**Database design**

**Constraints**

Its a strict rule or statement declared in a database that must be true or adhered to at all times.

Constraints exist in all parts of a database schema, for example:

* **Keys –** studentNumber uniquely identifies a single student.
* **Single-value constraints –** a person can only have one biological father.
* **Referential integrity constraints –** if you work for a company, a domain constraint has to exist, defining people’s ages between 0 and 150.
* **Cardinality constraints –** maximum of 100 students can enroll for a single course.

**Entity-Relationship model**

An (E-R) model defines the logical representation, structure and description of data in a relational database.

E-R model is expressed in terms of:

* **Entities**
* **Attributes**
* **Relationships**
* **Cardinalities**

**Entity-Relationship Diagrams**

General notations of ER diagrams:

* **Entity classes** are shown **in rectangles.**
* **Relationship** are shown **by diamonds.**
* The name of the **entity** is shown inside **the rectangle.**
* The name of the **relationship** is shown near the **diamond**.

**Tips on ERDs**

* Identify ***entity types*** by searching for nouns and noun phrase.
* Assume all entities are ***strong*** and check for weak ones at a later stage.
* Each strong entity needs an ***identifier.***
* Assume all relationship have ***optional*** participation and check for mandatory (total) ones at a later stage.

**Notations for ERDs**

**Chen’s notation**

**Entities**

Entities are represented by ***rectangles.***

Student

**Attributes**

Attributes are represented by ***ovals***

Flights

**Relationships**

A relationship is shown with ***diamonds***

Department

Employee

employs

Relationships which involve many ***employees*** is shown as an “m” at the ***far end*** of the relationship as we are reading it.

m

Department

Employee

employs

**Logical database design**

Logical database design focuses on the abstract representation of data and the relationships between the entities.

During the logical design phase:

* Entities and their attributes are identified.
* Relationships between entities are defined.

**Relational model**

Relational model represents data in the form of ***tables.***

Components of the relational model:

* **Data structure -**  data is organized in the form of tables with rows and columns.
* **Data manipulation –** powerful operations (using SQL) are used to manipulate data stored.
* **Data integrity –** Facilities are included to specify business rules that maintain the integrity of data.

A table must meet the following conditions to be a relation:

* Each relation must have a unique name.
* No multi-valued attributes in a relation.
* Each row is unique; there are no duplicate rows.
* Each attribute within a relation has a unique name.
* Sequence of columns is insignificant.

**Integrity constraints**

**Domain constraints**

A ***domain*** is a set of values defined by a domain name e.g. data type, length, format, range, allowable values and further restrictions (if applicable)

***Domain constraints*** require that all values in a column must be from the same domain.

Advantages of domains:

* Verifies that the value for an attribute in valid.
* Ensures the various data manipulation operations are logical.
* Domains help conserve effort in describing attribute characteristics.

**Entity integrity**

A ***NULL*** value in a column indicates that there is no value in that column.

* No ***primary key*** can be NULL.
* There is only one NULL value for all data types.

**Referential integrity**

A foreign key is used to associates two or more tables. Furthermore, a foreign key is defined as columns which contain data values.

Therefore, these values ***associate one or more rows*** in the table with the foreign key to a unique row in the table with the ***primary key***.

Ensures that ***relationships*** between tables are ***maintained, consistent, enforced and has integrity*** via the use of ***foreign key constraints***.

**Transforming ERD into relations**

**Mapping regular entities**

* The ***name*** of the relation is the same as the entity type.
* Each ***simple attribute*** of the entity type becomes an attribute of the relation.
* The ***identifier*** of the entity type becomes the ***primary key*** of the relation.

**Mapping multi-valued attributes**

* ***Two relations*** must be created.
* The ***1st relation*** contains all of the attributes of the ***entity type***, except the ***multi-valued*** attributes.
* The ***2nd relation*** contains two attributes that ***form the primary key.*** The ***first*** attribute contains the ***primary key of the*** ***1st******relation***