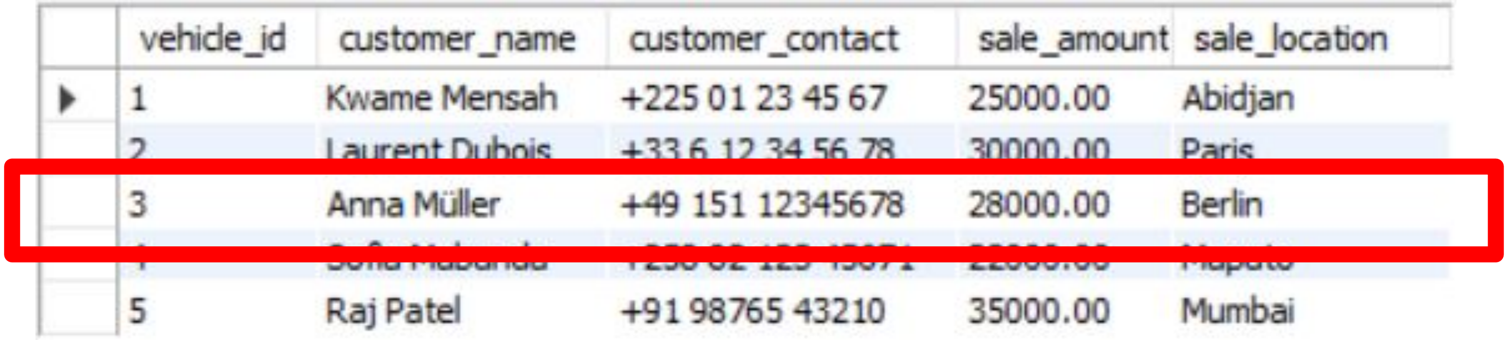


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Unveiling the Database Toolbox

Rows

Also known as a **record**, represents a **single instance of data** within a table, containing **specific values** corresponding to each column of the table. (e.g., a customer's name and address).

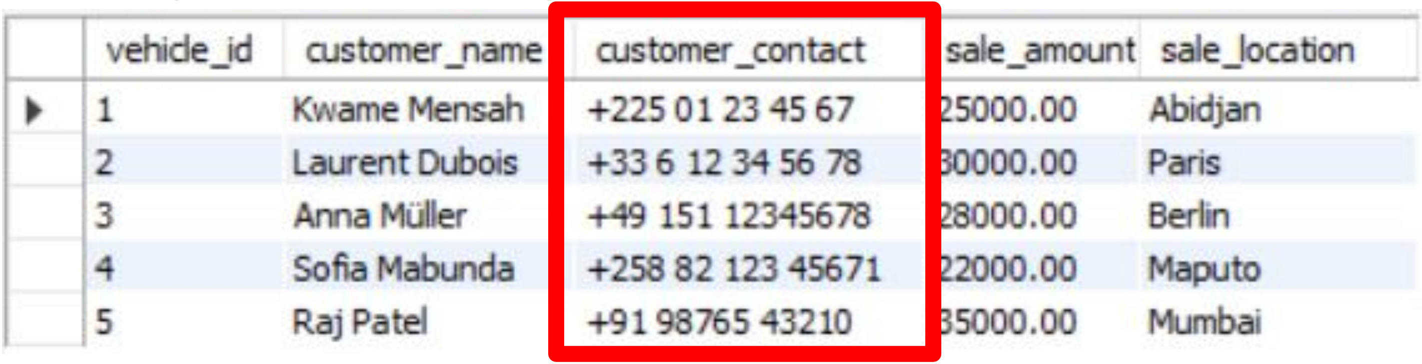


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Unveiling the Database Toolbox

Data Types

Specifies the **kind of data** a **column** can hold, such as **text (VARCHAR, TEXT)**, **numbers (INT, DECIMAL)**, **dates (DATE, TIMESTAMP)**, or **binary data (BIT)**, ensuring **consistency** and facilitating **efficient storage** and **retrieval** of information.

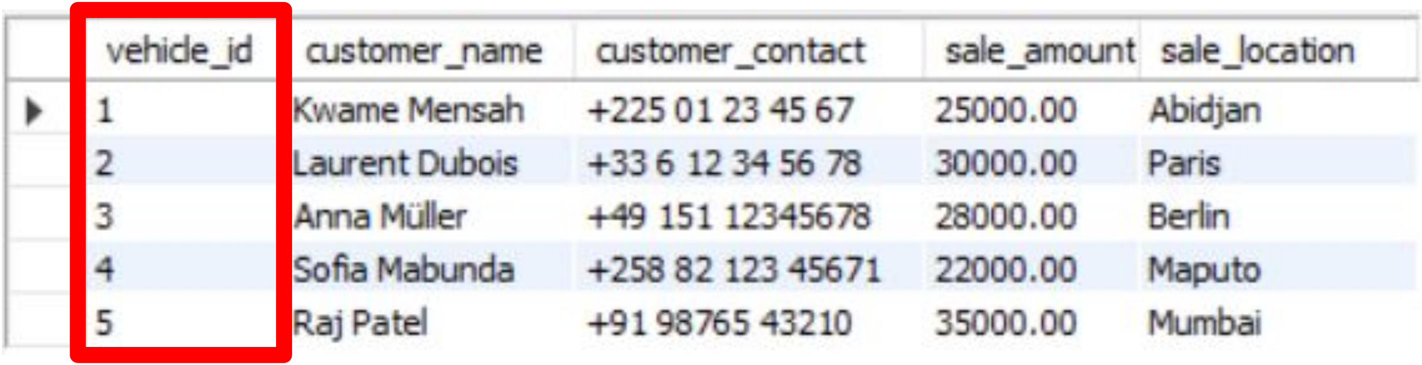


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Keys, Relationships, and CRUD

Primary Keys

A **unique** identifier for each record within that table, ensuring that each row is **uniquely identifiable**. It serves as a **reference point** for other tables and is used to enforce **data integrity** and facilitate efficient data **retrieval**.

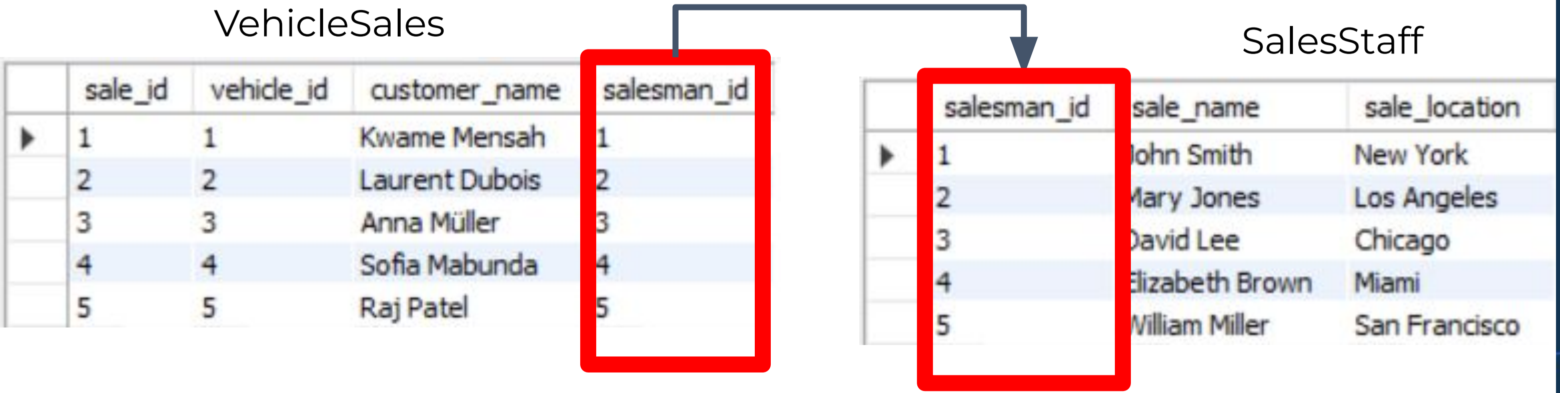


	vehicle_id	customer_name	customer_contact	sale_amount	sale_location
▶	1	Kwame Mensah	+225 01 23 45 67	25000.00	Abidjan
	2	Laurent Dubois	+33 6 12 34 56 78	30000.00	Paris
	3	Anna Müller	+49 151 12345678	28000.00	Berlin
	4	Sofia Mabunda	+258 82 123 45671	22000.00	Maputo
	5	Raj Patel	+91 98765 43210	35000.00	Mumbai

Keys, Relationships, and CRUD

Relationships

Foreign keys in a database are **columns** that establish a **relationship** between tables by referencing the **primary key** of another table



VehicleSales

	sale_id	vehicle_id	customer_name	salesman_id
▶	1	1	Kwame Mensah	1
	2	2	Laurent Dubois	2
	3	3	Anna Müller	3
	4	4	Sofia Mabunda	4
	5	5	Raj Patel	5

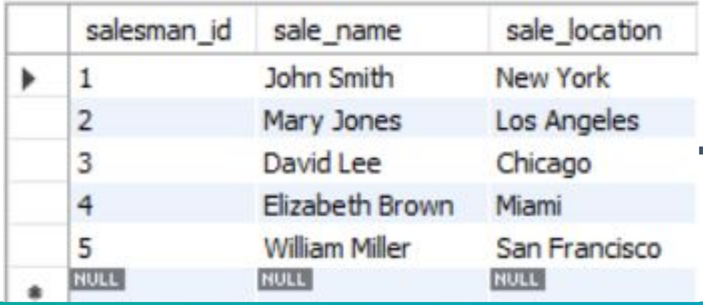
SalesStaff

	salesman_id	sale_name	sale_location
▶	1	ohn Smith	New York
	2	ary Jones	Los Angeles
	3	avid Lee	Chicago
	4	lizabeth Brown	Miami
	5	illiam Miller	San Francisco

Unveiling the Database Toolbox

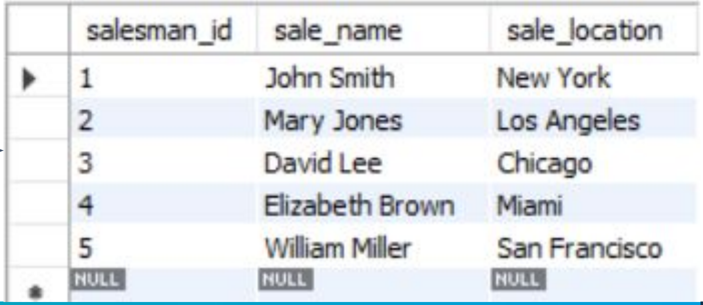
Views

It is a virtual table that represents the result set of a pre-defined query. It acts as a stored SQL query that can be queried like a regular table, simplifying complex queries, providing security by restricting access to certain columns or rows, and encapsulating business logic.



	salesman_id	sale_name	sale_location
▶	1	John Smith	New York
	2	Mary Jones	Los Angeles
	3	David Lee	Chicago
	4	Elizabeth Brown	Miami
	5	William Miller	San Francisco
*	NULL	NULL	NULL

View



	salesman_id	sale_name	sale_location
▶	1	John Smith	New York
	2	Mary Jones	Los Angeles
	3	David Lee	Chicago
	4	Elizabeth Brown	Miami
	5	William Miller	San Francisco
*	NULL	NULL	NULL

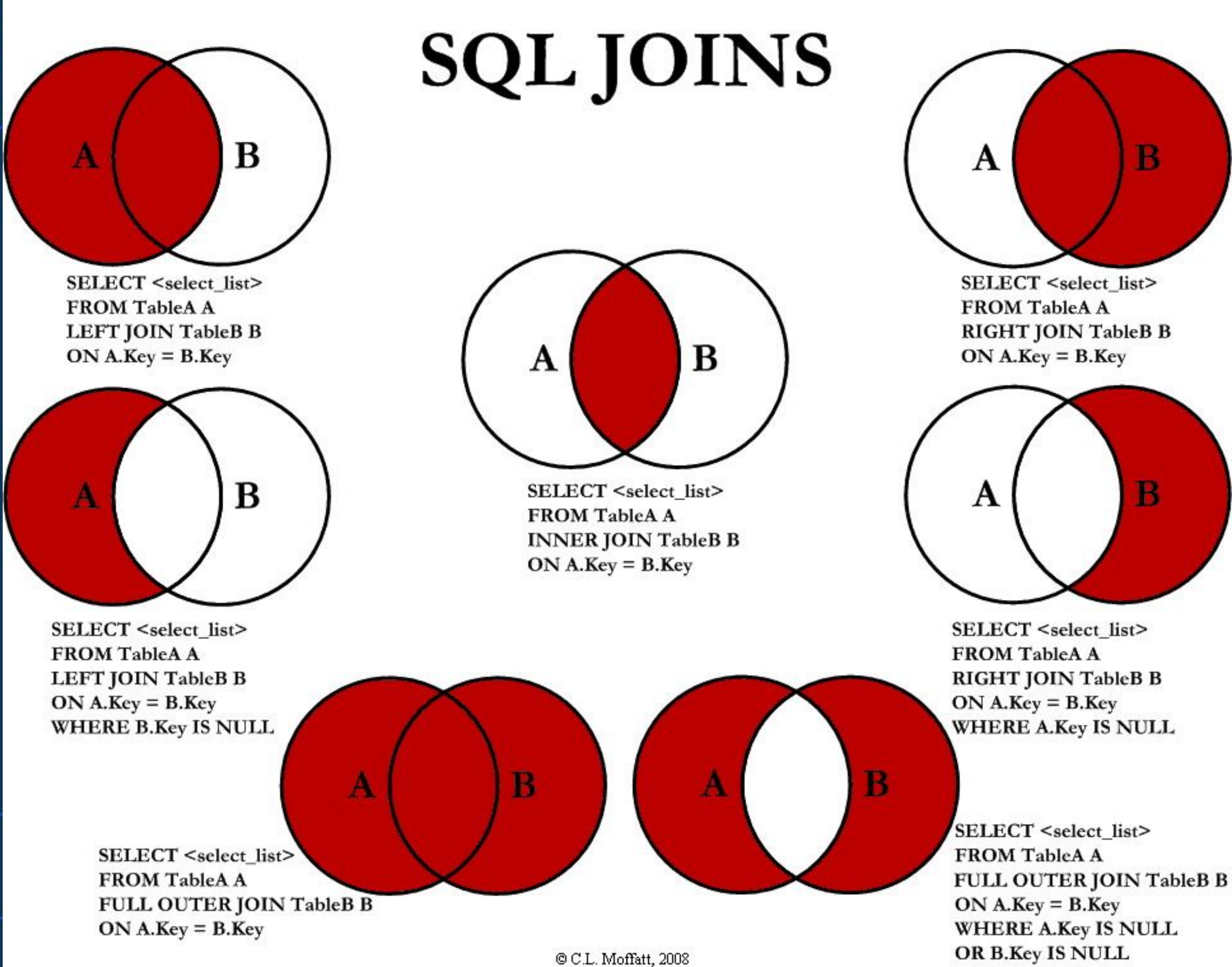
Unveiling the Database Toolbox

Indexes

Data structures that improve the performance of queries by allowing the database system to quickly locate rows based on the values of one or more columns, which is an essential component of the query execution plan.

JOINS

CoGrammar



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

BREAK!

5 mins

Types of Databases

Unveiling the Database Landscape

Relational Databases

- **Structured** data in tables with **rows** and **columns**.
- Enforces **data integrity** with constraints like **primary** and **foreign keys**.
- Supports **relationships** between tables through **foreign keys**.
- Utilizes **SQL** for **querying** and **manipulation**.
- Adheres to **ACID** properties for **transactional reliability** and **consistency**.
- Examples: MySQL, MariaDB, SQLite, PostgreSQL

Unveiling the Database Landscape

Non Relational Databases (NoSQL)

- Flexible **database schema**.
- Various **data models** (document-oriented, key-value, etc.).
- Ability to handle **unstructured** and **semi-structured data**.
- Support for **distributed transactions**.
- Handle **big data** or **large volumes** of data and **real-time analytics**.
- Examples: **MongoDB**, **Cassandra**

CoGrammar

Data Organization and Processing

Row-Oriented Databases

- Data for each **record (row)** is stored **together**.
- This structure is efficient for **retrieving** complete **records** and often used for:
  - **OLTP** (Online Transaction Processing): **Frequent updates** and **retrieval** of **individual** records (e.g., processing sales transactions, managing customer accounts).
- **Examples**: MySQL, SQLite, Amazon Aurora

Data Organization and Processing

Column-Oriented Databases

- Data for each **column** is stored **together**.
- Think of a library filing **books** by **genre** (all history books together, all novels together).
- This is an advanced topic, but offers benefits for:
  - **OLAP** (Online Analytical Processing): Analyzing **large** datasets and identifying trends (e.g., customer buying patterns, market analysis).
- **Examples**: PostgreSQL, Amazon Redshift, Apache Cassandra

Database Properties

CoGrammar

Transactions and Data Reliability

- **Transactions:** A series of database operations treated as a single unit.
- Imagine withdrawing money from your bank account
- All or nothing: either all operations succeed or none of them happen.

Transactions and Data Reliability

ACID Compliance

Ensures data consistency and reliability in transactions (ACID stands for):

- **Atomicity**: All operations are indivisible (all succeed or none happen).
- **Consistency**: Maintains data integrity (transforms database from one valid state to another).
- **Isolation**: Concurrent transactions are isolated from each other (prevents data inconsistencies).
- **Durability**: Committed changes are permanent (ensures data is not lost).

MySQL and MariaDB are ACID-compliant DBMS.

Speaking the Database Language

- Clear and consistent naming conventions are crucial for databases.
- Use descriptive and easy-to-understand names:
  - **customer_name** is better than **cust_nm**
  - **order_date** is clearer than **ord_dt**
- Consistency is key: choose a convention (e.g., lowercase_with_underscores) and stick to it.

Database Normalisation

CoGrammar

Streamlining Your Database: Normalisation

- Imagine storing a customer's address multiple times in a database instead of maintaining it in a separate table, that will lead to **duplicate** information.
- **Data normalisation** involves organizing data in a database to minimize redundancy and dependency.
- **Redundancy**: Repeated data across tables, which can lead to:
  - **Errors**: Updating one value in multiple places can lead to inconsistencies.
  - **Wasted Storage**: Duplicate data takes up unnecessary space.
  - **Inefficiency**: Queries become slower with redundant data to search through.

Normalisation Forms: A Step-by-Step Approach

Normalisation Form	Description
1NF (First Normal Form)	Eliminates repeating groups of data within a table. Each record (row) should be unique and identifiable by a primary key.
2NF (Second Normal Form)	Meets 1NF requirements and eliminates partial dependencies. All non-key attributes must depend on the entire primary key, not just a part of it.
3NF (Third Normal Form)	Meets 2NF requirements and eliminates transitive dependencies. No non-key attribute should depend on another non-key attribute.

Example

Unnormalised

EMPLOYEE	JOB	STATE_CODE	HOME_STATE
E001, Alice, J01	Chef	26	Michigan
E001, Alice, J02	Waiter	26	Michigan
E002, Bob, J02	Waiter	56	Wyoming

Example

1 NF

EMPLOYEE_ID	NAME	JOB_CODE	JOB	STATE_CODE	HOME_STATE
E001	Alice	J01	Chef	26	Michigan
E001	Alice	J02	Waiter	26	Michigan
E002	Bob	J02	Waiter	56	Wyoming

Example

2 NF

roles table

EMPLOYEE_ID	JOB_CODE
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

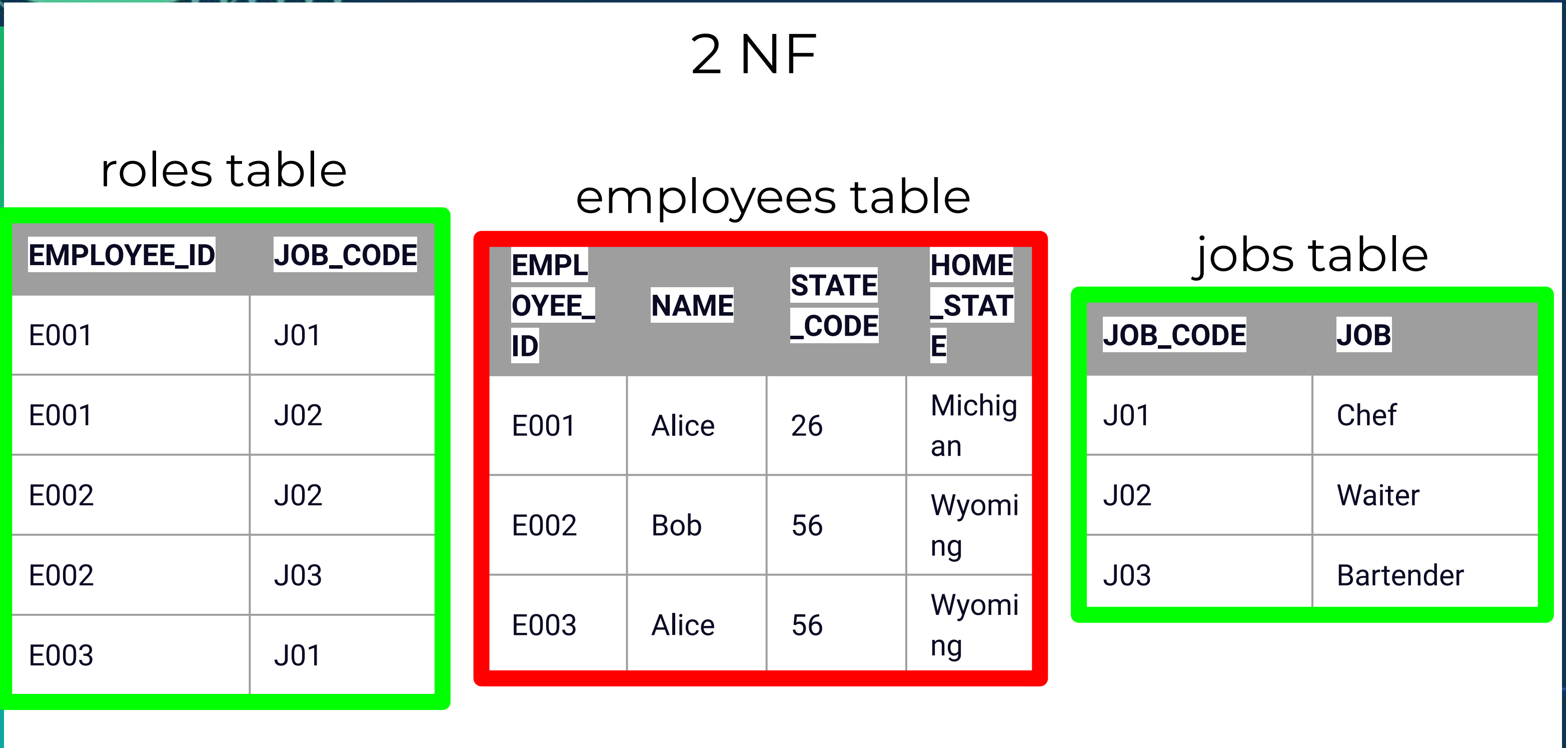
employees table

EMPLOYEE_ID	NAME	STATE_CODE	HOME_STATE
E001	Alice	26	Michigan
E002	Bob	56	Wyoming
E003	Alice	56	Wyoming

jobs table

JOB_CODE	JOB
J01	Chef
J02	Waiter
J03	Bartender

Example



2 NF

roles table

EMPLOYEE_ID	JOB_CODE
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

employees table

EMPLOYEE_ID	NAME	STATE_CODE	HOME_STATE
E001	Alice	26	Michigan
E002	Bob	56	Wyoming
E003	Alice	56	Wyoming

jobs table

JOB_CODE	JOB
J01	Chef
J02	Waiter
J03	Bartender

Example

3 NF

roles table

EMPLOYEE_ID	JOB_CODE
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

jobs table

JOB_CODE	JOB
J01	Chef
J02	Waiter
J03	Bartender

Example

3 NF

employees table

EMPLOYEE_ID	NAME	STATE_CODE
E001	Alice	26
E002	Bob	56
E003	Alice	56

states table

STATE_CODE	HOME_STATE
26	Michigan
56	Wyoming

The Benefits of Normalisation

Normalization is a powerful tool for:

- **Data Integrity:** Minimizes errors and inconsistencies.
- **Efficiency:** Improves query performance and reduces storage requirements.
- **Maintainability:** Makes databases easier to manage and update.

BREAK!

5 mins

CoGrammar

SQL Fundamentals

CoGrammar

Unveiling the Database Language

- Imagine a vast vault holding valuable information.
- SQL (Structured Query Language) is the key to accessing that data.
- SQL is a standardized language for interacting with **relational databases**.
- We use SQL statements to:
    - **Create** and **manage** database structures (tables).
    - **Retrieve**, **update**, and **delete** data within those structures.

The Building Blocks of SQL

Category	Description	Example
DDL (Data Definition Language)	**Creates** and **modifies** the structure of a database (**tables**, **views**, **indexes**).	* **CREATE** TABLE Customers (CustomerID INT PRIMARY KEY, Name VARCHAR(255), Email VARCHAR(255))
DML (Data Manipulation Language)	**Manages** data within the database (**insert**, **update**, **delete**).	* **INSERT** INTO Customers (Name, Email) VALUES ('John Doe', 'john.doe@email.com')
DCL (Data Control Language)	Controls **access privileges** for users within the database (**grant**, **revoke**).	* **GRANT** SELECT ON Customers TO reports_user

CoGrammar

Unveiling Your Data: The SELECT Query

Clause	Description	Example
SELECT	Specifies the columns you want to retrieve.	SELECT Name, Email
FROM	Identifies the table containing the data.	FROM Customers
WHERE (Optional)	Filters data based on a specific condition.	WHERE Email LIKE '%@gmail.com'
ORDER BY (Optional)	Sorts the results based on a specified column.	ORDER BY Name ASC
GROUP BY (Optional)	Groups rows based on a shared value in one or more columns.	GROUP BY Country
HAVING (Optional)	Filters groups based on a condition applied to aggregate functions.	HAVING COUNT(*) > 10

In Organised projects

CoGrammar

Navigating the Data Deluge: Storage Solutions

- The amount of data organizations generate is exploding.
- We need efficient ways to store and manage this data.
- Different storage solutions cater to various data needs.

Data Lakes: A Reservoir for All Your Data

- **Data lakes** are large, central repositories for storing raw, unstructured, and semi-structured data.
- They offer high scalability and flexibility, accommodating diverse data sources (sensor data, social media feeds, etc.).
- Ideal for exploratory analysis and uncovering hidden patterns within the data.

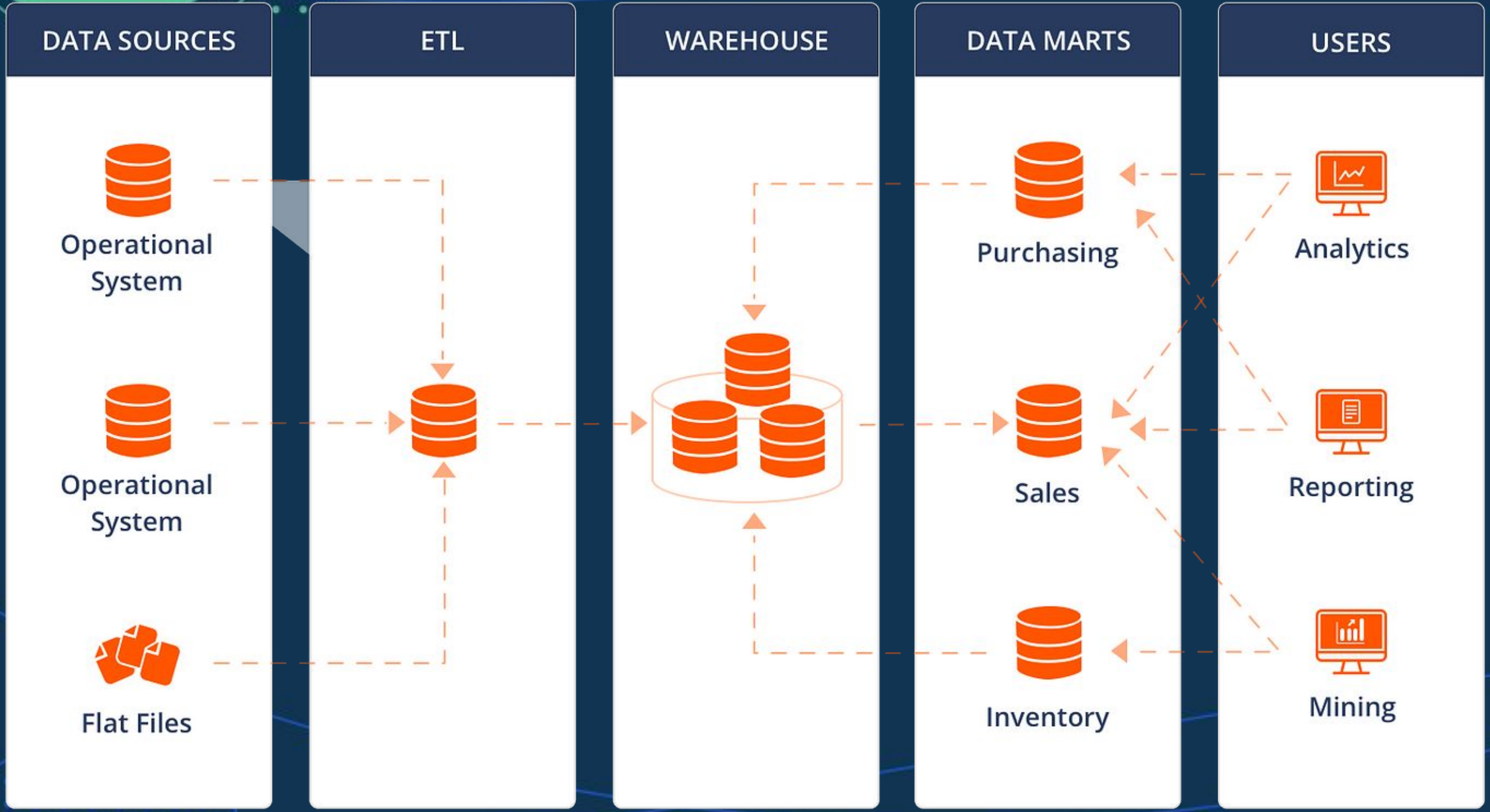
Data Warehouses: The Organized Analyst's Haven

- **Data warehouses** are subject-oriented, centrally managed repositories designed for analytical workloads.
- Data is pre-processed, cleaned, and transformed into a consistent, structured format.
- Optimized for querying, reporting, and data analysis (often integrated with data lakes).

Data Marts: A Tailored Approach to Data Analysis

- **Data marts** are focused subsets of data warehouses, tailored for specific business needs (e.g., marketing, sales).
- Offer a smaller, more manageable data footprint compared to data warehouses.
- Can be deployed more quickly and are often less expensive to maintain.

Example of Data Architecture



CoGrammar

Practical

CoGrammar

Practical

Use https://www.db-fiddle.com/ to create a simple database that will house 2 tables

- **Employees**
  - Employee ID
  - First name
  - Last name
- **Products**
  - Product ID
  - Product name

Query all the table using SELECT, WHERE, and ORDER by statement.

Final Assessment

CoGrammar

Poll

SQL (Structured Query Language): SQL is primarily used for what purpose in a relational database?

a. To design and define the structure of the database
b. To Create, Read, Update and Delete data within the database
c. To manage user access permissions for the database

CoGrammar

Poll

Data Warehouses: How do data warehouses typically differ from operational databases like OLTP systems?

a. Data warehouses are designed for real-time transaction processing, while OLTP systems focus on historical data analysis.
b. Data warehouses store detailed and integrated data from various sources, while OLTP systems manage current operational data.
c. Data warehouses are more scalable and handle larger datasets compared to OLTP systems.

Poll

Indexes: When designing a relational database, what is the primary purpose of creating indexes on specific columns?

a. To encrypt sensitive data stored within the database.
b. To improve the speed and efficiency of data retrieval based on those columns.
c. To define relationships between tables based on shared data points.

Lesson Conclusion and Recap

CoGrammar

Lesson Conclusion and Recap

- **Databases** store data in a structured format using tables with rows and columns.
- **SQL** (Structured Query Language) is the standard language for interacting with relational databases.
- **Data Modelling** defines the structure of a database to represent real-world entities and relationships.
- **Database Normalisation** is a process of reducing data redundancy within a database, improving efficiency and data integrity.
- We explored the basics of SQL categories: **DDL** (Data Definition Language) and **DML** (Data Manipulation Language).

Homework or Follow-up Activities

CoGrammar

Homework or Follow-up Activities

Objective: Practise writing basic SQL queries to retrieve, filter, and sort data.

- **Web-based SQL editor:** We'll be using a web-based SQL editor called "DB Fiddle" (or any other preferred platform you choose). This eliminates the need for Docker or local installations.
- **Sample Database:** A pre-populated sample database relevant to your field (e.g., library management system, e-commerce store) will be provided within the DB Fiddle environment.

Homework or Follow-up Activities

Instructions:

1. **Access DB Fiddle**: Visit **https://www.db-fiddle.com/** (or your chosen web-based SQL editor).

2. **Select Sample Database**: You'll be provided with a link or instructions to access the pre-populated sample database within DB Fiddle.

3. **Write SQL Queries**: Start by writing simple SELECT queries to retrieve specific data from the tables.

4. **Filter and Sort Results**: Experiment with WHERE and ORDER BY clauses to filter the retrieved data based on specific criteria and sort the results in various ways

References

- Database Design and Relational Theory, by C. J. Date
- FUNDAMENTALS OF Fourth Edition DATABASE SYSTEMS Ramez Elmasri
- Database Design for Mere Mortals, by Michael J Hernandez
- https://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar