Advanced Database Lecture 1

Dr. Marwa Al Enany

Database Types

- **Centralized Database**

Stores data at a centralized database system. Users access the stored data from different locations through several applications. These applications contain the authentication process to let users access data securely.

An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

Database Types

- **Distributed Database**

Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization.

These database systems are connected via communication links. Such links help the end-users to access the data easily

Database Types

- **Relational Database**

This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). Each table in the database carries a key that makes the data unique from others. **Examples** of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

Database Types

- **NoSQL Database**

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways.

It came into existence when the demand for building modern applications increased. Thus, NoSQL presented a wide variety of database technologies in response to the demands.

Database Types

We can further divide a NoSQL database into the following four types:

1. **Key-value storage:** It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.
2. **Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.
3. **Graph Databases:** It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.
4. **Wide-column stores:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

Database Types

- **Advantages of NoSQL Database**
- It enables good productivity in the application development as it is not required to store data in a structured format.
- It is a better option for managing and handling large data sets.
- It provides high scalability.
- Users can quickly access data from the database through key-value.

Database Types

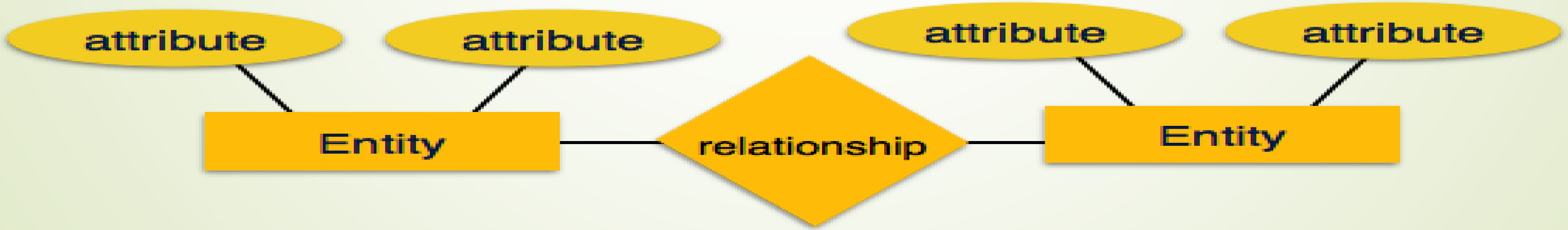
- **Cloud Database**

A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services for accessing the database. There are numerous cloud platforms, but the best options are:

- Amazon Web Services(AWS)
- Microsoft Azure
- Kamatera
- PhonixNAP
- ScienceSoft
- Google Cloud SQL, etc.

Entity-Relationship Model

- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.
- ER based on:
- **Entities** and their *attributes.*
- **Relationships** among entities.



Entity-Relationship Model

- **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Entity-Relationship Model

Types of Attributes

- **Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key.

- **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.

- **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. For example, average_salary, age can be derived from data_of_birth.

- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Entity-Relationship Model

- **Entity-Set and Keys**

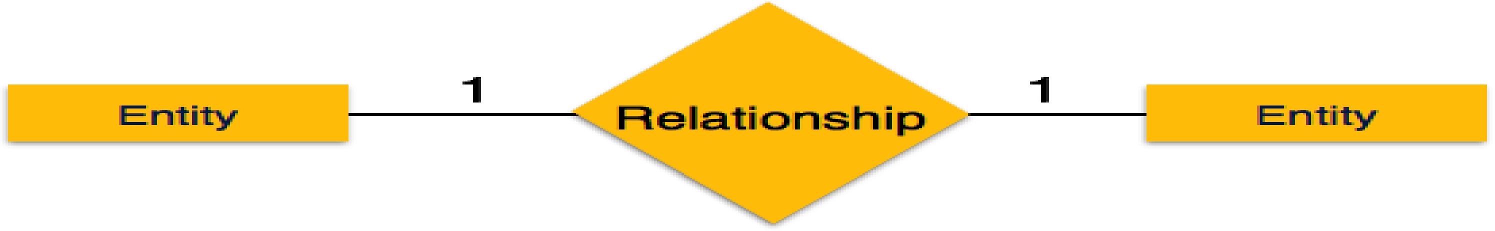
Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** – A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** – A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.
- **Foreign Key** – Foreign keys are the columns of the table used to point to the primary key of another table.

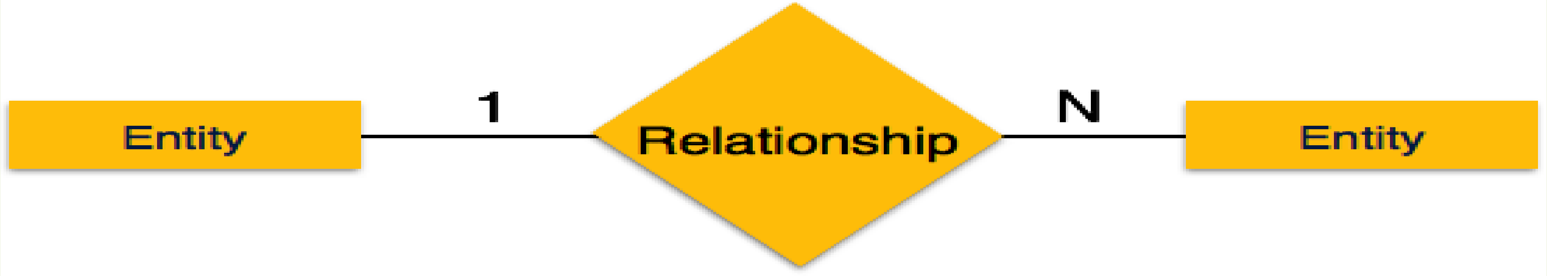
Entity-Relationship Model

- A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.
- **One-to-one** – When only one instance of an entity is associated with the relationship, it is marked as '1:1’.

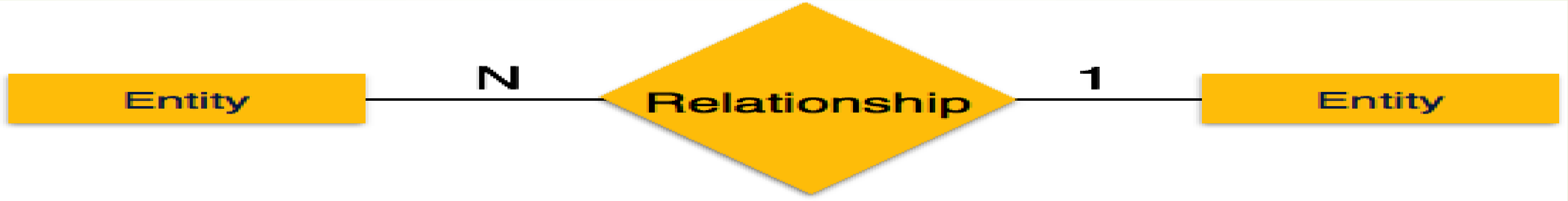


Entity-Relationship Model

- **One-to-many** – When more than one instance of an entity is associated with a relationship, it is marked as '1:N’.

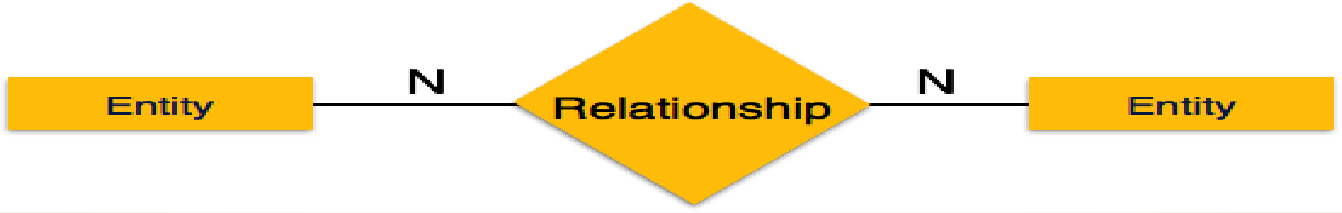


- **Many-to-one** – When more than one instance of entity is associated with the relationship, it is marked as 'N:1’.



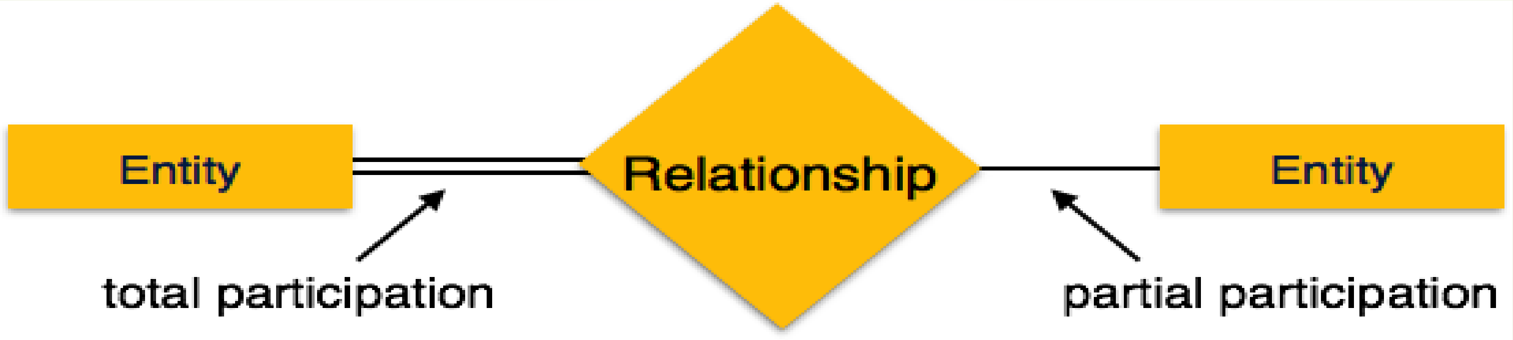
Entity-Relationship Model

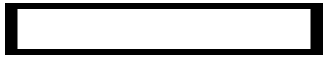
- **Many-to-many** – The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship.



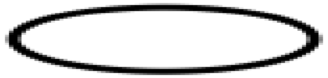
Entity-Relationship Model

- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

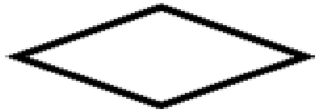




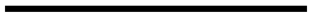
Represents Entity



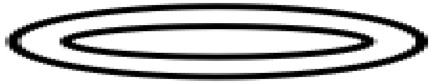
Represents Attribute



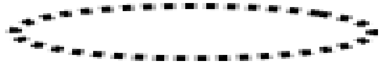
Represents Relationship



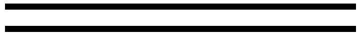
Links Attribute(s) to entity set(s) or
Entity set(s) to Relationship set(s)



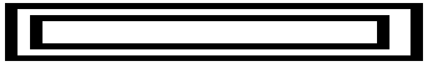
Represents Multivalued Attributes



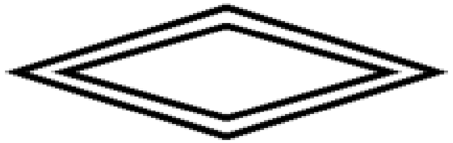
Represents Derived Attributes



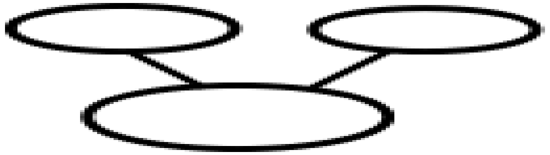
Represents Total Participation of Entity



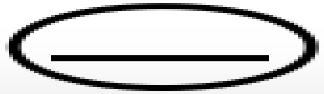
Represents Weak Entity



Represents Weak Relationships



Represents Composite Attributes



Represents Key Attributes / Single Valued
Attributes

Constraints

- Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints :

- **Key constraints**
- **Domain constraints**
- **Referential integrity constraints**

Constraints

- **Key Constraints**

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

- Key constraints force that :

- In a relation with a key attribute, no two tuples can have identical values for key attributes.
- A key attribute can not have NULL values.

- Key constraints are also referred to as **Entity Constraints**.

Constraints

- **Domain Constraints**

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation.

Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Constraints

- **Referential integrity Constraints**

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

Data Definition Language (DDL)

Data Definition Language (DDL) statements are used to classify the database structure or schema. Here are the lists of tasks that come under DDL:

- **CREATE** - used to create objects in the database
- **ALTER** - used to alters the structure of the database
- **DROP** - used to delete objects from the database
- **TRUNCATE** - used to remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT** - used to add comments to the data dictionary
- **RENAME** - used to rename an object

Data Manipulation Language (DML)

- **Data Manipulation Language (DML)**

- Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the lists of tasks that come under DML:

- **SELECT** - It retrieves data from a database
- **INSERT** - It inserts data into a table
- **UPDATE** - It updates existing data within a table
- **DELETE** - It deletes all records from a table, the space for the records remain
- **MERGE** - UPSERT operation (insert or update)
- **CALL** - It calls a PL/SQL or Java subprogram
- **EXPLAIN PLAN** - It explains the access path to data
- **LOCK TABLE** - It controls concurrency

The Data Control Language (DCL)

- The Data Control Language (DCL) is used to control privilege in Databases. To perform any operation in the database, such as for creating tables, sequences, or views, we need privileges. Privileges are of two types,
  - **System** - creating a session, table, etc. are all types of system privilege.
  - **Object** - any command or query to work on tables comes under object privilege. DCL is used to define two commands.

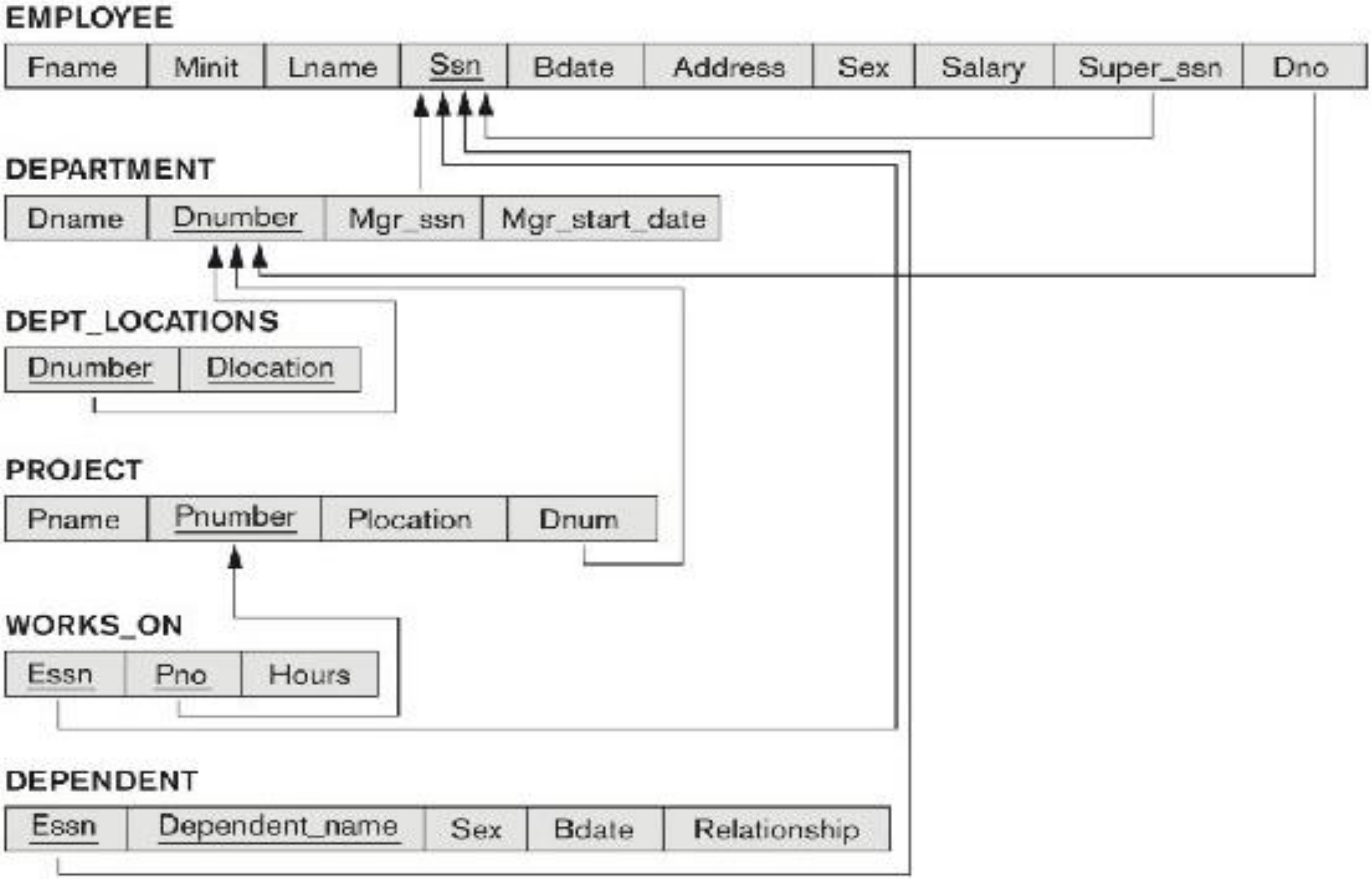
These are:

- **Grant** - It gives user access privileges to a database.
- **Revoke** - It takes back permissions from the user.

Transaction Control Language (TCL)

Transaction Control statements are used to run the changes made by DML statements. It allows statements to be grouped into logical transactions.

- **COMMIT -** It saves the work done
- **SAVEPOINT -** It identifies a point in a transaction to which you can later roll back
- **ROLLBACK -** It restores the database to original since the last COMMIT
- **SET TRANSACTION -** It changes the transaction options like isolation level and what rollback segment to use.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship