

Database System Architecture

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized or client server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. distributed database span multiple geographically separated machines.

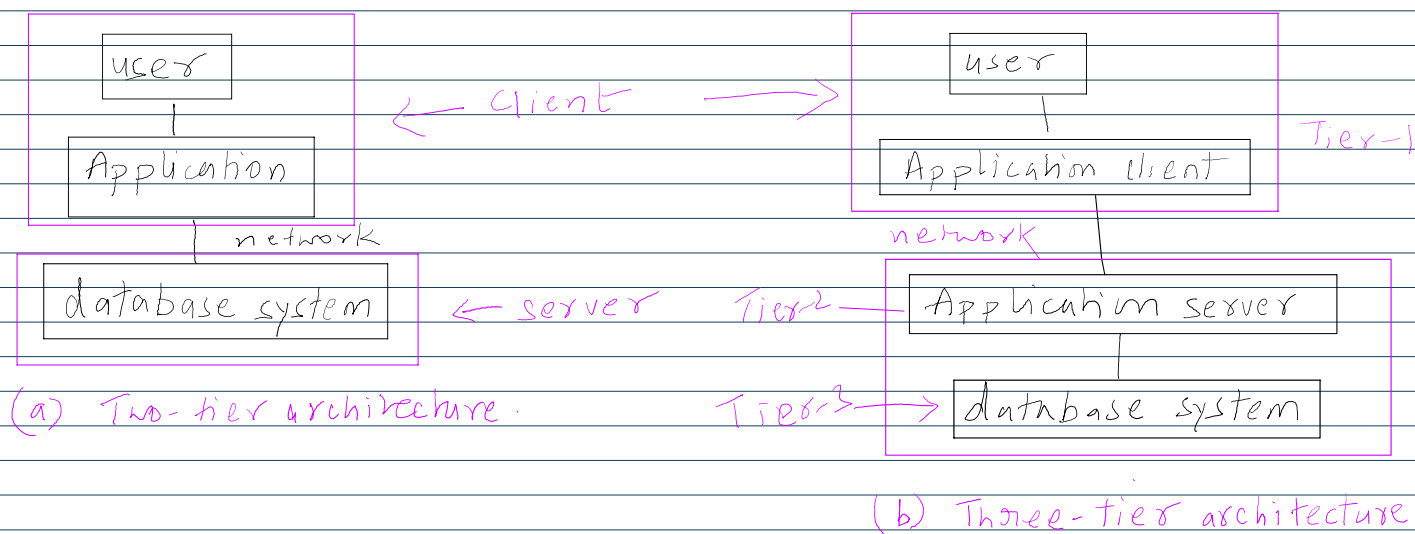
A database system is partitioned into modules that deals with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components. The storage manager is important because database typically requires a large amount of storage space. The query processor is important because it helps the database system simplify and facilitates access to data.

It is a job of a database system to translate updates and query written in the nonprocedural language at the logical level into an efficient sequence of operations at the physical level.

Database applications are usually partitioned into 2 or 3 parts as in above Figure. In a two tier architecture, the application resides at the client machine where it invokes database system functionality at the server machine through query language

statements. Application program interface standards like ODBC(Object Data-Base connectivity) and JDBC(Java Data-Base Connectivity) are used for interaction between the client and the server. In contrast, in a three tier architecture. the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface.

The application server in turn communicates with the database system to access data. The business logic of the application, which says, what action to carry out under what condition is embedded in the application server. Instead of being distributed across multiple clients, three tier applications are more appropriate for large applications and for applications that run on the worldwide web.



Query Processor:

The query processor components include:

1. **DDL interpreter:** Which interprets DDL statements and record the definition in the data dictionary.
2. **DML compiler:** Which translates DML statements in a query language into an evaluation plan consisting of low level instructions that query evaluation engine understands.

A query can usually be translated into a number of alternatives evaluation plan that all give us the same result. The DML compiler also performs query optimization, that is, it picks the lowest. cost evaluation plan from among the alternatives.

Query evaluation engine: Which executes low level instruction generated by DML compiler.

Storage manager:

A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system. The storage manager translates the various DML statements into low level file system commands. Thus, the storage manager is responsible for storing, retrieving and updating data in the database.

The storage manager components include:

Authorization and integrity manager: Which test for the satisfaction of integrity constraints and check the authority of the user to access data.

Transaction manager: Which ensures that database remains in a consistent(correct) State, despite system failures and that concurrent transaction-execution proceed without conflicting.

File Manager: Which manages the collection of space on disk storage and the data structures used to

represent information stored on disk.

Buffer Manager: Which is responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. The buffer manager is a critical part of a database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Conceptual Database Design - Entity Relationship(ER) Modeling:

Database Design Techniques

1. ER Modeling(Top Down Approach)
2. Normalization(Bottom up approach)

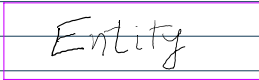
What is ER modeling?

A graphical technique for understanding and organizing the data independent of the actual database implementation.

What is an Entity?

Anything that has an independent existence and about which we collect data. It is also known as entity type.

In ER modeling, notion for entity is given below:



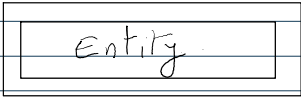
Entity

Entity instance: Entity instance is a particular member of the entity type. Example of entity instance: A particular employee.

Regular Entity: An entity which has its own key attributes is a regular entity. Example of a regular entity: Employee

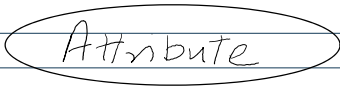
Weak entity: So an entity which depends on other entity for its existence and doesn't have any key attribute of its own. Is a weak entity. Example, for a weak entity: in a parent/child relationship, a parent is considered as a strong entity, and a child is a weak entity.

In ER modeling, notion for weak entity is given below:



Entity

Attributes: Properties or characteristics, which describe entities are called attributes. In ER modeling, notion for attribute is given below:



Attribute

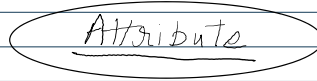
Domain of Attributes:

The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday,... Friday}. Hence, this set can be termed as the domain of the attribute day.

Key Attribute

The attribute (or combination of attributes) which is unique for every entity instance, is called key attribute. For example, the Employee_id of an employee, pan_card_number of a person, Etc. If the key

attribute consist of two or more attributes in combination, it is called a composite key. In er modeling notion for key attribute is given below:

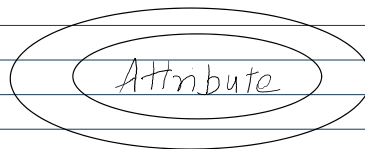


Simple attribute: If an attribute cannot be divided into simpler components, it is a simple attribute. For example, employee_ID of an employee.

Composite attribute: If an attribute can be split into components, it is called a composite attribute. For example name of an employee which can be split into first_name, middle_name and last_name.

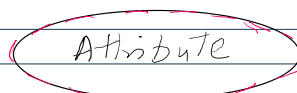
Single valued Attribute: If an attribute can take only a single value for each entity instance, it is a single valued attribute. For example, age of a student. It can take only one value for a particular student.

Multi-valued Attributes: If an attribute can take more than one value for each entity instance, it is a multi valued attribute. For example, mobile number of an employee. A particular employee may have multiple mobile numbers. in er modeling notion for multi valued attribute is given below:



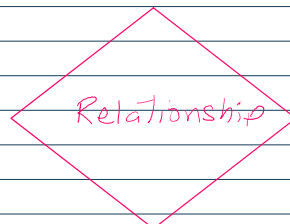
Stored Attribute: An attribute which need to be stored permanently is a stored attribute. For example, name of a student.

Derived Attribute: An attribute which can be calculated or derived based on other attributes. Is a derived attribute. For example, age, of an employee which can be calculated from the date of birth and current date. In ER modeling, notion for derived attribute is given below:

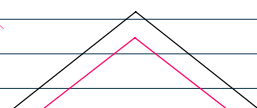


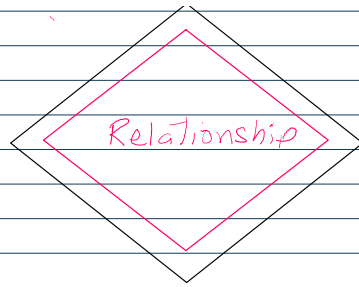
Relationships

Association between entities are called relationships. For example, an employee works for an organization. Here "works for" is a relation between the entities employee and organization. In ER modeling, notion for relationship is given below:



However, in ear modeling to connect a weak entity with others, you should use a weak relationship notion as given below:





Degree of a Relationship

Degree of a relationship is the number of entity types involved. The n-ary Relationship is the general form for degree n. Special cases are unary, binary and ternary, where degree is one, 2 and 3 respectively.

Example for unary relationships an employee is-a-manager of another employee.

Example of a binary relationship. an employee works-for a department.

Example of ternary relationship. a customer purchase items from a shopkeeper.

Cardinality of a relationship.

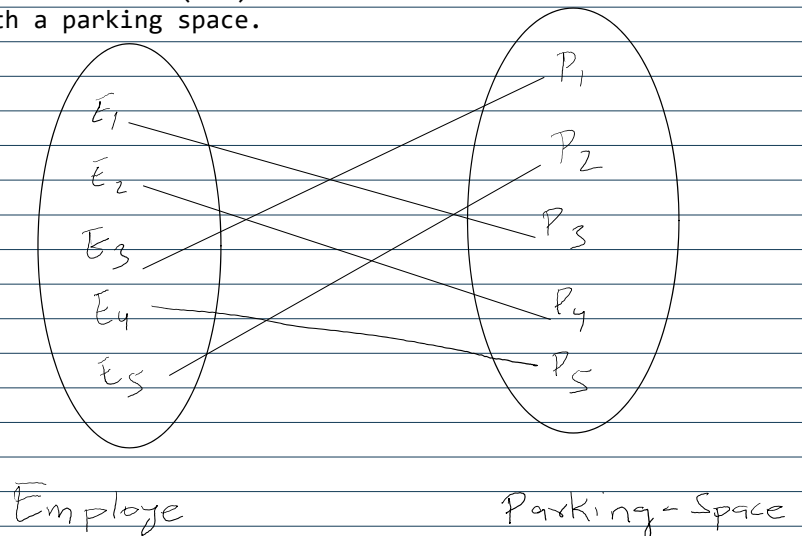
Relationship cardinalities specify how many of each entity type is allowed. Relationship can have four possible connectives as given below:

1. One to One (1:1) relationship
2. One to Many (1:N) relationship
3. Many to one (N:1) relationship
4. Many to many (N:N) relationship

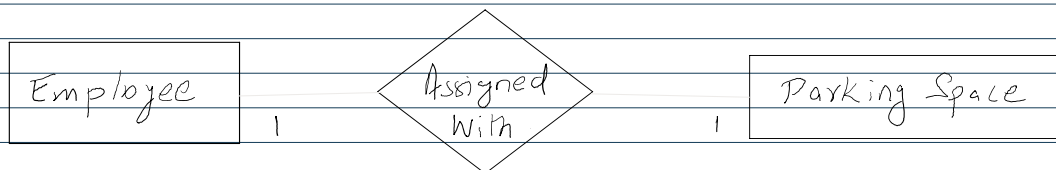
The minimum and maximum values of this connectivity is called the cardinality of the relationship.

Example for cardinality - One-to-one(1:1)

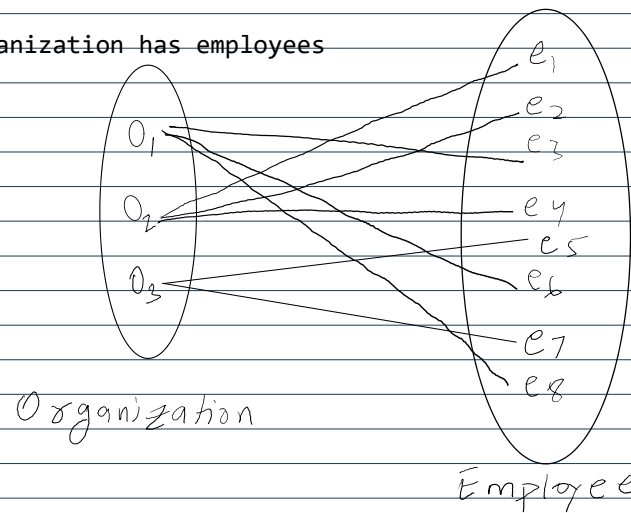
Employee is assigned with a parking space.



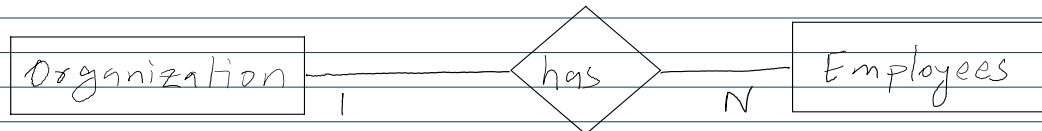
One employee assigned with only one parking space and one parking space is assigned to only one employee. Hence, it is a 1:1 relationship and cardinality is one-to-one(1:1).



One-to-Many (1:N): Organization has employees



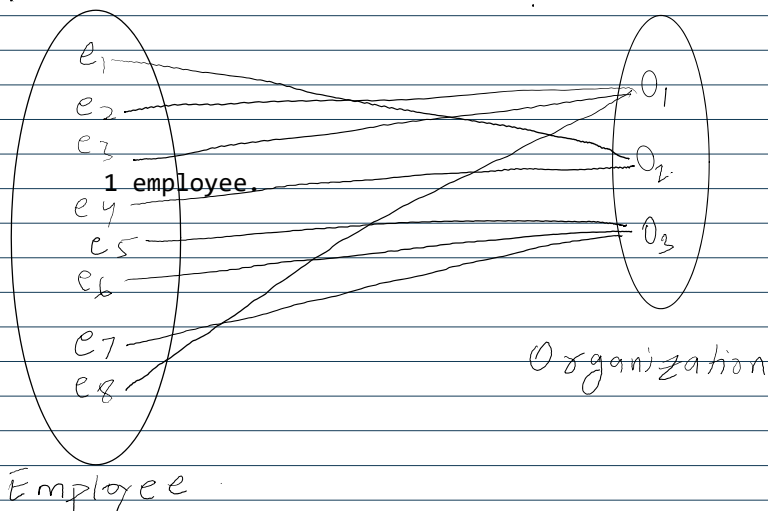
One organization can have many employees, but one employee works in only one organization. Hence it is a 1:N relationship and cardinality is one-to-many(1:N).



ER Modelling diagram (1:N)

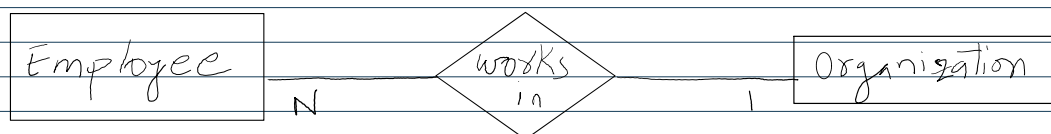
Many-to-one

It is a reverse of one to many relationship employees work in organization.

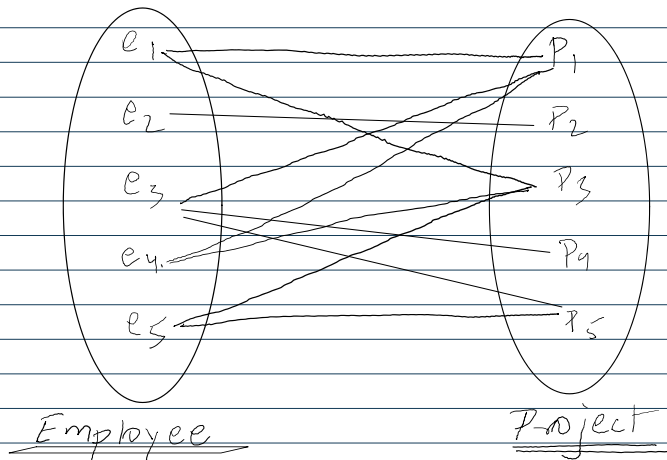


One employee works in one organization, but one organization can have many employees. Hence, it is N:1 relationship and cardinality is many-to-one(N:1).

In er modeling this can be mentioned using notation as given below:



Cardinality - Many-to-Many (M:N)
Employees works-in many projects



One employee can work in many projects, One project can have many employees.
Hence it is a M:N relationship and cardinality is Many-to-Many (M:N)
In er modeling this can be mentioned using notation as given below:

