step 1: create the table header (table schema) by using the DDL

```sql
create table Employee
(EmplId integer,
 Name varchar(25) not null,
 Birthdate date,
 Salary numeric(8,2),
 primary key (EmplId));
```

step 2: create the table body (table instance) by using the DML

```sql
insert into Employee values(567, 'Meyer', 25-MAR-1975, 23000);
insert into Employee values(123, 'Smith', 17-AUG-1959, 41000);
insert into Employee values(456, 'Kirby', 03-MAY-1966, 35000);
```

Error produced

- Kirby’s salary is $37000 and not $35000
- correction by update command

```sql
update Employee set Salary = 37000 where EmplId = 456
```
Some simple queries

- “Show the employee table.”
  
  ```sql
  select * from Employee
  ```

<table>
<thead>
<tr>
<th>Employee</th>
<th>EmpId</th>
<th>Name</th>
<th>Birthdate</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>567</td>
<td>Meyer</td>
<td>25-MAR-1975</td>
<td>23000</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>Smith</td>
<td>17-AUG-1959</td>
<td>41000</td>
</tr>
<tr>
<td></td>
<td>456</td>
<td>Kirby</td>
<td>03-MAY-1966</td>
<td>37000</td>
</tr>
</tbody>
</table>

- “Show the ids and salaries of all employees whose salary exceeds $27000.”
  
  ```sql
  select EmplId, Salary from Employee where Salary > 27000
  ```

<table>
<thead>
<tr>
<th>EmplId</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>41000</td>
</tr>
<tr>
<td>456</td>
<td>37000</td>
</tr>
</tbody>
</table>
Show only the employee ids and their salaries.

```
select EmplId, Salary from Employee
```

<table>
<thead>
<tr>
<th>EmplId</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>567</td>
<td>23000</td>
</tr>
<tr>
<td>123</td>
<td>41000</td>
</tr>
<tr>
<td>456</td>
<td>37000</td>
</tr>
</tbody>
</table>

How many employees are in the company.

```
select count (*) as total from Employee
```

<table>
<thead>
<tr>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
3. Conceptual Database Design

Introduction

- most known conceptual data model on a high abstraction level, easy to understand, independent of aspects of data organization and data management
- E-R model (besides **UML**) has great importance in practice
- two-phase procedure for DB design
  - phase 1: requirements analysis and design of an E-R model
  - phase 2: transformation of the E-R model into a concrete logical model
- goal: modeling of an interesting part of the “real world” by **abstraction** so that questions about it can be answered with the aid of the model

**E-R model describes the “real world” by**

- **entities** (objects)
- **attributes** (properties)
- **relationships** between entities
entities are distinguishable, independent, self-contained, physically or intellectually existing concepts of the mini-world to be modeled.

similar entities are collected in an entity set, e.g., the set of all books, the set of all cars.

an entity is described by a set of pertaining properties (attributes), e.g., each book has an ISBN number, an author, a publisher, ...

The values of an attribute are from domains like integer, real, string, ... e.g., the name of an author is of type string.

a minimal set of attributes whose values uniquely characterize the associated entity among all entities of its type is called key, e.g., ISBN number identifies a book, an article number an article.