Information Management Resource Kit

Module on Management of Electronic Documents

UNIT 5. DATABASE MANAGEMENT SYSTEMS

LESSON 5. RELATIONAL DATABASES AND SQL BASICS

NOTE

Please note that this PDF version does not have the interactive features offered through the IMARK courseware such as exercises with feedback, pop-ups, animations etc.

We recommend that you take the lesson using the interactive courseware environment, and use the PDF version for printing the lesson and to use as a reference after you have completed the course.

Objectives

At the end of this lesson, you will be able to:

- understand the principles on which relational databases and SQL are based.
- apply the **Extended Entity Relationship** (**EER**) model to a simple set of data.
- understand the function of the main SQL statements.

Documents which allow you to create and modify relational databases using SQL, are available for download and print at the end of the lesson.

Introduction

The most popular kind of database is the relational database, invented in the 70s, where the data is stored as relations.

The first step in building a relational database is deciding how to organize data, that is designing the database.

Then, the database can be created and manipulated using the Structured Query Language (SQL), which allows interaction with relational databases.

Principles

Relations can be viewed as **two-dimensional tables** where all the data is stored.

Let's consider for example this **Person table**.

Table: **PERSON**

Attribute

	ID	FirstName	Lastname	Work Phone	Home Phone
Tuple	1001	Laura	Williams	44.34.34.44	45.34.34.45
	1002	Paul	Smith	63.56.55.64	63.35.13.44
	1003	David	Fisher	21.86.77.24	21.79.33.99

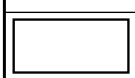
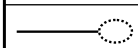
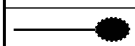
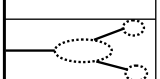
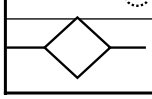
Primary Key

A row (or tuple) in a table is identified by a primary key. The other information in the row is referred to as attributes.

Specifically, the ID column in this table is the primary key.

EER modelling

The design phase must identify the relationships between entities that properly describe the collection.
The **Extended Entity Relationship** (**EER**) model is one of the most widely used frameworks for data modelling. It is based on three main categories:

EER Notation	Name	Description	It may be
	**ENTITY**	A class of real-world objects. It is normally a noun. An Entity would have one or many attributes.	**physical object** (e.g. person) **event** (e.g. appointment) **concept** (e.g. order).
	**ATTRIBUTE**	A property of an entity	**descriptor** (describe properties of the entity, e.g. lastname of the person)
			**identifier** (uniquely distinguish entity instances - primary key in relational context)
			**composite** (a group of attributes used as a single attribute)
	**RELATIONSHIP**	Describes the relationship between entities. Typically expressed in verbs (e.g. has).	There are no well defined standards of the EER notation.

EER modelling

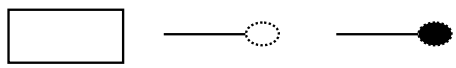
Table: **MOVIE**

Code	Title	Director	Year
100	Sophie's choice	Pakula	1982
101	The great dictator	Chaplin	1940
102	Dracula	Coppola	1992

Let's consider this example.

You have created a database for your video-rental store.

Can you identify the following parts of the table?



☐ ☐ ☐

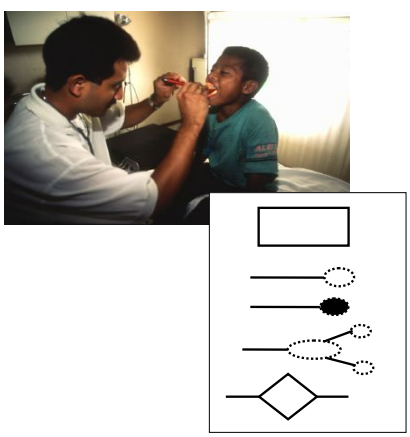
102 ☐ ☐ ☐

Movie ☐ ☐ ☐

Chaplin ☐ ☐ ☐

Choose your answers

EER modelling



Now, let's consider another example.

We want to develop a very simple appointment database for a doctor's office.

All we want to record are:

- patient details,
- doctor details, and
- appointments between the doctor and patients.

Let's look at the development step by step.

EER modelling

The first step in EER modelling is:

Identify entity sets and the relationships between them.

We can immediately identify two entities: **Patient** and **Doctor**.

A patient would have appointments with one or more doctors.

A doctor would have appointments with zero or more patients.

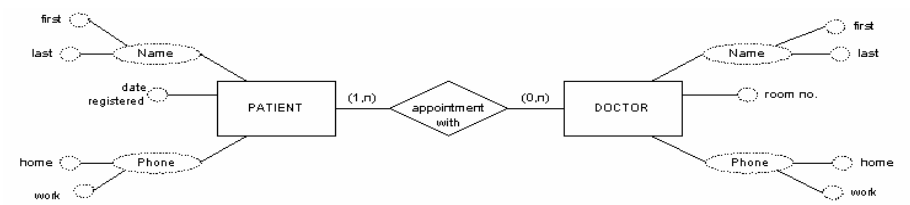
EER modelling

The second step is:

Identify attributes of each entity set.

We will now add all the patient and doctor details that we need.

In this example we will omit some obvious attributes (such as address) in order to simplify the model.



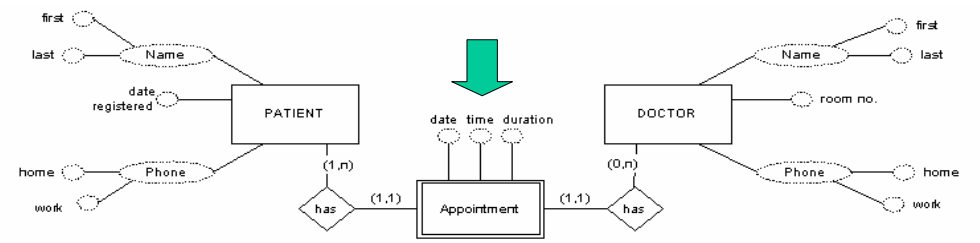
EER modelling

The third step is:

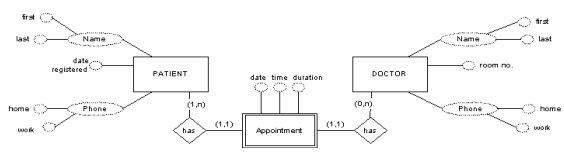
Identify any attributes associated with relationships. If any exist, create relationship sets.

The appointment would have a date, time and duration.

Appointment is a **weak entity,** that is its unique identifier is derived from some 'parent' entity (e.g. patient) and it will cease to exist if its parent is removed. In this case, if the patient is removed so is the patient's appointment.



EER modelling



The fourth step is:

Identify overlapping attributes.

These are the attributes that are repeated and therefore cause redundancy.

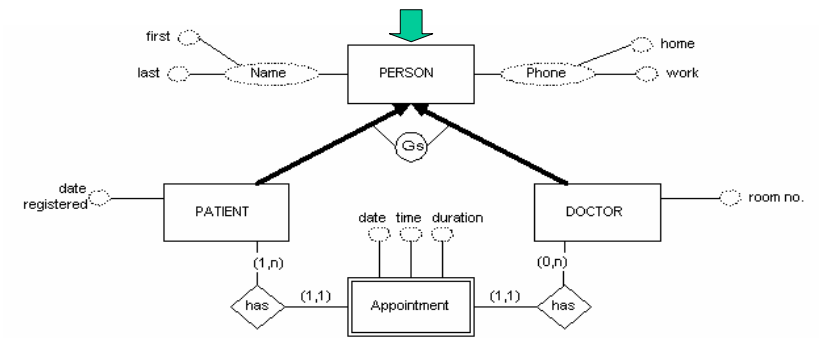
In your opinion, which of the following attributes are overlapping?

- ☐ Name
- ☐ Date
- ☐ Date registered
- ☐ Phone

Click on your answers

EER modelling

We can see that patient and doctor have a lot of similar attributes (name, phone, etc...). We could also have the case that most doctors also are patients from time to time. For these reasons, we will add an overlapping generalization hierarchy to our model by adding a **Person entity**.

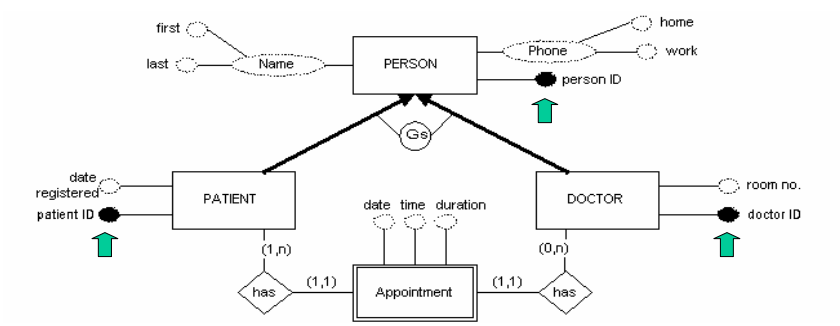


EER modelling

The fifth step is:

Select identifiers for each entity set.

None of the existing attributes for Person, Patient and Doctor can ensure that an instance would be unique. We will therefore need to include **id attributes**. Appointment is a relationship set, so it is not strictly necessary to define a key at this stage; although it will probably have a composite key made up of the patient id, date and time.



Normalization

Normalization is an alternative formal technique for **defining relations** that contain minimum redundancy.
It is a bottom-up design technique, which makes it difficult to use in large designs.
For this reason it has been largely superseded by the top-down approaches (e.g. EER), but it can be used as another method of checking the properties of a design arrived at through EER modelling. This can be done as follows:

1. Identify the key (simple or composite) of each relation.
2. Identify any foreign keys in a relation.
3. Check that the other attributes in the relation are determined by the relation's key.
4. Create a new relation comprising any attributes not determined by the relation's key.
5. Repeat steps 1-4 for every relation.

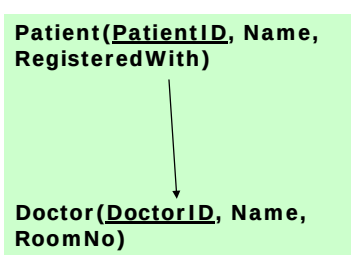
More information about normalization

Mapping a logical model to a relational schema

Based on our EER model, we have to design the **relational schema**.

In other words, we have to express the relationships between relations. To do this we use foreign keys.

A **foreign key** is an attribute in one relation that refers to the primary key in a related relation.



For example, consider the relation on the left. PatientID is the primary key and **RegisteredWith** is the foreign key which refers to the primary key of the Doctor relation (i.e. DoctorID).

The value in the foreign key can refer to only one tuple in the DOCTOR relation, although the DoctorID can be referenced by many foreign key values. That is, a patient instance can only be registered with one doctor, but many patients can be registered with the same doctor.

Mapping a logical model to a relational schema

How to express relationships in our schema? We will use **three separate relations** for Person, Patient and Doctor.

PERSON(PersonID, FirstName, LastName, HomePhone, WorkPhone)

PATIENT(PatientID, DateRegistered)

DOCTOR(DoctorID, RoomNo)

The PersonID, PatientID and DoctorID attributes act both as primary keys within their own relation, and as foreign keys to the matching rows in the other two relations.

APPOINTMENT(PatientID, DoctorID, Date, Time, Duration)

The relationship between Appointment and both Patient and Doctor is implemented using foreign keys (PatientID and DoctorID).

As primary key for Appointment, we choose the composite key "PatientID, Date, Time" allow for an appointment between one or many patients and one doctor.

Note that we choose to use this method because of the the complexity of a person being both a doctor and a patient. Other methods are available:

Comparison between the possible methods

From relations to tables

Based on our design, we can now create, manipulate and control our database.

Specifically, we have to:

- build the database structure (e.g. create tables and relations between them);
- add, delete, and modify data in the tables; and
- control what users may or may not do with the objects in the database.

To do this we need to use the Structured Query Language (**SQL**).

From relations to tables

SQL is the language used to interact with a relational database.

SQL is more than simply a query language; it is a database sub-language and is becoming the **standard interface** to relational and non-relational database management systems (DBMS).
The DBMS stores the data and retrieves or updates it **in response to SQL statements**.
SQL was originally designed as a query language based on the relational algebra. It started as a language called **Sequel** (Structured English QUEry Language) which was developed by IBM in the mid-1970s as the data manipulation language (DML) of one of their early attempts at a relational database.

This language allowed users to access and manipulate data stored in the database. During the early 1980s, IBM renamed the language SQL and based two of their relational database packages, SQL/DS and DB2, on this language.

SQL was adopted as an industry standard in 1986 (SQL-86). Since then there has been three more standards, SQL-89, SQL2 (or SQL-92) and SQL3 (or SQL-99).

All commercial relational database vendors now support some **variant of the SQL standard**. It is also the basis of most database interoperability products and proposals (e.g. ODBC).
You can find information about SQL validators at: http://developer.mimer.com/validator

Parts of SQL

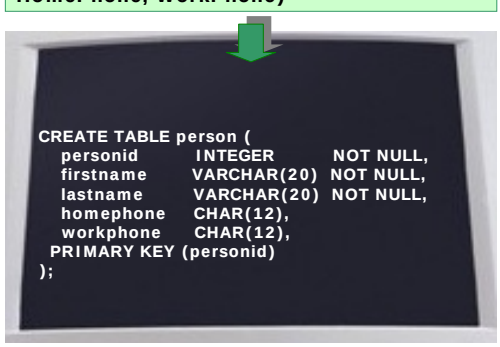
The Person relation can be translated into a SQL table using the **CREATE TABLE** statement.

As you can see in the example, the person identifier is an integer.

For first and last name, VARCHAR means that data values are expected to vary considerably in size, up to 20 (data length in bytes).
Home and work phone data values (CHAR) are expected to be consistently close to the same size (12), and they may be not present in the table.

The PersonId attribute is finally specified as primary key.

PERSON(PersonID, FirstName, LastName, HomePhone, WorkPhone)



```
CREATE TABLE person (
   personid      INTEGER       NOT NULL,
   firstname     VARCHAR(20)   NOT NULL,
   lastname      VARCHAR(20)   NOT NULL,
   homephone     CHAR(12),
   workphone     CHAR(12),
 PRIMARY KEY (personid)
);
```

Parts of SQL

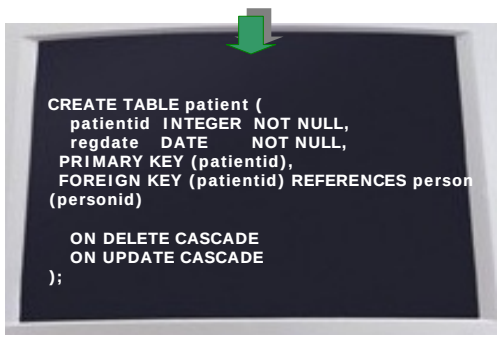
Now let's consider the **Patient relation**.

Note that the PatientID attribute acts both as primary key and as foreign key in relation with the Person entity.

The " references" clause is used to specify referential integrity constraints and, optionally, the actions to be taken if the related row is deleted (ON DELETE) or the value of its primary key is updated (ON UPDATE).

CASCADE means that:
• on update, a change to the primary key value in the related row is reflected in the foreign key;
• on delete, if the related row is deleted then so is the row containing the foreign key.

PATIENT(PatientID, DateRegistered)

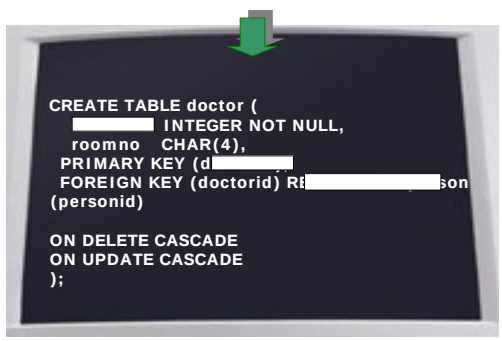


Parts of SQL

Our third relation is the **Doctor relation**.
Could you complete the creation of the Doctor Table?

Complete the SQL by typing the correct characters in the empty fields. Then click on the "check answer" button.

DOCTOR(DoctorID, RoomNo)



Check answer

Parts of SQL

Finally, let's have a look at the **appointment relation**.
Note that appointment has three primary keys: PatientId, Date and Time.

PatientId/PersonId
If the related Person row is deleted then so is the row containing PatientId, therefore the appointment is deleted.
If the PersonId value is updated nothing happens to the PatientId in this table (this means that the foreign key is here just a reference).

DoctorId/PersonId
If the related Person row is deleted then the DoctorId value is set to null, but the appointment is not deleted.
If the PersonId value is updated then the change is reflected in the DoctorId.

APPOINTMENT(PatientID, DoctorID, Date, Time, Duration)

```
CREATE TABLE appointment (
   patientid  INTEGER     NOT NULL,
   doctorid   INTEGER,
   appdate    DATE        NOT NULL,
   apptime    TIME        NOT NULL,
   duration   INTEGER     DEFAULT 15,
 PRIMARY KEY (patientid, appdate, apptime),
 FOREIGN KEY (patientid) REFERENCES person
(personid)
   ON DELETE CASCADE
   ON UPDATE NO ACTION,
 FOREIGN KEY (doctorid) REFERENCES person
(personid)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);
```

Parts of SQL

The **Create table** statement we have seen belong to the **Data Definition Language (DDL)**, a part of SQL's statements.

DDL comprises all the statements used to **define the data structures** such as creating, altering and deleting schemas and tables, etc.
The most important statements are **CREATE**, **DROP** and **ALTER**.

The **DROP** statement is used to remove a table. CASCADE drops all associated referential integrity constraints and views that depend on this base table, whereas RESTRICT raises an exception if any of these exist.

```
mysql> DROP TABLE person RESTRICT;
mysql> DROP TABLE person CASCADE;
```

The **ALTER** statement is used to add or delete a column in a table.
As above, CASCADE and RESTRICT determine the drop behaviour when constraints or views depend on the affected column.

```
mysql> ALTER TABLE person ADD COLUMN
address;
mysql> ALTER TABLE person DROP
COLUMN address RESTRICT;
mysql> ALTER TABLE person DROP
COLUMN address CASCADE;
```

Parts of SQL

Having created a structure for the tables in your database, you may want to add, delete, retrieve or **modify data in the tables**. This is done by using the SQL data manipulation statements.

This part of SQL is named the **Data Manipulation Language (DML)**.

Following are the DML essential statements:

SELECT	Used to retrieve information from tables.
INSERT	Used to add a new row or a set of rows.
UPDATE	Used to modify an existing row.
DELETE	Used to remove rows.

Parts of SQL

There is another part of SQL that allows the control of **what users may or may not do** with the objects in the database.

This set of statements is called the **Data Control Language**.

The primary mechanism for enforcing control issues in SQL is through the concept of a **view**.
Views are virtual tables which act as 'windows' on the database of real tables.
Access can be restricted on tables and views to particular users via the **GRANT** and **REVOKE** facilities of SQL.

Using SQL- tools

The following document describes the procedures to create a database, and how to manipulate tables and data using SQL:

You can also view only the sections you are interested in:

Creating a database

- Creating a new database

Creating and Listing Tables and Fields

- Data types on various database platforms
- Creating SQL tables
- Table Manipulations

Inserting data

- Adding new rows
- Modifying existing rows
- Removing rows

Selecting data

- The SELECT statement
- Joining tables
- UNION, EXCEPT and INTERSECT operators

Note: unless otherwise stated, all examples are in compliance with the SQL2 standard.

These documents describe procedures using MySQL Windows version 3.23. Click on the link to learn more about MySQL.

LINK TO MySQL

Using SQL- tools

MySQL Database Server

LINK TO MySQL

The MySQL database server is the world's most popular open source database. The MySQL Database Server can be obtained from the MySQL download portal at: http://www.mysql.com/downloads.

MySQL is available at zero price under the GNU General Public License (GPL), and is also sold under a commercial license to those who do not wish to be bound by the terms of the GPL. To learn more about the MySQL(TM) database server and other MySQL products visit: http://www.mysql.com/products/mysql/index.html. To learn more about the MySQL Licensing Policy please visit: http://www.mysql.com/products/licensing.html.

An HTML version of the MySQL Reference Manual can be found and searched at: http://www.mysql.com/doc/en/index.html. It is also available many other formats, including PDF and Windows HLP versions at: http://www.mysql.com/documentation.

Because all MySQL products are open source, free support provided by the MySQL community is available on the MySQL public mailing lists. You can subscribe to the MySQL mailinglist(s) using the MySQL Mailing Lists Subscription Tool available at: http://www.mysql.com/documentation/lists.php

MySQL AB also publishes a monthly email newsletter with articles about new products, new features, training, security issues, known bugs, and events of interest to the MySQL community. You can subscribe to the newsletter by using the online subscription form or by sending email to newsletter@mysql.com with 'subscribe' in the subject line.

Summary

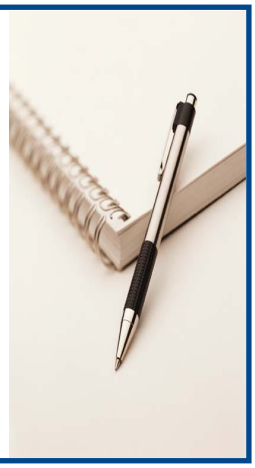
• **SQL** (**Structured Query Language**) is the language used to interact with a relational database.

• Relational databases, and the SQL standard, are based on a relational model, which derives from the concept of **relation**.

• Database design is fundamentally a task in data modelling; the **Extended Entity Relationship** (**EER**) model is one of the most widely used frameworks for data modelling.

• We can use the **normalization** technique to check the properties of our design.

• SQL statements are grouped into three main parts: Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL).



Exercises

The following five exercises will allow you to apply the principles of data modelling for a relational database.

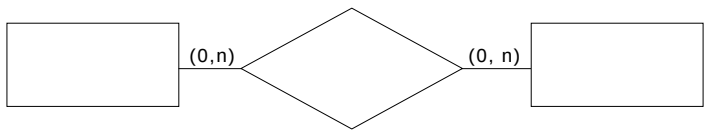
Good luck!

Exercise 1

Let's consider again the example of the video rental store.
In this case, you would create a database to register the actors and directors of the movies available.

All you want to record are:
- actor details,
- director details, and
- collaboration between actors and directors.

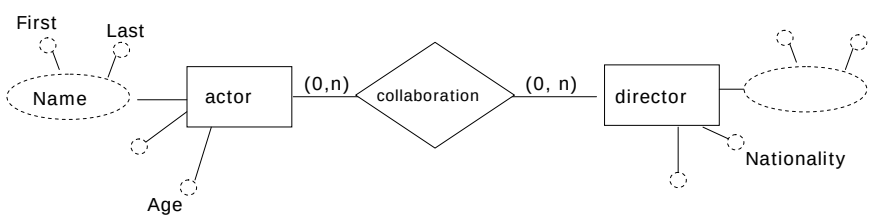
The first step in EER modelling is to identify entity sets and the relationships between them.
Can you do it?



Type the text in the relevant boxes. Then, click on Check answer.

Exercise 2

Second, you have to **identify the attributes of each entity set**.
You want to record the details listed here below. Can you add them to the diagram?

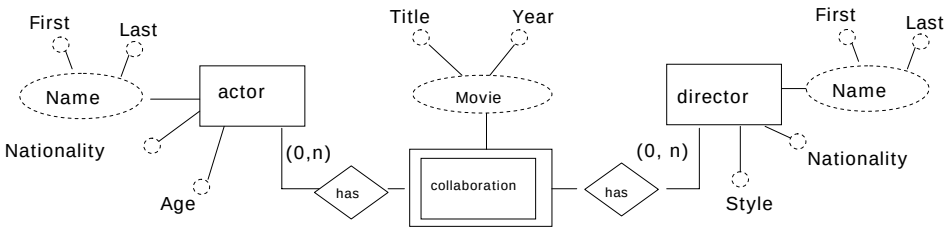


Nationality Style Name First

Select each option and drag it in the relevant fields.

Exercise 3

The composite attribute “movie” is associated to the relationship “collaboration”.

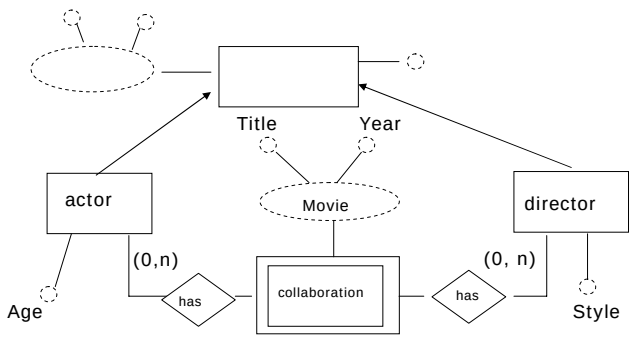


Could you identify the overlapping attributes?

Type your text in the relevant box.

Exercise 4

Which general entity would you add to the model?



Type your text in the relevant box.

Exercise 5

To create and manipulate your database it is important to know the main SQL statements.
Could you group them into the SQL categories?

a		1
Data Definition Language (DDL)		SELECT, INSERT, UPDATE, DELETE
Data Manipulation Language (DML)		GRANT, REVOKE
Data Control Language (DCL)		CREATE, DROP, ALTER

Standard message (drag and drop)

If you want to know more...

C.J. Date. An Introduction to Database Systems Addison Wesley; ISBN: 0201787229. The definitive book on database systems.

C.J. Date, Hugh Darwen. Foundation for Future Database Systems: The Third Manifesto Addison Wesley; ISBN: 0201709287.

Jim Melton, Alan Simon. SQL 1999: Understanding Relational Language Components. Morgan Kaufmann; ISBN: 1558604561

Joe Celko (Foreword), Michael J. Hernandez, John L. Viescas. SQL Queries for Mere Mortals: A Hands-on Guide to Data Manipulation in SQL. Addison Wesley; ISBN: 0201433362

SQL.ORG. Guide to online SQL resources. www.sql.org/online_resources.html

Intelligent Enterprise: A magazine dedicated to strategic business applications that turn information into intelligence. (www.intelligententerprise.com)

SQL Validation and resources from Mimer SQL.
http://developer.mimer.com/validator/

The MySQL Database Server and MySQL Reference Manual can be obtained from the MySQL download portal at: http://www.mysql.com along with other MySQL products.