Introduction to SQL and Databases

Introduction to Databases

A database is an organized collection of data.

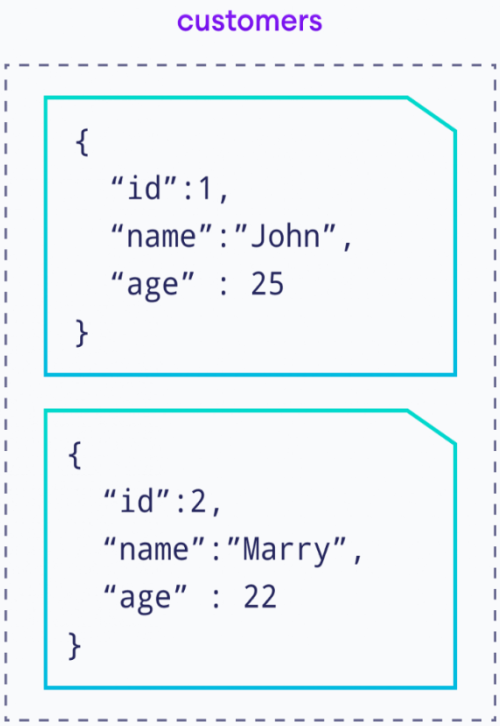
Types of Databases

In general, there are two common types of databases:

- Non-Relational
- Relational

Non-Relational Database

In a non-relational database, data is stored in **key-value pairs**. For example:



Example: Data stored in non-relational database

Here, customers' data are stored in key-value pairs.

Commonly used non-relational database management systems (Non-RDBMS) are MongoDB, Amazon DynamoDB, Redis, etc.

Relational Database

In a relational database, data is stored in **tabular format**. For example,

Table: customers

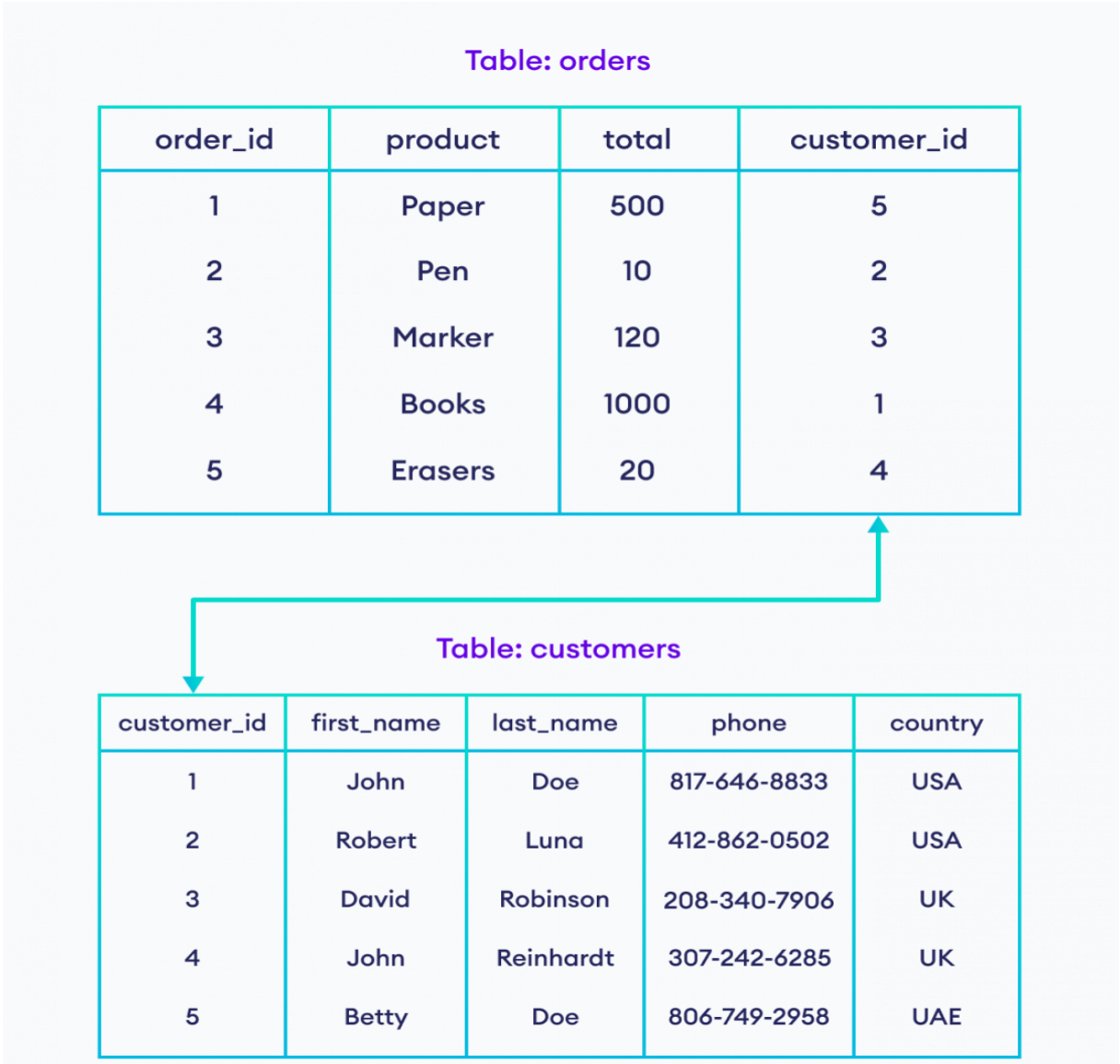
customer_id	first_name	last_name	phone	country
1	John	Doe	817-646-8833	USA
2	Robert	Luna	412-862-0502	USA
3	David	Robinson	208-340-7906	UK
4	John	Reinhardt	307-242-6285	UK
5	Betty	Taylor	806-749-2958	UAE

Example: Relational Database

Here, `customers` is a table inside the database.

The first row is the attributes of the table. Each row after that contains the data of a customer.

In a relational database, two or more tables may be related to each other. Hence the term "**Relational**". For example,



Example: Relational Database

Here, orders and customers are related through `customer_id`.

Commonly used relational database management systems (RDBMS) are MySQL, PostgreSQL, MSSQL, Oracle etc.

> **Note:** To access data from these relational databases, **SQL (Structured Query Language)** is used.

Introduction to SQL

Structured Query Language (SQL) is a standard query language that is used to work with relational databases.
We use SQL to

- create databases
- create tables in a database
- read data from a table
- insert data in a table
- update data in a table
- delete data from a table
- delete database tables
- delete databases
- and many more database operations

SQL commands are categorized into several groups:

Data Query Language (DQL): DQL commands like SELECT are used to retrieve data from the database.

Data Definition Language (DDL): DDL commands, including CREATE, ALTER, and DROP, are used to define and manage the structure of a database, including tables and schemas.

Data Manipulation Language (DML): DML commands such as INSERT, UPDATE, and DELETE allow users to modify and manipulate data within the database.

Data Control Language (DCL): DCL commands like GRANT and REVOKE are used to control access and permissions to the database.

CREATE TABLE [Students] ([StudentId] INTEGER NOT NULL PRIMARY KEY, [SubjectName] VARCHAR(30) NOT NULL, [DepartmentId] VARCHAR(30) NOT NULL, [DateofBirth] DATE NULL);

INSERT INTO Students (StudentId, SubjectName, DepartmentId, DateofBirth) VALUES (10005, "Prince", "IT002", "09-08-1989");

SELECT StudentId, SubjectName FROM Students;

SELECT * FROM Students;

UPDATE Students SET SubjectName = "Computer Science" WHERE StudentId = 10005;

Delete Table:
DROP TABLE tableName;

Rename Table:
ALTER TABLE oldTableName RENAME TO newTableName;

Add new field:
ALTER TABLE tableName ADD COLUMN newFieldName dataType;

Insert values:

```
INSERT INTO Tablename(colname1, colname2, ….) VALUES(valu1, value2, ….);

or

INSERT INTO Tablename VALUES(value1, value2, ….);

DELETE
DELETE FROM Students WHERE StudentId = 11;

Primary Key:
ColumnName INTEGER NOT NULL PRIMARY KEY;

PRIMARY KEY(ColumnName);

PRIMARY KEY(ColumnName1, ColumnName2);

Not Null constraints
ColumnName INTEGER  NOT NULL;

Default constraints
ColumnName INTEGER DEFAULT 0;

Unique constraints
EmployeeId INTEGER NOT NULL UNIQUE;

Check constraints
Quantity INTEGER NOT NULL CHECK(Quantity > 10);

Foreign Key
PRAGMA foreign_keys = ON;
```

```
CREATE TABLE [Departments] (
        [DepartmentId] INTEGER  NOT NULL PRIMARY KEY
AUTOINCREMENT,
        [DepartmentName] NVARCHAR(50)  NULL
);
CREATE TABLE [Students] (
        [StudentId] INTEGER  PRIMARY KEY AUTOINCREMENT NOT
NULL,
        [StudentName] NVARCHAR(50)  NULL,
        [DepartmentId] INTEGER  NOT NULL,
        [DateOfBirth] DATE  NULL,
        FOREIGN KEY(DepartmentId) REFERENCES
Departments(DepartmentId)
);

INSERT INTO Departments VALUES(1, 'IT');
INSERT INTO Departments VALUES(2, 'Arts');
```

Datatypes

Queries

JOIN

```
SELECT *
FROM Students
INNER JOIN Departments ON Students.DepartmentId =
Departments.DepartmentId;

SELECT Students.*
FROM Students
```

```
INNER JOIN Departments ON Students.DepartmentId =
Departments.DepartmentId;

Alias name - To column
SELECT StudentName AS 'Student Name' FROM Students;

or

SELECT StudentName 'Student Name' FROM Students;

Alias name - To table
SELECT s.* FROM Students AS s;

SELECT Students.StudentName, Departments.DepartmentName
FROM Students
INNER JOIN Departments ON Students.DepartmentId =
Departments.DepartmentId;

or

SELECT s.StudentName, d.DepartmentName
FROM Students AS s
INNER JOIN Departments AS d ON s.DepartmentId = d.DepartmentId;

Where - Startswith
SELECT StudentName FROM Students WHERE StudentName LIKE 'j%';

Where - Endswith
SELECT StudentName FROM Students WHERE StudentName LIKE '%y';

Where - Contains
```

```
SELECT StudentName FROM Students WHERE StudentName LIKE
'%n%';

AND
SELECT *
FROM Students
WHERE (StudentId > 5) AND (StudentName LIKE 'N%');

OR
SELECT *
FROM Students
WHERE (StudentId > 5) OR (StudentName LIKE 'N%');

BETWEEN
SELECT *
FROM Students
WHERE StudentId BETWEEN 5 AND 8;

LIMIT
SELECT * FROM Students LIMIT 4,3;

AGGREGATE
AVG
SELECT AVG(Mark) FROM Marks;

COUNT
SELECT COUNT(DepartmentId), COUNT(DISTINCT DepartmentId),
COUNT(*) FROM Students;

SUM
SELECT SUM(Mark) FROM Marks;
```