Databases and MS SQL

Databases are a structured set of data. MySQL is an open-source relational database management system.

Outline

1. Intro to Databases
2. ER Model
3. Relationships
4. Normalization
5. Intro to MS SQL
6. Data Creation
7. Transactions
8. Query Data
9. Joins
10. Set Operators
11. Subqueries
12. Procedures
13. Functions
14. Triggers

Prerequisites

1. Microsoft SQL Server: https://www.microsoft.com/en-us/sql-server/sql-server-downloads
2. Windows:
   - MS SQL Server
   - SQL Server Management Studio (IDE)
3. Mac:
   - Docker
   - MS SQL Server
   - Azure Data Studio

Introduction to Databases

What is a Database?

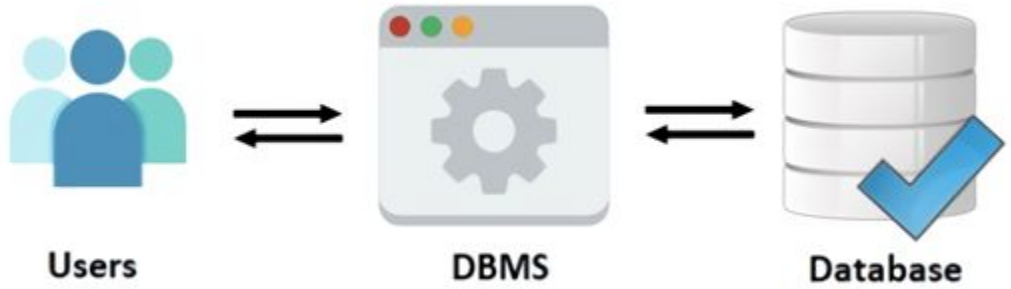
A **database** is a collection of information organized in such a way that a computer program can quickly select desired pieces of data.

Databases

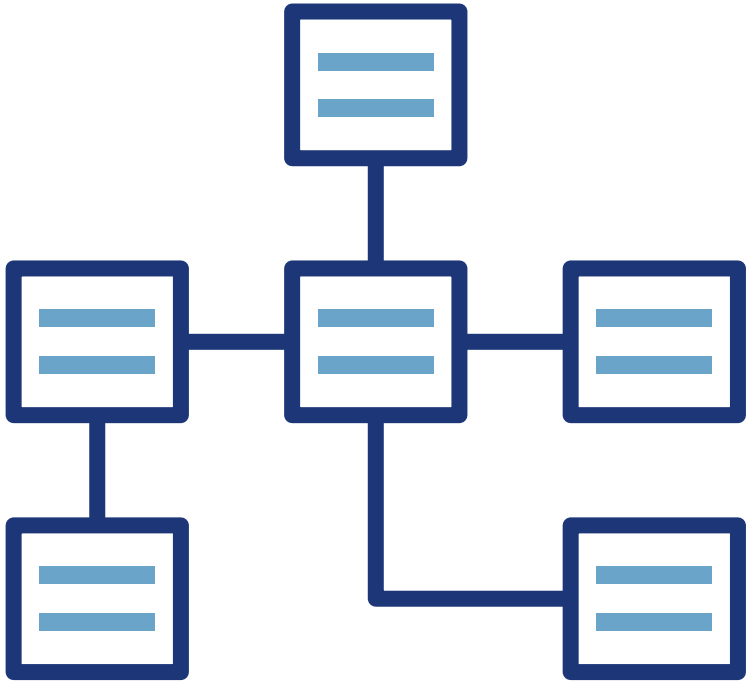
- Data stored on backend database server
- Database types:
  - ***Flat File***
  - ***Hierarchical***
  - ***Network***
  - ***Relational***
  - ***Object Oriented***
  - ***Object Relational***

Database Management System (DBMS)

A **DBMS** helps you create and manage databases—like MS Word helps you create and manage word document.



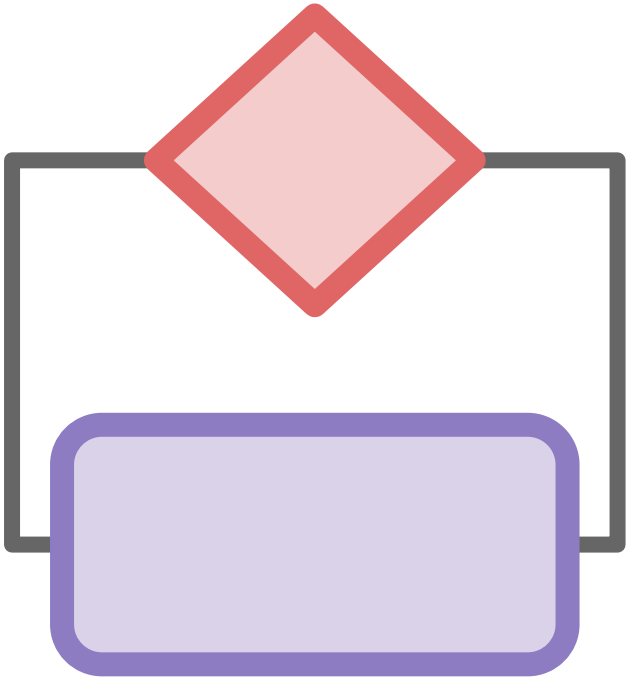
Schemas



Schema - description of the database

Subschema - describes a subset of the database and which users have access to this subset

ER Model



Entity-Relationship (ER) Data Model

An **ER Model** is made up of entities, attributes, and the relationships defined between entities.

Entity

An object in the real world that is distinguishable from other objects.

Ex: Employees, Places

Attribute

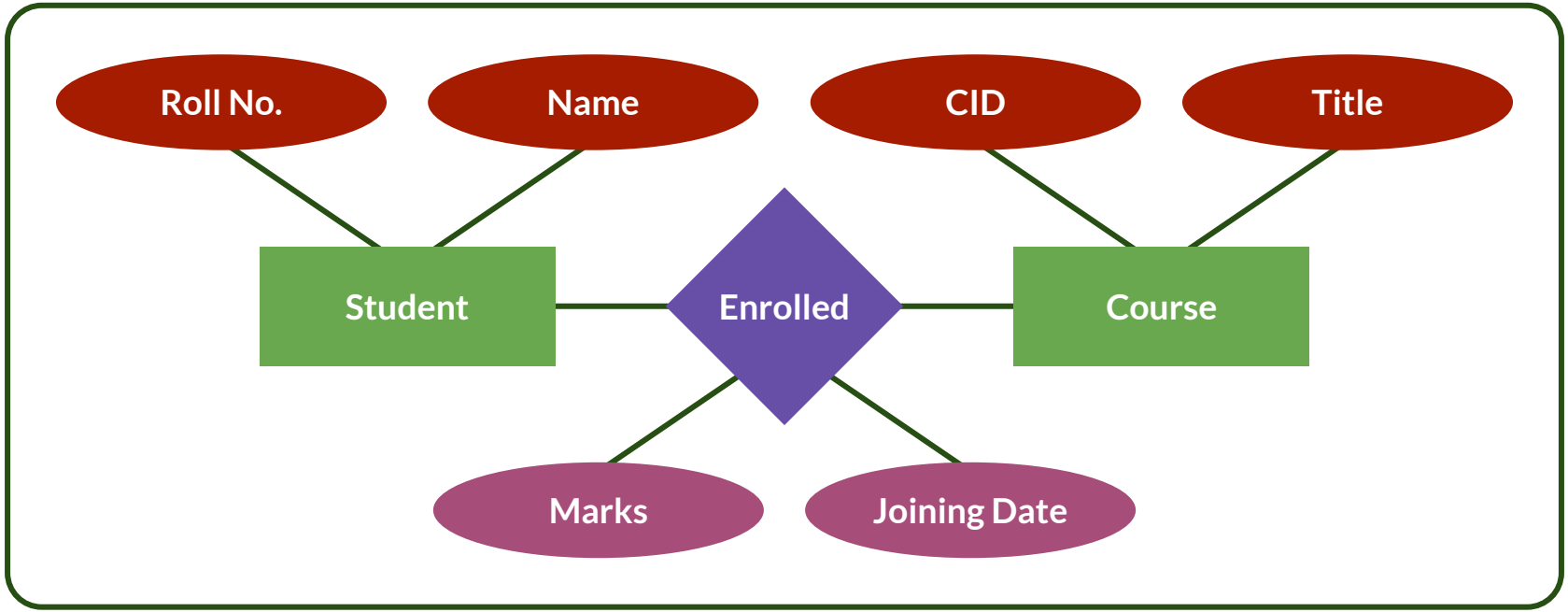
An entity is described in a database based on a set of attributes.

Ex: Name, Age, Gender

Relationship

A relationship links two entities that share one or more attributes.

Ex: Person lives in a House



Entities: Student, Course

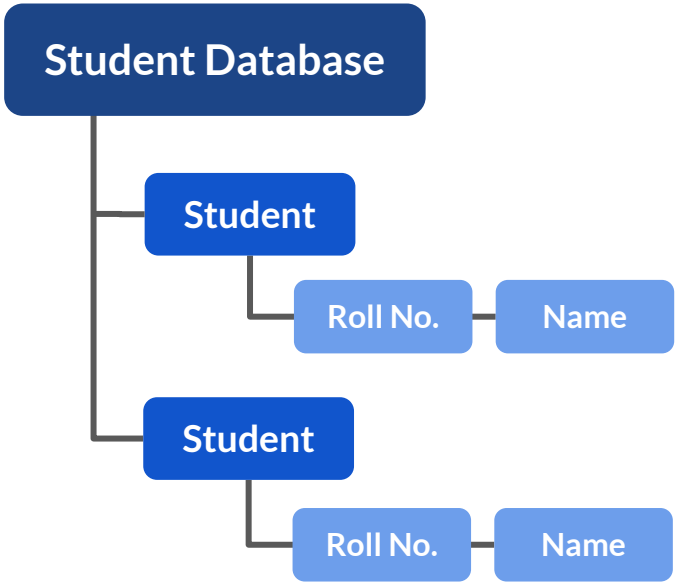
Attributes: Roll No, Name, CID, Title, Marks, Joining Date

Relationship: Enrolled

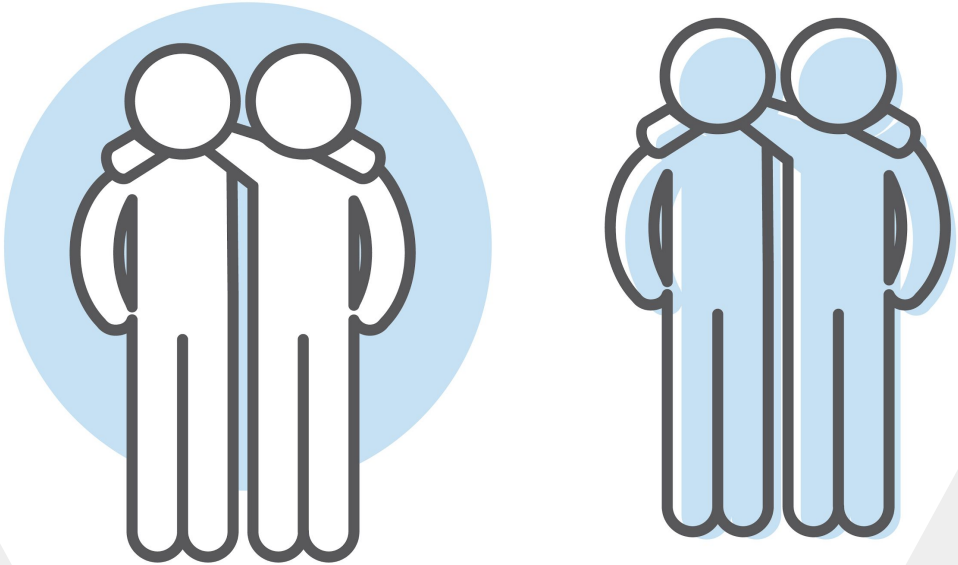
Defining Data With the ER Model

Instance (Record, Tuple) – single, specific occurrence of an entity

➔ You can have multiple students and each instance will have one student's RollNo and Name

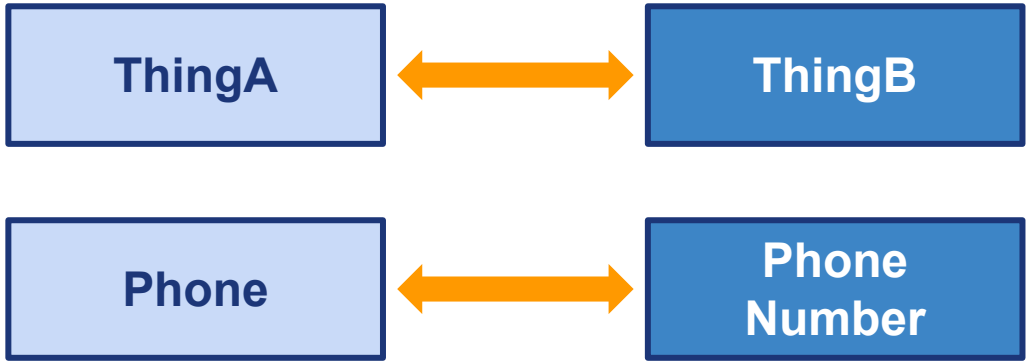


Relationships



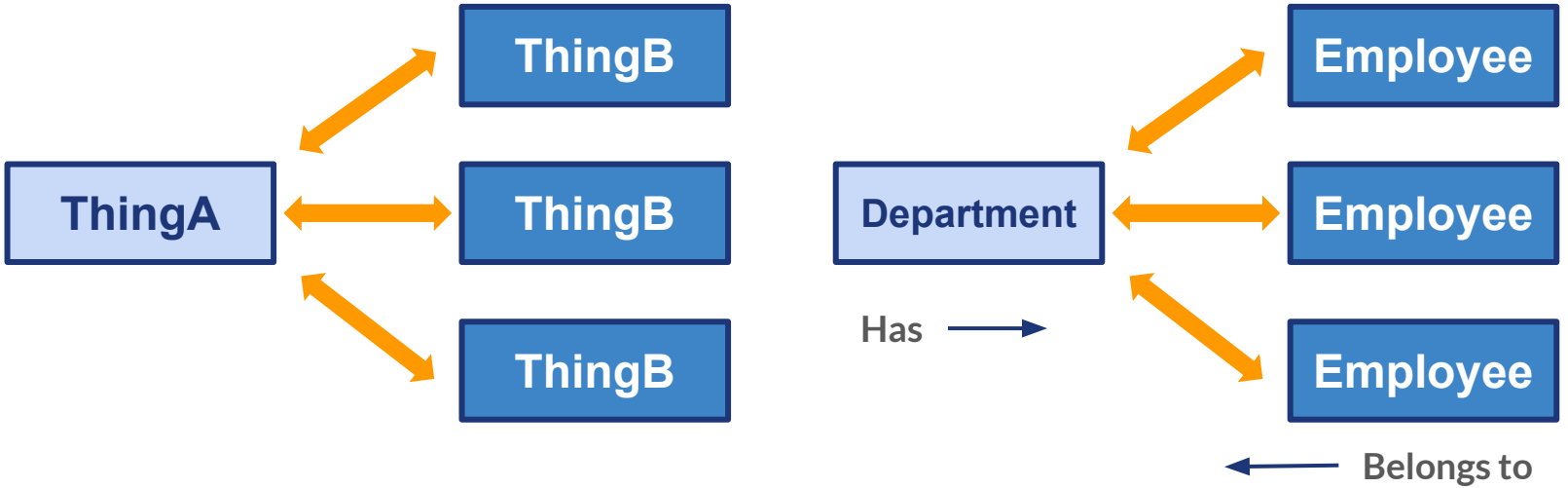
One-to-One

ThingA cannot be related to more than one ***ThingB***
ThingB cannot be related to more than one ***ThingA***



One-to-Many

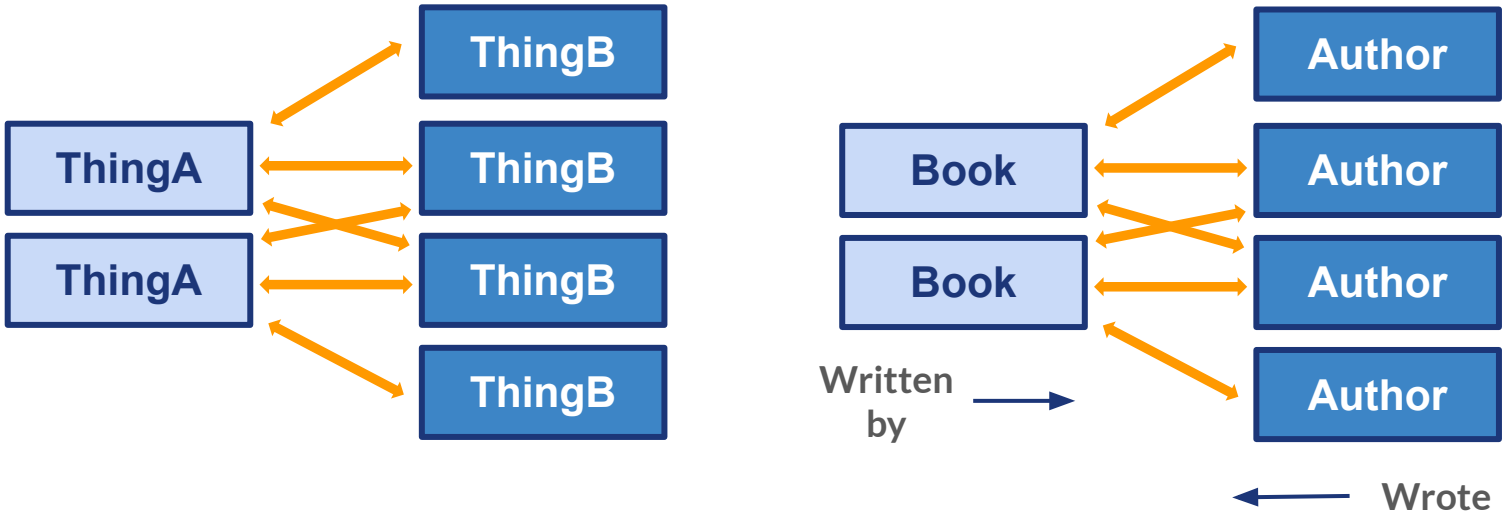
ThingA can be related to more than one ***ThingB***
ThingB cannot be related to more than one ***ThingA***



Many-to-Many

ThingA can be related to more than one ***ThingB***

ThingB can be related to more than one ***ThingA***



Relational Model

- ➔ Data is viewed as existing in a two dimensional table known as relations
- ➔ A relation (table) consists of unique attributes (columns) and tuples (rows)

Attributes/Fields/Columns

Emp_id	Emp_name	Emp_age	Emp_email
1000	Derek	24	dk@cognixia.com
1001	Maria	23	ma@cognixia.com
1002	Luis	25	ls@cognixia.com
1003	Janel	25	jl@cognixia.com

Rows/ Records/ Tuples

Keys in Relational Models

Primary Key - uniquely identifies each record in a database table

Emp_id	Emp_name	Emp_age	Emp_email
1000	Derek	24	dk@cognixia.com
1001	Maria	23	ma@cognixia.com
1002	Luis	25	ls@cognixia.com
1003	Janel	25	jl@cognixia.com

Keys in Relational Models

Foreign Key - an attribute in a table that references the primary key of another table

Emp_id	Emp_name	Emp_age	Emp_email	Fk_dept_id
1000	Derek	24	dk@cognixia.com	1
1001	Maria	23	ma@cognixia.com	3
1002	Luis	25	ls@cognixia.com	4
1003	Janel	25	jl@cognixia.com	3

Pk_dept_id	Dept_name
1	Accounting
2	Payroll
3	Marketing
4	HR

Review!

1. Answer some interview question style questions
2. Be ready to answer when your name is called

Normalization

Normalization

Normalization – method for organizing data into a database to eliminate data repetition and undesirable characteristics like

- ➔ ***Insertion anomalies***
- ➔ ***Delete anomalies***
- ➔ ***Update anomalies***

Problems Without Normalization

What are the issues we can run into with the table below?

rollNo	name	department	hod	office_tel
1000	Daisy	CS	Mr. X	53337
1001	Nick	CS	Mr. X	53337
1002	Ana	CS	Mr. X	53337
1003	James	CS	Mr. X	53337

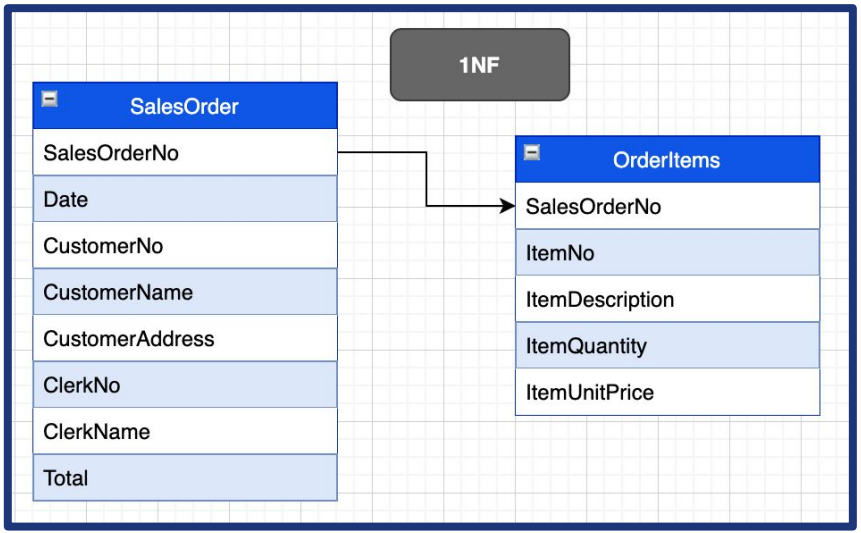
Normalization Example: Un-Normalized

SalesOrders		
SalesOrderNo	ClerkName	Item2UnitPrice
Date	Item1Description	Item3Description
CustomerNo	Item1Quantity	Item3Quantity
CustomerName	Item1UnitPrice	Item3UnitPrice
CustomerAddress	Item2Description	Total
ClerkNo	Item2Quantity	

Normalization Into First Normal Form

- ➔ **Separate repeating groups** into new tables
- ➔ Start a new table for repeating data
- ➔ The **primary key** for the repeating group is usually a **composite key**

Normalization Example: 1NF

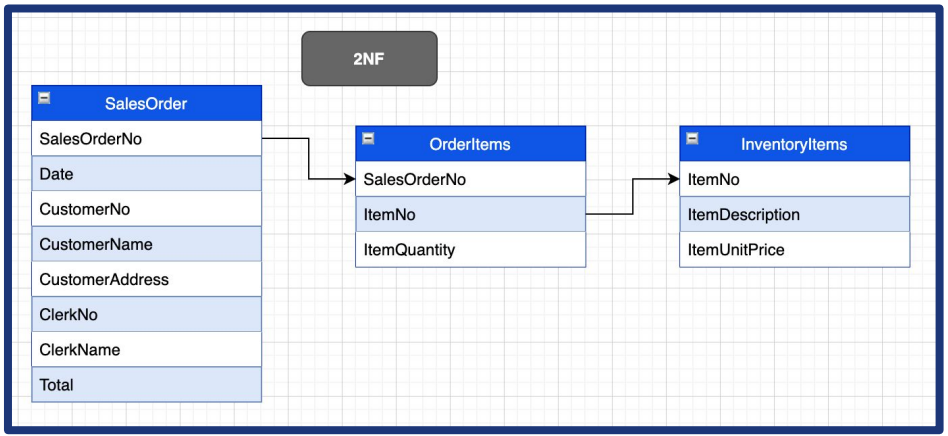


- ➔ **OrderItems** table created to separate repeating item information
- ➔ Create **ItemNo** and pair it with **SalesOrderNo** as primary key for new table to uniquely identify each item

Normalization Into Second Normal Form

- ➔ Remove **partial dependencies** – an attribute depends on only part of the primary key
- ➔ Start a new table for partially dependent data and part of key it depends on

Normalization Example: 2NF

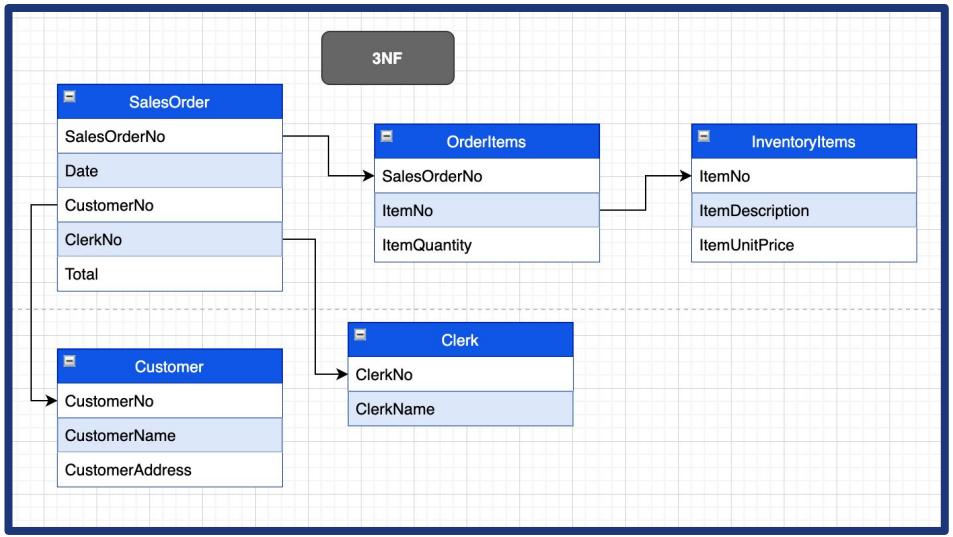


- ➔ Removed the partial dependency caused by Item
- ➔ Item description not duplicated
- ➔ Don’t need to insert sales order to add inventory item
- ➔ Item information stays even if sales order is deleted
- ➔ To change an item description, you don’t need to go through each sales order

Normalization Into Third Normal Form

- ➔ Remove **transitive dependencies** – attribute depends on an attribute other than the primary key
- ➔ Start a new table for transitive dependency and attributes that depend on it
- ➔ Keep a copy of the key attribute in the original table

Normalization Example: 3NF



- ➔ Removed the transitive dependencies with Customer and Clerk tables
- ➔ Now customer and clerk information isn't in every order
- ➔ If you delete a sales order, you aren't deleting customer or clerk information
- ➔ Don't need to change all orders when you need to update customer or clerk information

Database Planning

- ➔ **UML Diagrams** provide a way to visualize and design software or any system
- ➔ Flexibly used in development of applications but also to model how data will look
- ➔ Can plan out normalized database and determine if relationships between entities will work

Scenario: You are building an application for a music streaming company. They need to store information for artists names, album names, album release date, genre, song name, song length. What kind of UML do you build to model your data?

Review!

1. Answer some interview question style questions
2. Be ready to answer when your name is called

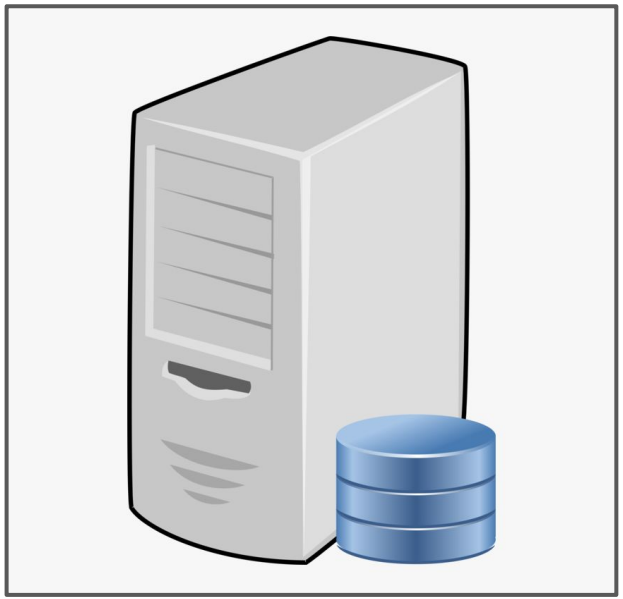
Introduction to Microsoft SQL

Microsoft®
SQL Server®

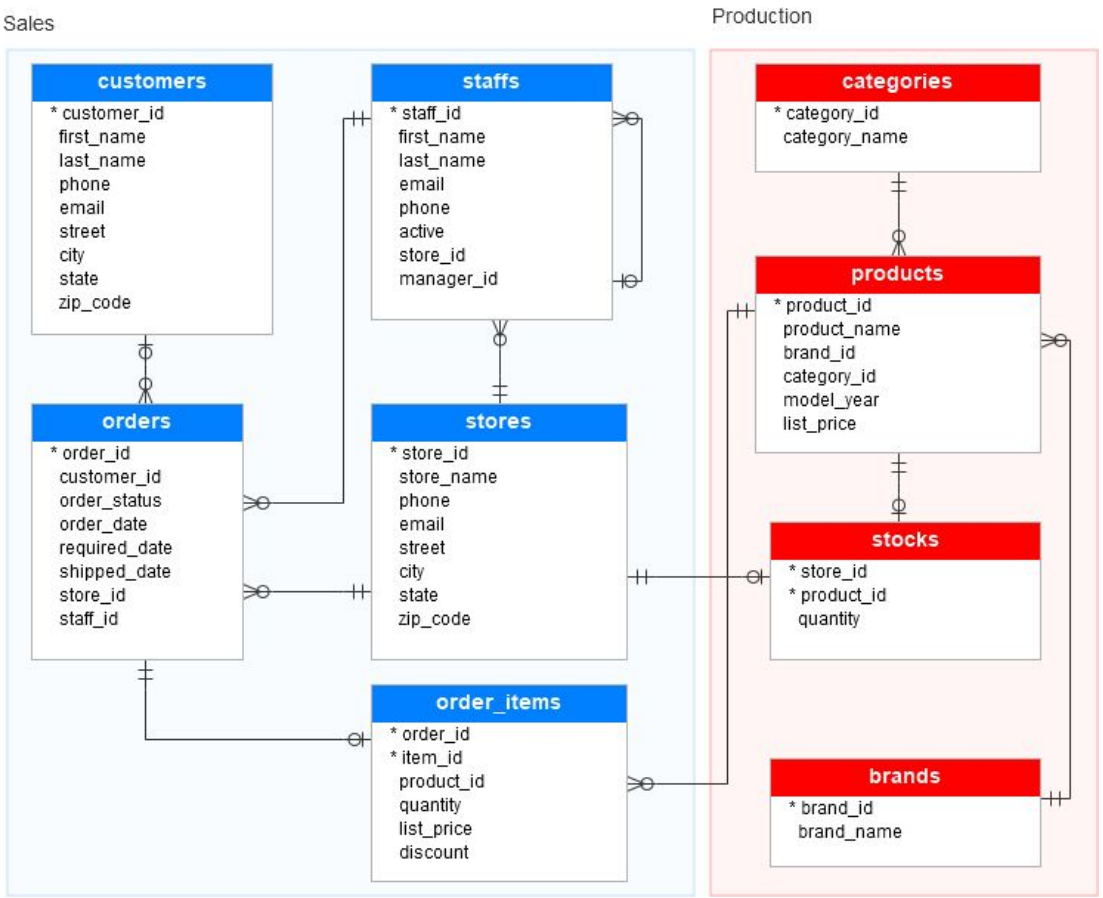
Microsoft SQL is a relational database management system (RDBMS) developed and marketed by Microsoft. SQL Server worked exclusively on Windows environment for more than 20 years. In 2016, Microsoft made it available on Linux.

SQL Server Architecture

- Consists of two main components:
  - **Database Engine** - core component of server, consists of ***relational engine*** that processes queries and a ***storage engine*** that manages data storage and retrieval.
  - **SQLOS** - stands for SQL Operating System, provides OS services like memory and I/O management



Data Creation



Before moving forward, for the following examples run, we will be using this sample database called ***BikeStores***. It keeps track of sales for a business that sells different types of bikes. Feel free to reference this slide and this site for information on this database. ***Script files will be given to you, please run them now.***

Exact Numeric Types

Table Header	Table Header
`bigint`, `int`, `smallint`, `tinyint`	Store whole numbers up to 8 bytes, 4 bytes, 2 bytes, and 1 byte in that order.
`bit`	Stores 0 or 1 or NULL, good for true or false values.
`decimal`, `numeric`	Used interchangeably to create decimal values between 5 and 17 bytes.
`money`	Stores currency values as high as 900 trillion.
`smallmoney`	Stores smaller currency values as high as 200 thousand.

Exact numeric data types store exact numbers such as integer, decimal, or monetary amount.

Approximate Numeric Types

Table Header	Table Header
`float`	Floating precision number data from -1.79E + 308 to 1.79E + 308. Can have a number precision of 7 digits. Memory allocation for value will vary based on how much is specified when initialized.
`real`	Floating precision number data from -3.40E + 38 to 3.40E + 38. Can have a number precision of 15 digits. Only needs 4 bytes of memory allocated per value.

Approximate numeric data types store floating point numeric data and are often used in scientific calculations.

Date & Time Types

Table Header	Table Header
`date`	Store a date only. From January 1, 0001 to December 31, 9999.
`time`	Store a time only to an accuracy of 100 nanoseconds.
`datetimeoffset`	The same as datetime2 but with time zone support.
`datetime2`	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds.

Date & time data types store store date and time data. The types above are what should be used for development of newer applications since have larger range and more precise than older types like ***datetime*** and ***smalldatetime***.

Character Strings Types

Table Header	Table Header
`char`	Store a fixed character string. Will store strings that have the exact number of characters specified.
`varchar`	Variable width character string, store up to the amount of characters specified.
`text`	Like varchar, stores a varied width of characters, but has a max of 2GB of text data it can store.

Character string data types allow you to store fixed length and variable length character data.

Binary String Types

Table Header	Table Header
`binary`	Fixed width binary string.
`varbinary`	Variable width binary string.
`image`	Variable width binary string (stores up to 2GB). Used specifically for storing images using binary.

Binary string data types allow you to store binary data. Usually this binary data is used when dealing with media like images, pdfs, or any data that can not be stored easily with numeric or character data types.

Constraints

Constraints define rules that allow/restrict values stored in columns

- ➔ **NOT NULL** - none of the values in this column can be null
- ➔ **UNIQUE** - each value in a column or group of columns must be unique, can accept one null value
- ➔ **PRIMARY KEY** - set column as primary key, no nulls accepted, must be unique
- ➔ **FOREIGN KEY** - set column as foreign key
- ➔ **DEFAULT** - if no value given, set default value for column
- ➔ **CHECK** - does a boolean check if values in that column meet condition given

Data Definition Language

- **Data Definition Language (DDL)** - define database structures and schemas
- ***Doesn't define actual data***, but the things that will hold/contain the data
- Commands:
  - `CREATE`
  - `ALTER`
  - `DROP`
  - `TRUNCATE`