INF.01014UF DATABASES - Report

Simone Franza 01530693
Lukas Meer 01430417

Table of content

1. Database schema

Domains:	**CId**, integer	Id-number of the customer
	CName, string	Name of the customer
	CCity, string	City where the customer lives
	CAge, integer	Age of the customer
	CNumber, string	Telephone number of the customer
	BId, integer	Id-number of kind of beer
	BName, string	Name of a beer
	BPrice, integer	Price of a beer
	TId, integer	Id-number of the transaction
	TDate, date	Date of the purchase
	TQnt, integer	Quantity of a kind of beer bought by a customer
	RRating, integer	Rating that a customer gave to a kind of beer

- **Relation: customer** (CId, CName, CCity, CAge, CNumber)
- **Relation: beer** (BId, BName, BPrice)
- **Relation: transaction** (TId, CId, BId, TDate, TQnt)
- **Relation: rating** (CId, BId, RRating)

2. Functional dependencies

Relation: customer (CId, CName, CCity, CAge, CNumber)
CId → CName
CId → CCity
CId → CAge
CId → CNumber
Relation: beer (BId, BName, BPrice)
BId → BName
BId → BPrice
Relation: transaction (TId, CId, BId, TDate, TQnt)
TId → CId
TId → BId
TId → TDate
TId → TQnt
Relation: rating (CId, BId, RRating)
(CId, BId)→ RRating

Definition of the 3rd normal form:
“Third normal form (3NF) is the third step in normalizing a database and it builds on the first and second normal forms, 1NF and 2NF.
3NF states that all column reference in referenced data that are not dependent on the primary key should be removed. Another way of putting this is that only foreign key columns should be used to reference another table, and no other columns from the parent table should exist in the referenced table.”

Source: https://www.techopedia.com/definition/22561/third-normal-form-3nf (visited: 05/06/2018)

As shown at the beginning of Chapter 2 all our domains in their relations only depend on their respective primary key, i.e. they don't depend on any other domains.

For the relation “beer” we assumed, that the entries of the column BName identify a single model of beer (for example Gösser Naturradler). This model can have different sizes, therefore BPrice doesn't depend of BName.

For the relation “transaction” we assigned to every transaction an Id. Every customer can buy multiple products a day and in different quantities so every column only depends of TId. The column TQnt shows the number of bottles of a certain beer model (BId) that a costumer bought.

For the relation “rating” we assumed, that every customer can assign only one rating to a certain model of beer (BId). Therefore RRating cannot depend only of CId or of BId.

Additionally all other requirements are also fulfilled (2nd normal form), so our database is in the 3rd normal form.

3. Current state of the database

customer				
CId	**CName**	**CCity**	**CAge**	**CNumber**
1	Mueller	Graz	20	00436601111222
2	Franza	Muenchen	30	00493149264872
3	Maar	Graz	35	00436602222333
4	Rossi	Rom	50	00393486805555
5	Pranger	Innsbruck	25	00436504262333
6	Hager	Wien	45	00436958728333
7	Muster	Heidelberg	25	00493149226872
8	Ferrari	Florenz	35	00393486195255
9	Heine	Graz	27	00436602182433
10	Musk	Innsbruck	97	00436605262339

beer		
BId	**BName**	**BPrice**
1	Goesser Naturradler	5
2	Goesser Naturradler	4
3	Stiegl Helles	3
4	Zillertaler Weizen	4
5	Murauer Helles	3

transaction				
TId	**CId**	**BId**	**TDate**	**TQnt**
1	1	1	2018-04-10	5
2	1	2	2018-04-10	3
3	2	1	2018-04-15	6
4	3	2	2018-04-15	1
5	6	5	2018-04-20	10
6	4	4	2018-04-25	6
7	8	5	2018-05-10	3
8	8	3	2018-05-17	7
9	10	4	2018-05-30	24

transaction				
TId	**CId**	**BId**	**TDate**	**TQnt**
10	5	2	2018-06-01	4
11	3	2	2018-06-03	9
12	6	3	2018-06-03	2

rating		
CId	**BId**	**RRating**
1	1	3
1	3	2
2	1	5
3	2	2
5	1	4
5	2	3
6	5	0
10	3	2

4. Database queries in terms of the Relational Algebra

Get the names of beer and ratings for consumer, which live in Graz

```
Select customer Where CCity = 'Graz' Giving A;
Join A And rating over CId Giving B;
Join B And beer over BId Giving C;
Project C Over BName, RRating Giving RESULT;
```

Get names of those customer, who bought “Stiegl Helles” or live in “Innsbruck”

```
Select customer Where CCity = 'Innsbruck' Giving A;
Project A Over CName Giving X;
Select beer Where BName = 'Stiegl Helles' Giving B;
Join B And transaction Over BId Giving C;
Join C And customer Over CId Giving D;
Project D Over CName Giving Y;
X Union Y Giving RESULT;
```

Get names of those customers, who bought both type of “Gösser Naturradler”

```
Select beer Where BName = 'Gösser Naturradler' Giving A;
Project A Over BId Giving X;
Project transaction Over CId, BId Giving Y;
Divide Y By X Giving C;
Join C And customer Over CId Giving D;
Project D Over CName Giving RESULT;
```

5. Database queries in terms of the Relational Calculus

Get the model of beer which sold the most on a single transaction

B -> beer
T1 -> transaction
T2 -> transaction

(B.BName): ∃T1 ∀T2 (B.BId = T1.BId & T1.TQnt >= T2.TQnt)

———

Get the names of the customers who bought “Murauer Helles”

C -> customer
T -> transaction
B -> beer

(C.CName): ∃T ∃B (C.CId = T.CId & T.BId = B.BId & B.BName = 'Murauer Helles')

———

Get the ratings of the model of beer which sold the least in a single transaction

R -> rating
T1 -> transaction
T2 -> transaction
B -> beer

(R.RRating): ∃T1 ∀T2 ∃B (R.BId = B.BId & B.Bid = T1.BId & T1.TQnt <= T2.TQnt)

6. Database queries in terms of the SQL

Get the names and the telephone numbers from all the customers who are older than 30 and don't live in “Innsbruck” and bought “Stiegl Helles”. Sort the results by name

```
SELECT CName, CNumber FROM
customer WHERE (CId IN
        (SELECT CId FROM transaction
        WHERE BId IN
                (SELECT BId FROM beer
                WHERE BName = 'Stiegl Helles')))
AND (CAge >= 30)
AND NOT (CCity = 'Innsbruck')
ORDER BY CName ASC;
```

———

Get the city of the customers who bought more than 10 products over all transactions.

```
SELECT CCity FROM
customer WHERE (CId IN
        (SELECT CId FROM transaction
        GROUP BY CId HAVING SUM(TQnt) > 10))
ORDER BY CCity DESC;
```

———

Get the name and ids of the beers of which were bought more than 15 units or are called “Murauer Helles”

```
SELECT BId, BName FROM
beer WHERE (BId IN
        (SELECT BId FROM transaction
        GROUP BY BId HAVING SUM(TQnt) > 15))
OR (BName = 'Murauer Helles')
ORDER BY BId ASC;
```

7. Database queries in terms of the SQL without nested SQL blocks

Get the names and the telephone numbers from all the customers who are older than 30 and don't live in “Innsbruck” and bought “Stiegl Helles”. Sort the results by name

```
SELECT CName, CNumber FROM customer, transaction, beer
WHERE customer.CId = transaction.CId
AND transaction.BId = beer.BId
AND BName = 'Stiegl Helles'
AND CAge >= 30
AND NOT CCity = 'Innsbruck'
ORDER BY CName ASC;
```

Get the city of the customers who bought more than 10 products over all transactions.

```
SELECT CCity FROM customer, transaction
WHERE customer.CId = transaction.CId
GROUP BY transaction.CId HAVING SUM(TQnt) > 10
ORDER BY CCity DESC;
```

Get the name and ids of the beers of which were bought more than 15 units or are called “Murauer Helles”

```
SELECT beer.BId, BName FROM beer, transaction
WHERE BName = 'Murauer Helles'
OR beer.BId = transaction.BId
GROUP BY beer.BId HAVING SUM(TQnt) > 15
ORDER BY beer.BId ASC;
```

8. Practical implementation of the database with SQL

```
CREATE DATABASE 01530693_beer;
USE 01530693_beer

# create all relations

CREATE TABLE customer (
        CId INTEGER NOT NULL,
        CName VARCHAR(30) NOT NULL,
        CCity VARCHAR(30) NOT NULL,
        CAge INTEGER NOT NULL,
        CNumber VARCHAR(20) NOT NULL,
PRIMARY KEY (CId));

CREATE TABLE beer (
        BId INTEGER NOT NULL,
        BName VARCHAR(40) NOT NULL,
        BPrice INTEGER NOT NULL,
PRIMARY KEY (BId));

CREATE TABLE transaction (
        TId INTEGER NOT NULL,
        CId INTEGER NOT NULL,
        BId INTEGER NOT NULL,
        TDate DATE NOT NULL,
        TQnt INTEGER NOT NULL,
PRIMARY KEY (TId),
FOREIGN KEY (CId) REFERENCES customer(CId),
FOREIGN KEY (BId) REFERENCES beer(BId));

CREATE TABLE rating (
        CId INTEGER NOT NULL,
        BId INTEGER NOT NULL,
        RRating INTEGER NOT NULL,
PRIMARY KEY (CId, BId),
FOREIGN KEY (CId) REFERENCES customer(CId),
FOREIGN KEY (BId) REFERENCES beer(BId));

# insert content into the relations:

INSERT INTO customer VALUES
        (1, 'Mueller', 'Graz', 20, '00436601111222'),
        (2, 'Franza', 'Muenchen', 30, '00493149264872'),
        (3, 'Maar', 'Graz', 35, '00436602222333'),
        (4, 'Rossi', 'Rom', 50, '00393486805555'),
        (5, 'Pranger', 'Innsbruck', 25, '00436504262333'),
        (6, 'Hager', 'Wien', 45, '00436958728333'),
        (7, 'Muster', 'Heidelberg', 25, '00493149226872'),
        (8, 'Ferrari', 'Florenz', 35, '00393486195255'),
        (9, 'Heine', 'Graz', 27, '00436602182433'),
        (10, 'Musk', 'Innsbruck', 97, '00436605262339');

INSERT INTO beer VALUES
        (1, 'Goesser Naturradler', 5),
        (2, 'Goesser Naturradler', 4),
        (3, 'Stiegl Helles', 3),
        (4, 'Zillertaler Weizen', 4),
        (5, 'Murauer Helles', 3);
```

```
INSERT INTO transaction VALUES
        (1, 1, 1, '2018-04-10', 5),
        (2, 1, 2, '2018-04-10', 3),
        (3, 2, 1, '2018-04-15', 6),
        (4, 3, 2, '2018-04-15', 1),
        (5, 6, 5, '2018-04-20', 10),
        (6, 4, 4, '2018-04-25', 6),
        (7, 8, 5, '2018-05-10', 3),
        (8, 8, 3, '2018-05-17', 7),
        (9, 10, 4, '2018-05-30', 24),
        (10, 5, 2, '2018-06-01', 4),
        (11, 3, 2, '2018-06-03', 9),
        (12, 6, 3, '2018-06-03', 2);

INSERT INTO rating VALUES
        (1, 1, 3),
        (1, 3, 2),
        (2, 1, 5),
        (3, 2, 2),
        (5, 1, 4),
        (5, 2, 3),
        (6, 5, 0),
        (10, 3, 2);

# print the content of every table (to test if everything is right)

SELECT * FROM customer;
SELECT * FROM beer;
SELECT * FROM transaction;
SELECT * FROM rating;

# now start the queries from chapters 6 and 7
```

9. Servlets (Database Modification)