Relational Databases 

- Press `Space` to navigate through the slides
- Use `Shift+Space` to go back

Module Readings 📖

Highly recommended:

- Lake, P. and Crowther, P. (2013). ***Concise Guide to Databases: A Practical Introduction*** (https://www.amazon.co.uk/Concise-Guide-Databases-Introduction-Undergraduate/dp/1447156005)
- Batra, R. (2018). ***SQL Primer: An Accelerated Introduction to SQL Basics*** (https://www.amazon.co.uk/SQL-Primer-Accelerated-Introduction-Basics/dp/1484235754)
- Tayloe, A. (2019). ***SQL For Dummies*** (https://www.amazon.co.uk/SQL-Dummies-Computer-Tech/dp/1119527074)

Classical database literature:

- Ramakrishnan, R. and Gehrke J. (2002). ***Database Management Systems*** (https://www.amazon.co.uk/Database-Management-Systems-Raghu-Ramakrishnan/dp/007123151X)
- Lake, P. and Crowther, P. (2014). ***Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition*** (https://www.amazon.co.uk/Database-Systems-Practical-Implementation-Management/dp/1292061189)
- Elmasri R. And S. Navathe (2016), ***Fundamentals of Database Systems, Global Edition*** (https://www.amazon.co.uk/Fundamentals-Database-Systems-Global-Elmasri/dp/1292097612)

SQL Tutorials:

- **W3Schools.com** – SQL Tutorial (https://www.w3schools.com/sql/)
- **SQLBolt.com** – Learn SQL (https://sqlbolt.com)
- **Codecademy.com** – SQL Tutorial (https://www.codecademy.com/learn/learn-sql)

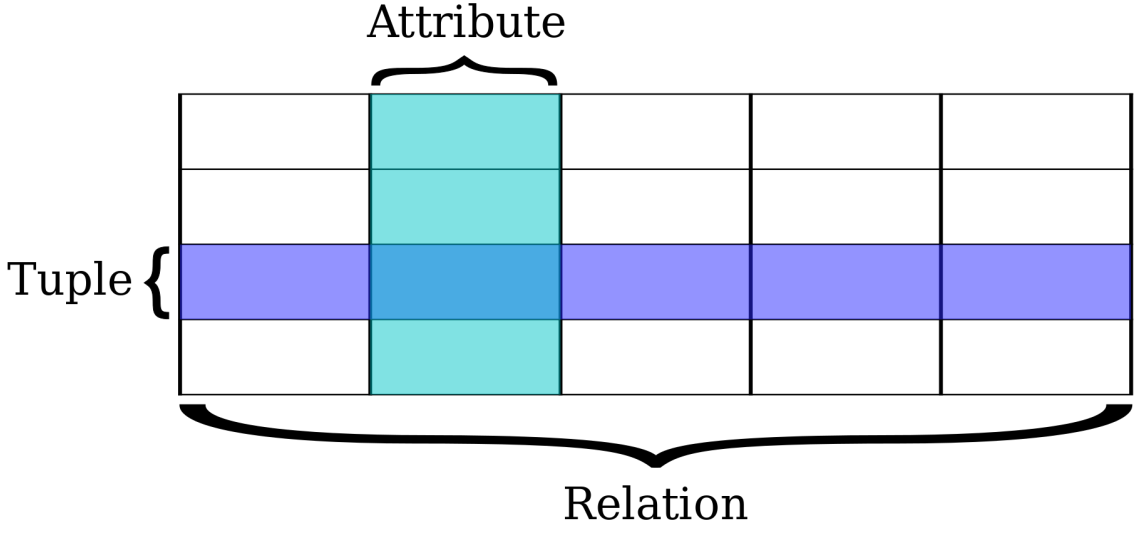
Continuous Assessment 

- **CA1 – 20%**
    - Design and implement a database
    - ON! Week 3
    - END Week 5
- **CA2 – 30%**
    - Querying and manipulating data using SQL
    - ON! Week 9
    - END Week 12

Quick Recap 🎒

- A **database** ia a self-describing collection of integrated records
- A **database management system (DBMS)** is a set of programs used to define, administer, and process databases and their associated applications.
- A **relational database** is a type of database where data is represented in tables, which stores and provides access to data points that are related to one another
- **SQL** is a flexible language used to communicate with a database (**DDL**, **DQL**, **DML**, **DCL**)

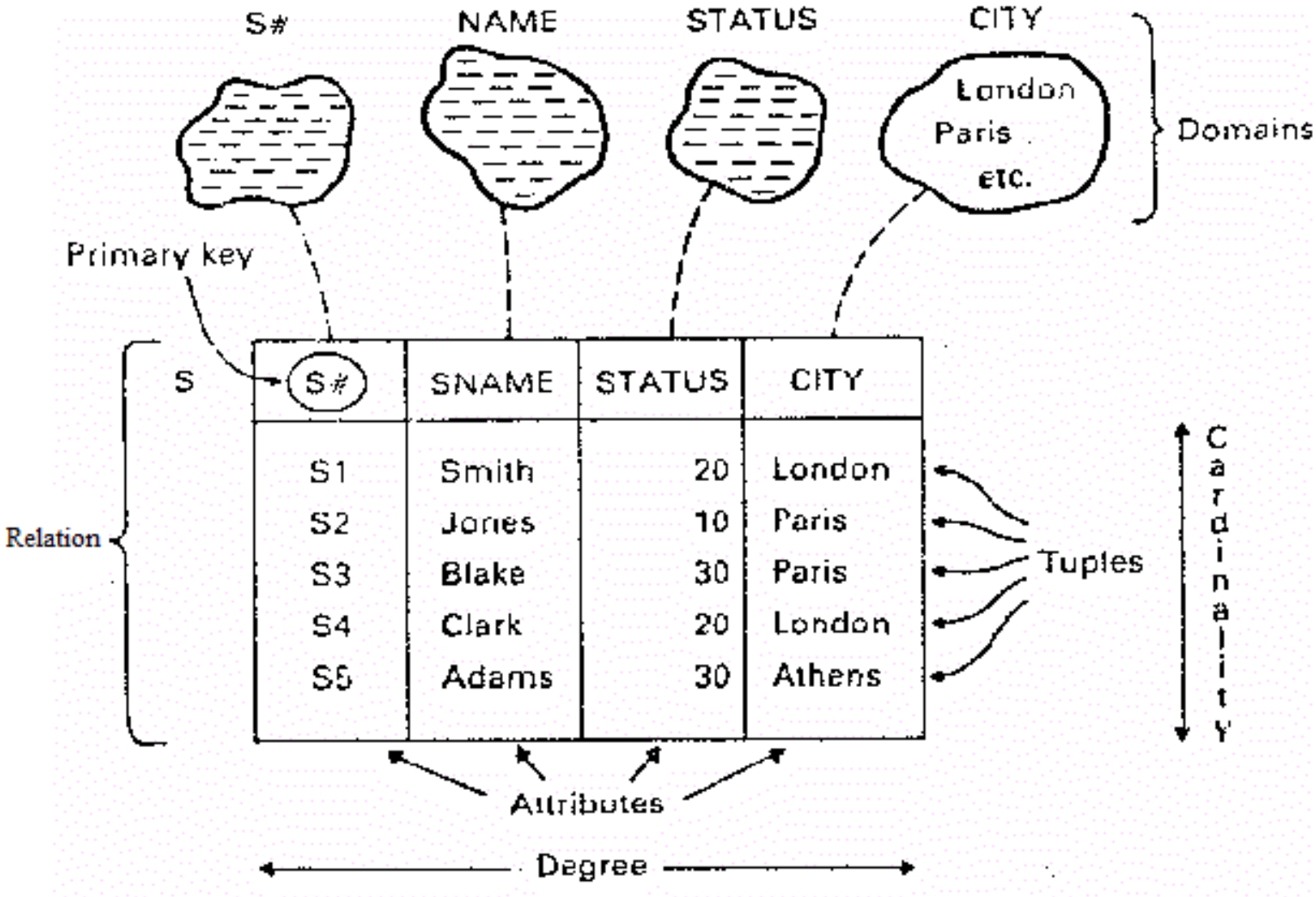
Relations 🔢 and Keys 🔑



SQL Term	Relational DB Term	Description
Row	Tuple / Record	A data set representing a single item
Column	Attribute / Field	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	Relation / Base relvar	A set of tuples sharing the same attributes; a set of columns and rows
View / Result set	Derived relvar	Any set of tuples; a data report from the RDBMS in response to a query

Terminology 📒

- **Relation** → Table
- **Tuple** → Row or record
- **Attribute** → Column or field
- **Cardinality** → Number of rows
- **Degree** → Number of columns
- **Primary key** → Unique identifier
- **Domain** → Pool of legal values


S#
NAME
STATUS
CITY
London
Paris
etc.
Domains
Primary key
S
S#
SNAME
STATUS
CITY
S1
Smith
20
London
S2
Jones
10
Paris
S3
Blake
30
Paris
S4
Clark
20
London
S5
Adams
30
Athens
Relation
Tuples
Cardinality
Attributes
Degree

Properties of Relations 🔢

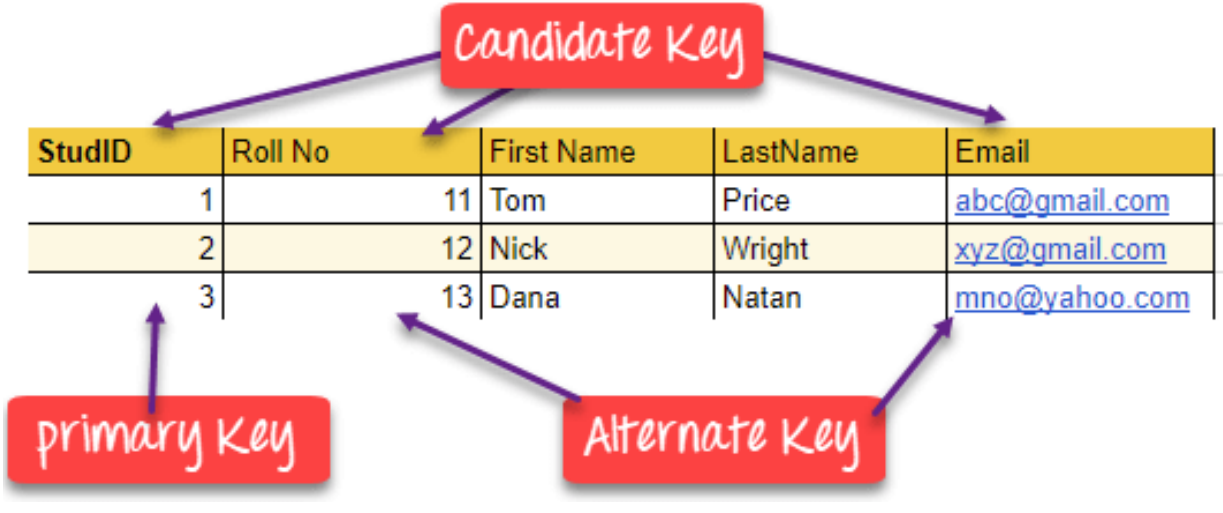
- **The name of a relation (table) is unique**
  - *No two relations may have the same name*
- **The name of an attribute is unique only within its relation**
  - *We can have two attributes called Name in separate relations (tables), but not in the same relation (table)*
- **The values of an attribute are all from the same domain**
  - *We should not allow a postcode to appear in a salary column for example*

- **The order of attributes within a relation has no significance**
    - *If we re-order the columns of a relation it does not become a different relation*
- **The order of rows within a relation has no significance**
    - If we re-arrange the rows of a relation, it does not become a different relation.
- **Each cell of a relation should contain at most one value**
    - *We cannot store two phone numbers in the same cell*
- **The records within a relation should all be distinct**
    - *If we examine the values in each row, no two rows should have exactly the same values (no duplicates)*

Candidate, Primary and Alternate Keys 🔐

- Keys with no repeated attribute values are called **candidate**
- Every table must have at least a single candidate key
    - *It must contain unique values*
    - *Candidate key may have multiple attributes*
    - *Must not contain null values*
    - *It should contain minimum fields to ensure uniqueness*
    - *Uniquely identify each record in a table*

- The **Primary key** should be selected from the candidate keys
- **Alternate keys** are candidate keys excluding the chosen primary key



Primary Keys 🔑

- A good rule for database design is to make sure that every row in a database table is distinguishable from every other row; each row should be ***unique***
- A **key** is an attribute (or combination of attributes) that uniquely identifies a row in a table
- To access a row in a database, you must have some way of distinguishing that row from all the other rows
- A **primary key** is a column or combination of columns in a table with values that uniquely identify the rows in the table

student_id	student	module
1	Lisa	Databases
2	Alan	HR
3	Julia	Mobile Dev

lecturer_id	lecturer	department
1	John	Business
2	Sarah	Computer Science
3	Michael	Computer Science

Foreign Keys 🗝️

- A **foreign key** is a column or group of columns in a table that corresponds to or references a primary key in another table in the database
- A **foreign key** doesn't have to be unique, but it must uniquely identify the column(s) in the table that the key references.
- A **foreign key** alternatively is called a **secondary key**

student_id	student	module	lecturer_id
1	Lisa	Databases	2
2	Alan	HR	1
3	Julia	Mobile Dev	2

lecturer_id	lecturer	department
1	John	Business
2	Sarah	Computer Science
3	Michael	Computer Science

Difference Between Primary key 🔑 & Foreign key 🗝️

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table	It is a field in the table that is the primary key of another table
Primary Key never accept null values	A foreign key may accept multiple null values
Primary key is a clustered index and data in the DBMS table are physically organised in the sequence of the clustered index	A foreign key cannot automatically create an index, clustered or non-clustered However, you can manually create an index on the foreign key
You can have the single Primary key in a table	You can have multiple foreign keys in a table

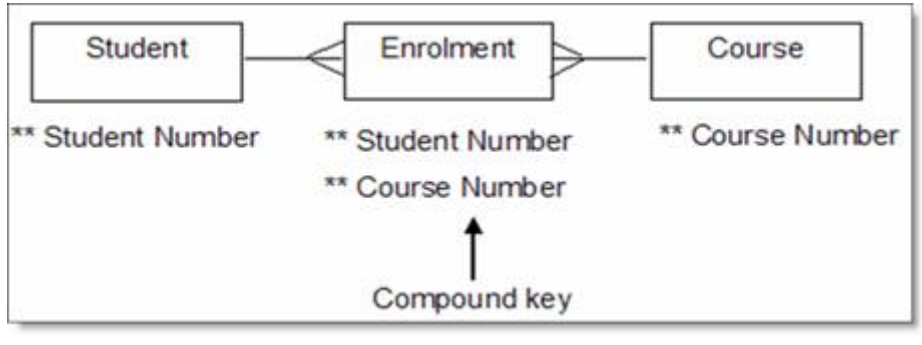
Simple, Compound and Composite Keys 🔐

- Any of the keys described before (ie: primary, foreign or secondary) may have ***one or more attributes***

- A **simple key** consists of a single attribute to uniquely identify an entity occurrence, for example, a student number, which uniquely identifies a particular student. No two students would have the same student number.

- A **compound key** consists of more than one attribute to uniquely identify an entity occurrence. Each attribute, which makes up the key, is also a simple key in its own right.

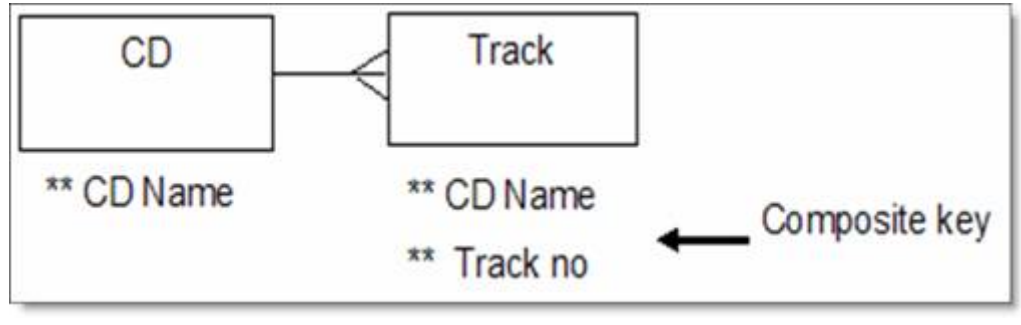
  For example, we have an entity named enrolment, which holds the courses on which a student is enrolled. In this scenario a student is allowed to enrol on more than one course. This has a compound key of both student number and course number, which is required to uniquely identify a student on a particular course.



Student number and course number combined is a compound primary key for the enrolment entity.

- A **composite key** consists of more than one attribute to uniquely identify an entity occurrence. This differs from a compound key in that one or more of the attributes, which make up the key, are not simple keys in their own right.

  For example, you have a database holding your CD collection. One of the entities is called tracks, which holds details of the tracks on a CD. This has a composite key of CD name, track number.



CD name in the track entity is a simple key, linking to the CD entity, but track number is not a simple key in its own right.

Internal, Conceptual, External Schema 

The term **schema** refers to the organisation of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

There are 3 types of schemas:

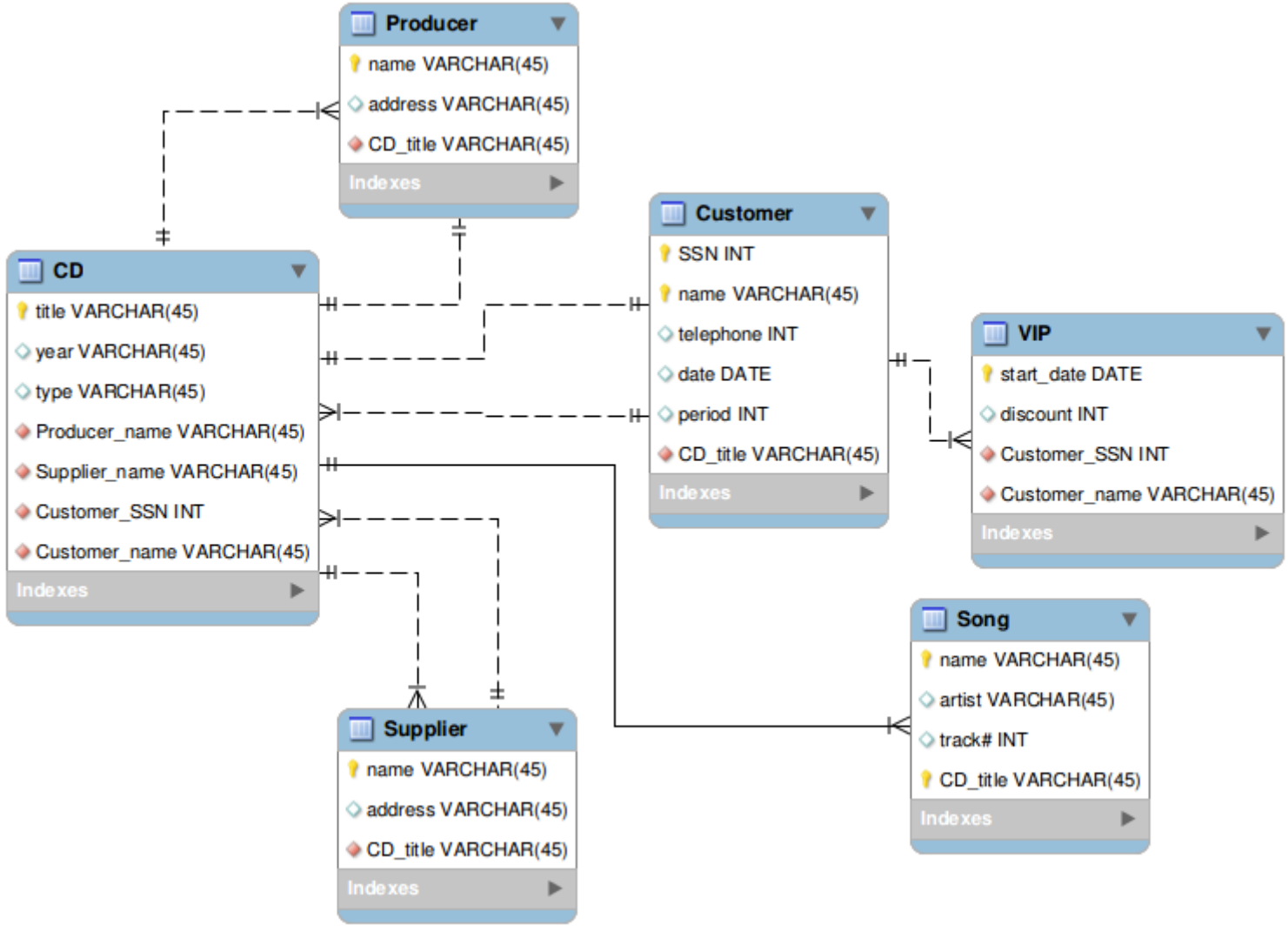
- **Internal**
- **Conceptual**
- **External**

Internal Schema 🏢

- The **internal schema** defines the physical storage structure of the database.
- The **internal schema** is a very low-level representation of the entire database.
- It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records.
- The **internal view** tells us what data is stored in the database and how
- It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages.

Conceptual Schema 🏛️

- The structure of an entire database is its **Conceptual schema** or **Conceptual view**
- This structure is sometimes also called the complete logical view of the database
- This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc


Producer
name VARCHAR(45)
address VARCHAR(45)
CD_title VARCHAR(45)
Indexes
CD
title VARCHAR(45)
year VARCHAR(45)
type VARCHAR(45)
Producer_name VARCHAR(45)
Supplier_name VARCHAR(45)
Customer_SSN INT
Customer_name VARCHAR(45)
Indexes
Customer
SSN INT
name VARCHAR(45)
telephone INT
date DATE
period INT
CD_title VARCHAR(45)
Indexes
VIP
start_date DATE
discount INT
Customer_SSN INT
Customer_name VARCHAR(45)
Indexes
Song
name VARCHAR(45)
artist VARCHAR(45)
track# INT
CD_title VARCHAR(45)
Indexes
Supplier
name VARCHAR(45)
address VARCHAR(45)
CD_title VARCHAR(45)
Indexes

External Schema 🏰

- An **external schema** describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database.
- Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.
- An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

Relational Integrity Constraints 🚫

- Relational Integrity constraints is referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.

  There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:

  - **Domain constraints**
  - **Key constraints**
  - **Referential integrity constraints**

Domain Constraints ✖

- **Constraints** are rules that determine what values the table attributes can assume
- By applying tight constraints to a column, you can prevent people from entering invalid data into that column
- Every value that is legitimately in the domain of the column must satisfy all the column's constraints
- A **Domain constraint** is a restriction on what a column may contain

```
Create DOMAIN CustomerName
CHECK (value not NULL)
```

Key Constraints ✖

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, **CustomerID=1** is only for the **CustomerName="Google"**.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

A **Primary key** it must be **unique** and must have the **NOT NULL** attribute.

Referential Integrity Constraints ✖

Referential integrity constraints is base on the concept of **Foreign Keys**. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

InvoiceNo	CustomerID	Amount
1	1	€100
2	1	€200
3	2	€150

Summary 🕶️

- Relations & Keys:
    - Terminology: **Relation**, **Tuple**, **Attribute**, **Cardinality**, **Degree**, **Primary key**, **Domain**
    - Properties of relations
    - Keys: **Candidate**, **Primary**, **Foreign**, **Alternate**
    - Keys constructs: **Single**, **Compound**, **Composite**
- Schemas:
    - **Internal**
    - **Conceptual**
    - **External**

- Relational Integrity Constraints:
    - **Domain constraints**
    - **Key constraints**
    - **Referential integrity**