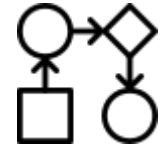
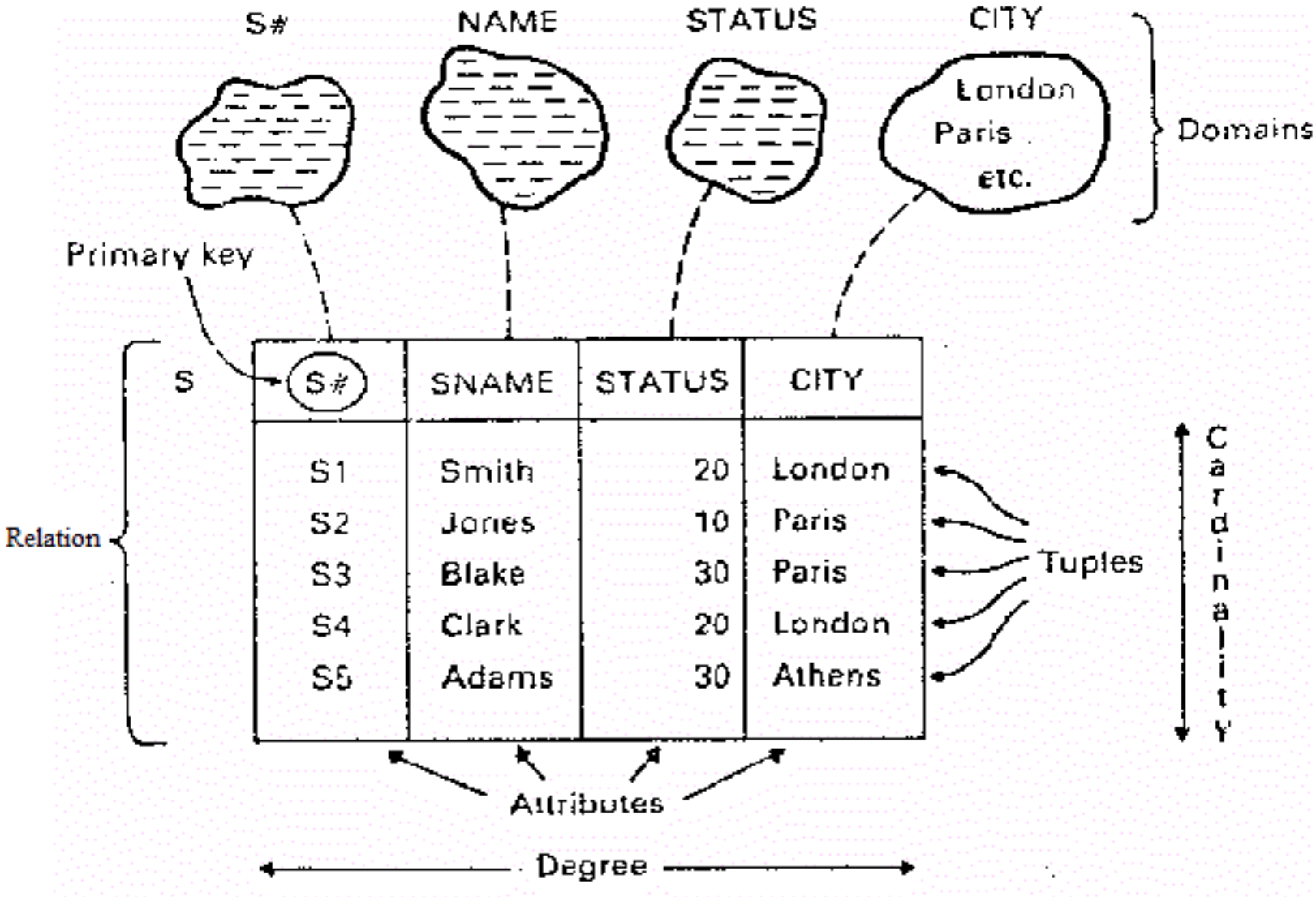
# Normalisation  & Data Modelling 

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

# Quick Recap 

- 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
- Keys:
  - **Candidate**, **Primary**, **Foreign**, **Alternate**
  - **Single**, **Compound**, **Composite**
- Schemas:
  - **Internal**, **Conceptual**, **External**


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

# Quick Recap 

- A **database** ia a self-describing collection of integrated records
- A **database management system (DBMS)** is system software for creating and managing databases
- A **relational database (RDB)** is a collective set of multiple data sets organised by tables, records and columns
- **SQL** is a standardised query language for requesting information from a database

# Normalisation 

- **Normalisation** is the process of reorganising data in a database so that it meets two basic requirements:

    1. There is no redundancy of data (all data is stored in only one place)
    2. Data dependencies are logical (all related data items are stored together)
- **Normalisation** is important, as allows databases to take up as little disk space as possible, resulting in increased performance.

---

- **Essentially**, this is the process of dividing larger tables into smaller tables and linking them using relationships.

# Normalisation 

- The inventor of the relational model Edgar Codd proposed the theory of normalisation with the introduction of **First Normal Form (1NF)**
- He continued to extend theory with **Second (2NF)** and **Third (3NF) Normal Form**
- Later he joined with Raymond F. Boyce to develop the theory of **Boyce-Codd Normal Form**

# Normalisation  Tutorial

- Assume we have a list of people and list their faviourite movies. Without any normalisation, all information is stored in one table as shown below.

| Name | Surname | Title | Address | Movies | Category |
|---|---|---|---|---|---|
| John | Smith | Mr. | Dublin 4 | Matrix, Terminator | Action, Action |
| Jane | Sanders | Mrs. | Dublin 1 | Titanic, Pirates of the Carribbean | Drama, Action |
| Jane | Sanders | Ms. | Dublin 8 | Donnie Darko | SciFi Thriller |

# First Normal Form (1NF)

- To achieve first normal form for a database, you need to make sure that no table contains multiple columns that you could use to get the same information.
- Each table should be organised into rows, and each row should have a primary key that distinguishes it as unique.

| Name | Surname | Title | Address | Movies | Category |
|---|---|---|---|---|---|
| John | Smith | Mr. | Dublin 4 | Matrix | Action |
| John | Smith | Mr. | Dublin 4 | Terminator | Action |
| Jane | Sanders | Mrs. | Dublin 1 | Titanic | Drama |
| Jane | Sanders | Mrs. | Dublin 1 | Pirates of the Carribbean | Action |
| Jane | Sanders | Ms. | Dublin 8 | Donnie Darko | SciFi |

# Second Normal Form (2NF)

## * To achieve second normal form, it would be helpful to split out the movies into an independent table and match them up using Person ID to remove partial functional dependency

**Rules**

- Tables should be in 1NF
- Have no **partial dependency**
    - There should be a **single primary key**
    - Remove any non-key attributes that only depend on part of the table key to a new table

| Person ID | Name | Surname | Title | Address |
|---|---|---|---|---|
| 1 | John | Smith | Mr. | Dublin 4 |
| 2 | Jane | Sanders | Mrs. | Dublin 1 |
| 3 | Jane | Sanders | Ms. | Dublin 8 |

| Person ID | Movies | Category |
|---|---|---|
| 1 | Matrix | Action |
| 1 | Terminator | Action |
| 2 | Titanic | Drama |

| Person ID | Movies | Category |
|---|---|---|
| 2 | Pirates of the Carribbean | Action |
| 3 | Donnie Darko | SciFi |

# Partial Dependency 

- **Partial dependency** occurs when a non-prime attribute is functionally dependent on part of a candidate key.

| Customer ID | Store ID | Purchase Location |
|---|---|---|
| 1 | 1 | Dublin |
| 1 | 3 | Cork |
| 2 | 1 | Dublin |
| 3 | 2 | Waterford |
| 4 | 3 | Cork |

- This table has a composite primary key **Customer ID, Store ID**.
- The non-key attribute is **Purchase Location**.
- In this case, **Purchase Location** only depends on **Store ID**, which is only part of the primary key.
- To bring this table to **2NF**, we break the table into two tables

- Now, in the new we created table, the column **Purchase Location** is fully dependent on the primary key of that table, which is **Store ID**.

| Customer ID | Store ID |
|---|---|
| 1 | 1 |
| 1 | 3 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |

| Store ID | Purchase Location |
|---|---|
| 1 | Dublin |
| 2 | Waterford |
| 3 | Cork |

# Transitive Dependency 

- A **transitive functional dependency** is when changing a non-key column, might cause any of the other non-key columns to change
- **Transitive dependency** must be removed to satisfy 3NF
- Exaple: Change of **Surname** and consequent change of **Title**

| Person ID | Name | Surname | Title | Address |
|---|---|---|---|---|
| 1 | John | Smith | Mr. | Dublin 4 |
| 2 | Jane | Sanders | Mrs. | Dublin 1 |
| 3 | Jane | Sanders | Ms. | Dublin 8 |

# Third Normal Form (3NF)

### Rules

- Tables should be in 2NF
- Have no **transitive dependencies**

---

- We have again divided our tables and created a new table which stores **Titles**
- There are no transitive functional dependencies, and hence our table is in **3NF**
- Now our little example is at a level that cannot further be decomposed to attain higher forms of normalisation

| Person ID | Name | Surname | Title ID | Address |
|---|---|---|---|---|
| 1 | John | Smith | 1 | Dublin 4 |
| 2 | Jane | Sanders | 3 | Dublin 1 |
| 3 | Jane | Sanders | 2 | Dublin 8 |

| Person ID | Movies | Category |
|---|---|---|
| 1 | Matrix | Action |
| 1 | Terminator | Action |
| 2 | Titanic | Drama |
| 2 | Pirates of the Carribbean | Action |
| 3 | Donnie Darko | SciFi |

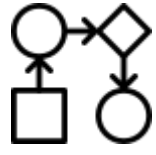
| Title ID | Title |
|---|---|
| 1 | Mr. |
| 2 | Ms. |
| 3 | Mrs. |

# Further Normalisation 

- **Boyce-Codd Normal Form (BCNF)**
    - Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate Key**
    - BCNF sometimes is reffered to as **3.5NF**
- **Other Normal Forms (including 4NF, 5NF, 6NF)**
    - https://en.wikipedia.org/wiki/Database_normalization (https://en.wikipedia.org/wiki/Database_normalization)
    - https://www.studytonight.com/dbms/database-normalization.php (https://www.studytonight.com/dbms/database-normalization.php)

# Normalisation  Summary

- Database designing is critical to the successful implementation of a DBMS that meets the data requirements of an enterprise system.
- Normalisation helps produce database systems that are cost-effective and have better security models.
- Functional dependencies are a very important component of the normalise data process
- Most database systems are normalised database up to the third normal forms.

# Data Modelling 

- Data modeling is the process of creating a data model for the data to be stored in a database.
- It describes:
    - Data objects
    - The associations between different data objects
    - The rules

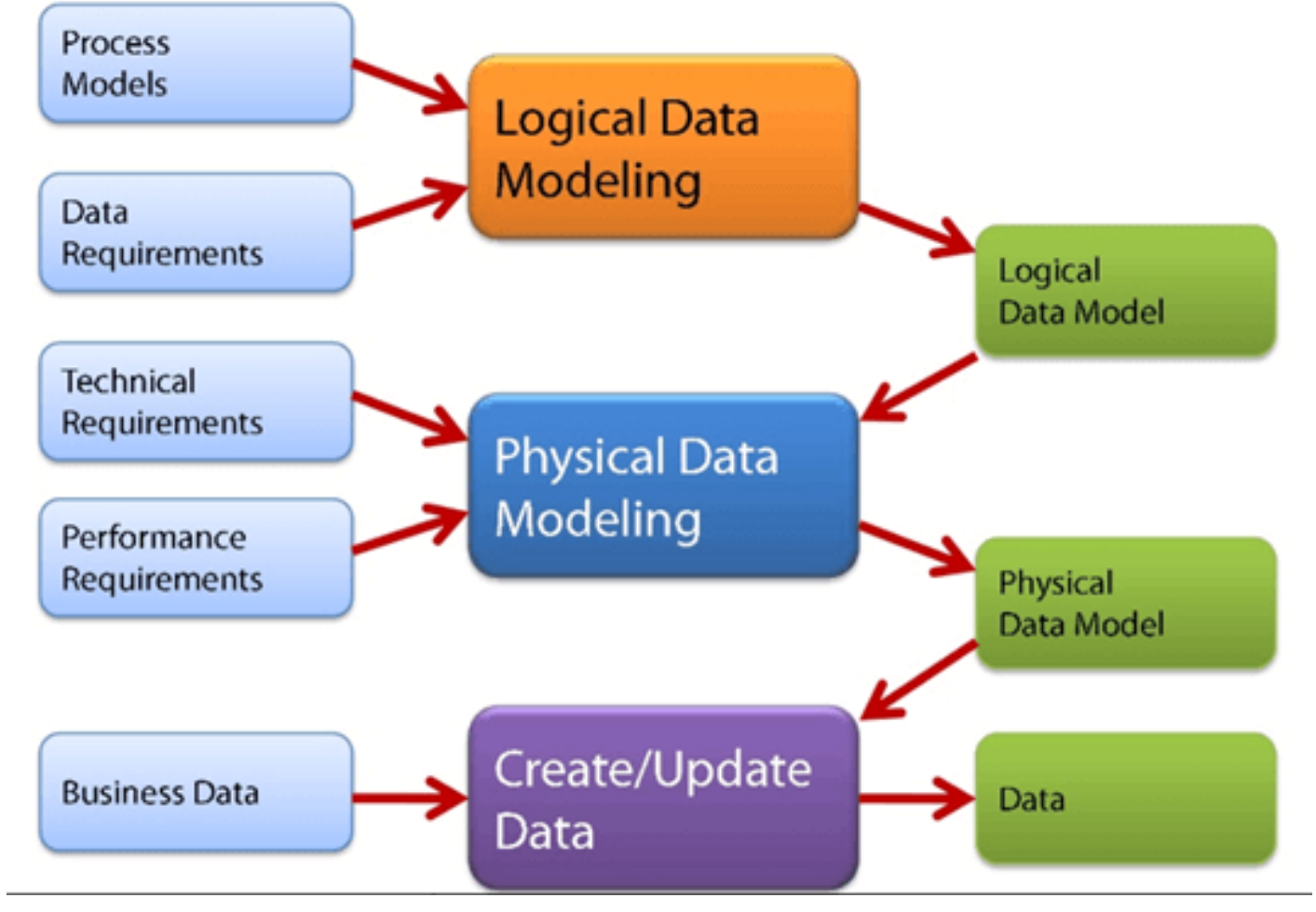
# Data Models 

- **Conceptual**
    - This Data Model defines **WHAT** the system contains.
    - This model is typically created by Business stakeholders and Data Architects.
    - The purpose is to organise, scope and define business concepts and rules.

- **Logical**
    - Defines **HOW** the system should be implemented regardless of the DBMS.
    - This model is typically created by Data Architects and Business Analysts.
    - The purpose is to developed technical map of rules and data structures.

- **Physical**
    - This Data Model describes **HOW** the system will be implemented using a specific DBMS system.
    - This model is typically created by DB analysts and developers.
    - The purpose is actual implementation of the database.

# Data Models

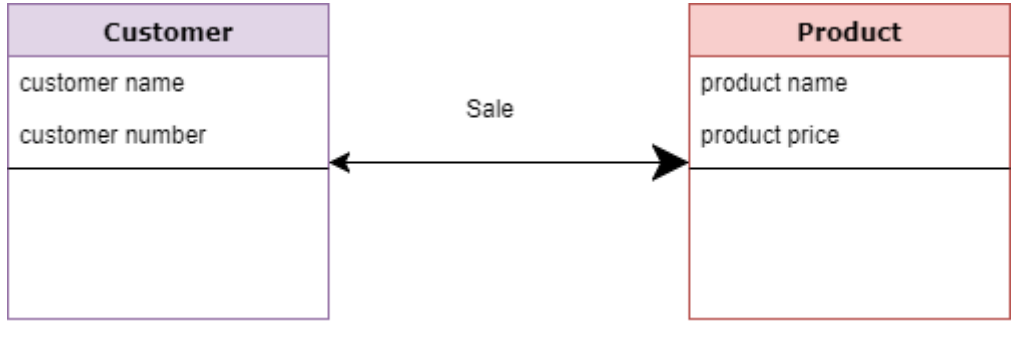

Process Models
Data Requirements
Technical Requirements
Performance Requirements
Business Data
Logical Data Modeling
Physical Data Modeling
Create/Update Data
Logical Data Model
Physical Data Model
Data

# Components of a Data Model

- **Entity**
    - Any object in the system that we want to model and store information about
    - Entities are usually recognisable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database
- **Attribute**
    - Characteristics or properties of an entity
- **Relationship**
    - Dependency or association between two entities

# Conceptual Model

- The main aim of this model is to establish the entities, their attributes and their relationships.
- At this level is hardly any detail available of the actual database structure.
- **Example**
    - Customer and Product are two entities
    - Customer number and name are attributes of the Customer entity
    - Product name and price are attributes of product entity
    - Sale is the relationship between the customer and product



# Conceptual Model Characteristics

- Offers organisation-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology.
- The focus is to represent data as a user will see it in the "real world."

# Logical Data Model

- Logical data models add further information to the conceptual model elements.
- It defines the structure of the data elements and set the relationships between them.

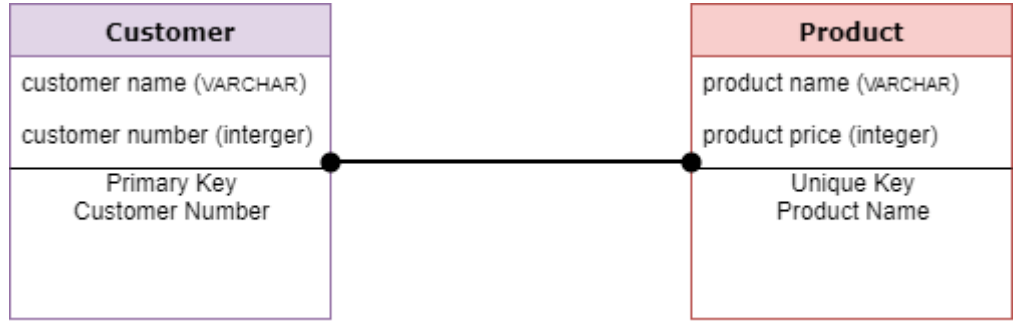

- The advantage of the Logical data model is to provide a foundation to form the base for the Physical model.
- However, the modeling structure remains generic.
- At this level, no primary or secondary key is defined.
- In this modelling phase, you need to verify and adjust the connector details that were set earlier for relationships.

# Logical Model Characteristics

- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have datatypes with exact precisions and length.
- Normalisation processes to the model is applied typically till 3NF.

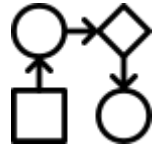
# Physical Data Model

- A Physical Data Model describes the database specific implementation of the data model.
- It offers an abstraction of the database and helps generate schema.
- This is because of the richness of meta-data offered by a Physical Data Model.

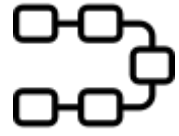


# Physical Model Characteristics

- The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, authorisations and other components are defined.

# Data Modelling  Summary

- Data modeling is the process of developing data model for the data to be stored in a Database.
- Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.
- There are three types of **conceptual**, **logical** and **physical**.
- The main aim of **conceptual model** is to establish the entities, their attributes and their relationships.
- A **logical model** defines the structure of the data elements and set the relationships between them.
- A **physical model** describes the database specific implementation of the data model.
- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.

# Entity-Relational Model 

- The ER or (Entity Relational Model) is a high-level **conceptual data model** diagram.
- Entity-Relation model is based on the notion of real-world entities and the relationship between them.
- ER modeling helps you to analyze data requirements systematically to produce a well-designed database.
- It is considered a best practice to complete ER modeling before implementing your database

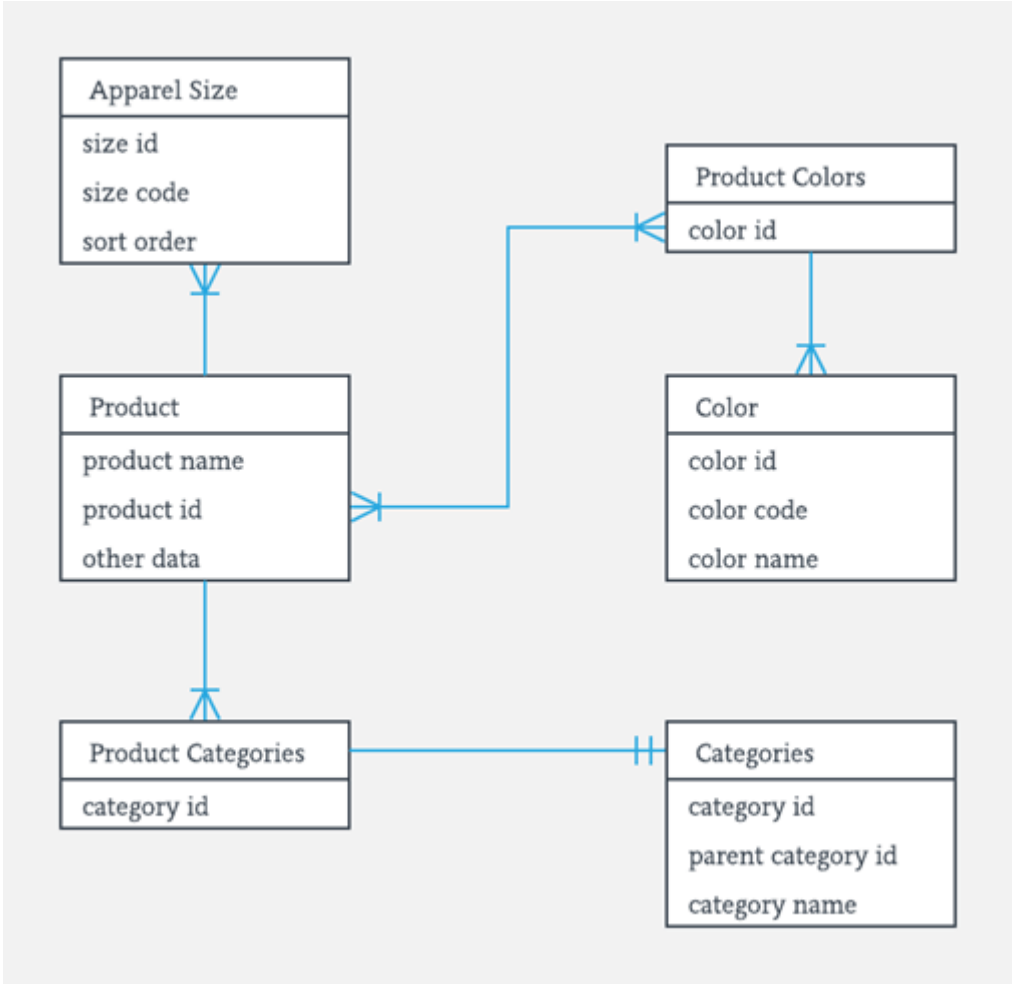
---

## History

- ER diagrams are a visual tool which is helpful to represent the ER model. It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network
- He aimed to use an ER model as a conceptual modeling approach

# What is ERD?

- Entity-relationship diagram (ERD) displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart.
- However, ER Diagram includes many specialized symbols, and its meanings make this model unique.


Apparel Size
size id
size code
sort order
Product Colors
color id
Product
product name
product id
other data
Color
color id
color code
color name
Product Categories
category id
Categories
category id
parent category id
category name

# Why use ER Diagrams?

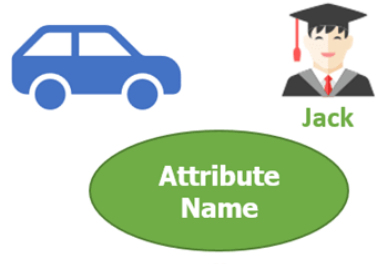
- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- ERD are translatable into relational tables which allows you to build databases quickly
- ERD can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram

# Components of ERD

- In a University database, we might have **entities** for **Students**, **Courses** and **Lecturers**.
- Students entity can have **attributes** like **Rollno**, **Name** and **DeptID**.
- They might have **relationships** with Courses and Lecturers.

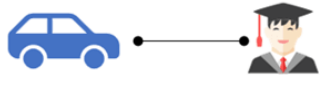

**Entity**

**Person,place,object,event or concept about which data is to be maintained**
**Example: Car, Student**




**Attribute**

**Property or characteristic of an entity**
**Example: Color of car Entity Name of Student Entity**



**Relation**


**Association between the instances of one or more entity types**
**Example: Blue Car Belongs to Student Jack**

# Summary 

- **Normalisation**
    - 1NF
    - 2NF (be 1NF and remove **partial dependency**)
    - 3NF (be 2NF and remove **transitive dependency**)
- **Data modelling**
    - Conceptual
    - Logical
    - Physical
- **Entity-Relational Model**
    - Conceptual representation
    - Introduction to ERD

# Continuous Assessment I 

- **CA1 – 20%** will be released on **Wendesday (02/10/2019)**
    - You will use visual database design tools to:
        - Create a **conceptual model** (ERD) that reflects:
            - Entities
            - Attributes and the attribute types
            - Relationships
    - You will transform the ER Model to create a **physical model**:
        - This will include:
            - All relations (i.e. tables)
            - All relationships, check integrity constraints (keys), domains, and use defined naming conventions.