Database Concepts

Table

A table is a data grid used to store similar information. It is made up of columns (also known as fields), which represent entity attributes or pieces of data, and rows, which represent individual records. All records in a table share the same fields. In the previous example, a TV set and a T-shirt are records in the table that stores products. The price and description of each product represent columns in this table. Usually, databases have more tables, each corresponding to one of the objects you are trying to represent.

Relation

In a database, tables are usually related in a logical way. A link between two tables is called a relation. A relation always involves two tables and one column from each. Linking tables is very important for avoiding data redundancy and for better organizing your database. Suppose the TV set product mentioned earlier was manufactured by a company called ACME, and you also store a CD player produced by ACME.

If there were not separate tables for products and manufacturers, you would need to store the same manufacturer twice in your database, once for each product. This not only takes up a lot of disk space, but maintaining several copies of the same data can make updates very difficult. What happens if ACME decides to relocate and you have to change its address?
From the following diagram, it's clear that the manufacturer name and address are not duplicated. However, in order to indicate that there is a relation between the two tables, you need two fields: a primary key in the manufacturers table and a foreign key in the products table. These concepts will be defined later on.

<table>
<thead>
<tr>
<th>Product ID</th>
<th>Product Name</th>
<th>Description</th>
<th>Price</th>
<th>Manufacturer</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TV set</td>
<td>This high-resolution plasma TV set will make a tempting choice for all viewers. CD-R/RW compatible. Programmable. Optical digital output. Silver Colour.</td>
<td>299</td>
<td>ACME</td>
<td>15 Sunset BD, Los Angeles</td>
</tr>
<tr>
<td>2</td>
<td>CD player</td>
<td></td>
<td>50</td>
<td>ACME</td>
<td>15 Sunset BD, Los Angeles</td>
</tr>
</tbody>
</table>

Look-up table

A look-up table is a special table containing additional attributes for features stored in an associated feature attribute table. It usually contains two fields: one storing a numeric identifier (the ID) and one storing the label (symbol). An example of a look-up table could be a categories table for products, storing the category ID and the name. To retrieve the complete information regarding the product, a JOIN operation must be performed between the two tables.

Types of relations
In practice, there are three types of relations between tables:

- **one-to-many** – Each manufacturer produces a variety of products, but a product belongs to a single manufacturer.
- **many-to-many** – Each student attends several different classes and each class is attended by several students.
- **one-to-one** – Each shopper has a delivery address.

However, not all these relations are translated into the actual database design. There is no point in having a one-to-one relation between two tables, since it would mean that the tables actually store different attributes of the same object. All similar data should be stored in the same structure, i.e. table. Also, in most situations, a many-to-many relation can be broken into two or more one-to-many relations, through the process of "normalization". Read the detailed presentation of how to normalize the information in the section "Plan your database".

---

**Keys**

A key is a special column in a table. While some keys are used for searching and indexing, other are used for linking related tables or uniquely identifying a record. A database table can survive without a primary key or an index, but this deters the whole purpose of a database: the easy retrieval of information.

Below, you can read about the types of keys used in databases: primary key, foreign key, unique key and index.

---

**Primary Key**

The purpose of a primary key is to uniquely identify each record in a table. In general, a primary key is defined on a single column, but it is not uncommon to have it defined on two columns. There are three basic rules that you should observe when defining primary keys:

1. Every record must have a value in the primary key. It cannot be null.
2. Primary key values must be unique.
3. Primary key values should not be reused. If a record is deleted, its primary key should not be reassigned to another record, as this can create errors and confusion.

---

**Foreign Key**

A foreign key is a field from a table that refers to (or targets) a specific key, usually the primary key, in another table. This is a convenient way of logically linking information from related tables. For instance, the products table contains a foreign key (idman_prd) that references the primary
key field (id_man) in the manufacturers table. This way, each product has an associated manufacturer – its foreign key points to the unique identifier of the manufacturer. Please note that the foreign key is not unique, but the referenced field (the primary key in the referenced table) usually is.

For instance, in the previous example, the CD player and the TV set are both manufactured by ACME company, therefore both records have the same foreign key value.

<table>
<thead>
<tr>
<th>product_prd</th>
<th>manufacturer_man</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>id_prd int (11)</td>
<td>id_man int (11)</td>
</tr>
<tr>
<td>idctg_prd int (11)</td>
<td>id_man varchar (200)</td>
</tr>
<tr>
<td>idman_prd int (11)</td>
<td>name_man varchar</td>
</tr>
<tr>
<td>name_prd varchar (200)</td>
<td>address_man varchar</td>
</tr>
<tr>
<td>price_prd real</td>
<td></td>
</tr>
<tr>
<td>description_prd varchar</td>
<td></td>
</tr>
</tbody>
</table>

**Unique Key**

A unique key is defined as having no two of its values the same. The columns of a unique key cannot contain null values. A table can have multiple unique keys or no unique keys at all. For instance, you might want to define a unique key on the manufacturer name field (name_man) from the previous example. This way, you can guarantee that users of your application will not create two separate records for the ACME company.

**Index**

An index is an alphabetical listing of the terms found on database records and the number of records on which the terms are found. The index is maintained by the database application and is used only by the database application. You never actually see the index in your database, and in fact, most modern database applications hide the physical storage location of the index altogether.

**Query**

A query is a question that is asked of the database server. Database queries can be one of two types:

- **Select queries** - they retrieve data from tables. For instance, getting a list of all products manufactured by the ACME Company that cost more than $100.
- **Action queries** - they manipulate data in some way (e.g. insert records, update or delete existing records). For instance, adding a new product to your catalog or changing the
address of the ACME Company.

The language used to query a database is called query language. The most well-known is the Structured Query Language (SQL).

**Recordset**

A recordset is basically "a set of records". Recordsets are the result of a query, and are used as a source of data for your dynamic web applications. For instance, if you query the sample database for all products manufactured by ACME, you will get a recordset containing two records: the CD player and the TV set.