Lesson: 10 (Advanced Database & Query)

The WHERE clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

Example "where.php":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons
WHERE FirstName='Saahil'");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
?>
```

The ORDER BY Keyword

The ORDER BY keyword is used to sort the data in a record set. The ORDER BY keyword sort the records in ascending order by default.

Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

Example "orderBy.php":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons ORDER BY age");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'];
  echo " " . $row['LastName'];
  echo " " . $row['Age'];
  echo "<br />";
  }

mysql_close($con);
?>
```

Update Data in a Database

The UPDATE statement is used to update existing records in a table.

Syntax:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Example "update.php":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);
```

```
mysql_query("UPDATE Persons SET Age = '88'
WHERE FirstName = 'Shafiul' AND LastName = 'Alam'");

mysql_close($con);
?>
```

Delete Data in a Database

The DELETE FROM statement is used to delete records from a database table.

Syntax:

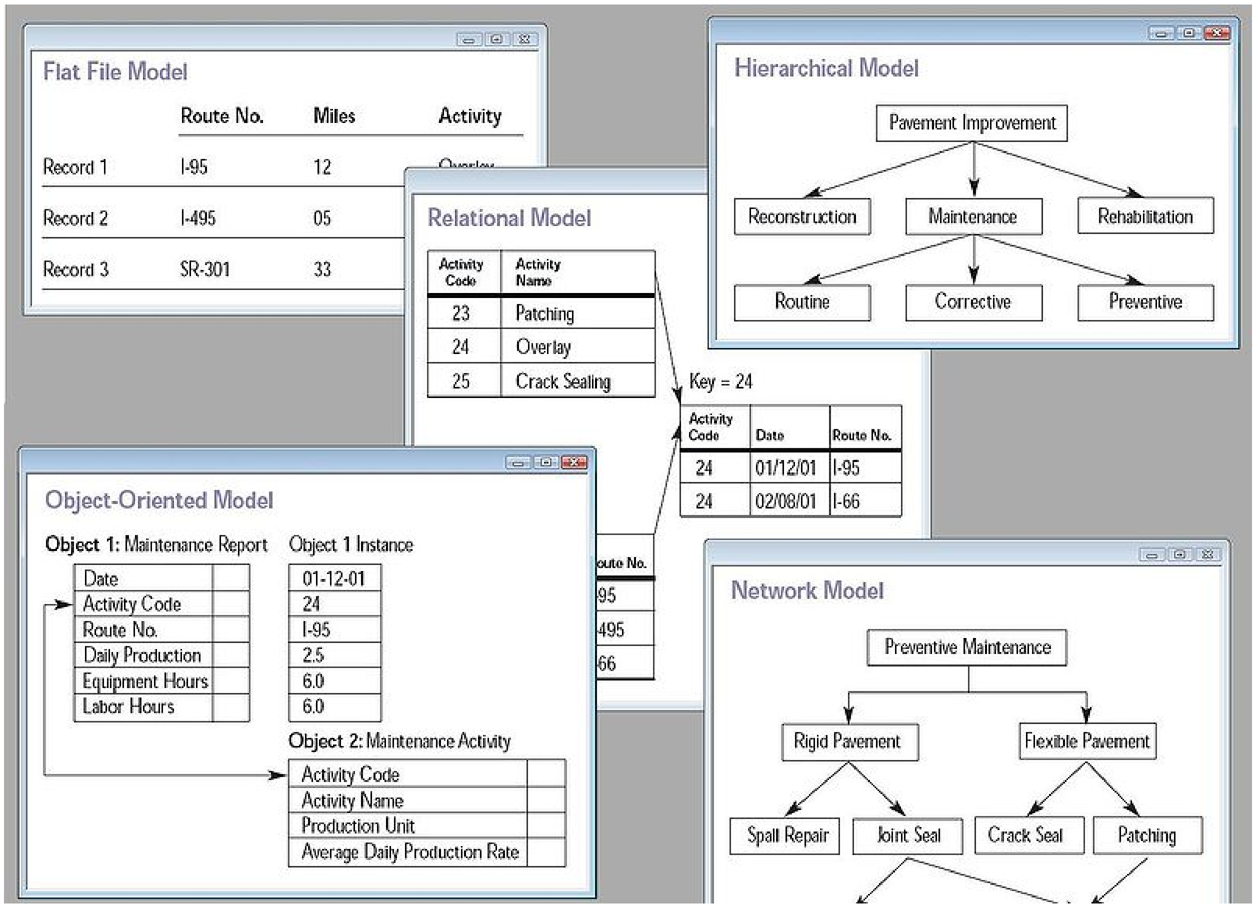
```
DELETE FROM table_name
WHERE some_column = some_value
```

Example "delete.php":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("DELETE FROM Persons WHERE LastName='Alam'");

mysql_close($con);
?>
```

Database Model

A database model or database schema is the structure or format of a database.



Flat model: The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. For example, the columns for name and password that might be used as a part of a system security database.

Hierarchical model: In a hierarchical model, data is organized into a tree-like structure.

Network model: The network model organizes data using two fundamental constructs, called records and sets. Records contain fields. Sets define one-to-many relationships between records: one owner, many members. A record may be an owner in any number of sets, and a member in any number of sets.

Relational model: The basic data structure of the relational model is the table, where information about a particular entity (say, an employee) is represented in columns and rows. Thus, the "relation" in "relational database" refers to the various tables in the database.

All relations in a relational database have to adhere to some basic rules to qualify as relations. A key that can be used to uniquely identify a row in a table is called a primary key. Keys are commonly used to join or combine data from two or more tables.

Object database models: An object-relational database (ORD), or object-relational database management system (ORDBMS), is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, it supports extension of the data model with custom data-types and methods.

Database Normalization

Normalization: Normalization is the process of efficiently organizing data in a database.

There are two goals of the normalization process:

1. Eliminating redundant data (for example, storing the same data in more than one table)
2. Ensuring data dependencies make sense (only storing related data in a table).

The Normal Forms: The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and 3NF. 4NF and 5NF are very rarely seen.

First Normal Form (1NF): First normal form (1NF) sets the very basic rules for an organized database:

- Eliminate duplicative columns from the same table.
- Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).

Second Normal Form (2NF): Second normal form (2NF) further addresses the concept of removing duplicative data:

- Meet all the requirements of the first normal form.
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Create relationships between these new tables and their predecessors through the use of foreign keys.

Third Normal Form (3NF): Third normal form (3NF) goes one large step further:

- Meet all the requirements of the second normal form.
- Remove columns that are not dependent upon the primary key.

Fourth Normal Form (4NF): Finally, fourth normal form (4NF) has one additional requirement:

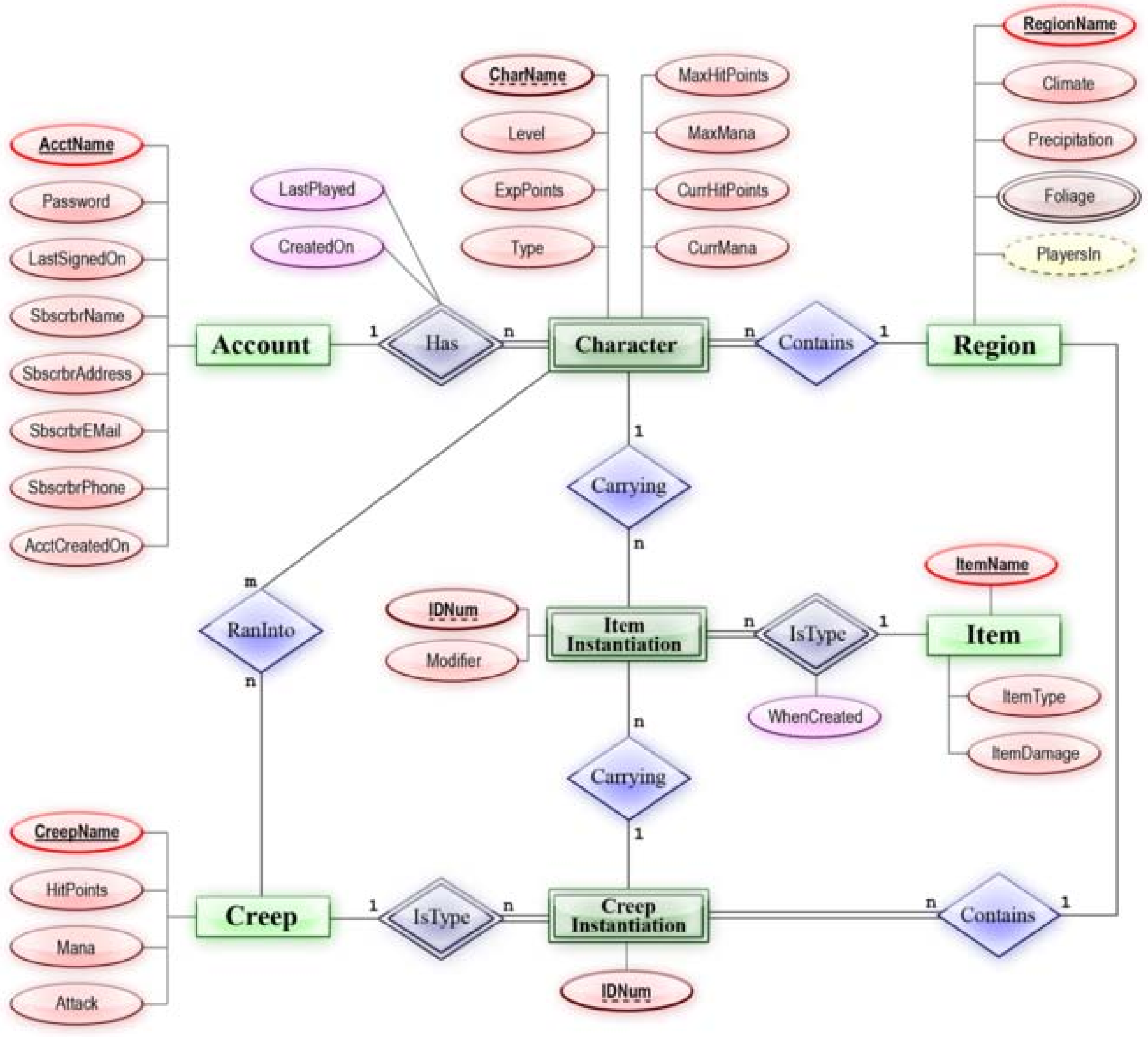
- Meet all the requirements of the third normal form.
- A relation is in 4NF if it has no multi-valued dependencies.

Database relationship

The entity-relationship model is a graphical representation of entities and their relationships to each other. An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database.

There are three types of relationships between entities:

- ***one-to-one:*** one instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).
- ***one-to-many:*** one instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.
- ***many-to-many:*** one instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.


RegionName
CharName
MaxHitPoints
Climate
AcctName
Level
MaxMana
Precipitation
LastPlayed
Password
ExpPoints
CurrHitPoints
Foliage
CreatedOn
LastSignedOn
Type
CurrMana
PlayersIn
SbscrbrName
Account
1
Has
n
Character
n
Contains
1
Region
SbscrbrAddress
SbscrbrEMail
1
Carrying
SbscrbrPhone
AcctCreatedOn
n
m
ItemName
IDNum
RanInto
Item Instantiation
n
IsType
1
Item
Modifier
n
ItemType
WhenCreated
n
ItemDamage
Carrying
1
CreepName
HitPoints
Creep
1
IsType
n
Creep Instantiation
n
Contains
1
Mana
Attack
IDNum