Databases Lab 03
Emily Antosch & Furkan Yildrim & Julian Moldenhauer
02.04.2025

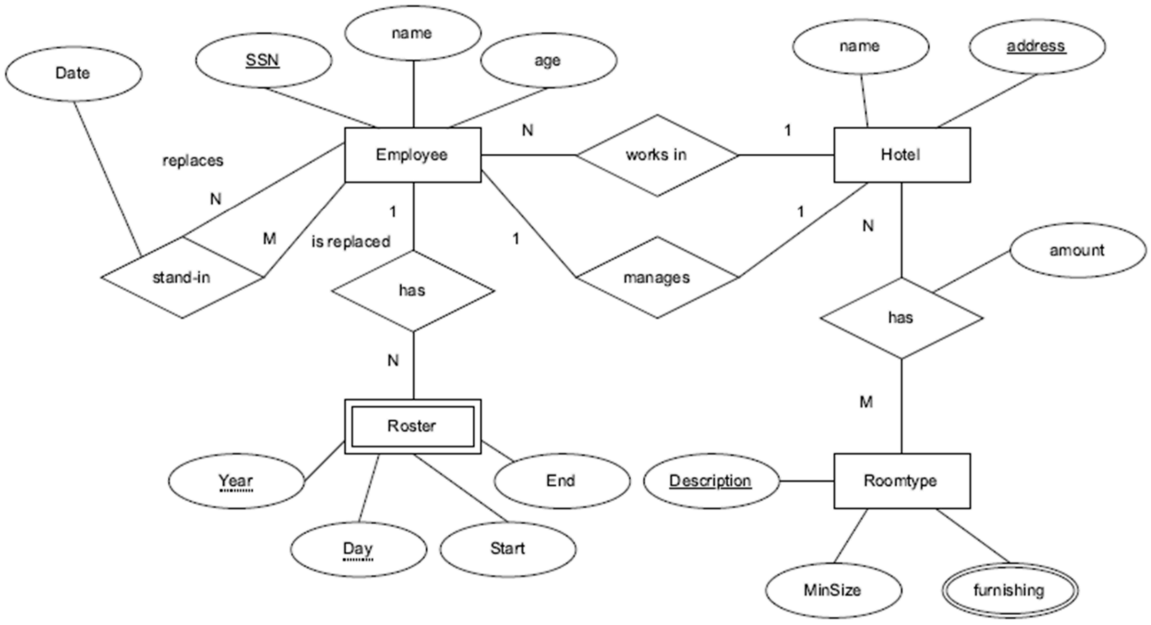
Databases Lab 03

This is the third lab of Databases. This lab is designed to help you deepen your understanding of database design and SQL. It is recommended that you familiarize yourself with the assignments to allow a more effective participation in Laboratory 2. If you have questions or need any support, help each other, or use the forum in our moodle room.

Contents

1. Assignment 1: Relational Model for a Hotel

A hotel chain wants to use a database to keep track of their hotels and employees, described in the following ERD:



In addition to the constraints in the ERD, take account of the following constraint: “Every stand-in is uniquely identifiable by the combination of the employee who replaces, the employee who is replaced and the date of the stand-in.” Convert the ERD into a relation schema. Make sure that every relation is in 3NF.

Solution

EMPLOYEE(SSN (PK), name, age, HotelID (FK))
HOTEL(Address (PK), SSN (FK), name)
ROOMTYPE(Description, minSize, furnishing)
ROSTER(year, day, SSN (FK), start, end)
HOTELHASROOMTYPE(Address (FK), Description (FK), amount)
STANDIN(SSN_Replacer (FK), SSN_Replacee (FK), date)

2. Assignment 2: Relational Algebra

The following excerpt of a database schema models a database of chess players. The relation PLAYER models a chess player, the relation GAME a game between two chess players. The attribute remis indicates whether the game is draw, it can be true or false.

PLAYER(PID (PK), firstName, lastName, dateOfBirth)
GAME(GID (PK), winnerID (FK), loserId (FK), remis)

The attributes winnerID and loserID are foreign keys to PID of PLAYER. Specify the following natural language queries as relational algebra queries.

- Which chess players were born before the 01.01.2000? (Output: lastName, dateOfBirth)

Solution

$$\text{PLAYER\_NAMES} \leftarrow \pi_{\text{dateOfBirth, lastname}}(\text{Player})$$
$$\text{PLAYER\_BEFORE} \leftarrow \sigma_{\text{dateOfBirth} < 01.01.2000}(\text{PLAYER\_NAMES})$$

- Which chess players have won at least once? (Output: first name and last name)

Solution

$$\text{PLAYER\_GAME} \leftarrow \text{PLAYER} \bowtie_{\text{PID = winnerID}}(\text{GAME})$$
$$\text{PLAYER\_NAMES} \leftarrow \pi_{\text{firstname, lastname}}(\text{PLAYER\_GAME})$$
$$\text{PLAYER\_WON} \leftarrow \sigma_{\text{remis = false}}(\text{PLAYER\_NAMES})$$

3. Assignment 3: Functional dependencies and normalization of a Furniture Database Version 1

A furniture company maintains a database that records information about orders, customers, and products. Note: In version 1, each order contains only one product. The sample database relation ORDER is as follows:

CustomerID	Name	Address	OrderID	Product	Quantity	Price
24	Maria Müller	Wegstraße 12b, Berlin	101	Table	2	150.0
24	Maria Müller	Wegstraße 12b, Berlin	102	Chair	5	80.0

18	Klaus Schmidt	Hauptstraße 4, Hamburg	103	Table	1	130.0
16	Petra Wagner	Lindenallee 7, Munich	104	Sofa	2	200.0

1. Determine the (full) functional dependencies. Keep in mind that functional dependencies are determined by the model, not just by the actual data in the database relations.
2. Determine potential candidate keys and a primary key for the given relation ORDER. Elaborate on your answer.
3. Transform the relational schema to 3NF. Your relation(s) should indicate PKs & FKs and contain all the data.
4. Create an entity relationship diagram of the 3NF schema.

Solution

1. **Functional Dependencies**:
   - CustomerID ⟶ Name, Address
   - OrderID ⟶ CustomerID, Name, Address, Product, Quantity, Price
   - Note: In version 1, each order contains only one product, so OrderID uniquely determines all other attributes.
2. **Candidate Keys and Primary Key**:
   - OrderID is a candidate key and the logical primary key since it uniquely identifies each order record.
   - No other attribute or combination of attributes can serve as a candidate key since OrderID functionally determines all other attributes.
3. **Transformation to 3NF**:
   - **CUSTOMER**(CustomerID (PK), Name, Address)
   - **ORDER**(OrderID (PK), CustomerID (FK), Product, Quantity, Price)
   - This decomposition removes the transitive dependency by separating customer information into its own relation.
4. **Entity Relationship Diagram**:

```md
[CUSTOMER] 1 -------- * [ORDER]
CustomerID (PK)        OrderID (PK)
Name                   CustomerID (FK)
Address                Product
                       Quantity
                       Price
```

4. Assignment 4: Functional dependencies and normalization of a Furniture Database Version 2

A furniture company maintains a database that records information about orders, customers, and products. Note: In version 2, each order may contain several products. The sample database relation ORDER is as follows:

CustomerID	Name	Address	OrderID	Product	Quantity	Price
24	Maria Müller	Wegstraße 12b, Berlin	101	Table	2	150.0
24	Maria Müller	Wegstraße 12b, Berlin	102	Chair	5	80.0
18	Klaus Schmidt	Hauptstraße 4, Hamburg	103	Table	1	130.0
16	Petra Wagner	Lindenallee 7, Munich	104	Sofa	2	200.0

1. Determine the (full) functional dependencies. Keep in mind that functional dependencies are determined by the model, not just by the actual data in the database relations.
2. Determine potential candidate keys and a primary key for the given relation ORDER. Elaborate on your answer.
3. Transform the relational schema to 3NF. Your relation(s) should indicate PKs & FKs and contain all the data.
4. Implement the schema in your database and insert sample data.

Solution

1. **Functional Dependencies**
   - CustomerID -> Name, Address
   - OrderID -> CustomerID, Name, Address
   - OrderID, Product -> CustomerID, Name, Address, Quantity, Price
2. **Candidate Key**
   - OrderID and Product together would make up the only feasible composite primary key, since those two define every other entry in a row in the database
3. **3NF**
   - **CUSTOMER**(CustomerID (PK), Name, Address)
   - **ORDER**(OrderID (PK), CustomerID (FK), Product (PK), Quantity, Price)
4. **Schema**

```SQL
CREATE TABLE ORDER (
  ORDER_ID INT NOT NULL,
  CUSTOMER_ID INT NOT NULL,
  PRODUCT TEXT NOT NULL,
  QUANTITY INT,
  Price REAL,
  CONSTRAINT "PK_ORDER_ORDER_ID_PRICE" PRIMARY KEY (ORDER_ID, PRODUCT),
  CONSTRAINT "FK_ORDER_CUSTOMER_CUSTOMER_ID_CUSTOMER_ID"
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID)
);

CREATE TABLE CUSTOMER (
  CUSTOMER_ID INT NOT NULL PRIMARY KEY,
  NAME TEXT,
  ADDRESS TEXT
);
```

5. Assignment 5: Relational Algebra vs. SQL query for a Weather Database

The following excerpt from a database schema models a database about a weather station. The following assignments are to be answered in the form of relational algebra.

- **WEATHER_DATA** (DataID, CityID (FK), Date, Temperature, Humidity, Precipitation, Wind-Speed)
- **CITY** (CityID, Name, Country)

1. Which cities have an average daily temperature above 25°C in August 2023?
2. Which cities experienced no precipitation on any day in July 2023?
3. On which particular day did the cities have the highest wind speed?

4. Which cities recorded the highest temperature?
5. Which cities had the lowest humidity in May 2023?

Solution

$$\text{CITY\_WEATHER} \leftarrow \text{WEATHER\_DATA} \bowtie_{\text{CityID} = \text{CityID(City)}}$$
$$\text{CITY\_TEMP} \leftarrow \sigma_{\text{TEMP} > 25(\text{CITY\_WEATHER})}$$
$$\text{CITY\_DATE} \leftarrow \sigma_{\text{Date} >= 01.08.2023 \ \& \ \text{Date} <= 31.08.2023(\text{City})}$$
$$\text{CITY\_TEMP\_DATE} \leftarrow \text{CITY\_TEMP} \cap \text{CITY\_DATE}$$
$$\text{CITY\_TEMP\_DATE} \leftarrow \pi_{\text{Name, TEMP(CITY\_TEMP\_DATE)}}$$

Solution

$$\text{CITY\_WEATHER} \leftarrow \text{WEATHER\_DATA} \bowtie_{\text{CityID} = \text{CityID(City)}}$$
$$\text{CITY\_JULY} \leftarrow \sigma_{\text{Date} >= 01.07.2023 \ \& \ \text{Date} <= 31.07.2023(\text{CITY\_WEATHER})}$$
$$\text{CITY\_PRECIPITATION} \leftarrow \sigma_{\text{Precipitation} = 0(\text{CITY\_WEATHER})}$$
$$\text{CITY\_TEMP\_DATE} \leftarrow \pi_{\text{Name, TEMP(CITY\_PRECIPITATION)}}$$

SQL

```sql
SELECT CITY, DATE, MIN(WindSpeed)
FROM WEATHER_DATA wd
LEFT JOIN CITY c ON c.cityid = wd.cityid
GROUP BY CITY, DATE
```

Solution

$$\text{CITY\_WEATHER} \leftarrow \text{WEATHER\_DATA} \bowtie_{\text{CityID} = \text{CityID(City)}}$$
$$\text{CITY\_WINDSPEED} \leftarrow \rho_{\text{windspeed/windspeed1}(\pi_{\text{windspeed(CITY\_WEATHER)}} - \pi_{\text{windspeed1(CITY\_WEATHER)}} \sigma_{\text{windspeed1<windspeed2}}($$
$$\text{CITY\_DAY} \leftarrow \pi_{\text{DAY(CITY\_WINDSPEED)}}$$

Solution

$$\text{CITY\_WEATHER} \leftarrow \text{WEATHER\_DATA} \bowtie_{\text{CityID} = \text{CityID(City)}}$$
$$\rho_{\text{temperature/temperature1}(\pi_{\text{temperature(CITY\_WEATHER)}} - \pi_{\text{temperature1(CITY\_WEATHER)}} \sigma_{\text{temperature1<temperature2}}(\rho_{\text{temperature1/temperature(CITY\_}}$$

Solution

$$\text{CITY\_WEATHER} \leftarrow \text{WEATHER\_DATA} \bowtie_{\text{CityID} = \text{CityID(City)}}$$
$$\text{CITY\_MAY} \leftarrow \sigma_{\text{Date} >= 01.05.2023 \ \& \ \text{Date} <= 31.05.2023(\text{CITY\_WEATHER})}$$
$$\rho_{\text{humidity/humidity1}(\pi_{\text{humidity(CITY\_MAY)}} - \pi_{\text{humidity1(CITY\_MAY)}} \sigma_{\text{humidity1>humidity2}}(\rho_{\text{humidity1/humidity(CITY\_MAY)}} \times \rho_{\text{humidity2/humidity(CITY\_M}}$$