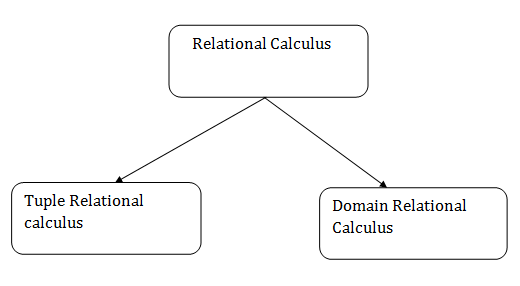
**Relational Languages**

Relational language is a type of programming language in which the programming logic is composed of relations and the output is computed based on the query applied. Relational language works on relations among data and entities to compute a result. Relational language includes features from and is similar to functional programming language.

**RELATIONAL CALCULUS**

* Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results.
* The relational calculus tells what to do but never explains how to do.



**TUPLE RELATIONAL CALCULUS (TRC)**

* The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.
* The result of the relation can have one or more tuples.

**Notation**

|  |
| --- |
| {T | P (T)} or {T | Condition (T)} |

Where

**T** is the resulting tuples

**P(T)** is the condition used to fetch T.

**Example**

**{ T.name | Author(T) AND T.article = 'database' }**

**OUTPUT:** This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

**DOMAIN RELATIONAL CALCULUS (DRC)**

* The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.
* Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ┓ (not).
* It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

**Notation**

|  |
| --- |
| **{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}** |

Where

**a1, a2** are attributes  
**P** stands for formula built by inner attributes

**Output:** This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

**DATABASE DESIGN AND THE E-R MODEL**

* It is a high-level conceptual data model diagram. ER modeling helps you to analyze data requirements systematically to produce a well-designed database. The Entity-Relation model represents real-world entities and the relationship between them. It is considered a best practice to complete ER modeling before implementing your database.
* 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.

**OVERVIEW OF THE DESIGN PROCESS**

* Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.
* The main objectives of database designing are to produce logical and physical designs models of the proposed database system.
* The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
* The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

**Why Database Design is Important?**

* It helps produce database systems
* That meet the requirements of the users
* Have high performance.
* Database designing is crucial to high performance database system.

**THE ENTITY-RELATIONSHIP MODEL WITH CONSTRAINS**

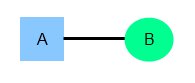
An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of [data](https://www.webopedia.com/TERM/D/data.html) within [databases](https://www.webopedia.com/TERM/D/database.html) or information systems. An entity is a piece of data-an [object](https://www.webopedia.com/TERM/O/object.html)or concept about which data is stored.

**Relationships Between Entities**

A relationship is how the data is shared between entities. There are three types of relationships between entities:

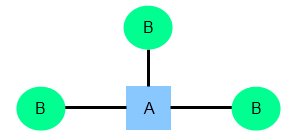
**1. One-to-One**

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



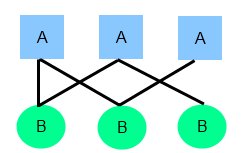
**2. One-to-Many**

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



**3. Many-to-Many**

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



**ENTITY RELATIONSHIP DIAGRAMS**

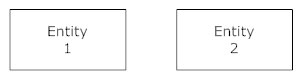
* An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.
* By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.
* ER diagrams are used to sketch out the design of a database.

****

**ER Diagram Uses**

When documenting a system or process, looking at the system in multiple ways increases the understanding of that system. ERD diagrams are commonly used in conjunction with a data flow diagram to display the contents of a data store. They help us to visualize how data is connected in a general way, and are particularly useful for constructing a relational database.

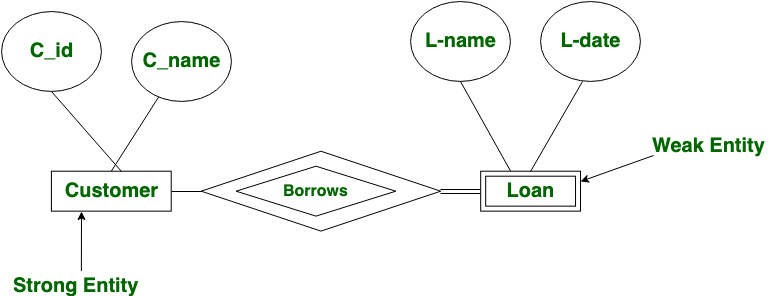
**Entity Relationship Diagram Tutorial**

* **Identify the entities.** The first step in making an ERD is to identify all of the entities you will use. An entity is nothing more than a rectangle with a description of something that your system stores information about. This could be a customer, a manager, an invoice, a schedule, etc. Draw a rectangle for each entity you can think of on your page. Keep them spaced out a bit.
* 
* **Identify relationships.** Look at two entities, are they related? If so draw a solid line connecting the two entities.
* **Describe the relationship.** How are the entities related? Draw an action diamond between the two entities on the line you just added. In the diamond write a brief description of how they are related.
* **Add attributes.** Any key attributes of entities should be added using oval-shaped symbols.
* **Complete the diagram.** Continue to connect the entities with lines, and adding diamonds to describe each relationship until all relationships have been described. Each of your entities may not have any relationships, some may have multiple relationships.

**WEAK ENTITY SET IN ER DIAGRAMS**

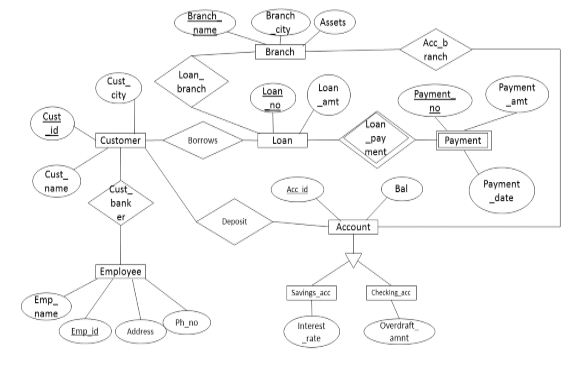
* An entity type should have a key attribute which uniquely identifies each entity in the entity set, but there exists some entity type for which key attribute can’t be defined. These are called Weak Entity type.
* The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets and the entity sets which have a primary key are known as strong entity sets.
* As the weak entities do not have any primary key, they cannot be identified on their own, so they depend on some other entity (known as owner entity). The weak entities have total participation constraint (existence dependency) in its identifying relationship with owner identity. Weak entity types have partial keys. Partial Keys are set of attributes with the help of which the tuples of the weak entities can be distinguished and identified.

Weak entity is **depend on strong entity** to ensure the existence of weak entity. Like [strong entity](https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/), weak entity does not have any primary key, It has partial discriminator key. Weak entity is represented by double rectangle. The relation between one strong and one weak entity is represented by double diamond.



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

**DATABASE DESIGN FOR BANKING ENTERPRISE**



**Relational Models**

