

# Database Management System

## Data Models : Entity Relationship Data Model

Tulasi Prasad Sariki

VIT

January 7, 2020

## Module-2

- Entity Relationship Model.
- Types of Attributes and Relationship.
- Structural Constraints.
- Relational Model and its constraints.
- Mapping ER model to a relational schema.
- Integrity constraints
- Relational Algebra and Calculus

## Text Books

- R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7 th Edition, 2015
- Raghu Ramakrishnan, Database Management Systems, Mcgraw-Hill, 4 th edition, 2015

## References

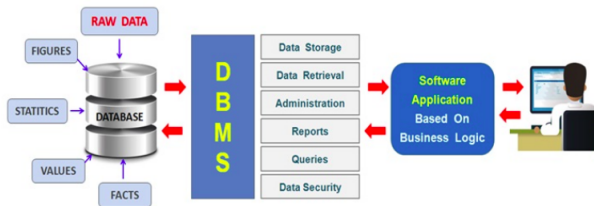
- A. Silberschatz, H. F. Korth & S. Sudershan, Database System Concepts, McGraw Hill, 6 th Edition 2010
- Thomas Connolly, Carolyn Begg, Database Systems : A Practical Approach to Design, Implementation and Management, 6 th Edition, 2012
- Pramod J. Sadalage and Marin Fowler, NoSQL Distilled: A brief guide to merging world of Polyglot persistence, Addison Wesley, 2012.
- Shashank Tiwari, Professional NoSql, Wiley, 2011.

## Outline....!

- Modelling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design of the University Database (Case Study)
- Design of the Company Database (Case Study)
- Reduction to Relation Schemas
- Database Modelling Tools

# Overview of DBMS

# Overview of DBMS



- A database can be modeled as:
  - A Collection of Entities
  - Relationship among Entities.
- An entity is an object that exists and is distinguishable from other objects.  
**Example: Specific person, Company, Event, Plant**
- Entities have attributes (defines the properties of Entity)  
**Example: People have Names and Sddresses**
- An entity set is a set of entities of the same type that share the same properties.  
**Example: Set of all persons, companies, trees, holidays**

## What is ER-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. So, it is considered a best practice to complete ER modeling before implementing your database.

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.

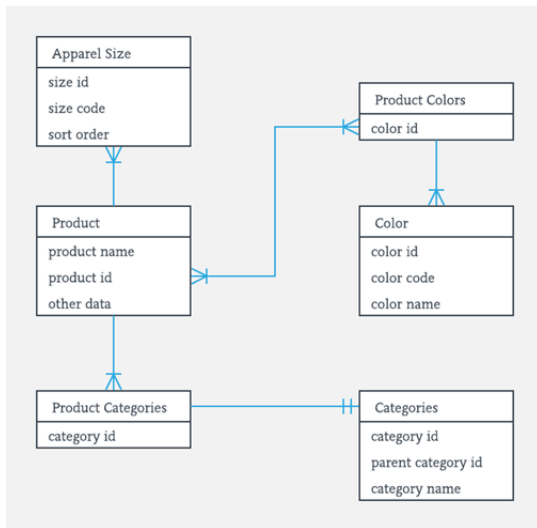


It displays the relationships of entity set stored in a database. In other words, It helps you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, It includes many specialized symbols, and its meanings make this model unique.

## Facts about ER Diagram Model:

- 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

# ER-Diagram



Here, are prime reasons for using the ER Diagram

- 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 is allowed you to communicate with the logical structure of the database to users

# Components of ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships



Entity Name

**Entity**

Person, place, object, event  
or concept about which  
data is to be maintained

**Example:** Car, Student



Jack

Attribute  
Name

**Attribute**

Property or characteristic of  
an entity

**Example:** Color of car Entity  
Name of Student Entity



**Relation**

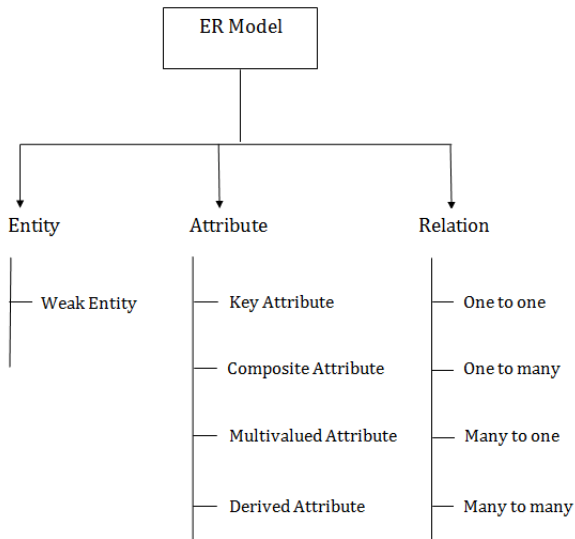
Verb  
Phrase

Association between the instances of one or  
more entity types

**Example:** Blue Car Belongs to Student Jack



# Components of ER Diagram



## What is Entity?

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. 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 Attributes are the characteristics of entities.

Examples :

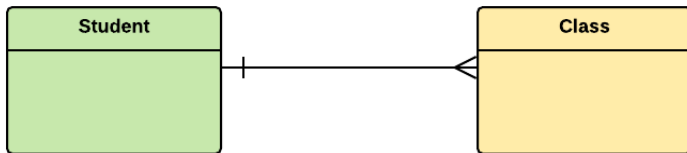
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

# Entity Set

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.



# University Example

## Example :

A university may have some **Departments**.

All these Departments employ various **Lecturers** and offer several **Programs**.

Some **Courses** make up each program. **Students** register in a particular program and enroll in various courses.

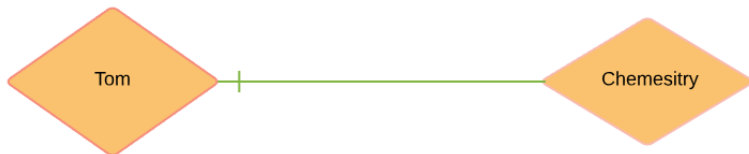
A Lecturer from the specific department takes each course, and each lecturer teaches a various group of students.



# Relationships

Relationship is 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.

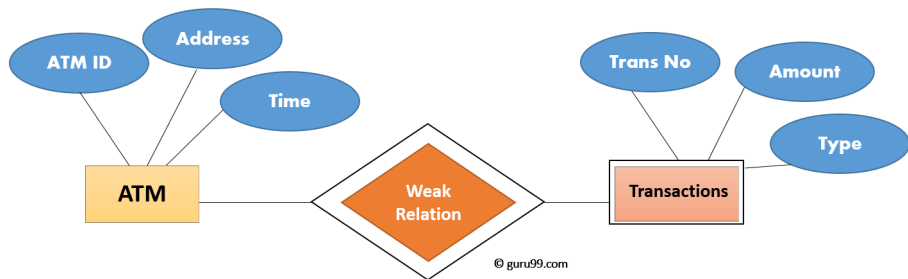
There are two types of Entities :

- Strong Entity
- Weak Entity

# Weak Entity

It is a type of entity which doesn't have its key attribute.

It can be identified uniquely by considering the primary key of another entity. Weak entity sets need to have participation.



Here, "**Trans No**" is a discriminator within a group of transactions in an ATM.

# Strong vs. Weak Entity

## Comparison of Strong and Weak Entity

<b>Strong Entity</b>	<b>Weak Entity</b>
It set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.

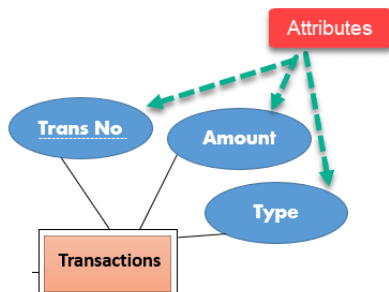
# Strong vs. Weak Entity

## Comparison of Strong and Weak Entity

<b>Strong Entity</b>	<b>Weak Entity</b>
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

# Attributes

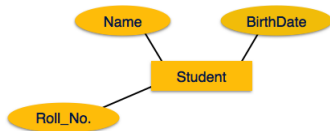
- It is a single-valued property of either an entity-type or a relationship-type.
- For example, a lecture might have attributes: time, date, duration, place, etc.
- An attribute is represented by an Ellipse



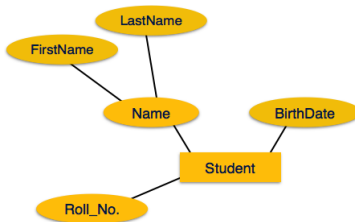
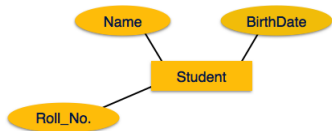
# Types of Attributes

- **Simple Attribute** : are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits. Composite Attribute
- **Composite Attribute** : are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
- **Derived Attribute** : are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
- **Single Valued Attribute** : contain single value. For example : Social\_Security\_Number.
- **Multi Valued Attribute** : contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

# Types of Attributes

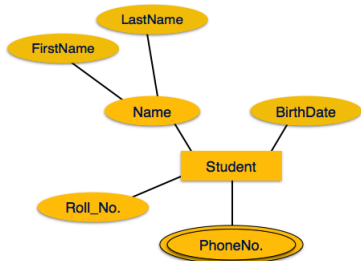
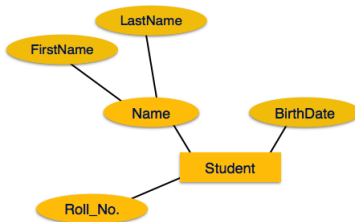
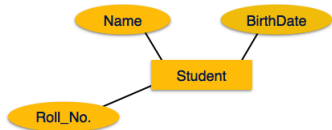


# Types of Attributes

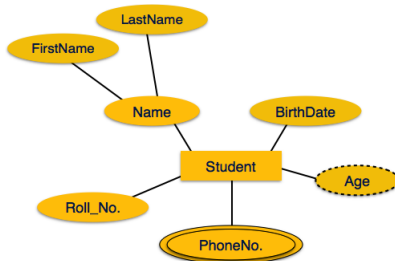
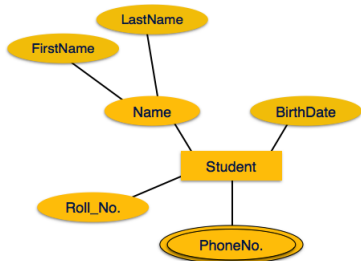
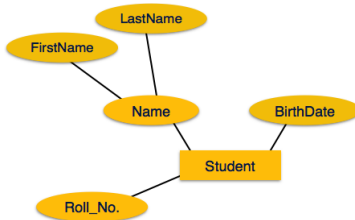
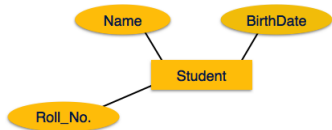




# Types of Attributes



# Types of Attributes

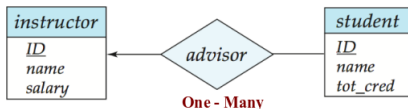


# Cardinality

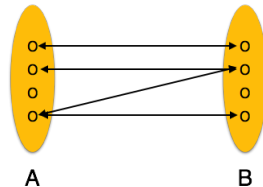
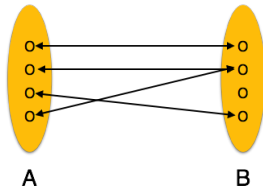
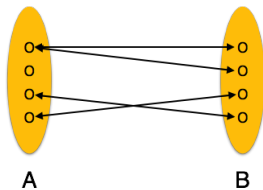
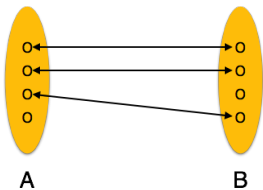
Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

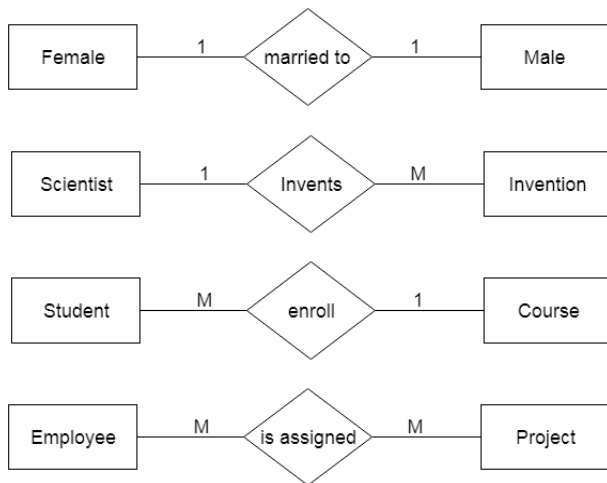
- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships



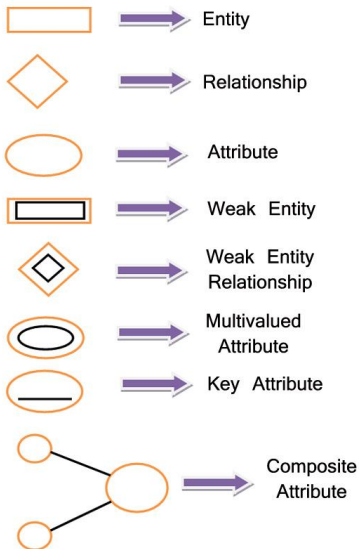
# Cardinality's...!



# Cardinality's...!



# ER-Diagram Notations



# ER-Diagram Notations



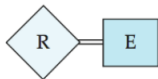
**Entity**



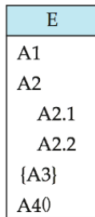
**Relationship**



**Relationship against weak entity**



**Total Participation**



**Attributes :**

**Simple** A1

**Composite** A2

**Multivalued** A3

**Derived** A4

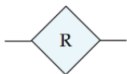


**Primary Key**



**Weak Entity**

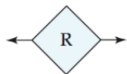
# ER-Diagram Notations



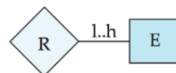
**Many-Many**



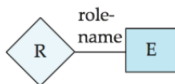
**Many-One**



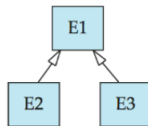
**One-One**



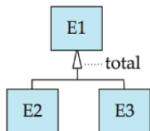
**Cardinality Limits**



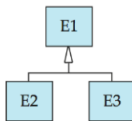
**Role Indicator**



**ISA :  
Generalization/  
Specialization**



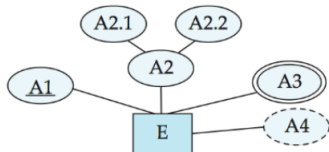
**Total  
Generalization**



**Disjoint  
Generalization**



# ER-Diagram Notations



entity set E with simple attribute A1, composite attribute A2, multivalued attribute A3, derived attribute A4, and primary key A1

## Generalization



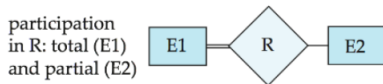
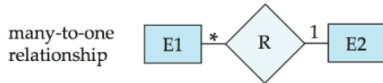
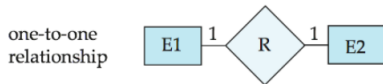
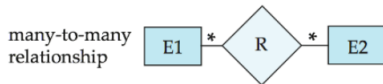
Week Entity



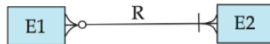
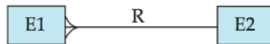
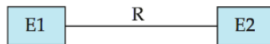
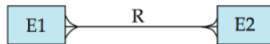
Total Generalization

# ER-Diagram Notations

## Chen



## IDE1FX Crow feet Notation



# Steps to Draw ER-Diagram

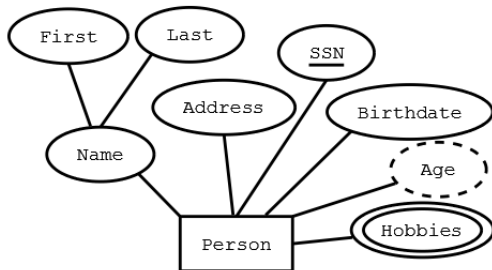
Following are the steps to create an ERD.

- **Entity Identification**
- **Relationship Identification**
- **Cardinality Identification**
- **Identify Attributes**
- **Create ERD**

**Example:** In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course.

# ER - Diagram

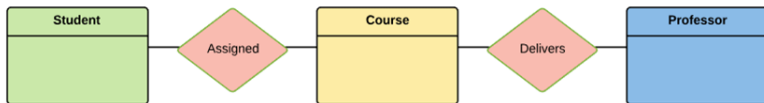
- **Entity Identification : Student, Course and Professor.**



- **Relationship Identification :** We have the following two relationships

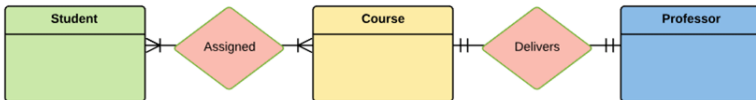
The student is assigned a course

Professor delivers a course

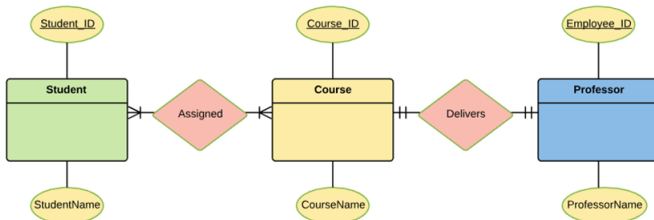


# ER - Diagram

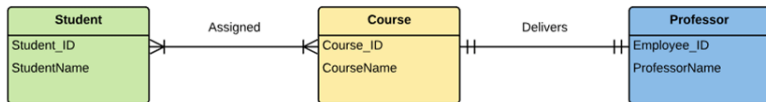
- **Cardinality Identification** : For the problem w.k.t :  
A student can be assigned multiple courses  
A Professor can deliver only one course



- **Identify Attributes** :



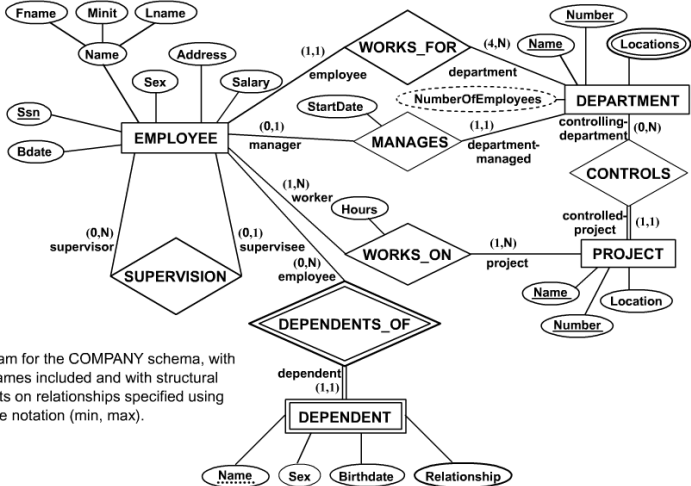
- **Create ERD** : A more modern representation of ERD Diagram



## Best Practices for Developing Effective ER Diagrams

- Eliminate any redundant entities or relationships
- You need to make sure that all your entities and relationships are properly labeled
- There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
- You should assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on your diagram
- Never connect relationships to each other
- You should use colors to highlight important portions of the ER diagram

## Alternative ER Notations

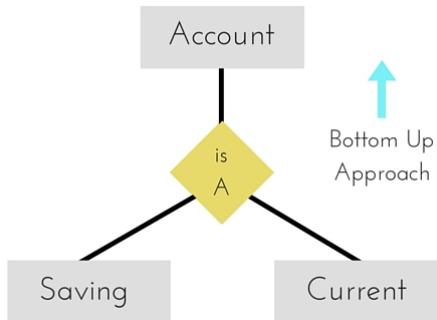


ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

# EER (Enhanced/Extended - ER Diagram)

Three new concepts were added to the existing ER Model, they were:

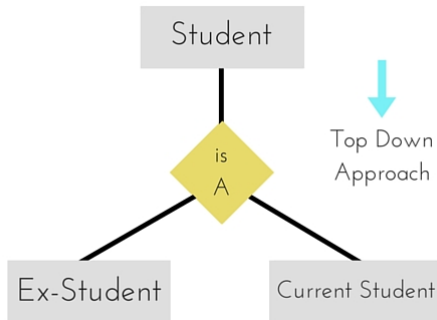
- Generalization : It is a bottom-up approach in which two lower level entities combine to form a higher level entity.





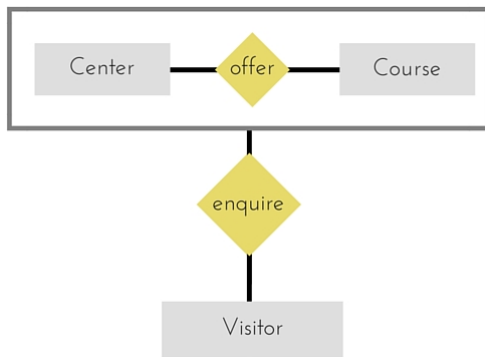
# EER (Enhanced/Extended - ER Diagram)

- Specialization : It is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity.

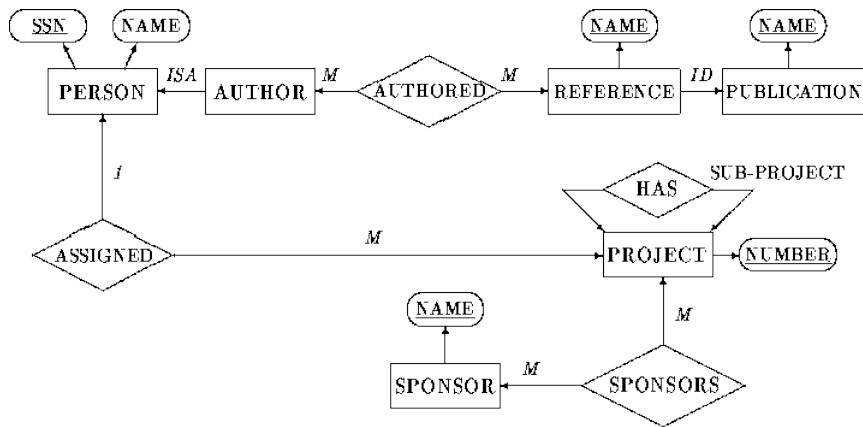


# EER (Enhanced/Extended - ER Diagram)

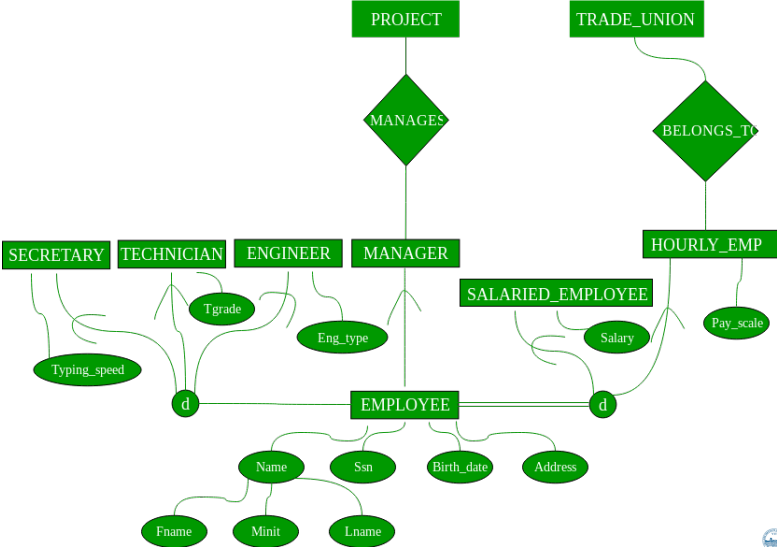
- Aggregation : It is a process when relation between two entities is treated as a single entity.



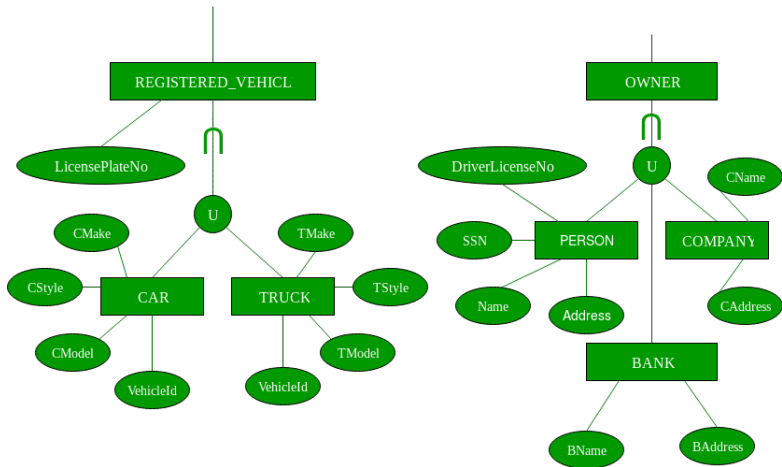
# EER - Example1



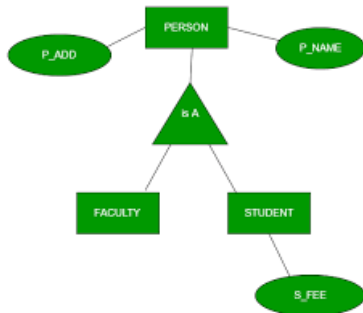
# EER - Example2



# EER - Example3



# EER - Example3



- A number of popular tools that cover conceptual modeling and mapping into relational schema design.  
Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, etc.
- POSITIVES: serves as documentation of application requirements, easy user interface - mostly graphics editor support

## Problems with Current Modeling Tools

- **DIAGRAMMING**
  - Poor conceptual meaningful notation.
  - To avoid the problem of layout algorithms and aesthetics of diagrams, they prefer boxes and lines and do nothing more than represent (primary-foreign key) relationships among resulting tables.
- **METHODOLOGY**
  - Lack of built-in methodology support. poor tradeoff analysis preferences.
  - Poor design verification and suggestions for improvement.

# Automated Database Modeling Tools

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration and space and security management
Oracle	Developer 2000 and Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum Technology	Platinum Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwrtier	Mapping from O-O to relational model
Rational	Rational Rose	Modeling in UML and application generation in C++ and JAVA
Rogue Ware	RW Metro	Mapping from O-O to relational model
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design and reengineering Visual Basic and Visual C++

## Automated Database Modeling Tools



# Database Management System

## Data Models : Relational Data Model

Tulasi Prasad Sariki

VIT

January 7, 2020

# Contents....!

- What is Relational Model
- Relational Model Concepts
- Relational Integrity constraints
- Operations in Relational Model
- Best Practices for creating a Relational Model
- Advantages of using Relational model
- Disadvantages of using Relational model
- Relational Query Languages
- Relational Operations

# What is Relational Data Model?

- The relational model represents the database as a collection of relations(Tables).
- A relation is nothing but a table of values. Every row in the table represents a collection of related data values.
- These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations.
- In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.
- Popular RDBMS are : **DB2 and Informix Dynamic Server(IBM), Oracle and RDB(Oracle), SQL Server and Access(Microsoft)**

# Relational Model Concepts

- **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student\_Rollno, NAME,etc.
- **Tables** – In the Relational model, the relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
- **Tuple** – It is nothing but a single row of a table, which contains a single record.
- **Relation Schema:** A relation schema represents the name of the relation with its attributes.
- **Degree:** The total number of attributes which in the relation is called the degree of the relation.
- **Cardinality:** Total number of rows present in the Table.
- **Column:** The column represents the set of values for a specific attribute.

# Relational Model Concepts

- **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
- **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

## Table also called Relation

© guru99.com

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

**Primary Key** (points to CustomerID)

**Domain**  
Ex: NOT NULL (points to CustomerName)

**Column OR Attributes**  
Total # of column is **Degree**

**Tuple OR Row**  
Total # of rows is **Cardinality**

# 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:
  - ① **Domain constraints**
  - ② **Key constraints**
  - ③ **Referential integrity constraints**

- 1 Domain Constraints : Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

It specifies that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

**Example:**

*Create DOMAIN CustomerName CHECK (value not NULL)*

The example shown demonstrates creating a domain constraint such that **CustomerName** is **not NULL**

- ② **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.

## Example:

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



- 3 **Referential integrity constraints** : It is based on the concept of **Foreign Keys**.

A foreign key is an important attribute of a relation which should be referred to in other relationships.

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

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	\$100
2	1	\$200
3	2	\$150

In the above example, we have 2 relations, **Customer** and **Billing**.  
 Tuple for **CustomerID =1** is referenced twice in the relation **Billing**.  
 So we know **CustomerName=Google** has billing amount **\$300**

# Operations in Relational Model

Four basic update operations performed on relational database model are **Insert, update, delete and select**.

- **Insert** is used to insert data into the relation
- **Delete** is used to delete tuples from the table.
- **Modify** allows you to change the values of some attributes in existing tuples.
- **Select** allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

# Operations in Relational Model with Example

- 1 **Insert Operation** : The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

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



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

- 2 **Update Operation**: You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

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




CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

# Operations in Relational Model with Example

- ③ **Delete Operation** : To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.  
In the given example, **CustomerName= "Apple"** is deleted.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

- ④ **Select Operation** :

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
2	Amazon	Active

In the given example, **CustomerName="Amazon"** is selected

# Best Practices to Create Relational Model

- 1 Data need to be represented as a collection of relations
- 2 Each relation should be depicted clearly in the table
- 3 Rows should contain data about instances of an entity
- 4 Columns must contain data about attributes of the entity
- 5 Cells of the table should hold a single value
- 6 Each column should be given a unique name
- 7 No two rows can be identical
- 8 The values of an attribute should be from the same domain

# Merits of Relational Model

- 1 **Simplicity:** A relational data model is simpler than the hierarchical and network model.
- 2 **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- 3 **Easy to use:** The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
- 4 **Query capability:** It makes possible for a high-level query language like SQL to avoid complex database navigation.
- 5 **Data independence:** The structure of a database can be changed without having to change any application.
- 6 **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

# De-Merits of Relational Model

- 1 Few relational databases have limits on field lengths which can't be exceeded.
- 2 Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- 3 Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.



# Summary of Relational Model

- It represents the database as a collection of relations (tables)
- Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation instance, are some important components of Relational Model
- Relational Integrity constraints are referred to conditions which must be present for a valid relation
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type
- Insert, Select, Modify and Delete are operations performed in Relational Model
- The relational database is only concerned with data and not with a structure which can improve the performance of the model
- Advantages of relational model is simplicity, structural independence, ease of use, query capability, data independence, scalability.
- Few relational databases have limits on field lengths which can't be exceeded.

# Database Management System

## Mapping the ER Model to Relational Model

Tulasi Prasad Sariki

VIT

January 7, 2020

# Mapping the ER Model to Relational DBs

## Database Design :

- Goal of design is to generate a formal specification of the database schema

# Mapping the ER Model to Relational DBs

## Database Design :

- Goal of design is to generate a formal specification of the database schema

## Methodology :

- Use E-R model to get a high-level graphical view of essential components of enterprise and how they are related
- Then convert E-R diagram to SQL Data Definition Language (DDL), or whatever database model you are using
- E-R Model is not SQL based.

**The E-R Model:** The database represented is viewed as a graphical drawing of

- Entities and attributes
- Relationships among those entities
- –not tables!

# Mapping the ER Model to Relational DBs

**The E-R Model:** The database represented is viewed as a graphical drawing of

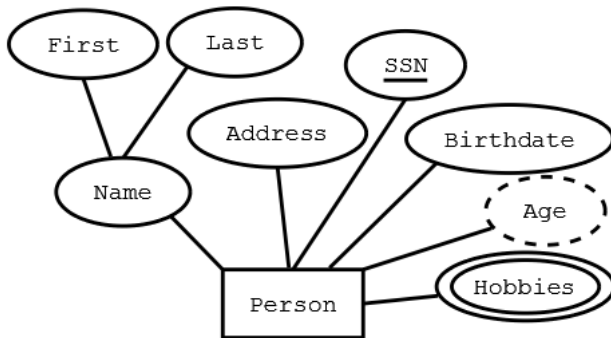
- Entities and attributes
- Relationships among those entities
- –not tables!

**Relational Model:** The database is viewed as a

- Tables and their attributes (keys)
- –we could include constraints but will not at this stage.

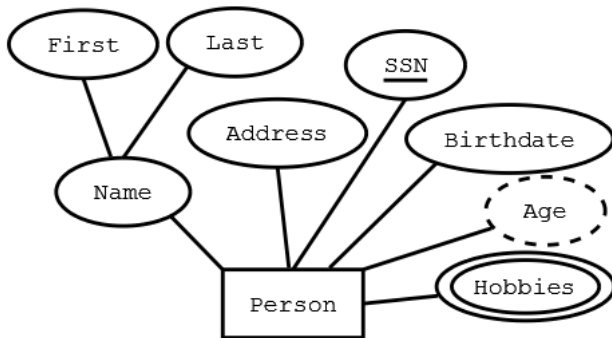
# Representation of Entity type in Relational Model

Mapping #1: Each entity type always corresponds to a relation



# Representation of Entity type in Relational Model

Mapping #1: Each entity type always corresponds to a relation



Solution : **Person**(.....)



# Representation of Entity type in Relational Model

## Mapping #2: The Relation attributes are the simple attributes of that entity type

- Attributes are single valued
- There may be additional attributes (foreign keys)
- **Person**(SSN, *FirstName*, *LastName*, *Address*, *Birthdate*)

**Problem** : Recall that the entity type can have multi-valued attributes.

**Possible Soln** : Use several rows to represent a single entity

- (50396, Pradeep K V, VIT Chennai, Cricket)
- (50396, Pradeep K V, VIT Chennai, Caroms)

**Problems with this solution:**

- Redundancy of the other attributes (never good)
- Key of entity type no longer can be key of relation

# Representation of Entity type in Relational Model

So, the resulting relation must be further transformed (Normalization). Normalization is the process we will study to help deal with this and would result in:

- **Persons**(SSN, *FirstName*, *LastName*, *Address*, *Birthdate*)
- **Hobbies**(SSN, *Hobby*)

# Relationship Mapping

**Relationship:** Connects two or more entities into an association.

- "Deepu" majors in Computer Science

# Relationship Mapping

**Relationship:** Connects two or more entities into an association.

- "Deepu" majors in Computer Science

**Relationship Type:** Set of similar relationships

- Student (entity type) related to Department (entity type) by **MajorsIn** (relationship type).

# Relationship Mapping

**Relationship:** Connects two or more entities into an association.

- "Deepu" majors in Computer Science

**Relationship Type:** Set of similar relationships

- Student (entity type) related to Department (entity type) by **MajorsIn** (relationship type).

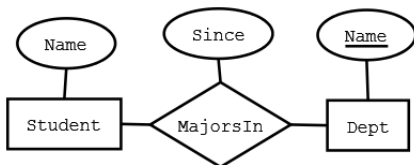
**Distinction**

- Relation (relational model) - set of tuples
- Relationship (E-R Model) – describes relationship between entities of an enterprise

# Relationship Mapping

**Mapping #3: 1-1 and 1-many relationships between separate entities need not be mapped to a relation;**

**The primary key attributes of the "1" relation become foreign key attributes of the "many" relation**



If no "Since" attribute, the relations could be (with some appropriate attribute renaming and additions)

- **Students**(StudId, Name, Dept)
- **Departments**(Dept, Chair)

Relationship Types may also have attributes in the E-R model.

# Relationship Mapping

**Mapping #4: Any attributes of the 1-1 or 1-many relationship may be attached to the "many" relation.**

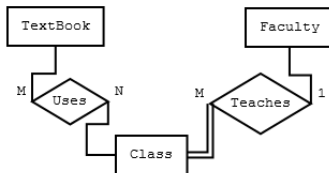
- **Students**(StudId, Name, Dept, Since)
- **Departments**(Dept, Chair)

# Relationship Mapping

**Mapping #4:** Any attributes of the 1-1 or 1-many relationship may be attached to the "many" relation.

- **Students**(StudId, Name, Dept, Since)
- **Departments**(Dept, Chair)

**Mapping #5:** many-many relationships are always mapped to a separate relation.



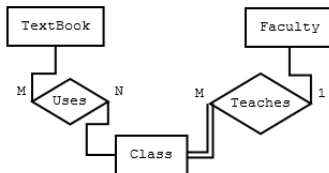


# Relationship Mapping

**Mapping #4: Any attributes of the 1-1 or 1-many relationship may be attached to the "many" relation.**

- **Students**(StudId, Name, Dept, Since)
- **Departments**(Dept, Chair)

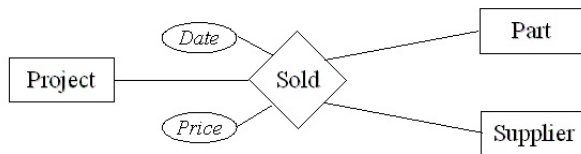
**Mapping #5: many-many relationships are always mapped to a separate relation.**



- **Textbooks**(ISBN, Title, Author, Copyright, Edition, Price)
- **Class**(ClassNo, Name, Room, Days, Time)
- **TextUses**(ISBN, ClassNo)

# Relationship Mapping

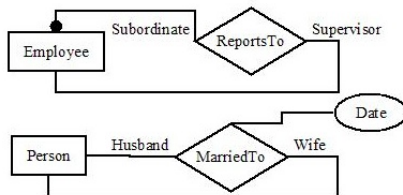
**Mapping #6:** The attributes of many-many relationships become part of the relationship type relation, as well as the primary key attributes of the related entity types.



- Projects(ProjId, Name, TotalCost, StartDate)
- Parts(UPC, PartName, Weight, WSPPrice)
- Suppliers(SupId, Name, Address)
- Sold(ProjId, UPC, SupId, Date, Price)

# Relationship Mapping

**Mapping #7:** If the cardinality is 1-many or 1-1 of a recursive relationship, then a second attribute of the same domain as the key may be added to the entity relation to establish the relationship. Attributes of the relationship can also be added to the entity relation, but may be a good reason to create a separate relation with the attributes and keys of the entities.



- Employees(EmpID, Name, Address, Salary, SupervisorID)
- Persons(PID, Name, Address, SpouseID, Mdate)

**Mapping #8: for many-many recursive relationships, you create a relation including the attributes of the relation but with the primary keys of the entity included twice, one for each role.**

Assume multiple marriages are now recorded, thus many-to-many.

- MarriedTo(HusbandID, WifeID, MarDate, DivDate)

# ER Mapping Examples

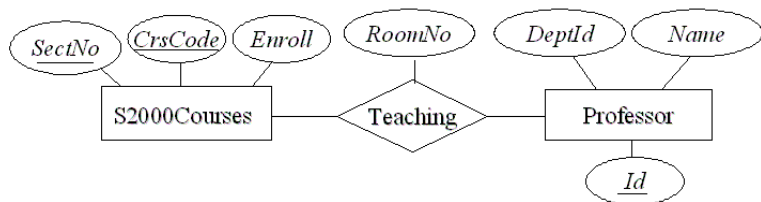


Figure: Example-1

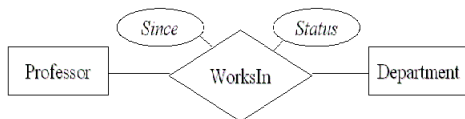
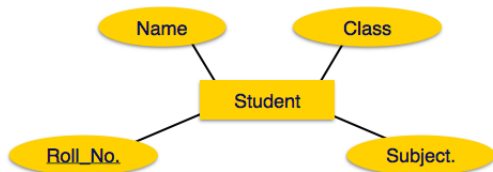


Figure: Example-2

# Mapping Entity

An entity is a real-world object with some attributes.

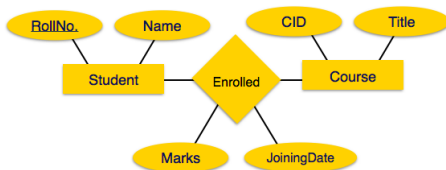


Mapping Entity Procedure :

- Create table for each entity.
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.

# Mapping Relationship

A relationship is an association among entities.

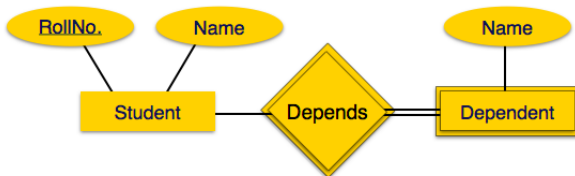


Mapping Relationship Procedure :

- Create table for a relationship.
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

# Mapping Weak Entitys

A weak entity set is one which does not have any primary key associated with it.



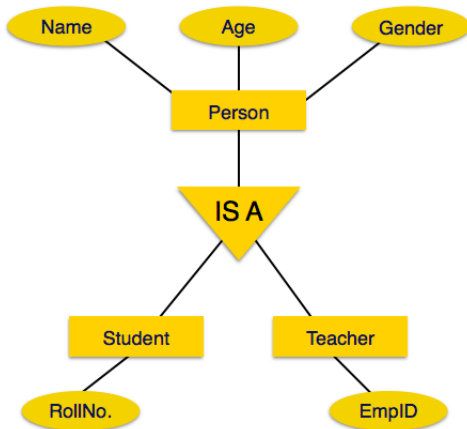
Mapping Weak Entity's Procedure :

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.



# Mapping Hierarchical Entities

ER specialization or generalization comes in the form of hierarchical entity sets.



# Mapping Hierarchical Entity's

## Mapping Hierarchical Procedure :

- Create tables for all higher-level entities.
- Create tables for lower-level entities.
- Add primary keys of higher-level entities in the table of lower-level entities.
- In lower-level tables, add all other attributes of lower-level entities.
- Declare primary key of higher-level table and the primary key for lower-level table.
- Declare foreign key constraints.

# Database Management System

## Relational Query Languages

Tulasi Prasad Sariki

VIT

January 7, 2020

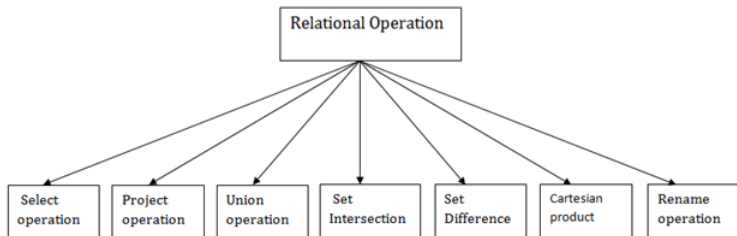
# Relational Query Languages

- Relational database systems comes with a query language, which assists users to query the database instances.
- There are two kinds of query languages:
  - 1 **Relational Algebra**
  - 2 **Relational Calculus**

# Relational Algebra

- Relational algebra is a widely used procedural query language.
- It collects instances of relations as input and gives occurrences of relations as output.
- It uses various operation to perform this action.
- Relational algebra operations are performed recursively on a relation.
- The output of these operations is a new relation, which might be formed from one or more input relations.
- Basic Relational Algebra Operations :
  - 1 **Unary Relational Operations** : SELECT( $\sigma$ ), PROJECT( $\Pi$ ), RENAME( $\rho$ )
  - 2 **Binary Relational Operations** : JOIN( $\bowtie$ ), DIVISION( $/$ )
  - 3 **Set Theory Operations** : UNION( $\cup$ ), INTERSECTION( $\cap$ ), DIFFERENCE( $-$ ) AND CARTESIAN PRODUCT( $\times$ )

# Relational Operations



# SELECT( $\sigma$ ) Operation

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition.

Sigma( $\sigma$ ) Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition.

Select operation selects tuples that satisfy a given predicate.

For Example :  $\sigma_p(\mathbf{r})$  :

- $\sigma$  is a predicate
- p is a propositional logic
- r stands for relation(Table)

# SELECT( $\sigma$ ) Operation

## Example-1:

$\sigma_{\text{topic}=\text{"Database"}}(\text{CSE2004})$

**Output** : Selects tuples from Tutorials where topic = 'Database'.

## Example-2:

$\sigma_{\text{topic}=\text{"Database"} \text{ and } \text{author}=\text{"Pradeep"}}(\text{CSE2004})$

**Output** : Selects tuples from CSE2004 where the topic is 'Database' and 'author' is "Pradeep".

## Example-3:

$\sigma_{\text{sales} > 50000}(\text{Customer})$

**Output** :Selects tuples from Customers where sales is greater than 50000



# Projection( $\Pi$ ) Operation

It eliminates all attributes of the input relation but those mentioned in the projection list. It defines a relation that contains a vertical subset of Relation.

It helps to extract the values of specified attributes to eliminates duplicate values.

( $\Pi$ ) symbol used to choose attributes from a relation.

This operation helps you to keep specific columns from a relation and discards the other columns.

# Projection( $\Pi$ ) Operation

Example : Consider the following table.

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

Here, the projection of **CustomerName** and **Status** from **Customer** Table will give :

$\Pi_{CustomerName, Status}(Customer)$

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

# Examples of SELECT( $\sigma$ ) & PROJECTION ( $\Pi$ )

Notation:  $\sigma_p(\mathbf{r})$

Where  $\sigma$  stands for selection predicate and  $\mathbf{r}$  stands for relation,  $p$  is predicate logic formula which may use connectors like **and**, **or**, and **not**.

These terms may use relational operators like: =,  $\neq$ ,  $\geq$ ,  $<$ ,  $>$ ,  $\leq$ .

For example:

$\sigma_{\text{subject}=\text{"database"}}(\mathbf{Books}) :$

Selects tuples from **Books** where subject = 'database'.

$\sigma_{\text{subject}=\text{"database"} \text{ and } \text{price}=\text{"450"}}(\mathbf{Books}) :$

Selects tuples from **Books** where subject is 'database' and 'price' is 450.

$\sigma_{\text{subject}=\text{"database"} \text{ and } \text{price} < \text{"450"} \text{ or } \text{year} > \text{"2010"}}(\mathbf{Books}) :$

Selects tuples from **Books** where subject is 'database' and 'price' is 450 or those books published after 2010.

# Examples of SELECT( $\sigma$ ) & PROJECTION ( $\Pi$ )

Notation:  $\Pi_{A_1, A_2, \dots, A_n}(r)$

Where  $A_1, A_2, \dots, A_n$  are attribute names of relation  $r$ .

Duplicate rows are automatically eliminated, as relation is a set.

For example:

$\Pi_{\text{subject, author}}(\text{Books})$

selects and projects columns named as **subject** and **author** from the relation Books.

# Union ( $\cup$ ) Operation

UNION is symbolized by  $\cup$  symbol.

It includes all tuples that are in tables **A** or in **B**.

It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:  $A \cup B$ .

The Following condition must hold:

- A and B must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

# Example : Union ( $\cup$ ) Operation

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

A  $\cup$  B gives

Table A $\cup$ B	
column 1	column 2
1	1
1	2
1	3

# Set Difference (−)

– Symbol denotes it.

The result of  $A - B$ , is a relation which includes all tuples that are in  $A$  but not in  $B$ .

The Following Conditions must satisfy :

- The attribute name of  $A$  has to match with the attribute name in  $B$ .
- The two-operand relations  $A$  and  $B$  should be either compatible or Union compatible.
- It should be defined relation consisting of the tuples that are in relation  $A$ , but not in  $B$ .

# Example : Set Difference (−)

For Example : Consider the Below Relations :

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

The Operations on Relations  $A - B$  gives :

Table A - B	
column 1	column 2
1	2



# Intersection $\cap$ Operation

An intersection is defined by the symbol  $\cap$ .

Defines a relation consisting of a set of all tuple that are in both A and B.  
However, A and B must be union-compatible

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

. The intersection of  $A \cap B$  is :

Table $A \cap B$	
column 1	column 2
1	1

# Cartesian Product (X) Operation

This type of operation is helpful to merge columns from two relations.

Generally, a Cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations.

Example : Let us consider the below Relations A and B.

The  $\sigma_{column2='1'}(A \times B)$  is :

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

$\sigma_{column 2='1'}(A \times B)$	
column 1	column 2
1	1
1	1

# Join ( $\bowtie$ ) Operation

Join operation is essentially a Cartesian product followed by a selection criterion.

Join operation denoted by  $\bowtie$ .

JOIN operation also allows joining variously related tuples from different relations.

Types of Join

- **Inner Join** : Theta/EQUI/Natural Joins
- **Outter Join** : Left/Right/Full Outer Joins

# Inner Join Operations

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded.

- 1 Theta Join** : The general case of JOIN operation is called a Theta join. It is denoted by symbol  $\theta$   
Example :  $A \bowtie_{\theta} B$ ,  $A \bowtie_{A.Column2 > B.Column2} B$ .
- 2 EQUI join** : When a theta join uses only equivalence condition, it becomes a equi join  
Example :  $A \bowtie_{A.Column2 = B.Column2} B$ .
- 3 NATURAL JOIN ( $\bowtie$ )** : Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.  
Example :  $A \bowtie B$ .

# Example of Inner Join

Example-1 (**Theta Join**) :  $A \bowtie_{A.Column2 > B.Column2} B$

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

$A \bowtie_{A.column 2 > B.column 2} (B)$	
column 1	column 2
1	2

Example-2 (**EQUI Join**) :  $A \bowtie_{A.Column2 = B.Column2} B$

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

$A \bowtie_{A.column 2 = B.column 2} (B)$	
column 1	column 2
1	1

In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

- 1 **Left Outer Join** ( $\bowtie$ ) : In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.
- 2 **Right Outer Join**( $\bowtie$ ) : In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.
- 3 **Full Outer Join**( $\bowtie$ ) : In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

# Join Operation Examples

## 1. Left Join ( $A \bowtie B$ )

A		B		$A \bowtie B$		
Num	Square	Num	Cube	Num	Square	Cube
2	4	2	8	2	4	4
3	9	3	18	3	9	9
4	16	5	75	4	16	-

## 2. Right Join ( $A \bowtie B$ )

A		B		$A \bowtie B$		
Num	Square	Num	Cube	Num	Cube	Square
2	4	2	8	2	8	4
3	9	3	18	3	18	9
4	16	5	75	5	75	-

# Join Operation Examples

## 3. Full Join ( $A \bowtie B$ )

A	
Num	Square
2	4
3	9
4	16

B	
Num	Cube
2	8
3	18
5	75

A $\bowtie$ B		
Num	Cube	Square
2	4	8
3	9	18
4	16	-
5	-	75



# Summary of Relational Algebra Operations

- 1 **Select**( $\sigma$ ) : It is used for selecting a subset of the tuples according to a given selection condition
- 2 **Projection**( $\Pi$ ) : The projection eliminates all attributes of the input relation but those mentioned in the projection list.
- 3 **Union**( $\cup$ ) : It includes all tuples that are in tables A or in B.
- 4 **Intersection**( $\cap$ ) : It is a relation which includes all tuples that are in A but not in B.
- 5 **Intersection**( $\cap$ ) : It a relation consisting of a set of all tuple that are in both A and B.
- 6 **Cartesian Product**( $\times$ ) : It is helpful to merge columns from two relations.

# Summary of Relational Algebra Operations

- 1 **Theta Join**( $\theta$ ) : The general case of JOIN operation is called a Theta join
- 2 **EQUI Join** : When a theta join uses only equivalence condition, it becomes a equi join.
- 3 **Natural Join**( $\bowtie$ ) : can only be performed if there is a common attribute (column) between the relations.
- 4 **Left Outer Join**( $\bowtie\lrcorner$ ) : operation allows keeping all tuple in the left relation.
- 5 **Right Outer Join**( $\lrcorner\bowtie$ ) : operation allows keeping all tuple in the right relation.
- 6 **Full Outer Join**( $\bowtie\lrcorner\lrcorner$ ) : all tuples from both relations are included in the result irrespective of the matching condition.

Thanks

*Thank  
you*

A close-up image of a fountain pen nib, which is gold-colored with a black barrel. The nib is positioned at the end of the word 'you' in a cursive script, appearing to have just finished writing it.