Knowledge Leveling

Introduction to Databases

DATABASE DESIGN

Overview

Data

1950

It's a symbolic representation. By itself, it has no meaning.

Information

1950

- Meaningful data, which has been processed and converted into information.

Data - It's a symbolic representation. By itself, it has no meaning. Difference between Data and Information. Example with a number, which can be one year, subject code, door number, etc.

Information: Meaningful data, which has been processed and converted into information.

GeneXus

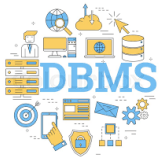
Database

Related data set.

Requirements:

- Represent some aspect of reality
- Have an implicit meaning
- Serve a specific purpose.

Database Management System (DBMS)



Program that provides a set of services for the maintenance and construction of databases.

A **Database** is a set of **data related to each other**. It is NOT a DBMS. It is wrong to say that Oracle is a database. You don't need a DBMS to have a database, e.g., a folder, Excel spreadsheets, sequential files, comma separated files, etc.

By **data** we mean facts that can be recorded. This definition may be too broad, so to refer to the term "database" that we'll be using, let's add to that definition that the data must also meet the following properties:

- Represent some aspect of reality
- Have an implicit meaning
- Serve a specific purpose for a specific user group.

Example: Phone book, search by person or by phone

CONCEPTUAL MODEL DESIGN

Model Entity Relationship (MER)

The design of a **database** consists of three stages:

- Conceptual Model Design
- Relational Model Design
- Physical Model Design

Model Entity Relationship (MER)

It's a high-level conceptual model and is used for data definition. It is based on the graphic representation of objects (entities) and relationships between these objects.

Entity


Entity

It is a distinguishable object from reality, of which we are interested in storing data.

For example: Client, Country, Product, etc.

An **entity-relationship model or entity-relationship diagram** is a data modeling tool that allows you to represent the relevant entities of an information system as well as their interrelationships and properties.

It allows for:

- Specific description of the information requirements (data types, relationships, restrictions that the data must comply with).
- Easy to understand by non-technical users.

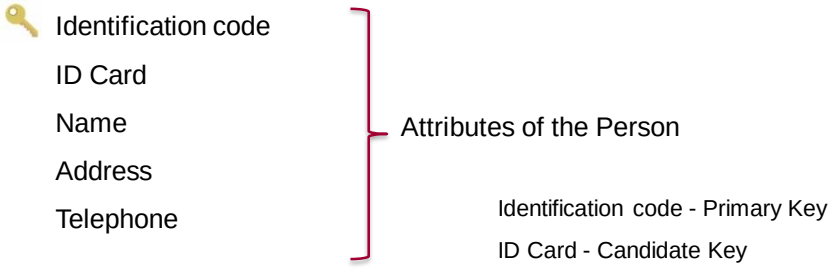
Entity is a real-world object that we can distinguish from other objects, and from which we are interested in storing data.
Attribute: All those characteristics of the entity that we are interested in knowing.

Entities may be related to each other, and there may be attributes that correspond to that relationship.

Attribute

It's a characteristic of the entity that we are interested in knowing.

Example: Entity Person



Consider the entity Person.
There are certain details that are common to all people:

- Identification code
- ID Card
- Name
- Address
- Phone

Within this set of attributes, there will be one (or a set) that will represent and make each person unique. In this example we have an identification code of the person and the identity card.

If we take the code of the person, we can say that it will be the **primary key** of the entity, and that there can't be two people with the same identification code.
The Identity Card could also be chosen as the primary key since there are no two people with the same card, but in this example it will play the role of a **candidate key**, since due to its functional characteristics it operates as a unique attribute but is not the primary key of the entity.

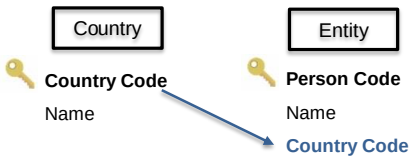
Attribute: Primary Key

It's the attribute (or set of attributes) that uniquely identifies each element of the Entity.

A Person has an identification code that is unique

Attribute: Foreign Key

Attribute that is the primary key in entity A and participates as an attribute in entity B with reference to entity A.



- Country code is the primary key in the entity Country.
- Country code is the foreign key in the entity Person.

INTER-ENTITY RELATIONSHIPS

Cardinality

It is the way in which entities relate to each other.

INTER-ENTITY RELATIONSHIPS

- 1-1 relationship

- 1-N relationship

- N-N relationship

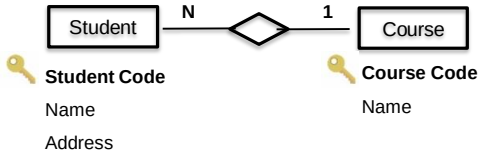
The Cardinality between two entities is the way in which these entities relate to each other.

There are 4 types of relationships that can be established between entities, which establish how many type B entity occurrences a type A entity occurrence can be related to:

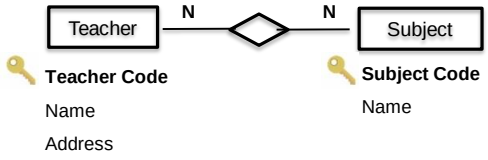
- **One-to-one (1-1) relationship:** One element of entity A relates to one element of entity B, and one element of entity B relates to one element of entity A.
- **One-to-many (1-N) relationship:** One element of entity A relates to many elements of entity B, and one element of entity B relates to one element of entity A.
- **Many-to-many (N-N) relationship:** One element of entity A relates to many elements of entity B, and one element of entity B relates to many elements of entity A.

Examples

1) A Student enrolls in one Course, and in one Course many Students are enrolled.

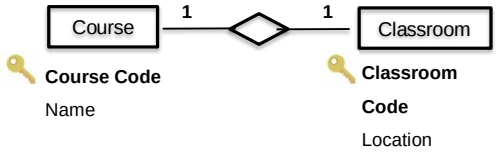


2) One Teacher teaches many Subjects, and one Subject is taught by many Teachers.



Examples

3) A Course is taught in one classroom, and in one Classroom there is only one Course.



Weak entities

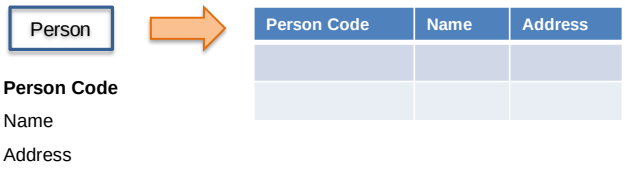
They are those entities that express dependence on another. A weak Entity does not exist on its own but depends on the existence of another.

DESIGN OF THE LOGICAL MODEL

Relational Model (RM)

Logical Model

Every strong Entity declared in the MER will be a physical table in the Logical Model.

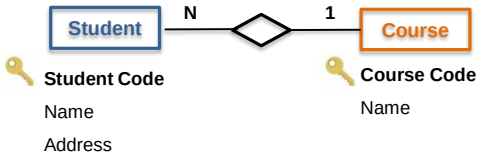


Each strong MER entity will be translated into a table in the RM. Remember that a strong entity is an element of reality that we need to know and store its data.

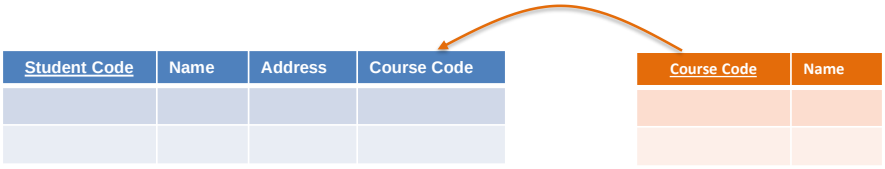
It is always necessary to consider the cardinality aspects that exist between the entities defined in the MER for the creation of the tables.

Representation of 1-N relationship

A Student enrolls in one Course, and in one Course many Students are enrolled.



The primary key of the Course is added as the foreign key in Student.



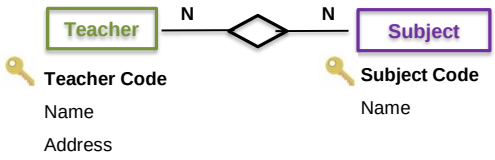
The Code Course attribute is the primary key in the Course and the foreign key in the Student. Only one course may be indicated per student.

Thus, it is represented that a Student is associated with a Course, but that a Course can be associated with many Students.

This way we may have:
María Rodríguez attends a Web Design course
Luis Gómez attends a Network Technician course
Andrés García attends a Web Design course
And so on.

Representation of N-N relationship

One Teacher teaches many Subjects, and one Subject is taught by many Teachers.



The "teach" relationship is translated into a third table that combines the primary key of the Teacher with the primary key of the Subject, and is able to add other attributes of its own.

Compound key

Teacher Code	Name	Address

Subject Code	Name

Teacher Code	Subject Code	Date

First, we must make sure that the Teacher and Subject exist with all their data.

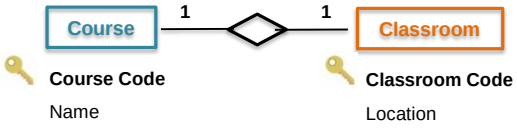
As a Teacher can teach many Subjects, and a Subject can be taught by many Teachers, the relationship "teaches" is translated as a third table where at least the Teacher primary key and the Subject primary key are contained. It may also contain other attributes such as the date of teaching.

In this example, the primary key of the relationship is made up of:
Teacher Code, Subject Code

This way we may have:
Juan Pérez teaches Databases, Programming and GeneXus.
María Gómez teaches Operating Systems and Programming.
Ricardo López teaches GeneXus and Networks.
And so on.

Representation of 1-1 relationship

A Course is taught in one classroom, and in one Classroom there is only one Course.



It can be represented in two ways:

- For each Course, reference the Classroom Code.
- For each Classroom, reference the Course Code.

Course Code	Location	Classroom Code

VS.

Classroom Code	Location	Course Code

This relationship can be represented in two ways:

- Adding the Classroom Code as a foreign key in the Course.
- Adding the Course Code as a foreign key in the Classroom.

In either case, in some way, it should be checked that these assignments are not repeated. That is, that no Classroom Code is repeated in the Course, or that no Course Code is repeated in the Classroom.

The choice will have to do with the way in which one wishes to have the data available, and the order to enter it:

- Enter the courses first so that you can then refer to them from the Classroom.
- First enter the Classrooms and then make the reference from the Course.

GeneXus

CONSTRUCTION

Physical Model (PM)

Construction

It determines how the data is stored.

- Data type
- Relationships between data (integrity)
- Restrictions
    - Domain Restrictions
    - Key Uniqueness Restrictions
    - Referential Integrity Restrictions

For the tables to accurately reflect reality, a set of concepts is defined to determine how the data is stored.

- Data Type - Characteristic that the data must meet to be accepted (if it must be numerical, date, character data, etc.).
- Relationships between data (integrity): Primary and foreign keys. This is what defines the integrity and relationship between the data.
- Restrictions:
    - Domain: An attribute will only admit the type of data that is indicated, and not another one.
    - Uniqueness of the key: The value of a primary key attribute cannot be repeated or changed.
    - Referential Integrity: Determined by existing foreign keys. It is not possible to reference data that does not exist, and it is not possible to delete data referenced in other tables.

NORMALIZATION

Normalization

- Process during which unsatisfactory table structures are broken down.
- Ensures that no abnormal situations occur when entering, modifying or deleting data.
- It involves converting the tables to first, second and third normal form.

The content of all columns in the table is a unique value and a list of values.

The table is in First Normal Form and, in addition, all the values declared in it depend on the primary key.

The table is in Second Normal Form and all the columns that do not depend on the primary key are independent of each other.

There are more levels of normalization, but a table is considered normalized if it is converted to at least the Third Normal Form.

INDEXES

- Indexes are a database structure that helps improving the speed of operations. An index basically serves to quickly find data, without having to run through the entire table sequentially in search of a particular row.

TABLE CLIENT:

Order by ClientId (Default)

Index: ClientId

ClientId	ClientName
1	Leo
2	Jacob
3	Theo
4	Jonah
5	Alex
6	Henry
7	Nora

Order by Name

Index: ClientName

ClientId	ClientName
5	Alex
6	Henry
2	Jacob
4	Jonah
1	Leo
7	Nora
3	Theo

Database indexes work in a similar way to a book index, where the item to be indexed and its position will be stored. This way, to search for an element that is indexed, only that element will have to be searched for in the index, thus avoiding running through the entire table being navigated to search for the necessary data.

Unique indexes can also be created, in which no two rows can have the same value in the key column of the index. This means that duplicate values are not allowed.

DATABASE OPERATIONS

Unidad Lógica de Trabajo (UTL)

- A Logical Unit of Work (LUW) is a sequence of database operations performed as a single unit, which involves statements that modify data in one or more of its tables. LUWs correspond to the concept of “database transactions.”
- If a LUW is successful, all changes to the data made during the transaction are confirmed and converted into permanent changes. (**Commit**)
- If, during the course of a LUW, any statement finds errors, all the changes made in the transaction are cancelled. (**Rollback**)

Commit and **Rollback** SQL statements to confirm or undo the LUW:

Commit:

It confirms the changes made to a LUW as permanent, and indicates that the LUW has ended correctly.
It ensures that all changes to the transaction become a permanent part of the database.

Rollback:

If an error occurs in a LUW or the user decides to cancel it, the changes made so far must be rolled back. This is done with the ROLLBACK
instruction, which returns the data to the state it was in at the beginning of the transaction.
It marks an incorrect end to a transaction, and aborts all changes made since it started.

Rollbacks are important for database integrity, because they mean that the database can be restored to a consistent state even after erroneous operations
have been performed. When a database has this capability, it is said to have “Transactional Integrity.”

GeneXus™

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certificactions	training.genexus.com/certifications