2. Database Management: Handling Tables

Goals:

- rough view on databases as a collection of tables
- enable you to start with Oracle as soon as possible
  - creation of tables
  - insertion and modification of records (so-called tuples) in a table
  - show a few simple queries on a single table
- postpone the ‘theoretical’ discussion to a later time

Concept of ‘table’ used in

- books, papers, newspapers, etc.
  - purpose: summarize information, give an overview
- spreadsheet programs
  - purpose: give an overview, make numerical computations (e.g., aggregations)
- databases
  - purpose: allow to retrieve (parts of the) data, make computations, derive new information (e.g., selections, combining information)
What is a table?

Example

<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>03/25/1975</td>
<td>23000</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>Smith</td>
<td>08/17/1959</td>
<td>41000</td>
</tr>
<tr>
<td></td>
<td>456</td>
<td>Kirby</td>
<td>05/03/1966</td>
<td>37000</td>
</tr>
</tbody>
</table>

How can we create this table (in Oracle)?

- use the DDL (data definition language to create the table)
- remember: the DDL is used to define the structure (i.e., the static part) of a database as part of the database’s meta data information
step 1: create the table header (table schema) by using the DDL

```
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

```
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

```
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>EmpId</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
Entity

- **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.
Relationship

- a **relationship** describes a connection between several entities, e.g., student Smith attends lecture COP 4720, teaching assistant Benson works for professor Meyer

- a homogeneous set of relationships is collected in a **relationship set**, e.g., relationship sets `attends_lecture` or `works_for`

- formal: relationship set $R$ between the entity sets $E_1, E_2, ..., E_n$ as a relation, i.e.,
  
  $$R \subseteq E_1 \times E_2 \times ... \times E_n,$$

  $n$ degree of relationship set $R$

  - `attends_lecture` $\subseteq$ students $\times$ lectures
  - `works_for` $\subseteq$ TAs $\times$ professors

- attributes may characterize relationships, e.g.
  - frequency as an attribute for `attends_lecture`

- an entity set can occur more than once in a relationship set

- if there is only one entity set $E$ participating in a binary relationship $R(E, E)$, each of these entity sets can be assigned **roles**
  - e.g., `is_precondition_of` $\subseteq$ lectures $\times$ lectures
  - first lecture / second lecture has the role of a predecessor / successor
Constraints of binary relationship sets

- **1:1-relationship** (one-to-one relationship)
  
  if for a binary relationship set \( R(E_1, E_2) \) each entity in \( E_1 \) is associated with *at most* one entity in \( E_2 \), and vice versa

- **1:m-relationship** (one-to-many relationship)
  
  if for a binary relationship set \( R(E_1, E_2) \) each entity in \( E_1 \) is associated with any number (zero or more) of entities in \( E_2 \), and each entity in \( E_2 \) is associated with *at most* one entity in \( E_1 \)

- **m:1-relationship** (many-to-one relationship)
  
  analogous to the 1:m-relationship

- **m:n-relationship** (many-to-many relationship)
  
  if for a binary relationship set \( R(E_1, E_2) \) each entity in \( E_1 \) is associated with any number (zero or more) of entities in \( E_2 \), and vice versa

- constraints considered as *partial functions*, e.g.
  
  for 1:1-relationship: has_husband: women \( \rightarrow \) husbands, has_wife: men \( \rightarrow \) wives
  
  for m:1-relationships: employed_by: persons \( \rightarrow \) companies
E-R diagrams

- graphical representation of entity sets, relationship sets, and their attributes by means of a graph

Notations

- rectangles represent entity sets: \( E \)

- ellipses represent attributes: \( A \)
  - they are connected with their entity set by undirected edges
  - key attributes are underlined

- relationship sets are represented by diamonds: \( R \)
  - relationship sets are connected with their pertaining entity sets by edges
  - edges carry information about cardinality according to imposed constraints

- a role of a relationship set is attached to the corresponding edge
Example: conceptual university schema

- **students**
  - reg-id
  - name
  - sem
  - attends** lectures**
  - grade

- **lectures**
  - id
  - credits
  - title
  - is_precondition_of
  - predecessor
  - successor

- **tests**
  - gives

- **assistants**
  - pers-id
  - name
  - room
  - works_for** professors**
  - pers-id
  - name
  - room

- **professors**
  - rank