TechLift
KICKSTART YOUR CAREER

WEB ENGINEERING

.Net

P@SHA
Pakistan Software Houses Association for IT & ITES

PAKISTAN
PSEB
SOFTWARE EXPORT BOARD

Week 4 - Day 20

Recap Assignment (10 marks)(Time: 1 hr)

Students will Work in pairs and perform a Simple

Crud Operation Using
ADO.NET

They may take help from this video to complete the project.

ASP.NET Web Application

Learning Objectives

By the end of this session, the students will have developed an understanding of:

- Types of Relationship
- Implementing One to One Relationship
- Implementing One to Many Relationship
- Implementing Many to Many Relationship

Types of Relationships

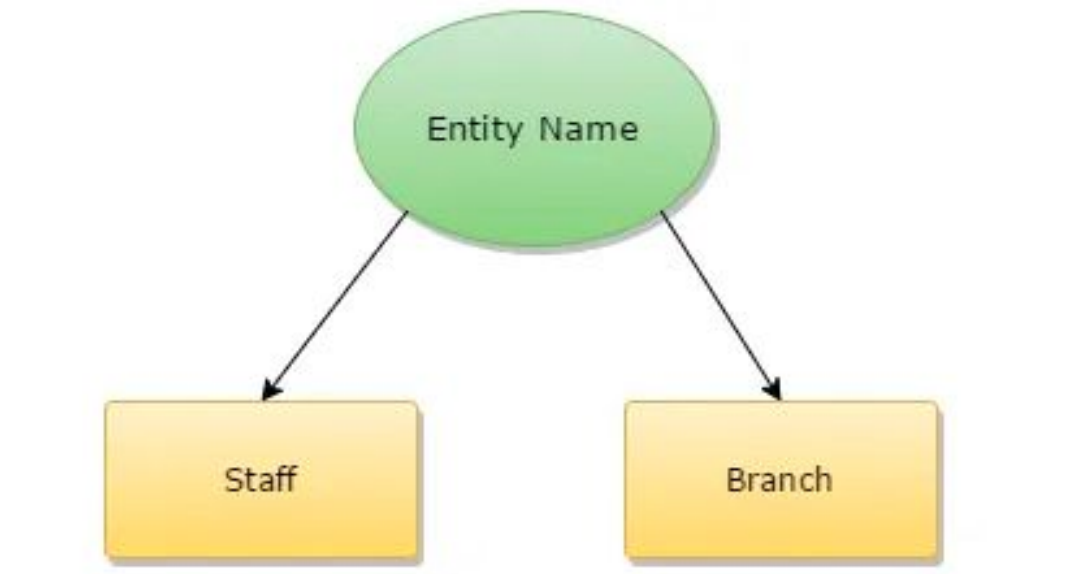
One of the most important things in databases is to understand the types of relations in the databases. That stands for both – a process of designing a database model as well as when you're analyzing your data. Understanding these relations is somehow natural and not so complex but is still essential in the database theory (and practice).

There are 3 different types of relations in the database:

- one-to-one
- one-to-many, and
- many-to-many

Entity Relationship Model

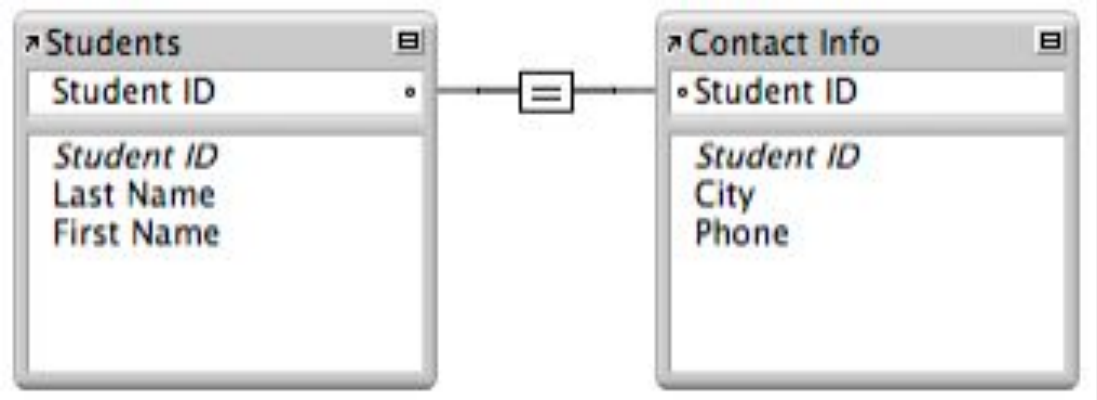
ER-Diagram is a pictorial representation of data that describes how data is communicated and related to each other. Any object, such as entities, attributes of an entity, sets of relationship, and other attributes of relationship, can be characterized with the help of the ER diagram.

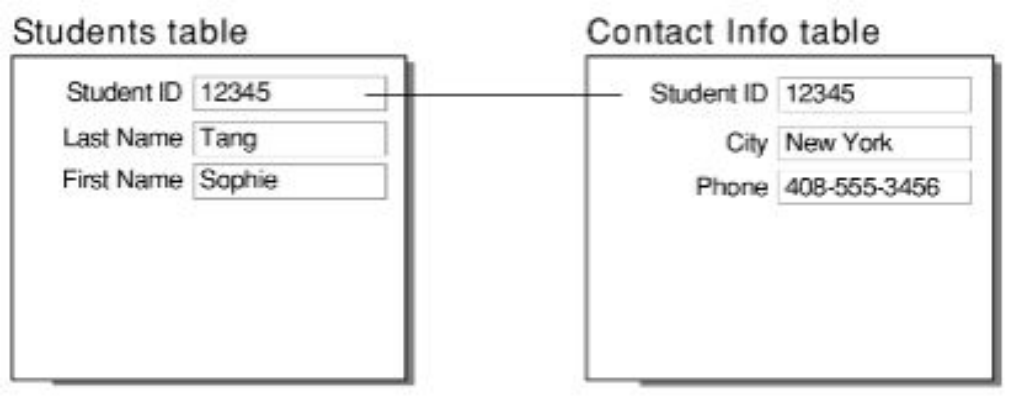


Diagrammatic representation of the Staff and Branch entity types.

One to One Relationship

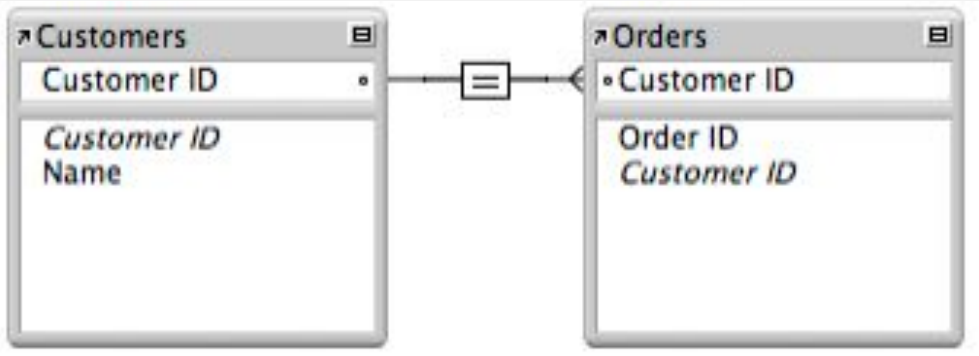
In a one-to-one relationship, one record in a table is associated with one and only one record in another table. For example, in a school database, each student has only one student ID, and each student ID is assigned to only one person.

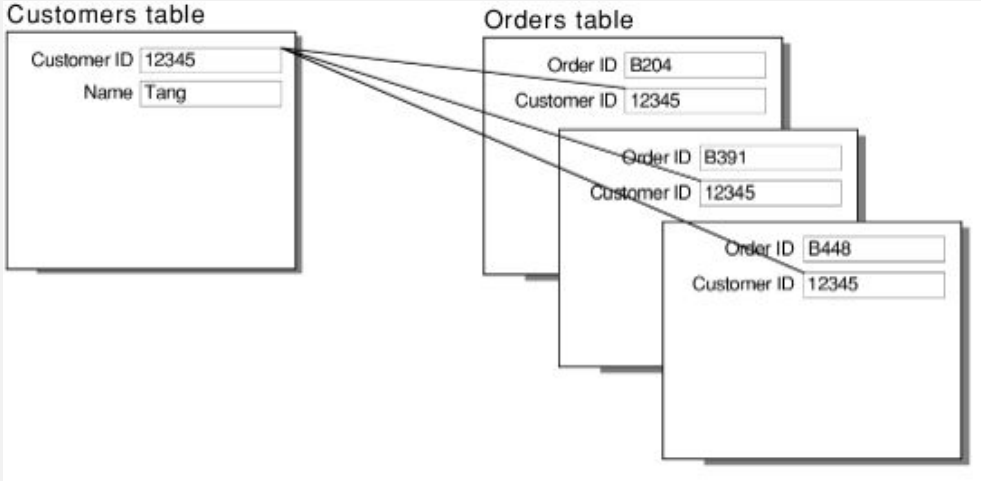




One to Many Relationship

In a one-to-many relationship, one record in a table can be associated with one or more records in another table. For example, each customer can have many sales orders.





Many to Many Relationship

A *many-to-many relationship* occurs when multiple records in a table are associated with multiple records in another table. For example, a many-to-many relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers.

Relational database systems usually don't allow you to implement a direct many-to-many relationship between two tables. Consider the example of keeping track of invoices. If there were many invoices with the same invoice number and one of your customers inquired about that invoice number, you wouldn't know which number they were referring to. This is one reason for assigning a unique value to each invoice.

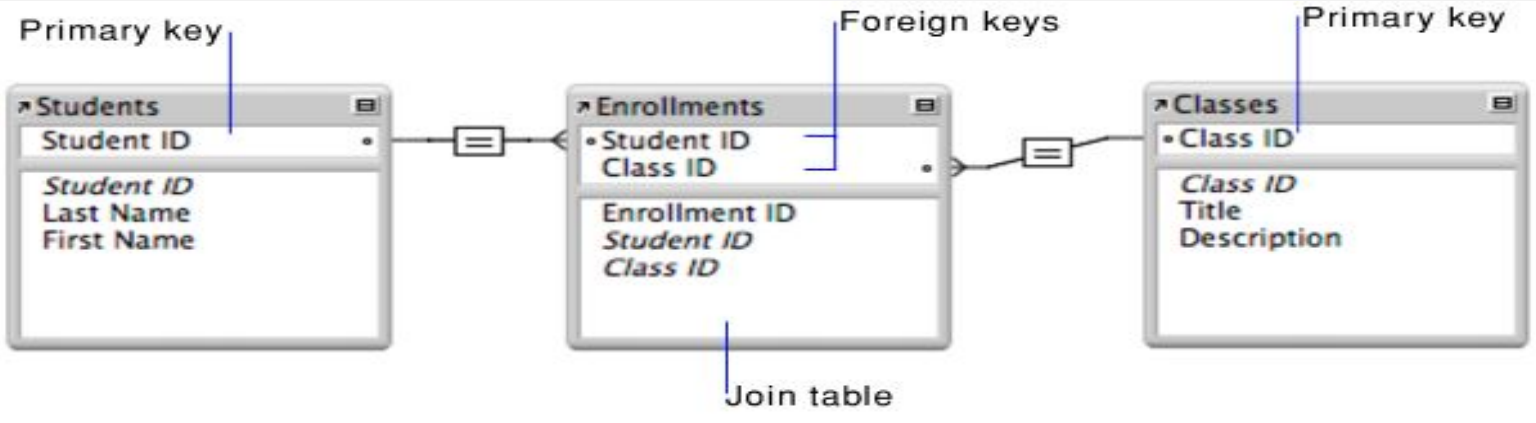
To avoid this problem, you can break the many-to-many relationship into two one-to-many relationships by using a third table, called a *join table*. Each record in a join table includes a match field that contains the value of the primary keys of the two tables it joins. (In the join table, these match fields are foreign keys.) These foreign key fields are populated with data as records in the join table are created from either table it joins.

Many to Many Relationship (Example)

A typical example of a many-to many relationship is one between students and classes. A student can register for many classes, and a class can include many students.

The following example includes a Students table, which contains a record for each student, and a Classes table, which contains a record for each class. A join table, Enrollments, creates two one-to-many relationships—one between each of the two tables.

The primary key Student ID uniquely identifies each student in the Students table. The primary key Class ID uniquely identifies each class in the Classes table. The Enrollments table contains the foreign keys Student ID and Class ID.



Exercise

To set up a join table for a many-to-many relationship:

1. Using the example above, create a table named Enrollments. This will be the join table.
2. In the Enrollments table, create a Student ID field and a Class ID field.

   Join tables typically hold fields that might not make sense to have in any other table. You can add fields to the Enrollments table, such as a Date field to keep track of when someone started a class, and a Cost field to track how much a student paid to take a class.
3. Create a relationship between the two Student ID fields in the tables. Then create a relationship between the two Class ID fields in the tables.

Using this design, if a student registers for three classes, that student will have one record in the Students table and three records in the Enrollments table—one record for each class the student enrolled in.

For help click here

Learning Objectives

By the end of this session, the students have practised

- Types of Relationship
- Implementing One to One Relationship
- Implementing One to Many Relationship
- Implementing Many to Many Relationship

Conclusion & Q/A

See you tomorrow!