I.E.S Rodrigo Caro

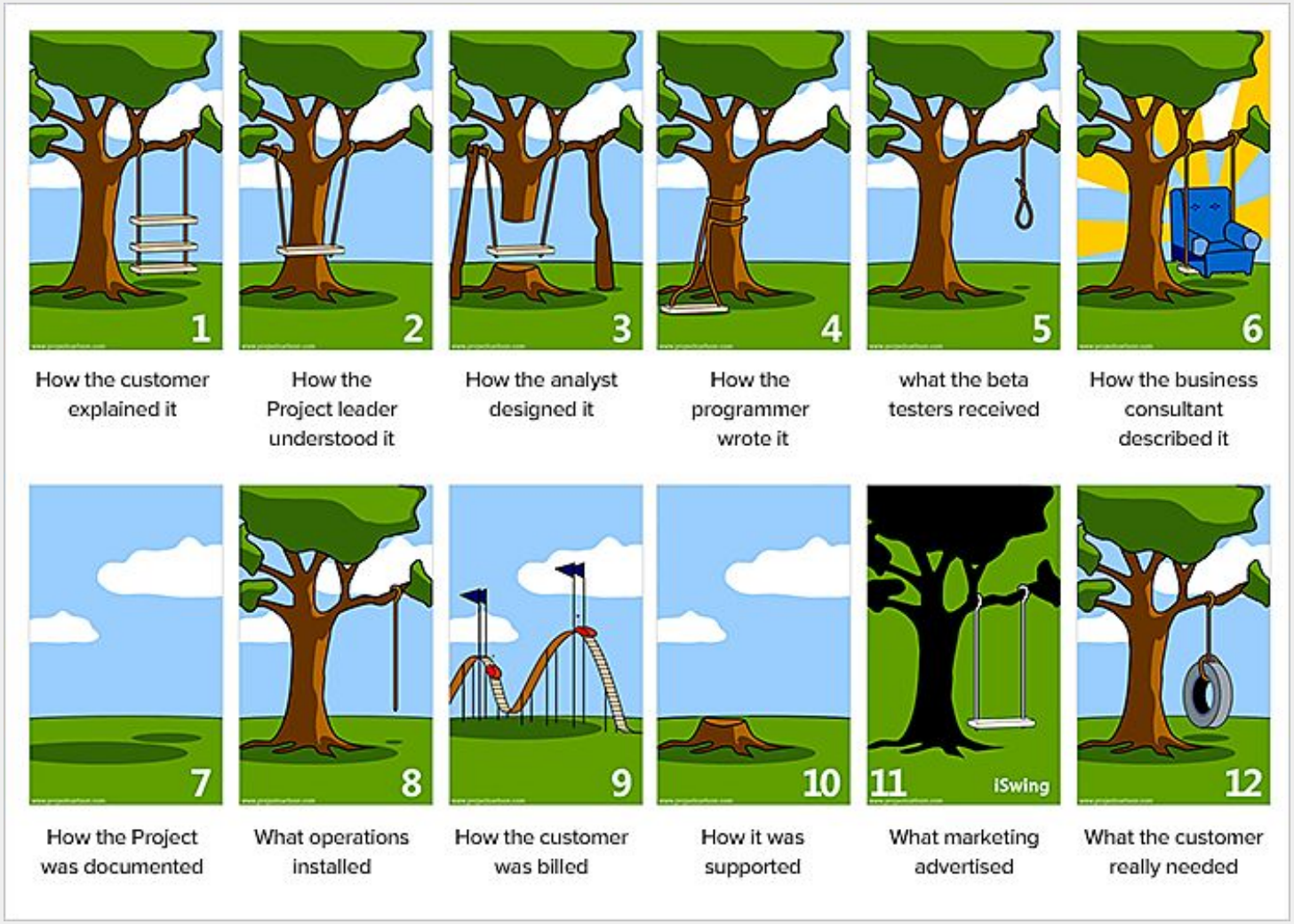
Departamento de Informática
www.informaticarodrigocaro.es
Informática Rodrigo Caro

Unit 2: The Entity-Relationship Model

ASIR

Index

Database Lifecycle

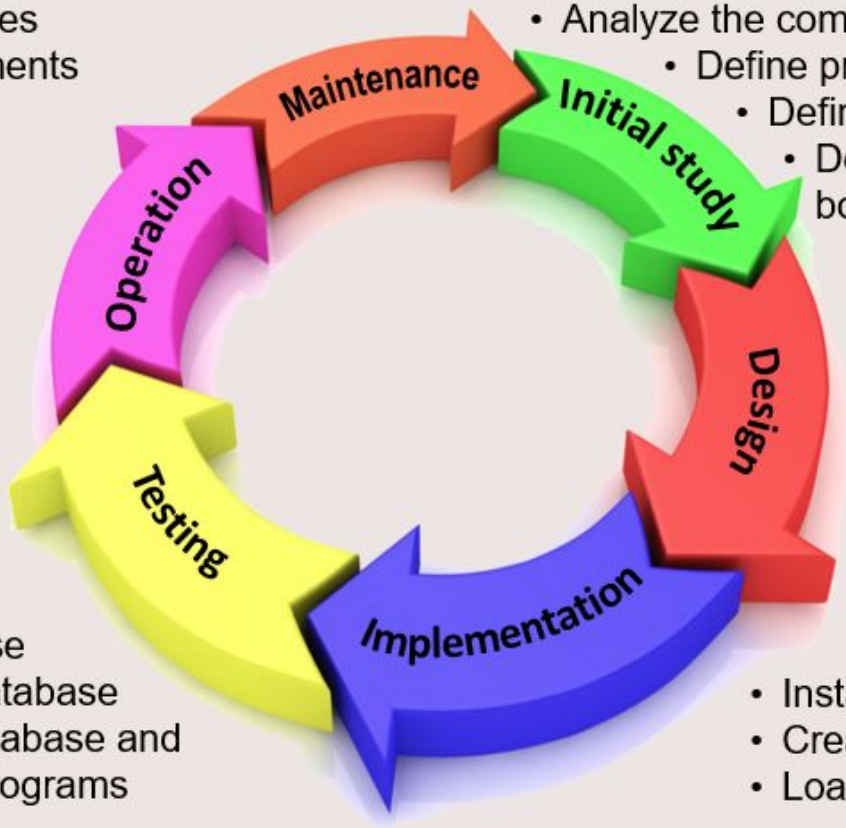


Database Lifecycle

According to Coronel and Morris (2019), the database life cycle consists of six phases, namely:

1. Database initial study
2. Database design
3. Implementation and loading
4. Testing and evaluation
5. Operation
6. Maintenance and evolution.

Database Lifecycle



- Introduce changes
- Make enhancements

- Analyze the company situation
- Define problems and constraints
- Define objectives
- Define scope and boundaries

- Produce the required information flow

- Create the conceptual design
- DBMS software selection
- Create the logical design
- Create the physical design

- Test the database
- Fine-tune the database
- Evaluate the database and its application programs

- Install the DBMS
- Create the database(s)
- Load or convert the data

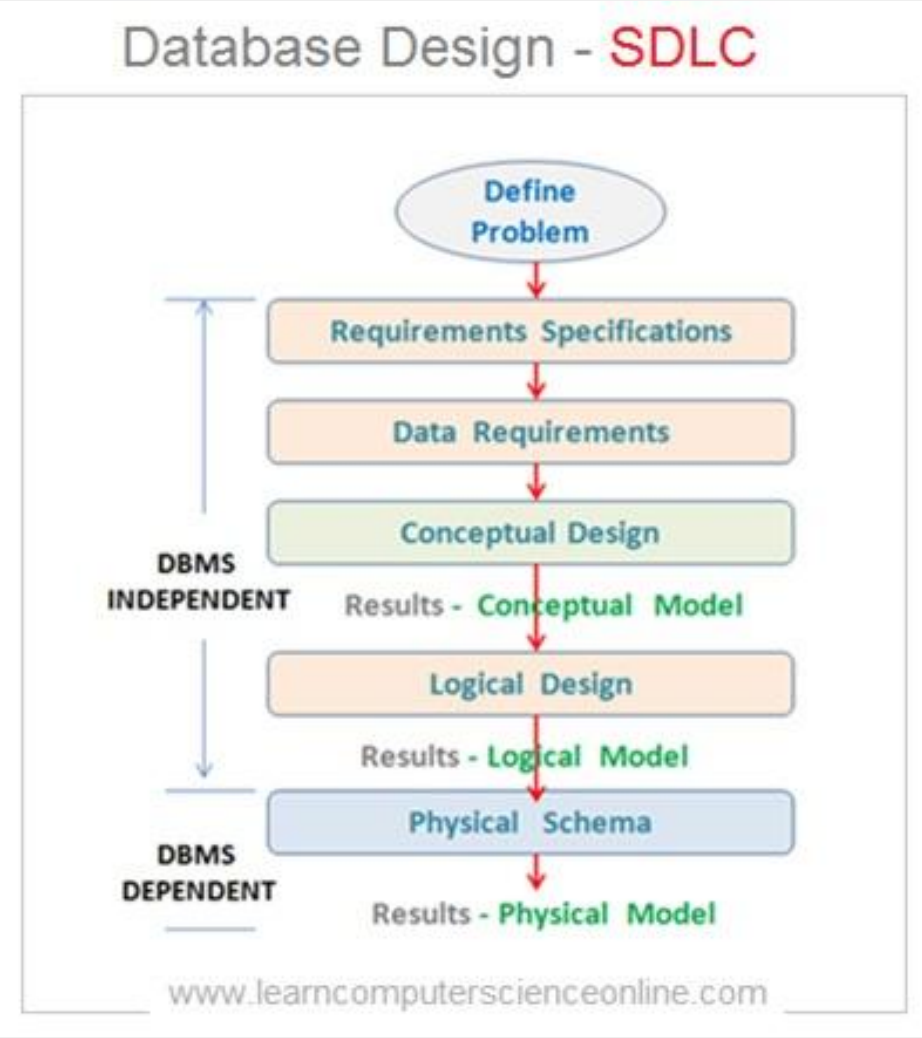
Database Lifecycle: Initial Study

The first we should think about is:

1. **Purpose**. The purpose of a database depends on the nature of the data and how they relate to each other.
2. **Universe discourse.** This is the context. It can be defined as the set of needs to be fulfilled. They are also called information requirements, as they are the needs of the data.
3. **Planning** – This stages of database design concepts are concerned with planning of entire Database Development Life Cycle.

Exercises

Database Lifecycle: Design



Database Lifecycle: Design
Requirements Collection

- **Functional requirements** are the document that includes the detailed requirements provided by the users. Phases:
  a. **Customers interviews**. The database designers have to interview the customers (database users) to understand the proposed system.
  b. Build **data requirements document.** It should describe in natural language:
     - what the data items are
     - what attributes they have
     - what constraints apply
     - the relationships that hold between the data items.
  c. Review the final document with the customers to get the global **agreement**.

Database Lifecycle: Design
Conceptual Data Model

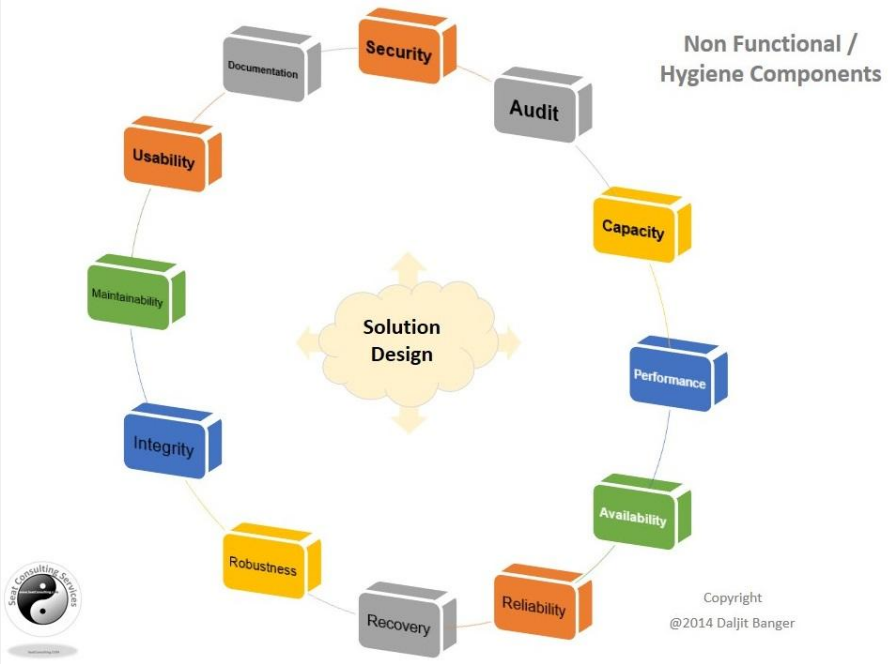
- **Data analysis** begins with the statement of data requirements and then **produces a conceptual data model**.
- The **conceptual data model** then is a **formal representation** of what data a database should contain and the constraints the data must satisfy. This should be expressed in terms that are independent of how the model may be implemented. As a result, analysis focuses on the questions, "**What is required**?" not "How is it achieved?"
- Here we could build de **Entity-Relationship model**.

Database Lifecycle: Design
Logic Design

- Database design starts with a conceptual data model and **produces a specification of a logical schema**; this will determine the **specific type of database system** (network, relational, object-oriented) that is required.
- The relational representation is still independent of any specific DBMS; it is another conceptual data model.
- We could do some transformation to optimize the model. Normalized database.

Database Lifecycle: Design
Physical Schema

- The goal of this phase is to get the **internal schema.**
- In this phase, it is necessary to know the hardware and software resources available. **Non-functional requirements** such as data security policies will be taken into account.



Conceptual Data Model: ER model

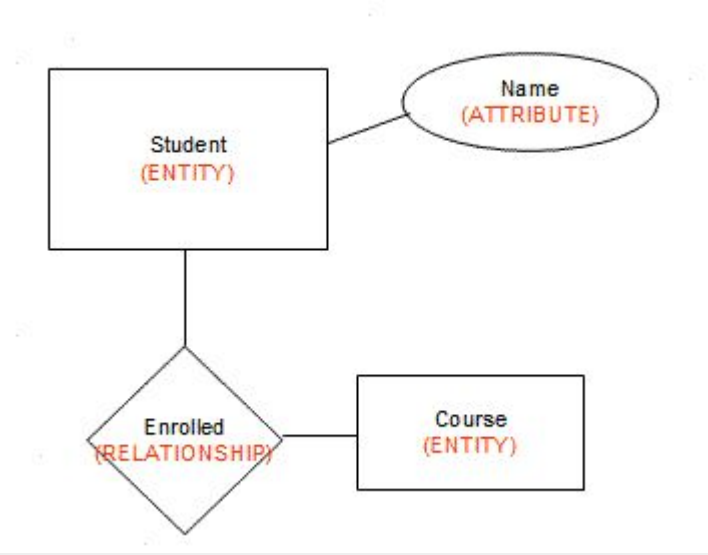
- **Definition**: It is also called the entity-relationship model and it is a special graphical representation technique that incorporates information about the data and and the relationship between them.
- It is independent of its final structure. (1976, Codd)
- ER modelling is based on two concepts:
  a. **Entities with attributes (data)**
  b. **Relationships**, defined as the associations or interactions between entities

E/R Model

- **Definition**: It is also called the entity-relationship model and it is a special graphical representation technique that incorporates information about the data and and the relationship between them. (1976, Codd). Main
    i) It reflects only the existence of the data, not what is done with it.
    ii) It is independent of specific databases and operating systems.
    iii) It does not take into account non functional requirements.

E/R Model

- ER modelling is based on two concepts:
  a) **Entities with attributes (data)**
  b) **Relationships**, defined as the associations or interactions between entities



E/R Model: Entities

An entity is an object in the real world with an independent existence that can be differentiated from other objects.

- An entity might be:
  - An object with physical existence
  - An object with conceptual existence.
- All instances of an entity type must have the same properties.
- Each instance, occurrence or instance of a type of entity must be distinguishable from the others.
- It is recommended to write the name of the type of entity in singular.

E/R Model: Entities

Weak Entities

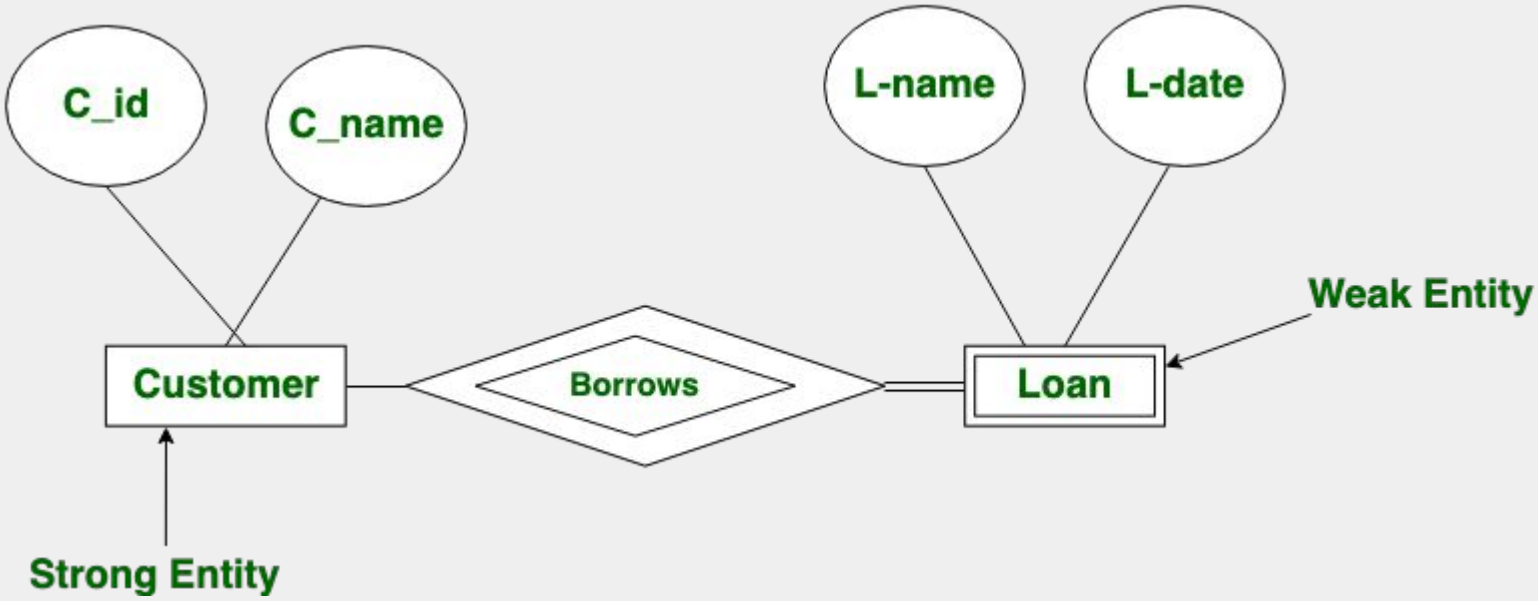
A **weak entity** is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key.

Weak entity is depend on strong entity to ensure the existence of weak entity.

Weak entities are represented with double rectangular box in the ER Diagram and the identifying relationships are represented with double diamond. Partial Key attributes are represented with dotted lines.

E/R Model: Entities

Weak Entities



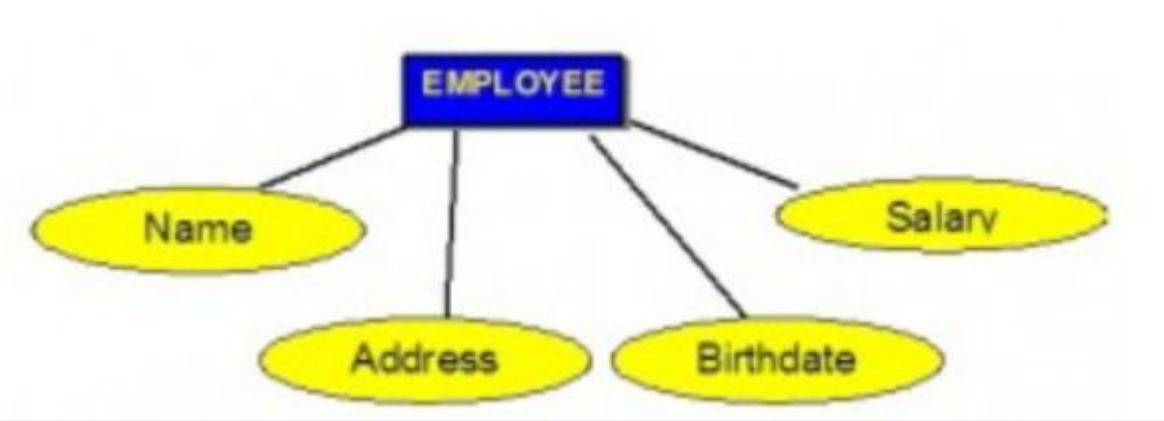
A loan entity can not be created for a customer if the customer doesn't exist

E/R Model: Attributes

An entity **contains a set of attributes**. They are the properties which define the entity.

Each **attribute** has a **name** and is associated with an entity and a **domain of legal values**. However, the information about attribute domain is not presented on the ERD.

Each attribute is represented by an **oval with a name inside.**

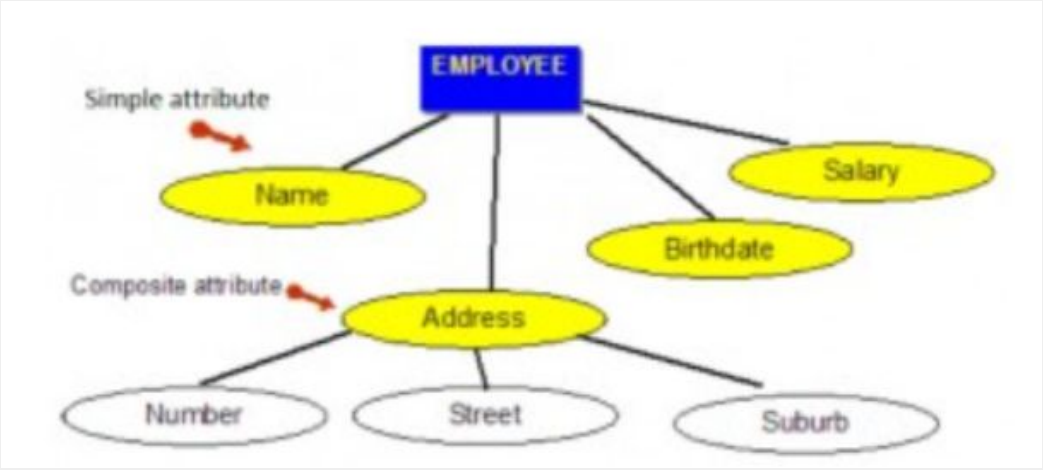


E/R Model: Types of Attributes

- **Simple attributes** are those drawn from the atomic value domains; they are also called single-valued attributes. In the COMPANY database, an example of this would be:

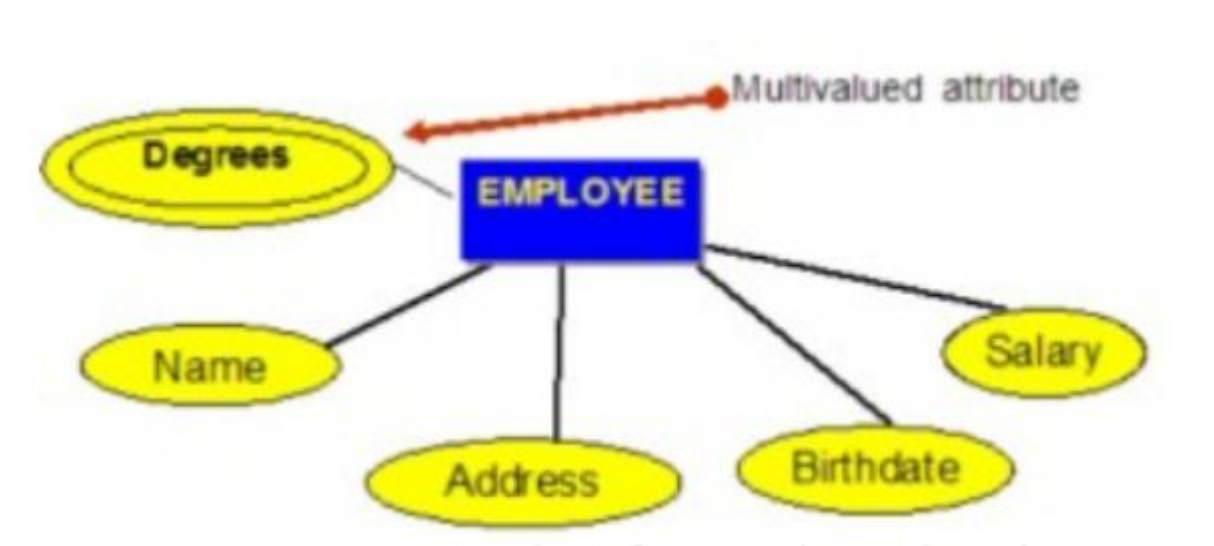
  Name = {John} ; Age = {23}

- **Composite attributes** are those that consist of a hierarchy of attributes.



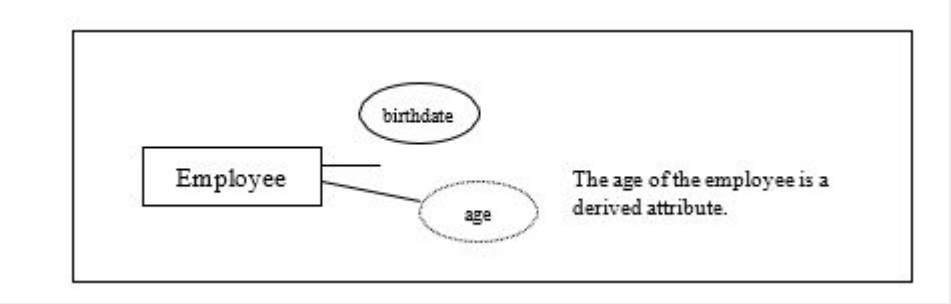
E/R Model: Types of Attributes

- **Multivalued attributes** are attributes that have a set of values for each entity.



E/R Model: Types of Attributes

- **Derived attributes** are attributes that contain values calculated from other attributes. Age can be derived from the attribute Birthdate. In this situation, Birthdate is called a stored attribute, which is physically saved to the database.



E/R Model: Keys

- **Key attribute** is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set.
- **Types:**
  - **Candidate Key** is a simple or composite key that is unique and minimal. It is unique because no two rows in a table may have the same value at any time. It is minimal because every column is necessary in order to attain uniqueness.
  - **Composite Key** is composed of two or more attributes, but it must be minimal. For example: first and last name.

E/R Model: Keys

- **Key attribute** is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set.
- The list of attributes which are key attributes are the candidate attributes.
- **Primary Key (PK)** From all possible candidate keys which identifies the whole entity set, the primary key is the chosen candidate. It must uniquely identify tuples in a table and not be null. The primary key is indicated in the ER model by underlining the attribute.

E/R Model: Keys

- **A foreign key (FK)** is an attribute in a table that references the **primary key in another table OR it can be null.** Both foreign and primary keys must be of the same data type.
- A **secondary key** is used strictly for **retrieval purposes** and accessing records . These keys **do not have to be unique**. The term secondary key is also occasionally used as a synonym for alternate key. Example: first and last name
- **Nulls** is a special symbol, independent of data type, which means either unknown or inapplicable. It does **not mean zero or blank**. It represents an unknown attribute value.

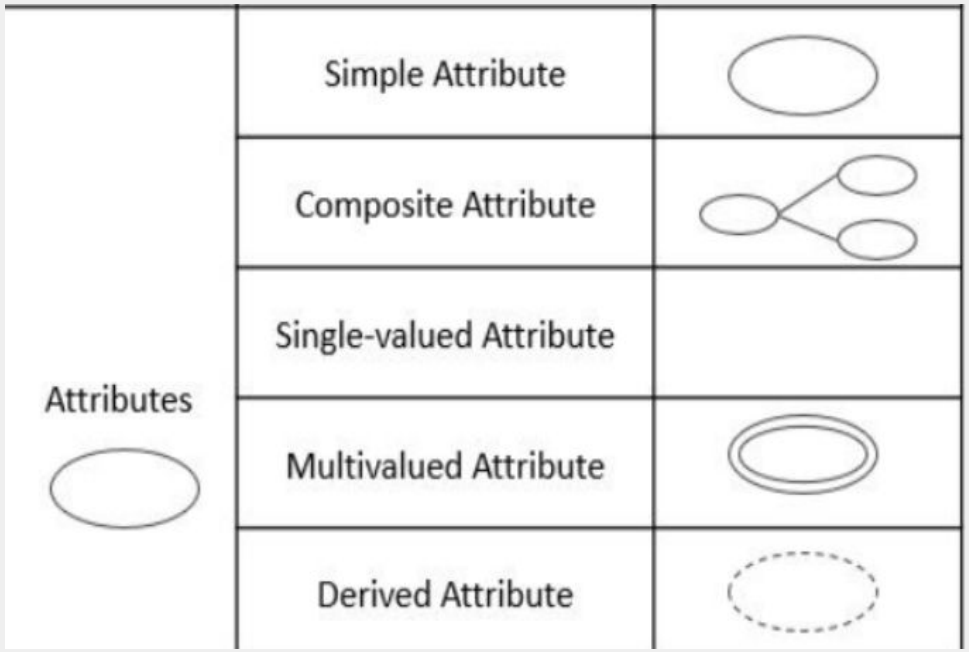
E/R Model: Keys

- **A foreign key (FK)** is an attribute in a table that references the **primary key in another table OR it can be null.** Both foreign and primary keys must be of the same data type.
- A **secondary key** is used strictly for **retrieval purposes** and accessing records . These keys **do not have to be unique**. The term secondary key is also occasionally used as a synonym for alternate key. Example: first and last name
- **Nulls** is a special symbol, independent of data type, which means either unknown or inapplicable. It does **not mean zero or blank**. It represents an unknown attribute value.

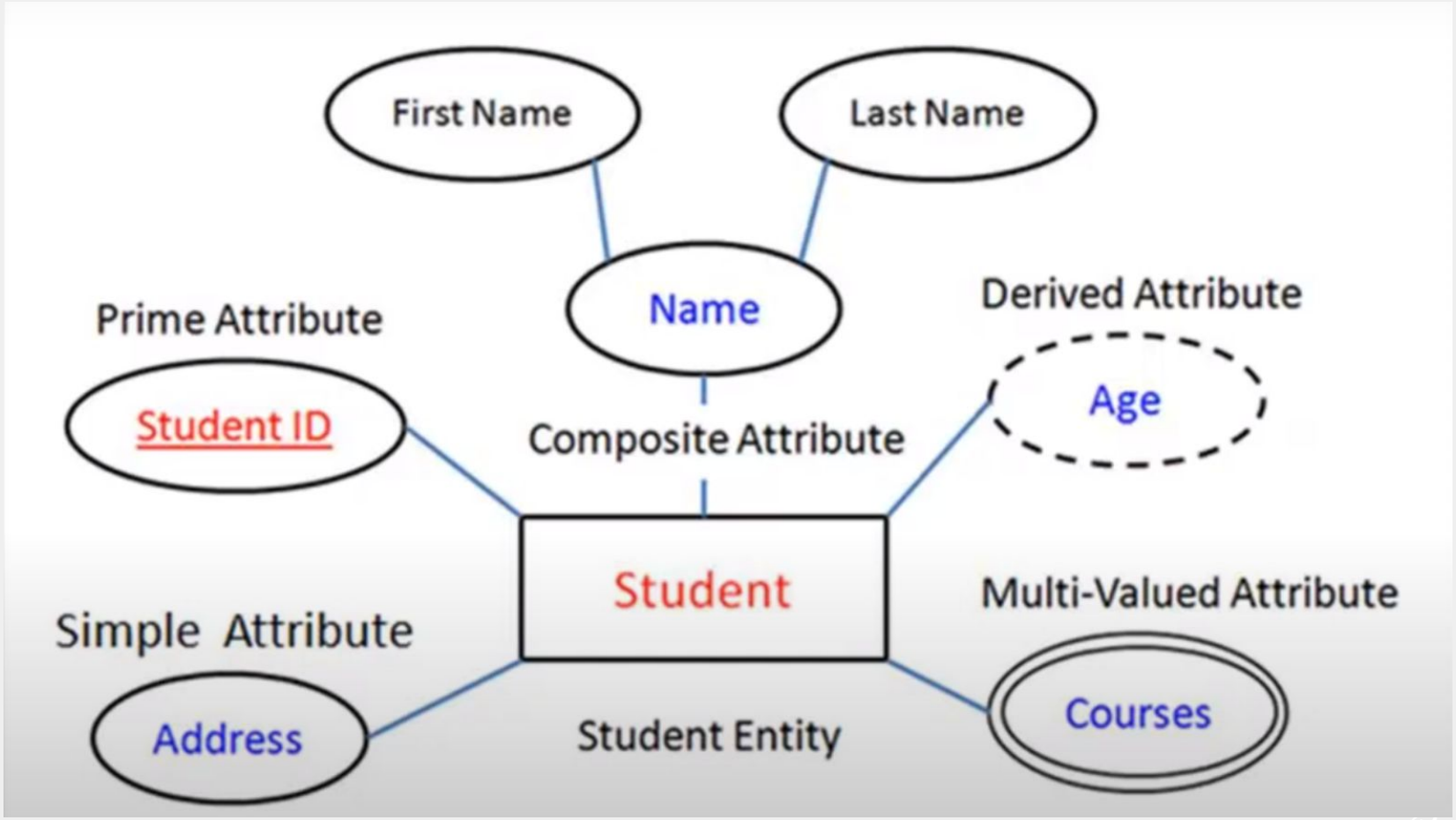
E/R Model: Exercise

Design the E/R for Student entity.
You should:

- Identify PK
- Classify each attribute according to the following schema.



E/R Model: Exercise: Solution



E/R Model: Relationships

- Relationships are are used to connect related information between entities.
- Relationship strength is based on how the primary key of a related entity is defined.
  - **A weak,** or non-identifying, **relationship** exists if the primary key of the related entity does **not contain a primary key** component of the parent entity. Example: company database examples include: Customer(**CustID**, CustName), Order(**OrderID**, CustID, Date)
  - A **strong**, or identifying, **relationship** exists when the primary key of the related entity **contains the primary key** component of the parent entity. Examples: Course(**CrsCode**, DeptCode, Description) Class(**CrsCode, Section**, ClassTime…)

E/R Model: Relationships

- Relationships are are used to connect related information between entities.
- Relationship strength is based on how the primary key of a related entity is defined.
  - **A weak,** or non-identifying, **relationship** exists if the primary key of the related entity does **not contain a primary key** component of the parent entity. Example: company database examples include: Customer(**CustID**, CustName), Order(**OrderID**, CustID, Date)
  - A **strong**, or identifying, **relationship** exists when the primary key of the related entity **contains the primary key** component of the parent entity. Examples: Course(**CrsCode**, DeptCode, Description) Class(**CrsCode, Section**, ClassTime…)

E/R Model: Type of Relationships

Cardinality: the cardinality of a relationship is the number of tuples (rows) in a relationship. Cardinality ratio means to denote **the number of entities to which another entity can be linked through a certain relation set.**

Let's start by focusing on binary relationships (2 entities) and their cardinalities. We could consider these types:

- one-to-one
- one-to-many
- many-to-one
- many-to-many
- Unary (recursive)

E/R Model:

- **Referential integrity** refers to the relationship between tables. Because each table in a database must have a **primary key**, this primary key can appear in other tables because of its relationship to data within those tables. When a primary key from one table appears in another table, it is called a **foreign key**.
- Foreign keys join tables and establish dependencies between tables.
- Referential integrity is the logical dependency of a foreign key on a primary key.
- For example, you can not delete a row with a primary key if there are other rows in other tables that contain the same key as foreign key. To do keeping the referential integrity: you should delete on cascade, for example.

E/R Model: Type of Relationships

One to one (1:1) relationship is the relationship of **one entity to only one other entity, and viceversa**.

It should be rare in any relational database design. In fact, it could indicate that two entities actually belong in the same table.

A person can have only one passport in force and a passport belongs to only one person.

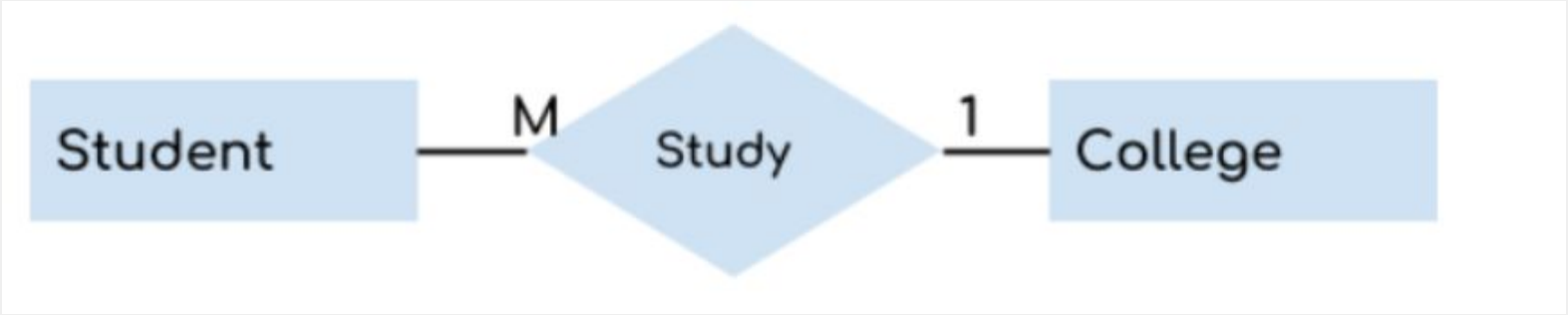
E/R Model: Type of Relationships

One to many (1:m) When a **single instance of an entity is associated with more than one instances of another** entity then it is called one to many relationship.

A customer can place many orders but a order cannot be placed by many customers.

E/R Model: Type of Relationships

Many to one (m:1) When **more than one instances of an entity is associated with a single instance** of another entity then it is called many to one relationship.



Many students can study in a single college but a student cannot study in many colleges at the same time.

E/R Model: Type of Relationships

Many to many (m:n) When **more than one instances of an entity is associated with more than one instances of another entity** then it is called many to many relationship.

An student can be assigned to many projects and a project can be assigned to many students.

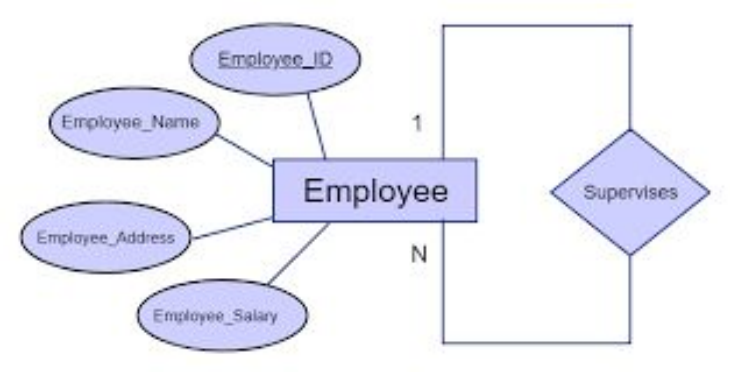
E/R Model: Type of Relationships

- It cannot be implemented as such in the relational model!!
- To model these relations we could create a third table.
- The primary key of this third table is composed of the both other primary keys
- Additional attributes may be assigned as needed.
- In our previous example, we should create a third table called projectAssignment. Its primary key is composed of idStudent and idProject

E/R Model: Type of Relationships

Unary relationship, it is also called **recursive**, is one in which a **relationship exists between occurrences of the same entity se**t.

In this relationship, the primary and foreign keys are the same, but they represent two entities with different roles.



For some entities in a unary relationship, a separate column can be created that refers to the primary key of the same entity set.

E/R Model: TODO Ternary Relationships

TODO

References

- Rafael Lozano, Introducción a las Bases de Datos
- Antonio Postigo Palacios, Gestión de base de datos Editorial Paraninfo Madrid 2022
- https://opentextbc.ca/dbdesign01/chapter/chapter-13-database-development-process/
- https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/comment-page-1/
- https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/
- https://jcsites.juniata.edu/faculty/rhodes/dbms/ermodel.htm
- https://prepinsta.com/dbms/weak-entity-and-strong-entity/
- https://www.datasciencecentral.com/key-attributes-in-er-diagrams/
- https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/
- https://beginnersbook.com/2015/04/e-r-model-in-dbms/
- https://www.geeksforgeeks.org/cardinality-in-dbms/
- https://www.ibm.com/docs/en/informix-servers/14.10?topic=integrity-referential