Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Datenbanksysteme I

WS 2019/20
Torsten Grust, Christian Duta

Assignment #8

Submission Deadline: December 17, 2019 - 10:00

Please note that students currently have the opportunity to **evaluate lectures**. Please help us to improve **your** courses by providing precious feedback. Check your Mailbox now and participate **today**.

Exercise 1: Aggregate Queries (6 Points)

Again, we use the now familiar university schema and data from assignment #6. In case you miss any schemata or instances, we provided you with an archive `uni.zip` containing the university data from assignment #6 again.

Please solve the following tasks by formulating SQL queries. Make sure that your result tables match the **result** schemas.

1. Compute the average age of students for each type of degree, i.e. BSc and MSc students.

   ```
   result(pursueddegree, average_age)
   ```

2. Compute the number of different major subjects for which students are registered.

   ```
   result(subject_count)
   ```

3. Compute the students with maximal age.

   ```
   result(name, age)
   ```

4. Compute the number of classes in which each student is enrolled.

   ```
   result(name, classes_count)
   ```

Exercise 2: From NF^2 to 1NF (12 Points)

The table in Figure 1 contains information about the departments of a company, including each department's contacts as well as its employees and their tasks.[1] Boolean column `client` indicates whether a contact is a client (as opposed to a staff member, for example). The table is given in Non-First Normal Form (NF^2).

1. Transform the NF^2 schema into an equivalent 1NF database schema using algorithm `nf2to1nf()` from slide 10, slide set 8 "Database Design". Then, formulate suitable SQLDDL statements to create the resulting flat tables.
2. Populate the flat tables so that they provide the same information as the NF^2 table in Figure 1.
3. Formulate SQL queries for the following tasks on your 1NF representation:
   (a) Compute the number of employees per department.

   ```
   result(department, employees_count)
   ```

   (b) Find departments without contacts.

   ```
   result(department)
   ```

   (c) Return the names and department of employees without tasks.

   ```
   result(name, department)
   ```

   (d) Compute the number of clients per department.

   ```
   result(department, clients_count)
   ```

   Note: Obviously, some departments don't have any clients. Nevertheless, they should be included in the result with their correct number of clients. Think about using a *correlated subquery*.

Exercise 3: LEGO Data Warehouse (12 Points)

We provided you with a dataset that you already know from the lecture: The LEGO data warehouse `legodw.sql`. Load the file into a PostgreSQL database. Then, formulate queries for the following tasks:

1. For each store in Germany, compute the turnover per day of the week.

   ```
   result(store, city, dow, turnover)
   ```

2. For each store in Germany, compute the day(s) with the highest turnover.

   ```
   result(store, city, dow, turnover)
   ```

3. Compute the most popular set(s) per country, i.e. those sets that have had the most sold *items*.

   ```
   result(country, set)
   ```

[1] Due to the widespread use of employee-department schemas in database literature, it has been called the *Drosophila melanogaster* of database systems research.

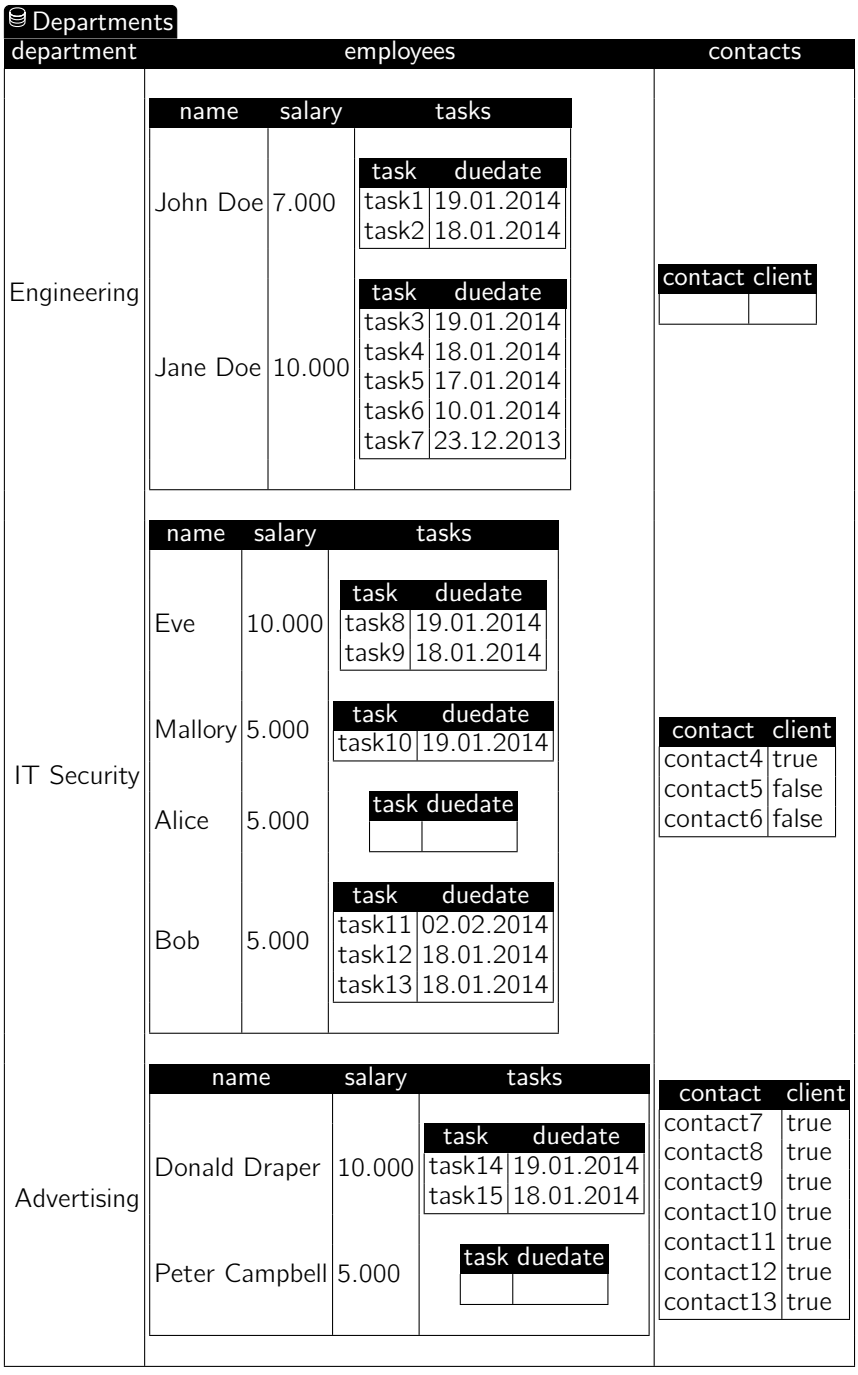

Figure 1: Table `Departments` in NF^2