Let's design a simple database in order to address the following question:

I have a set of co-regulated genes. I would like to know if the enzymes coded by those genes are involved in the same metabolic pathway.

We can first create a "Gene" table that contain expression data.

Because different genes might have the same name, we will associate to each gene a unique identifier (the ID\_gene). This ID constitutes the primary key of the table.
In order to retrieve the reaction, we would like to know the enzyme. We therefore need information about proteins.

Adding a column "protein" in the "gene" table is not a good solution because several proteins can be encoded by the same gene (due to alternative splicing, post-translationnal modifications,...). This would lead to multivalued attributes, which should be avoided in relational databases. In addition, many genes do not code for proteins and it is not useful to have a table containing many NULL values.

Solution: create a new table:

Similarly, several proteins can have the same name and it is necessary to defined a unique ID as a primary key. All information pertaining to a given protein (sequence,...) as well as links to other databases (like PDB) can then be given as attributes.
Finally, we need information about enzyme-catalyzed reactions.

Adding a column "reaction" in the "protein" table is not a good solution because several reaction can be catalysed by the same protein. In addition, many proteins are not enzymes and it is not useful to have a table containing many NULL values.

Solution: create a new table:

Here again, each reaction is characterized by a unique ID and some attributes (such as the reaction rates, etc).
Entity tables

An entity defines a type of object.

In our simple database, we have 3 entities: "gene", "protein", and "reaction". Each data of our database must belong to one of those categories.

Each entity has attributes.

The entity "protein" has the attributes prot_names, sequence,...
Relational databases: example

Our entity tables must be related

Gene codes Protein catalyses Reaction
Such a scheme is the *conceptual scheme* of our database.
Relational databases: example

The relations (represented by diamonds in our conceptual scheme) are themselves tables.
Example of a real database

Conceptual scheme of the aMAZE database

Having all our tables defined and filled, we can query them.

A query is a question like:

What is the expression of gene PutA in the condition of lactose-rich medium?

What are all the genes whose expression is above "3" in condition of lactose-rich medium?

What is the protein sequence coded by gene Gal1?

What are the reactions catalysed by all the genes whose expression is above "3" in condition of lactose-rich medium?

Those questions can be asked to the database using a specific language, called SQL (structured query language).
The SQL command to extract data is **SELECT**.

**Examples:**

**What are all the genes present in my Gene table?**

**SELECT * FROM** Gene  
returns the full Gene table

**SELECT gene_name, expr_galactose FROM** Gene  
returns a 2-column table with all the gene names and their corresponding expression in galactose-rich medium

**What is the sequence of protein PutA?**

**SELECT seq FROM** Protein **WHERE** prot_name = 'putA'  
returns the sequence of the protein "PutA".
The SQL command to extract data is `SELECT`.

Examples:

What is the expression of gene PutA in the condition of lactose-rich medium?

```
SELECT expr_lactose FROM Gene WHERE gene_name = 'putA'
```
returns the expression level of gene "PutA" in the condition of lactose rich medium.

What are all the genes whose expression is above "3" in condition of lactose-rich medium?

```
SELECT gene_name FROM Gene WHERE expr_lactose > 3
```
returns a 2-column table with all the genes having an expression level greater than 3 (one column gives the gene name and the second column gives the actual expression level).
The SQL SELECT command can also be used to query data by combining data from different tables. Examples:

**What is the protein associated to each gene?**

```
SELECT Gene.gene_name, Protein.protein_name
FROM Gene, Protein, Gene-Protein
WHERE Gene.ID_gene = Gene-Protein.ID_gene AND
Protein.ID_prot = Gene-Protein.ID_prot AND
Gene-Protein.ID_gene = Gene-Protein.ID_prot
returns a 2-column table with the gene name and the associated protein (note that the same gene or protein can appear on several lines).
```

**What is the protein sequence coded by gene Gal1?**

```
SELECT Protein.seq
FROM Gene, Protein, Gene-Protein
WHERE Gene.ID_gene = Gene-Protein.ID_gene AND
Protein.ID_prot = Gene-Protein.ID_prot AND
Gene-Protein.ID_gene = Gene-Protein.ID_prot AND Gene.gene_name = "Gal1"
returns the sequence of the protein coded by "Gal1"
```
The SQL SELECT command can also be used to query data by combining data from different tables.

Examples:

What are the reactions catalyzed by genes which code for enzymes and which are overexpressed in galactose-rich medium.

```sql
SELECT Reaction.reaction FROM Gene, Protein, Reaction, Gene-Protein, Protein-Reaction
WHERE Gene.ID_gene = GeneProtein.ID_gene
AND Protein.ID_protein = GeneProtein.ID_protein
AND Protein-Reaction.ID_protein = Protein-Reaction.ID_reac
AND Protein-Reaction.ID_reac AND Protein-Reaction.ID_reac
AND Gene.exp_galactose > 3.
```

returns the list of reactions catalysed by enzyme whose gene expression level is greater than 3.
Some other useful SQL commands:

**CREATE TABLE:**

allows to create a new table in the database. The name of the columns and their type should be specified.

**Example:** `CREATE TABLE MRNA (mrna_name varchar(10) NOT NULL, mrna_sequence varchar(1000));`

**INSERT**

allows to insert new data in a table.

**Example:** `INSERT INTO Gene (ID_gene, gene_name, expr_galactose, expr_lactose) values ('G239', 'gal80', '3.9', '1.2');`

**UPDATE**

allows to update data in a table.

**Example:** `UPDATE Reaction SET v_max = 15 WHERE ID_reaction=R239;`
Some other useful SQL commands:

**DELETE**

allows to delete data from a table.

Example: `DELETE FROM Gene WHERE gene_name = LEXB`

**DROP**

allows to delete a whole table from the database.

Example: `DROP TABLE MRNA`

Other commands can be used to perform more refined query, to sort of group the results of a select, to add or remove columns in a table, to display the description of the tables, etc.

Some examples of SQL query are described here:

http://homepages.ulb.ac.be/~dgonze/INFO/sql.html