Data Models and Design

Tushar B. Kute,
http://tusharkute.com

Data Model

- Data modeling (data modelling) is the process of creating a data model for the data to be stored in a database.
- This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules.
- Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data.
- Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.

Data Model

- The Data Model is defined as an abstract model that organizes data description, data semantics, and consistency constraints of data.
- The data model emphasizes on what data is needed and how it should be organized instead of what operations will be performed on data.
- Data Model is like an architect's building plan, which helps to build conceptual models and set a relationship between data items.
- The two types of Data Modeling Techniques are
  - Entity Relationship (E-R) Model
  - UML (Unified Modelling Language)

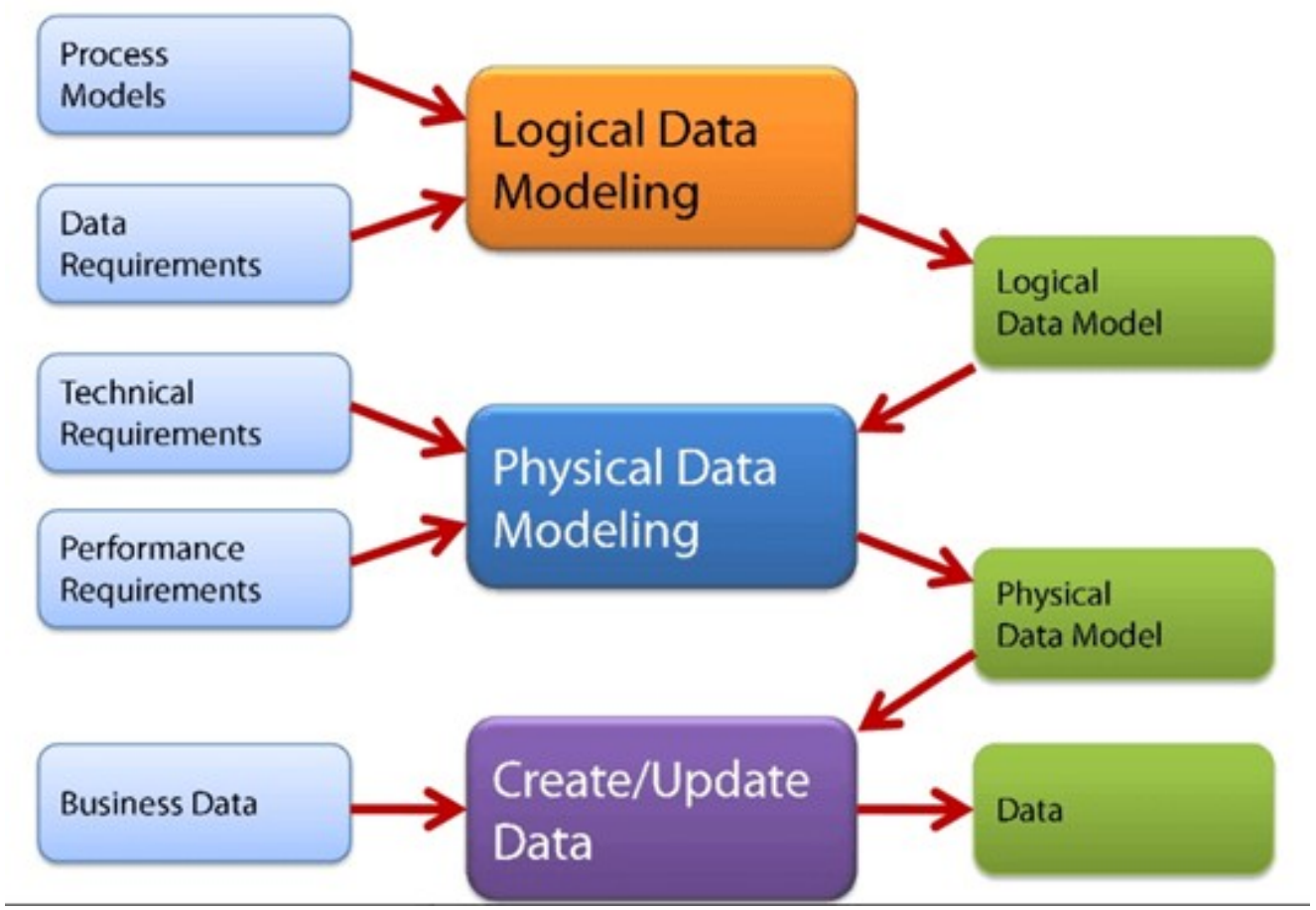
Why Data Model?

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

Types of Models

- Conceptual Data Model: This Data Model defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.
- Logical Data Model: 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 Data Model: This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

Types of Models



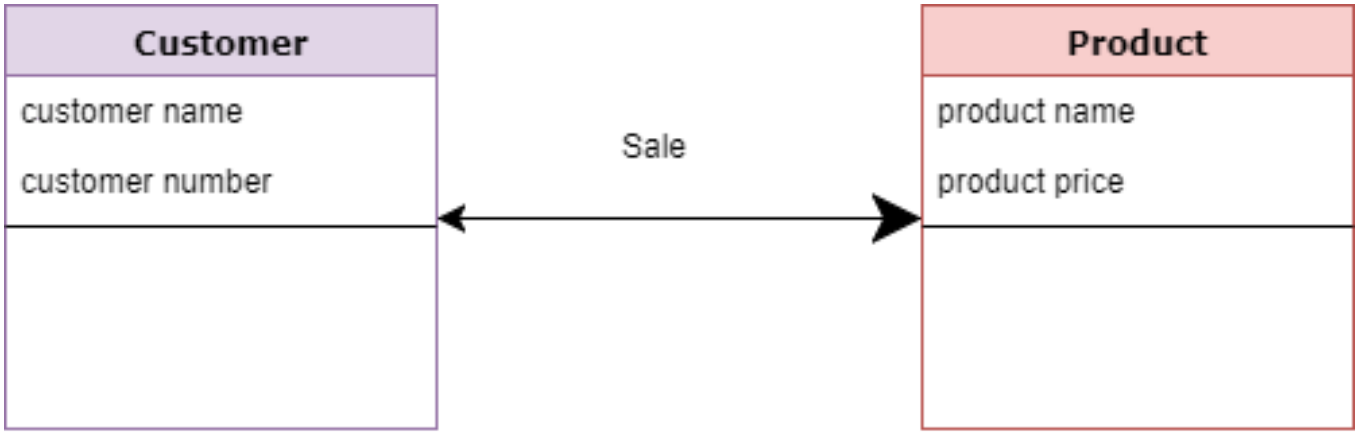
Conceptual Data Model

- A Conceptual Data Model is an organized view of database concepts and their relationships. The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships.
- In this data modeling level, there is hardly any detail available on the actual database structure. Business stakeholders and data architects typically create a conceptual data model.
- The 3 basic tenants of Conceptual Data Model are
  - Entity: A real-world thing
  - Attribute: Characteristics or properties of an entity
  - Relationship: Dependency or association between two entities

Conceptual Data Model

- Data model 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 Data Model



Conceptual Data Model: Features

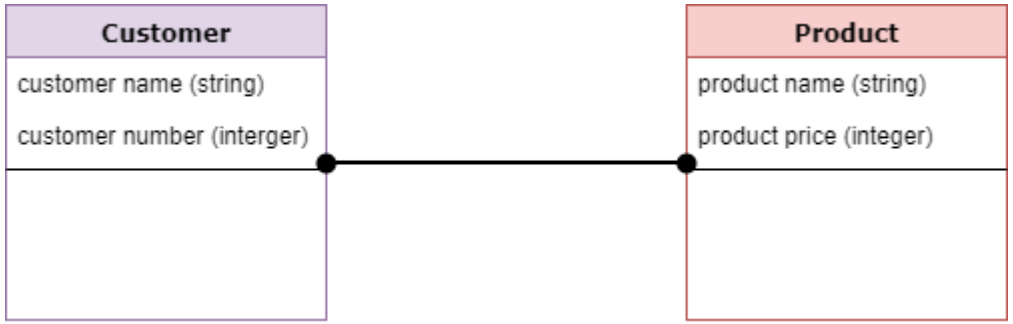
- Offers Organisation-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and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

Logical Data Model

- The Logical Data Model is used to define the structure of data elements and to set relationships between them.
- The logical data model adds further information to the conceptual data model elements.
- The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.

Logical Data Model

- At this Data Modeling level, no primary or secondary key is defined.
- At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.



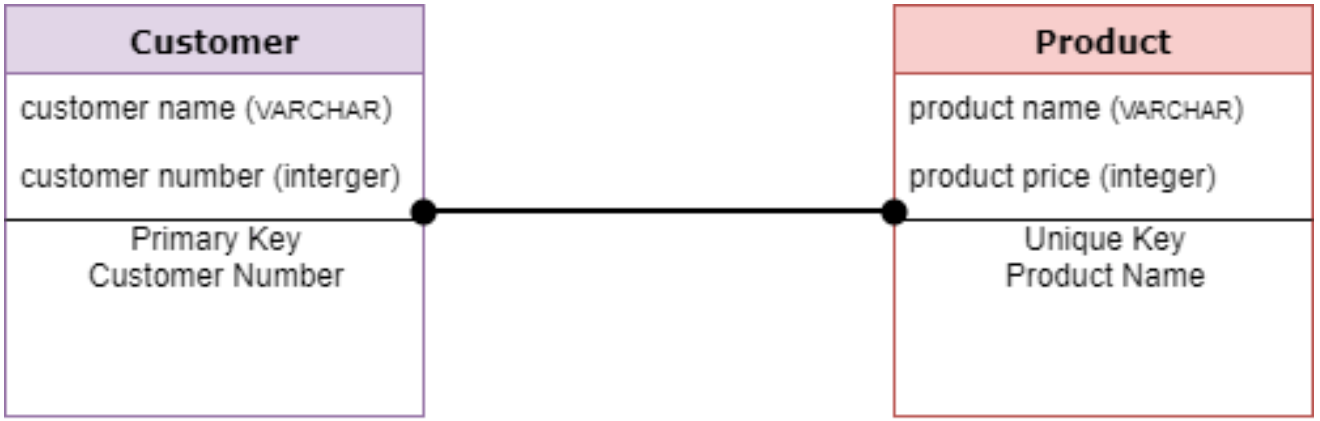
Logical Data Model: Features

- 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.
- Normalization processes to the model is applied typically till 3NF.

Physical Data Model

- A Physical Data Model describes a database-specific implementation of the data model.
- It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model.
- The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.

Physical Data Model



Physical Data Model: Features

- 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, and authorizations, etc. are defined.

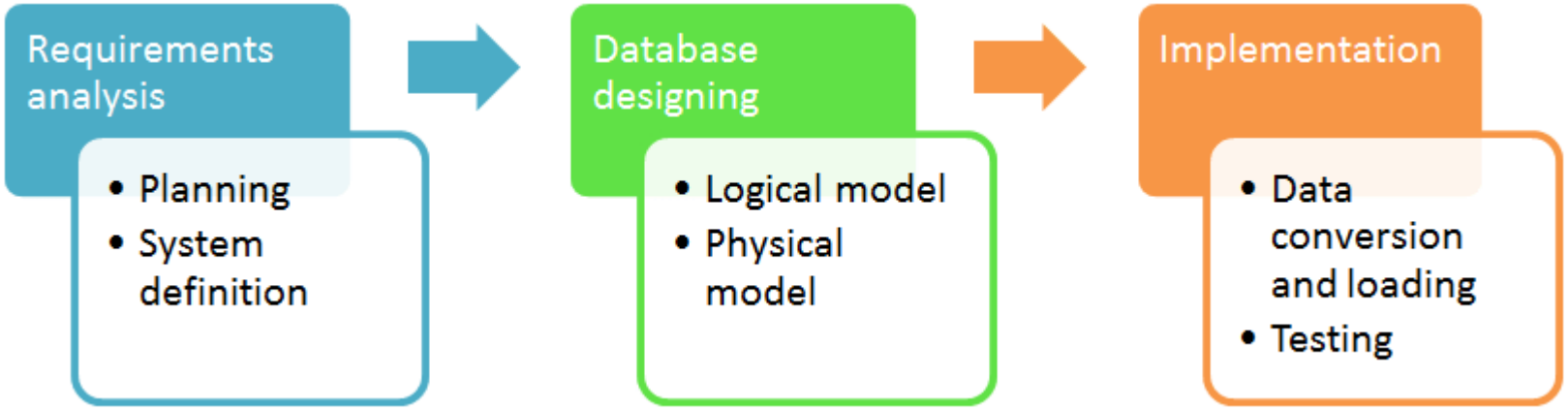
DBMS Design

- Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems.
- Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space.
- The database designer decides how the data elements correlate and what data must be stored.

DBMS Design

- The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.
- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Database Development Life Cycle



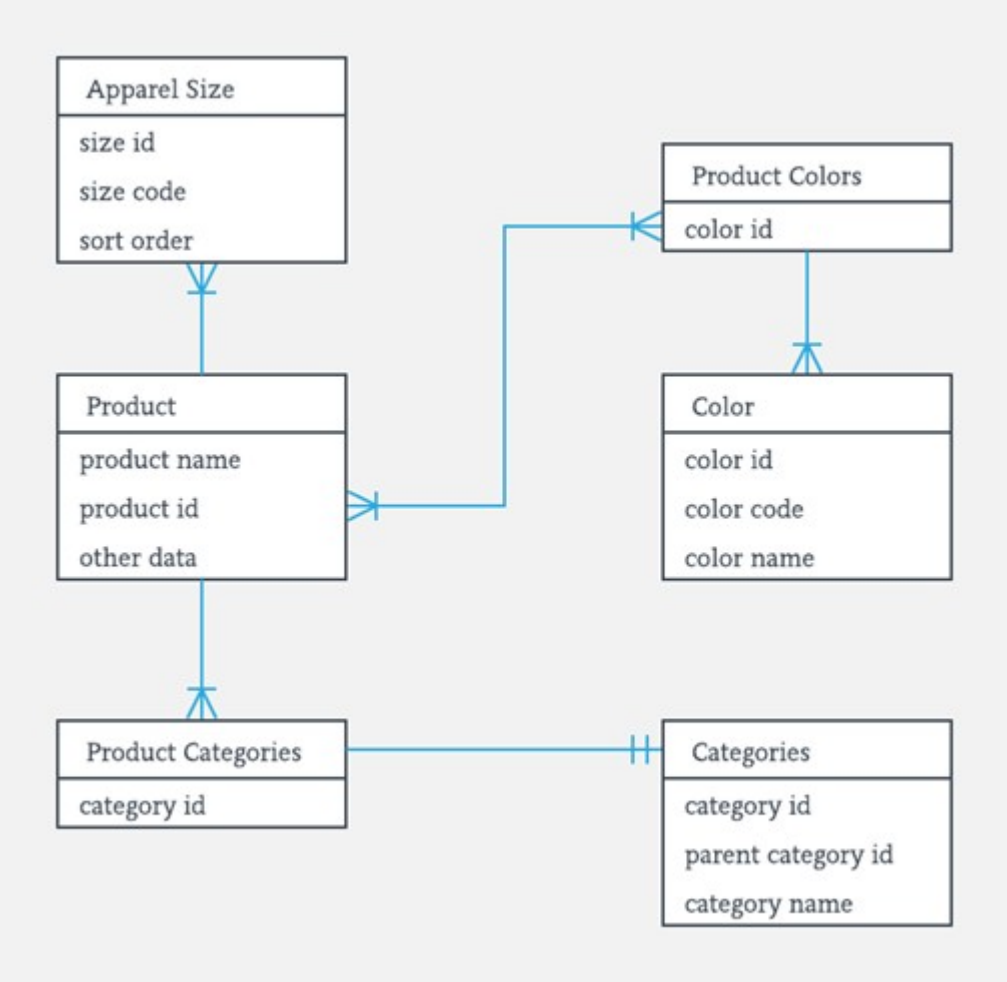
Database Design Techniques

- Two Types of Database Techniques
  - Normalization
  - ER Modeling

ER Diagram

- ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database.
- In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.
- ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

ER Diagram


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

ER Diagram: Why?

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams 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
- ERD Diagram allows you to communicate with the logical structure of the database to users

ER Diagram: Symbols and Notations

- Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes.
- There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

ER Diagram: Symbols and Notations

- Following are the main components and its symbols in ER Diagrams:
  - Rectangles: This Entity Relationship Diagram symbol represents entity types
  - Ellipses : Symbol represent attributes
  - Diamonds: This symbol represents relationship types
  - Lines: It links attributes to entity types and entity types with other relationship types
  - Primary key: attributes are underlined
  - Double Ellipses: Represent multi-valued attributes

ER Diagram: Symbols and Notations

ER Diagram: Symbols and Notations

Entity Name

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

Entity

- A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database.
- It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database.
- The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some ‘attributes’ which represent that entity.

Entity

- Examples of entities:
  - Person: Employee, Student, Patient
  - Place: Store, Building
  - Object: Machine, product, and Car
  - Event: Sale, Registration, Renewal
  - Concept: Account, Course

Entity Set

- Student
  - An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values.
  - Entities are represented by their properties, which also called attributes. All attributes have their separate values.
  - For example, a student entity may have a name, age, class, as attributes.

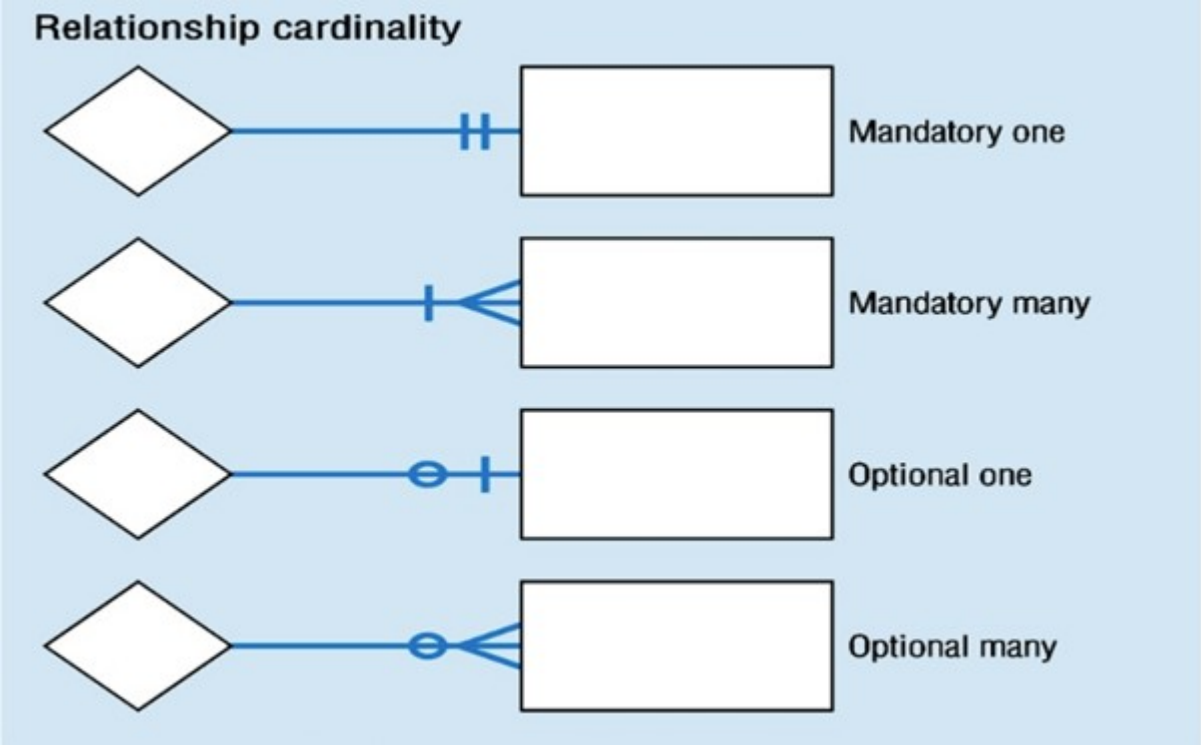
Relationships

- Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.
- Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

Cardinality

- Defines the numerical attributes of the relationship between two entities or entity sets.
- Different types of cardinal relationships are:
  - One-to-One Relationships
  - One-to-Many Relationships
  - Many to One Relationships
  - Many-to-Many Relationships

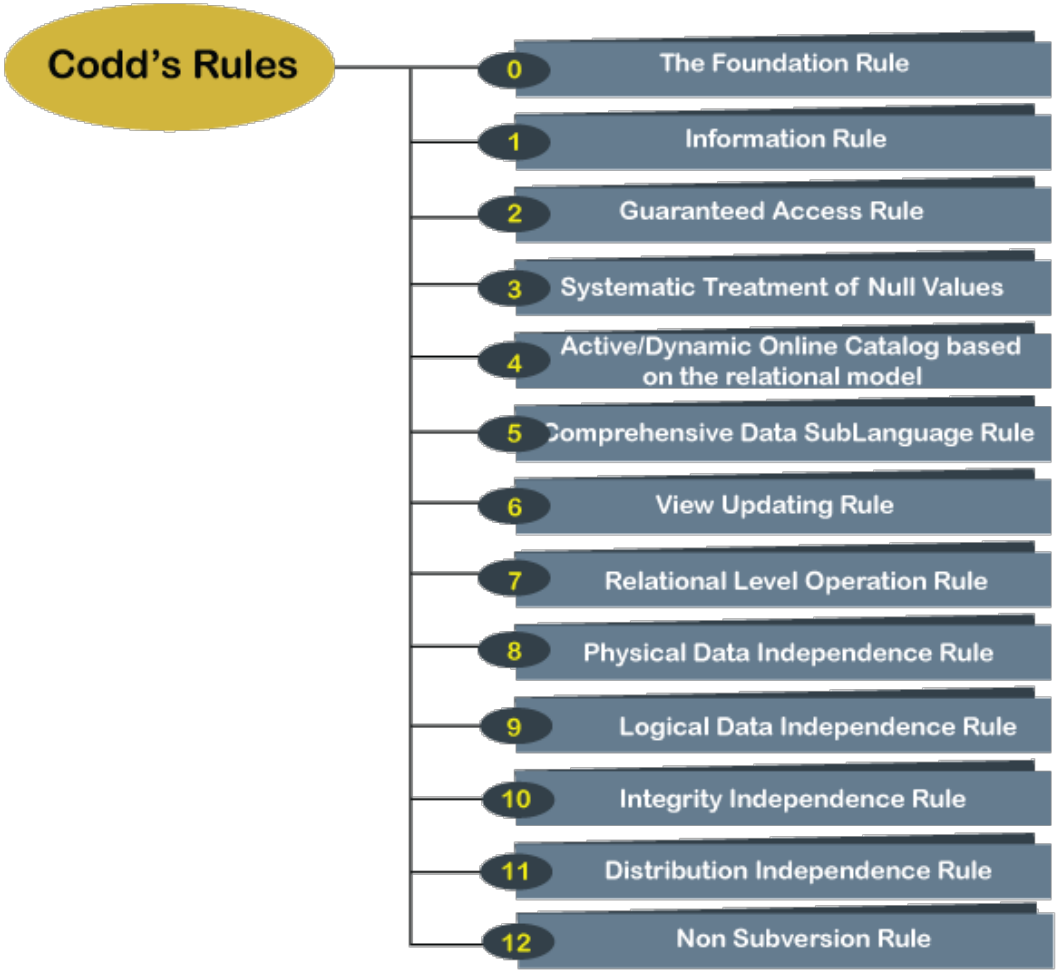
Cardinality



Codd's 12 Rules for RDBMS

- Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a Relational Database System (RDBMS).
- So, some rules define a database to be the correct RDBMS. These rules were developed by Dr. Edgar F. Codd (E.F. Codd) in 1985, who has vast research knowledge on the Relational Model of database Systems.
- Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS).

Codd's 12 Rules for RDBMS



Codd's 12 Rules for RDBMS

- Rule 0: The Foundation Rule
  - The database must be in relational form. So that the system can handle the database through its relational capabilities.
- Rule 1: Information Rule
  - A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.
- Rule 2: Guaranteed Access Rule
  - Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Codd's 12 Rules for RDBMS

- Rule 3: Systematic Treatment of Null Values
  - This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.
- Rule 4: Active/Dynamic Online Catalog based on the relational model
  - It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Codd's 12 Rules for RDBMS

- Rule 5: Comprehensive Data SubLanguage Rule
  - The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations.
  - If the database allows access to the data without any language, it is considered a violation of the database.

Codd's 12 Rules for RDBMS

- Rule 6: View Updating Rule
  - All views table can be theoretically updated and must be practically updated by the database systems.
- Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule
  - A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Codd's 12 Rules for RDBMS

- Rule 8: Physical Data Independence Rule
  - All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application.
  - If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

Codd’s 12 Rules for RDBMS

- Rule 9: Logical Data Independence Rule
  - It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application).
  - For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

Codd’s 12 Rules for RDBMS

- Rule 10: Integrity Independence Rule
  - A database must maintain integrity independence when inserting data into table's cells using the SQL query language.
  - All entered values should not be changed or rely on any external factor or application to maintain integrity.
  - It is also helpful in making the database-independent for each front-end application.

Codd's 12 Rules for RDBMS

- Rule 11: Distribution Independence Rule
  - The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users.
  - Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site.
  - The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

Codd's 12 Rules for RDBMS

- Rule 12: Non Subversion Rule
  - The non-submersion rule defines RDBMS as a SQL language to store and manipulate the data in the database.
  - If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com