**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DBMS Theory Assignment #5**

**Entity relational model**

**Submitted by:**

Arjun Neupane

013BSCCSIT009

**Submitted to:**

|  |  |
| --- | --- |
| Er. Sanjay K. Yadav  Lecturer, St. Xavier’s College |  |

**Date of Submission:** 27th August 2015

**Entity relational model:**

**1.What do you mean by entity-relational model? Explain.**

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 within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

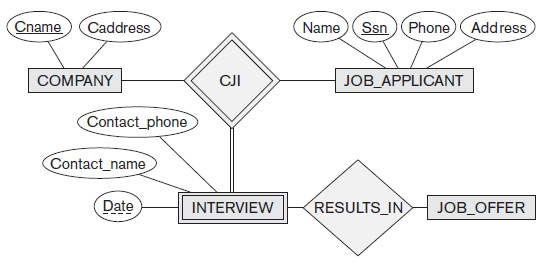


Fig: ER diagram

While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ERD is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Three main components of an ERD are the entities, which are objects or concepts that can have data stored about them, the relationship between those entities, and the cardinality, which defines that relationship in terms of numbers.

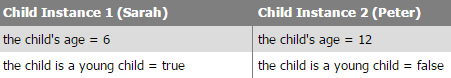
**2. Define entity and give an example.**

Entity is defined as the real world object that can be seen and are around us. It is generally an existing or real thing. In relation to a database, an entity is a single person, place, or thing about which data can be stored. In data modeling, an entity is some unit of data that can be classified and have stated relationships to other entities.

A member of the entity group is called an entity instance. For example, if a family had 2 children, Sarah and Peter, Sarah would be one instance of "the child" entity and Peter would be another instance of "the child" entity. By creating an entity to represent "the child", information such as the child's age can be collected for each child.

We can also infer attributes from this data just as we can in a normal rule. Therefore, the attribute "the child is a young child" can hold a different value for each child, depending on the child's age. For instance, if we had a rule:

the child is a young child if the child's age < 8.



**3. Explain the different between an entity class and an entity instance.**

The difference between an entity and entity instance are:

|  |  |
| --- | --- |
| **Entity class** | **Entity instance** |
|  |  |

**4. Define attribute and its types.**

In general, an attribute is a characteristic property of an entity. In a database management system (DBMS), an attribute refers to a database component, such a table. It also may refer to a database field. Attributes describe the instances in the row of a database.

The types of attributes are:

**Required or Optional Attributes:**

A required attribute is an attribute that must have a value in it, while an optional attribute may not have a value in it and can be left blank. The reasoning for making an attribute required is to put emphasis on what is important in that entity and what makes it stand out from other entities.

Example**:** Consider the entity Student above stud\_LastName and studFirstName would be required attributes as it uniquely defines that table and we assume all students have a first and last name. Optional attributes in the table Student could be stu\_MiddleName, stu\_Email, and stu\_Phone since some students may not have a middle name, a phone number, or an email address .

**Keys and non-keys Attributes:**

In every entity an attribute or grouped attributes uniquely identify that entity. These attributes are the key attributes and range from Primary key (single attribute identifier) to a Composite Key (Multi attribute Identifier). The rest of the attributes after the identifier are considered the non-key attributes or descriptors, which just describe the entity.

Example: Above in the table Student there is only one unique identifier, stu\_LastName, which is the primary key of the table. The rest of the attributes are descriptors.

**Single and Composite Attributes:**

Attributes can be classified as having many parts to them or just a single unbreakable attribute. The composite attribute is an attribute that can be subdivided into other single attributes with meanings of their own. A single attribute is just an attribute that cannot be subdivided into parts.

Example: Imagine from the entity Student that instead of having the three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName it had one attribute called stu\_Name. The attribute stu\_Name would be considered a composite attribute since it can be subdivided into the other three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName. The rest of attributes would be consider single attributes since they can't be subdivided into parts.

**Single-valued and multi-valued Attributes:**

Attributes can be classified as single or multi-value. The single-value attribute can only have one value, while the multi-valued attributes usually can store multiple data in them.

Example: In the entity Student, stu\_Address could be considered a multi-value attribute since a student could have multiple addresses where he lives at. An example of a single-value attribute would be stu\_LastName since a student usually has one last name that uniquely identifies him/her.

**Derived Attributes:**

Derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example: In the entity Student, stu\_Age would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.﻿﻿

**5. What is derived attributes?**

The attribute from which another attribute value is derived is called derived or stored attribute.

Derived attributes are usually created by a formula or by a summary operation on other attributes.

There may be a case when two or more attributes values are related. Take the example of age. Age of a person can be calculated from person’s date of birth and present date. Difference between the two gives the value of age. In this case, age is the derived attribute.

**6. Define relationship and give an example.**

A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items.

For example, in a bank database a CUSTOMER\_MASTER table stores customer data with a primary key column named CUSTOMER\_ID; it also stores customer data in an ACCOUNTS\_MASTER table, which holds information about various bank accounts and associated customers. To link these two tables and determine customer and bank account information, a corresponding CUSTOMER\_ID column must be inserted in the ACCOUNTS\_MASTER table, referencing existing customer IDs from the CUSTOMER\_MASTER table. In this case, the ACCOUNTS\_MASTER table’s CUSTOMER\_ID column is a foreign key that references a column with the same name in the CUSTOMER\_MASTER table. This is an example of a relationship between the two tables.

**7. Explain the difference between a relationship class and a relationship instance.**

The differences between relationship class and relationship instance are:

|  |  |
| --- | --- |
| **Relationship class** | **Relationship instance** |
|  |  |

**8. Define degree of relationship.**

The degree of relationship is the number of occurrences in one entity which are associated or linked to the number of occurrences in another.

The types of relationship are:

One to one (1:1)

One to many (1:M)

Many to one (M:1)

Many to many (M:M).

**9. List and give an example of the three types of binary relationships. Draw an E-R diagram**

**for each.**

Three types of binary relationships are:

1:1 - a single entity instance of one type is related to a single-entity instance of another type.

1:N - a single entity instance of one type is related to many-entity instances of another type.

M:N - many-entity instances of one type relate to many-entity instances of another type.

**Reference:**

<http://www.webopedia.com/TERM/E/entity_relationship_diagram.html>