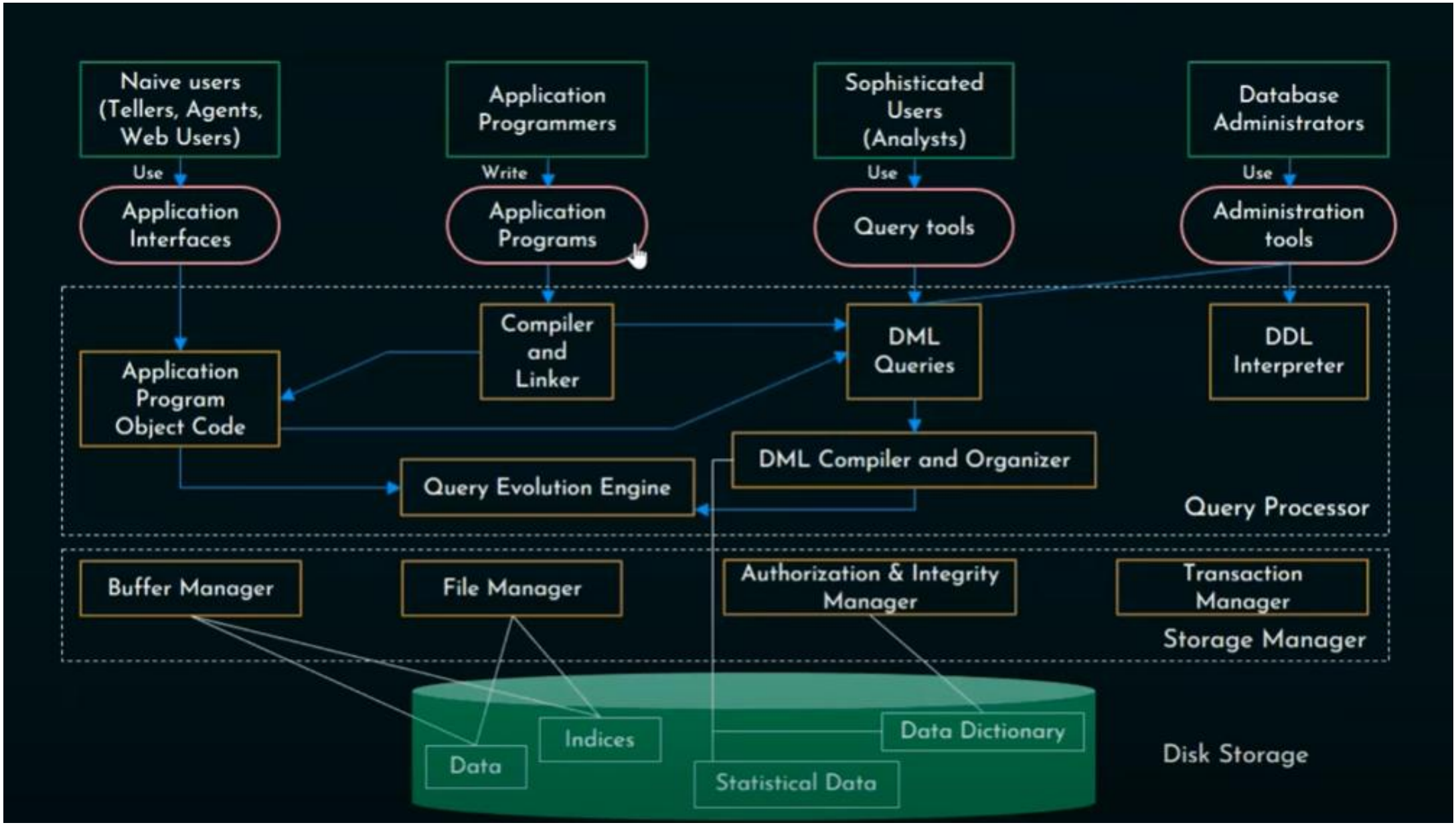

Naive users (Tellers, Agents, Web Users)
Application Programmers
Sophisticated Users (Analysts)
Database Administrators
Use
Write
Use
Use
Application Interfaces
Application Programs
Query tools
Administration tools
Compiler and Linker
DML Queries
DDL Interpreter
Application Program Object Code
DML Compiler and Organizer
Query Evolution Engine
Query Processor
Buffer Manager
File Manager
Authorization & Integrity Manager
Transaction Manager
Storage Manager
Indices
Data Dictionary
Data
Statistical Data
Disk Storage

Attributes Types

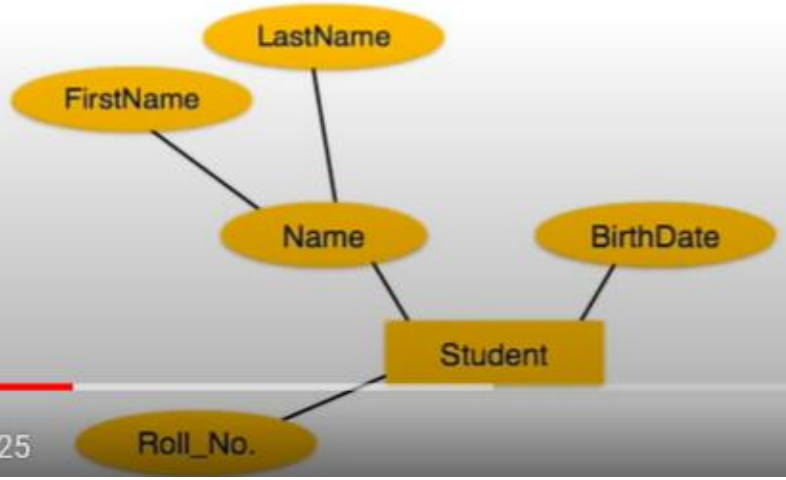
- **Simple** and **Composite attributes**
- **Single-valued** and **Multi-valued attributes**
- **Stored** and **Derived attributes**
- **Key Attribute**

1. Simple attributes

- **Simple attributes** are atomic values, which cannot be divided further.
- For example:
  - a student's mobile number is an atomic value of 10 digits.
  - a Birth Date is an atomic value of date-month-year

2. Composite attributes

- **Composite attributes** are made of more than one simple attribute.
- A composite attribute is divided in a tree like structure.
- For example:
  - A student's complete ***name*** may have **first_name** and **last_name.**
  - An ***address*** may have **street, city, state, country** and **pin code**



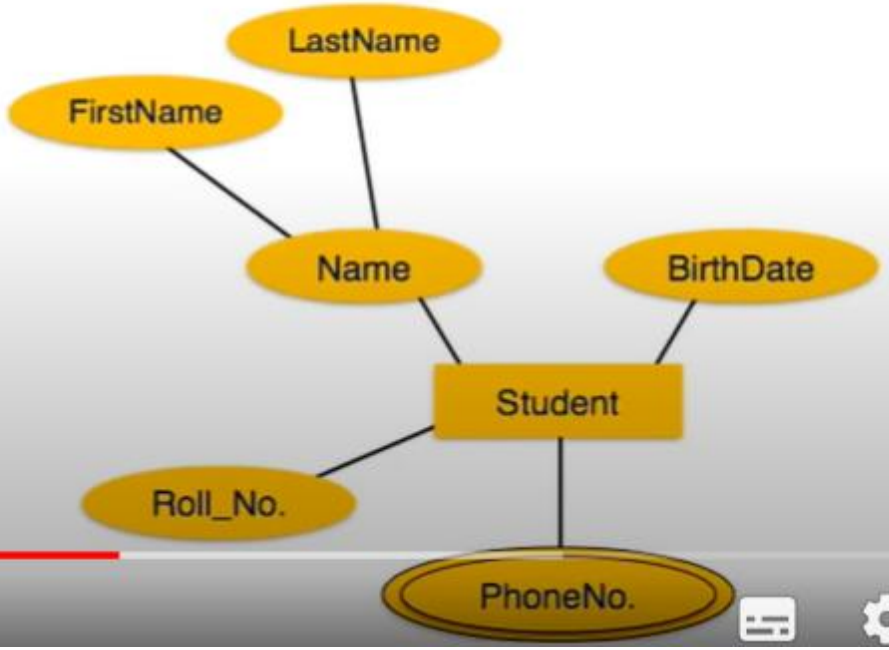
3. Single-valued attribute

- **Single-value attributes** contain single value.
- For example:
  - Social_Security_Number, Aadhar_card_no, Roll_no

4. Multi-valued attribute

- **Multi-valued attributes** may contain more than one values.
- Represented by **double-ellipse** 

- For example: a person can have more than one phone number, email_address, etc.



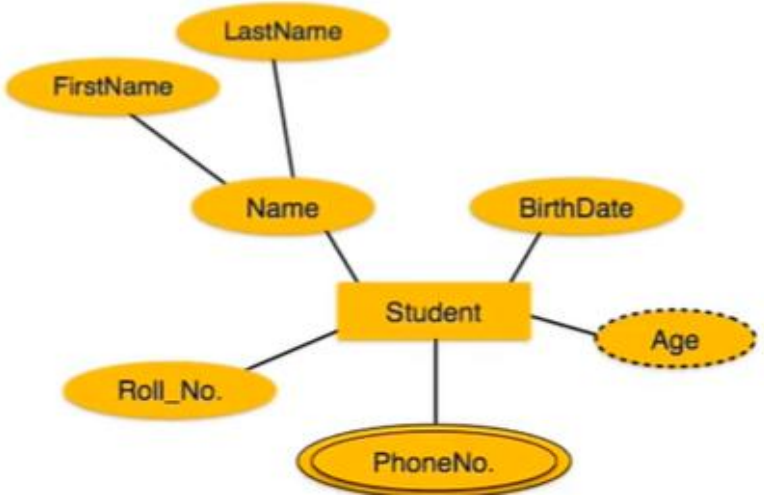
5. Stored attribute

- **Stored attributes** are physically stored in the database
- Mostly all attributes are stored in database except few ones
- For example: Roll_no, Name, birth_date, phone_no

6. Derived attribute

- **Derived attributes** are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
- **Derived** attributes are depicted by **dashed ellipse.**
- For example:
  - **age** can be derived from *data_of_birth*

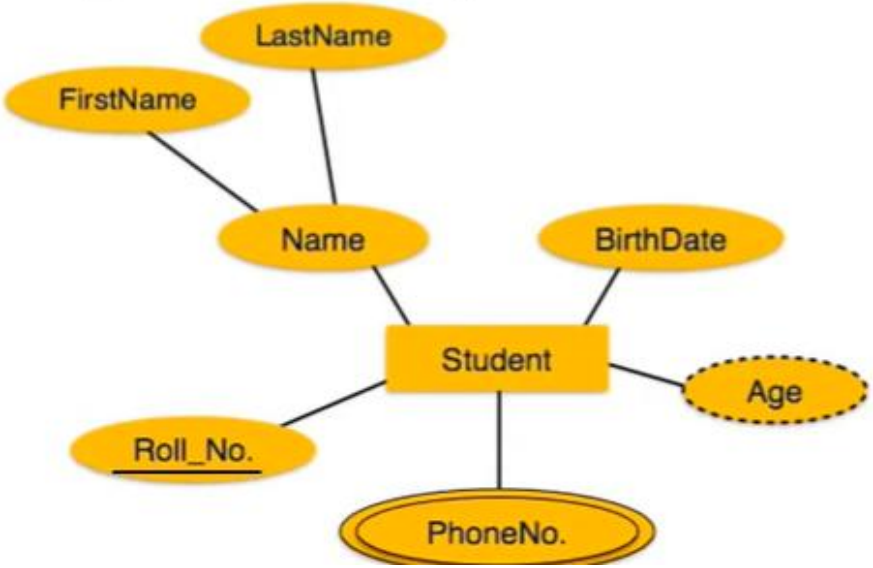
  - **average_salary** in a department should not be saved directly in the database, instead it can be derived.



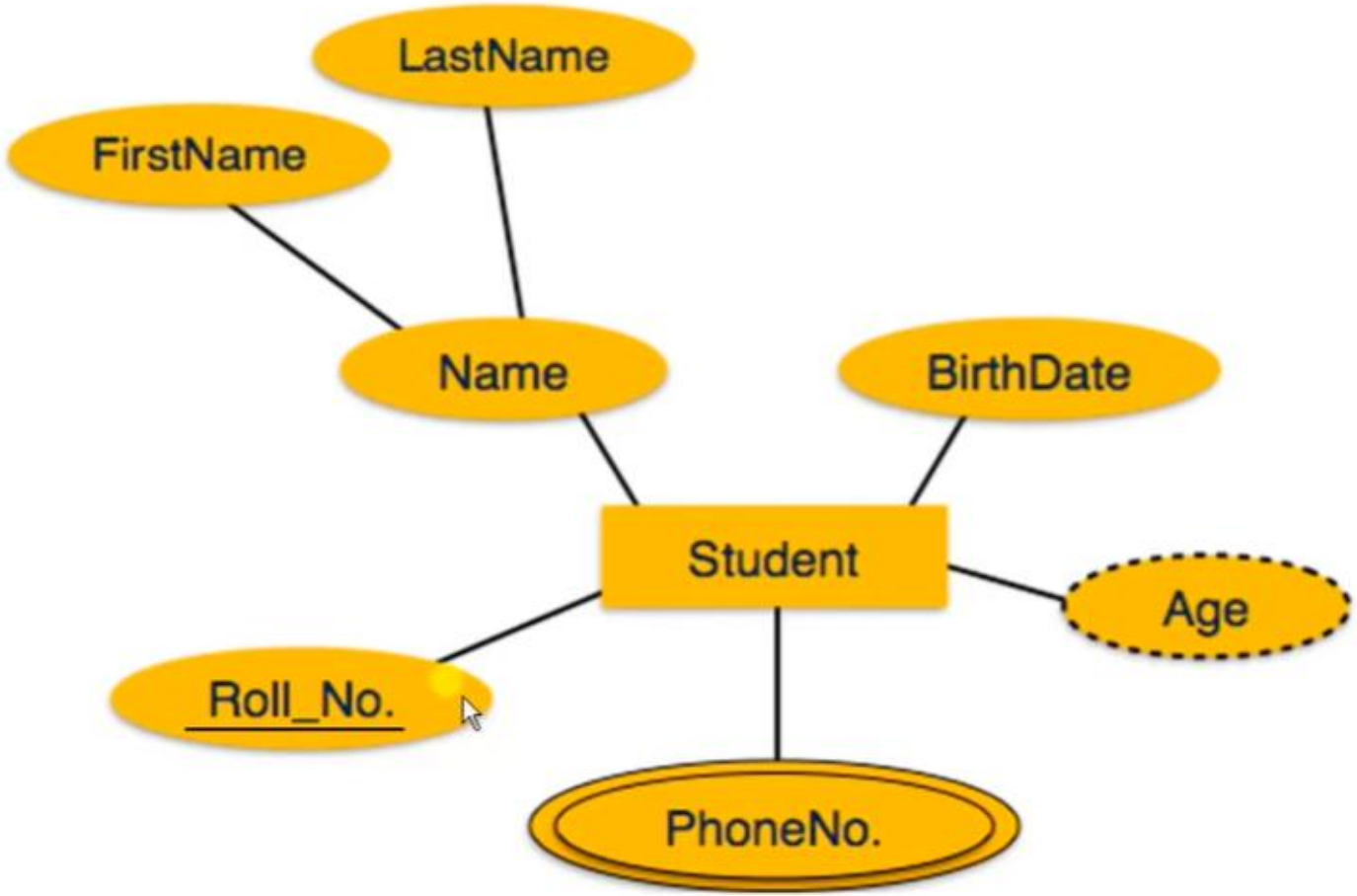
7. Key Attribute

- **Key Attribute**: The attribute which **uniquely identifies each entity** in the entity set is called **key attribute**
- It represents a primary key
- Key attribute is represented by an ellipse with underlying lines

- For example: Roll_No will be unique for each student.



E-R diagram of **Student** entity type with its attributes can be represented as:



Other possible combinations

These attribute types can come together in a way like:

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Extended Entity Relationship (ER) Features

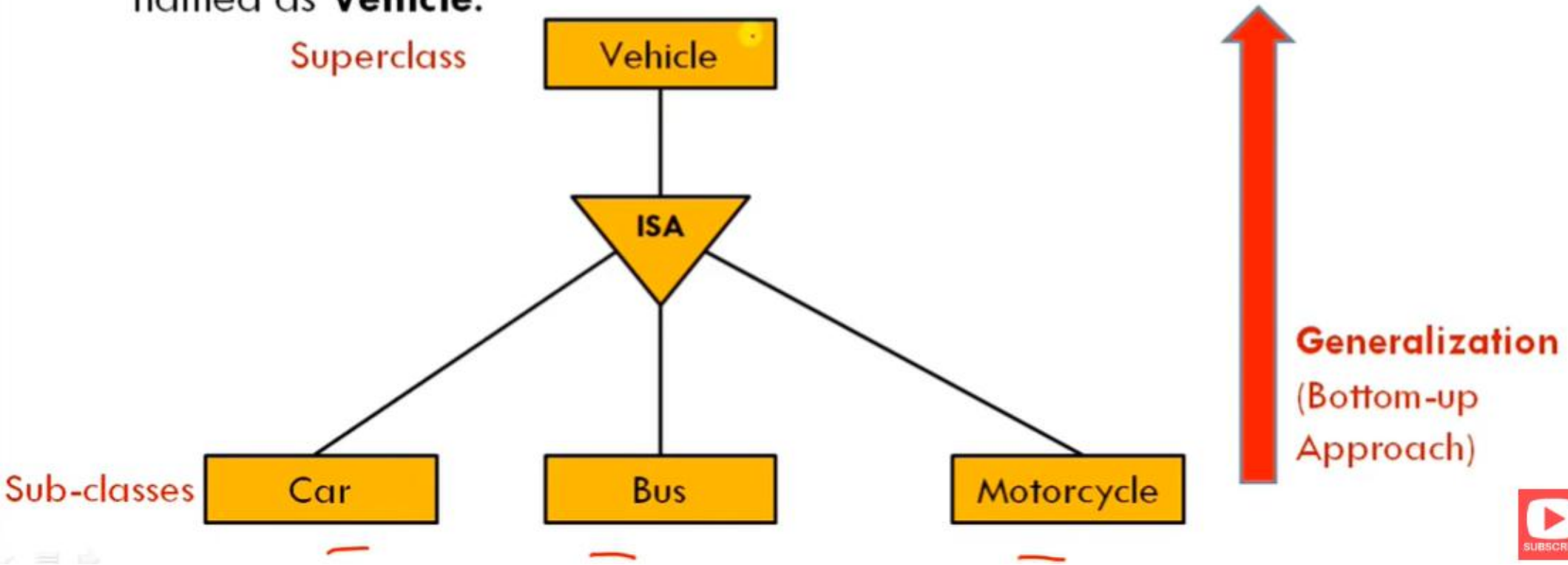
- As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.
- Hence, as part of the **Extended ER Model**, along with other improvements, three new concepts were added to the existing ER Model:
  1. **Generalization**
  2. **Specialization**
  3. **Aggregation**

Generalization

- **Generalization** is the process of extracting common properti from a set of entities and create a generalized entity from it
- **Generalization is a "bottom-up approach"** in which two or more entities can be combined to form a higher level entity i they have some attributes in common.
  - subclasses are combined to make a superclass.
- **Generalization is used** to emphasize the similarities among lower-level entity set and to hide differences in the schema

Example: Generalization

- Consider we have 3 sub entities Car, Bus and Motorcycle. Now these three entities can be generalized into one higher-level entity (or super class) named as **Vehicle**.

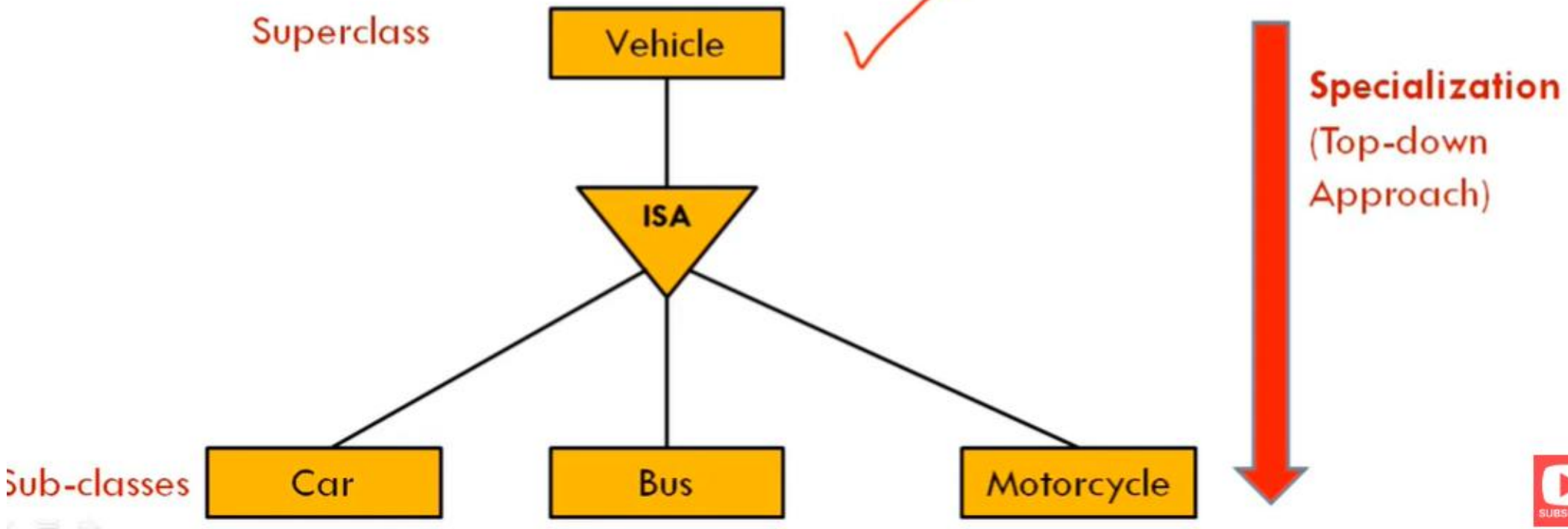


Specialization

- Specialization is **opposite** of Generalization
- In **Specialization**, **an entity is broken down into sub-entities based on their characteristics.**
- **Specialization is a "Top-down approach"** where higher level entity is specialized into two or more lower level entities.
- **Specialization is used** to identify the subset of an entity set that shares some distinguishing characteristics.
- **Specialization** can be repeatedly applied to refine the design of schema
- depicted by triangle component labeled **ISA**

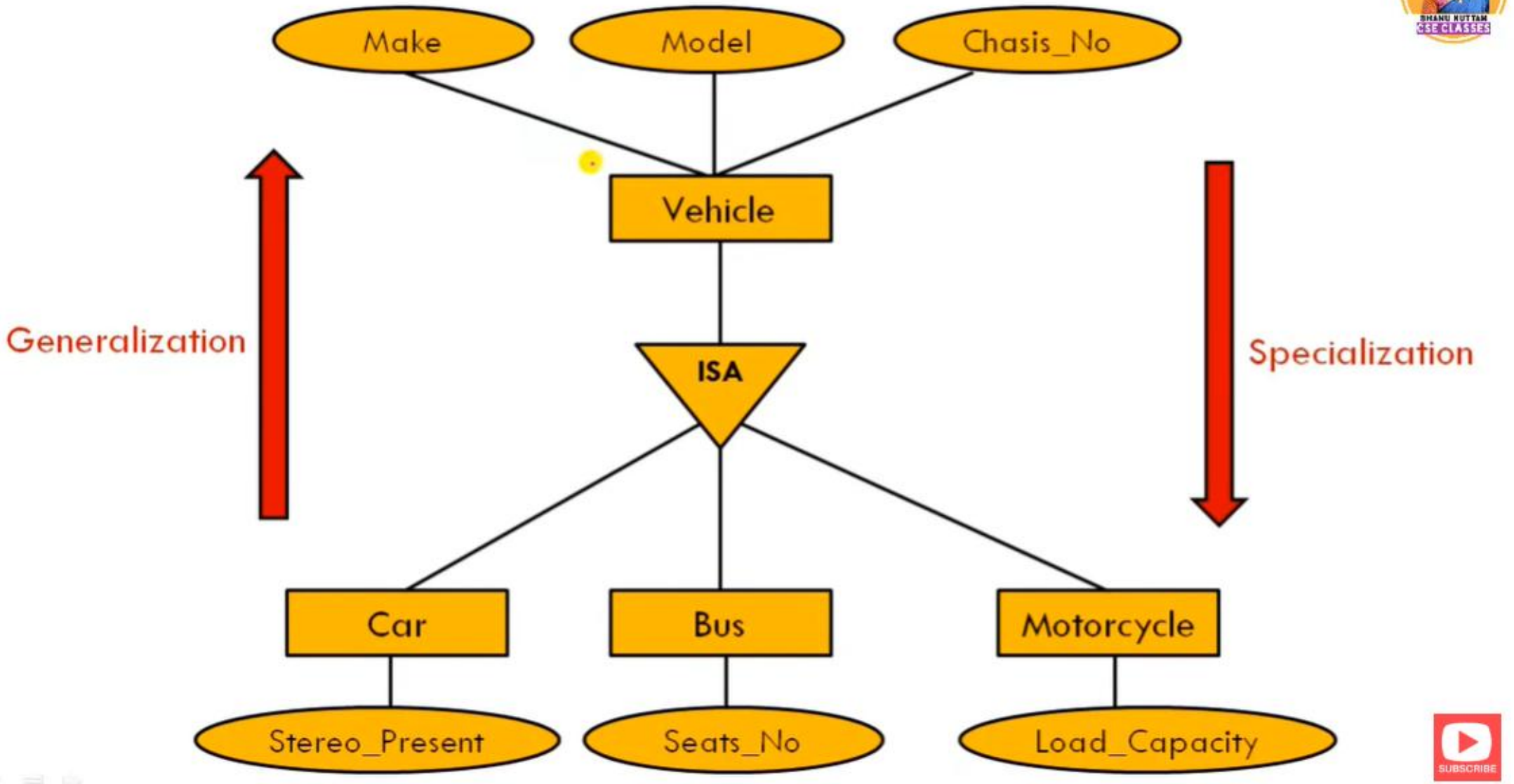
Example: Specialization

- **Vehicle** entity can be a Car, Truck or Motorcycle.
  - Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

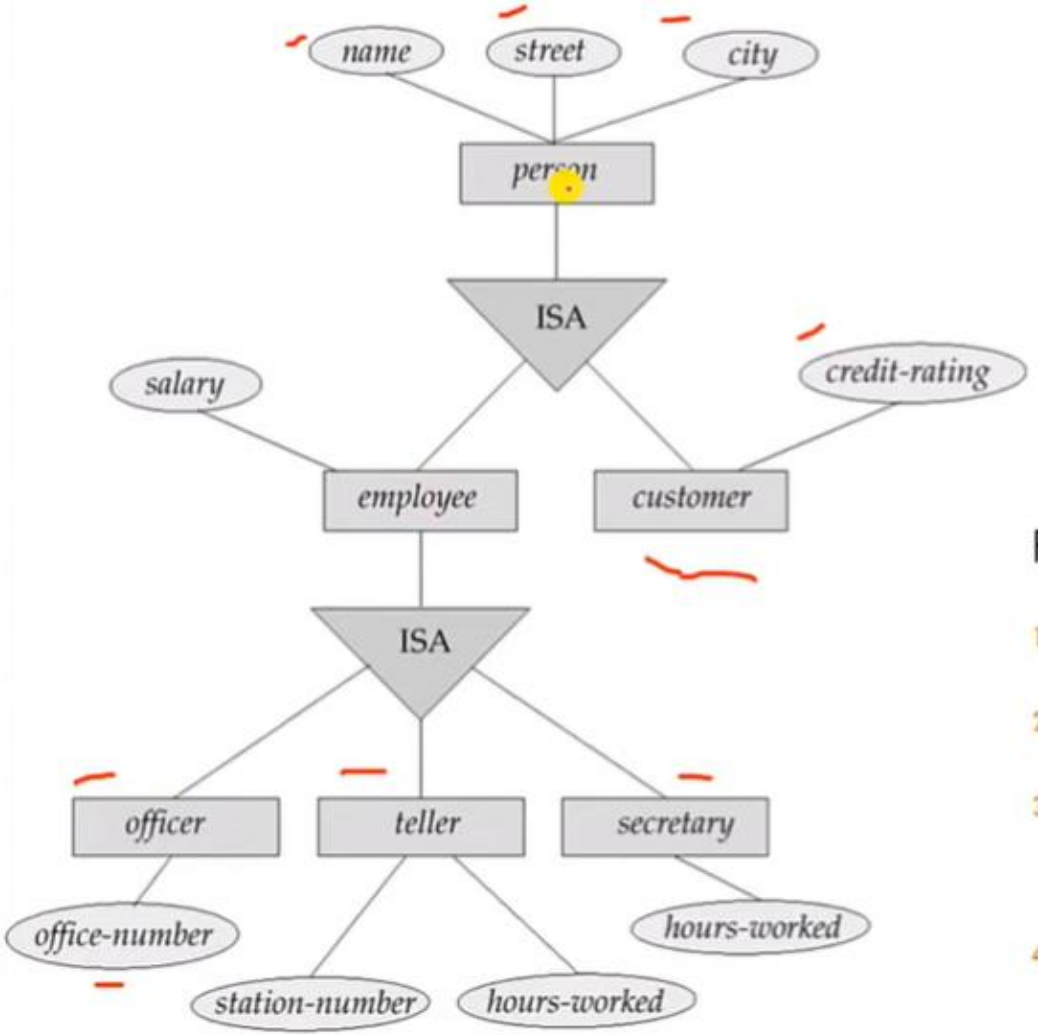


Inheritance

- **Inheritance** is an important feature of generalization and specialization.
- **Attribute inheritance allows lower level entities to *inherit the attributes* of higher level entities.**
  - **For example,** Consider relations Car and Bus inheriting the attributes of Vehicle. Thus, Car is described by attributes of super-class Vehicle as well as its own attributes.
- This also extends to **Participation Inheritance** in which ***relationships involving higher-level entity-sets are also inherited by lower-level entity-sets.***
  - A lower-level entity-set can participate in its own relationship-sets, too


Make
Model
Chasis_No
Vehicle
ISA
Generalization
Specialization
Car
Bus
Motorcycle
Stereo_Present
Seats_No
Load_Capacity

How Schema or Tables can be formed?



Four tables can be formed:

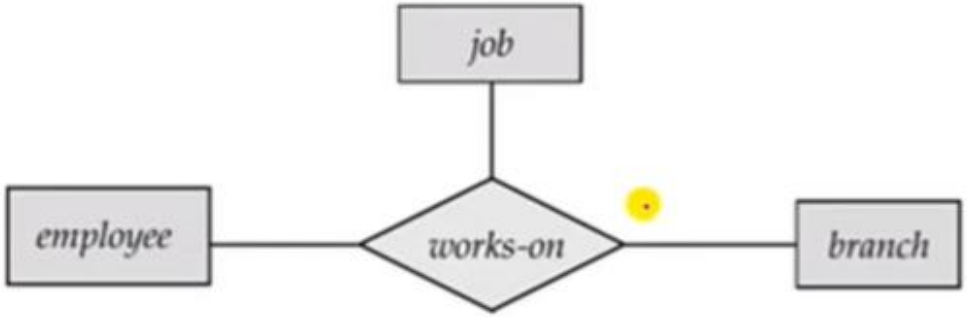
1. **customer** (name, street, city, credit_rating)
2. **officer** (name, street, city, salary, office_number)
3. **teller** (name, street, city, salary, station_number, hours_worked)
4. **secretary** (name, street, city, salary, hours_worked)

Aggregation

- **Aggregation** is used when we need to express a **relationship among relationships**
- **Aggregation** is an **abstraction** through which **relationships are treated as higher level entities**
- **Aggregation** is a process when a **relationship between two entities is considered as a single entity** and again this single entity has a **relationship** with **another entity**

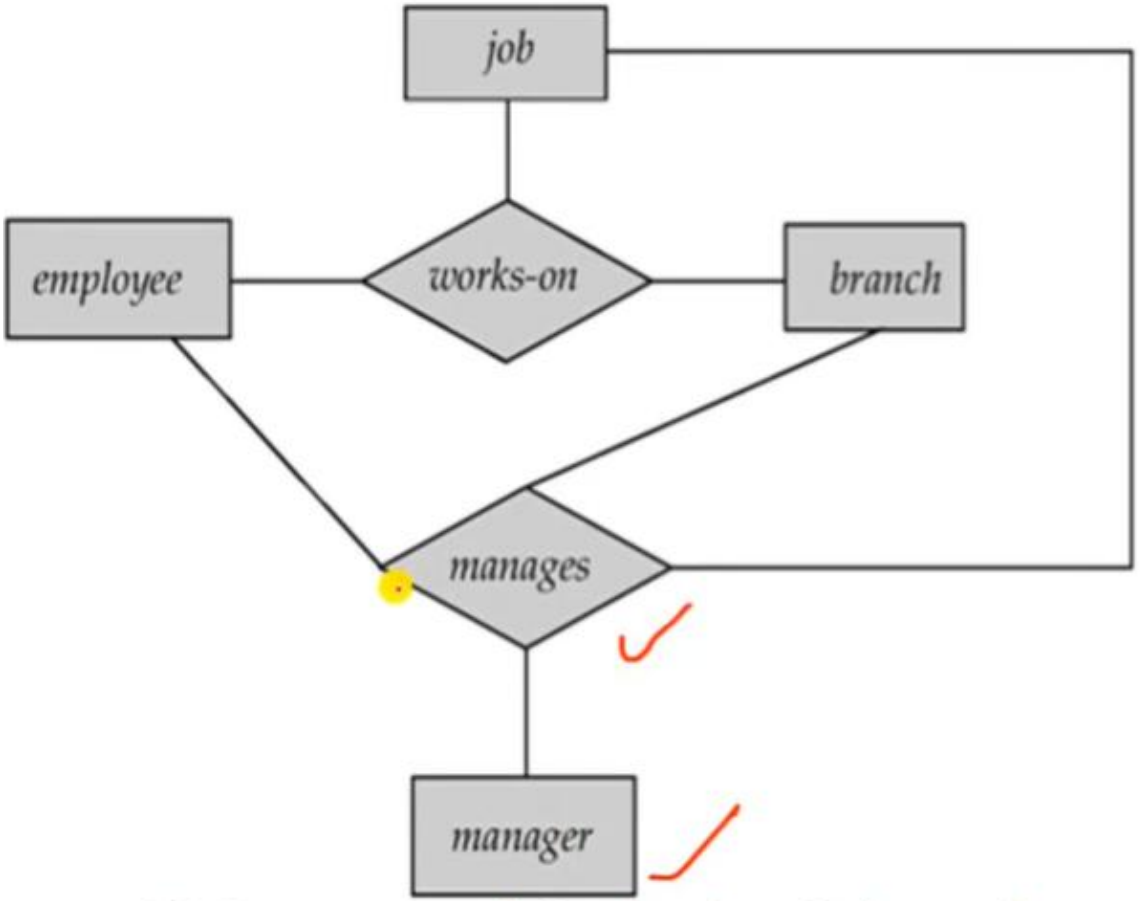
Example: (Relationship of Relations)

- Basic E-R model can't represent relationships involving other relationships
- Consider a ternary relationship **works_on** between **Employee, Branch and Job.**
  - Am **employee** ***works on*** a particular **job** at a particular **branch**



- Suppose we want to assign a **manager** for jobs performed by an employee at a branch (i.e. want to assign managers to each employee, job, branch combination)
  - Need a separate manager entity-set
  - Relationship between each manager, employee, branch, and job entity

Example: (Redundant Relationship)



ER Diagram with Redundant Relationship

- Relationship sets ***works-on*** and ***manages*** represent overlapping (redundant) information
  - Every *manages* relationship corresponds to a *works-on* relationship
  - However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works-on* relationship

Aggregation

- Eliminate this redundancy via ***aggregation***
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

Example: (Aggregation)

- With **Aggregation** (without introducing redundancy) the ER diagram can be represented as:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager