
# Database Theory And Applications

**Lecturer: Dr Chris Anagnostopoulos**

# Assessed Coursework

**Student: Theodoros Vrakas**

| | |
|---|---|
| **2593566v** | **2593566V@student.gla.ac.uk** |

**Date: 01/04/2021**

# Task 1

## Task 1.1

**//M1**

Number of tuples: We have to INSERT 2 tuples.

Explanation: In order to add a new language for McDonald, we have to insert 2 tuples since there is a need to include one language for each COMP_SKILL of the employee. This is an Insertion Anomaly.

**//M2**

Number of tuples: We have to UPDATE 4 tuples.

Explanation: To enforce consistency, we have to update all the tuples related to BROOKS. Address is Partial Dependant with SNN Primary Key only. This is an Updating Anomaly.

**//M3**

Number of tuples: We have to INSERT 2 tuples.

Explanation: To enforce consistency, we have to insert the new computer skill as many languages BROOKS speaks, one for French and one for English. This is an Insertion Anomaly.

**//M4**

Number of tuples: We have to INSERT 600 tuples.

Explanation: For the relation to remain in 1NF we shouldn't have nested and multivalued attributes. Per employee, we have to insert the 3 new computer skill twice, as many are employee's spoken languages. So we have 100x3x2=600 tuples. This is an Insertion Anomaly.

**//M5**

Number of tuples: We have to DELETE 4 tuples.

Explanation: After task M1, we have 4 tuples for McDonald projecting Programming skill for each language. To enforce consistency, we have to delete all these 4 tuples. This is a Deletion Anomaly.

## Task 1.2

**//SQL1**

```
SELECT surname, COUNT(distinct language) as languages_count
FROM employee
GROUP BY surname;
```

**//SQL2**

```
SELECT surname, COUNT(distinct language) as languages_count
FROM employee
WHERE SSN=2
GROUP BY surname;
```

**//SQL3**

```
SELECT COUNT(distinct ssn) as SQL3
FROM employee
WHERE language = 'Greek'
```

**//SQL4**

```
SELECT COUNT(*) as SQL4
FROM(
    SELECT COUNT(distinct ssn) as CountSkills
    FROM employee
    GROUP BY ssn
    HAVING COUNT(distinct comp_skills) > 1
) AS whatever
```

**//SQL5**

```
SELECT COUNT(*) as SQL5
FROM(
    SELECT COUNT(distinct ssn)
    FROM employee
    GROUP BY ssn, salary
    HAVING COUNT(distinct comp_skills) > 1 AND salary > 50000
) AS whatever
```

**//SQL6**

```
SELECT distinct ssn, surname
FROM employee
GROUP BY ssn, surname, comp_skills
HAVING COUNT(distinct language) > 2 AND comp_skills = 'Data Analysis'
```

**//SQL7**

```
SELECT COUNT(*)
FROM(
    SELECT COUNT(distinct ssn)
    FROM employee
    WHERE department IN (
            SELECT department
            FROM employee
            GROUP BY department
            HAVING COUNT(distinct ssn) > 2
    )
    GROUP BY ssn, salary
    HAVING COUNT(distinct language) > 2 AND salary > 51000
) AS whatever
```

The most challenging query for me was the SQL7. It took me more time to write it correctly than all the other queries together probably. I had to do a quick revision of all the SQL

operators again. The double nested queries were a little tricky for my mind at first, to figure what I wanted to pass to nested SELECT statement each time.

# Task 2

## Task 2.1

Employee relation is in 1NF. It has only single values attributes, unique names for attributes and columns and the values that are stored in a column are of the same type.

The relation is not in 2NF since 2NF doesn't allow Partial Dependency (PD). It was decided that the Primary Key (PK) will be the composition of three columns, SSN, LANGUAGE, COMP_SKILLS. We can see though that SURNAME, ADDRESS, SALARY and DEPARTMENT only depend on the SSN. Hence, we have PD for four attributes, since they depend only to a part of the PK.

## Task 2.2

A Multivalued Dependency (MVD) is an declaration of two attributes or sets of attributes that are independent of one another. The Fourth Normal form is used when multivalued dependency occurs in any relation.

Some relations have constraints that cannot be specified as functional dependencies and therefore Boyce-Codd Normal (BCNF) is not violated. For this situation, the concept of multivalued dependency was suggested and based on this dependency, the fourth normal form was defined.

Multivalued dependencies are a consequence of first normal form (1NF), which forbids an attribute in a tuple to have a set of values. If more than one multivalued attribute is present, the second option of normalizing the relation introduces a multivalued dependency.

In order for a relation to be in 4NF, it should satisfy the conditions of being in the BCNF and it should not have any MVD. Therefore, in 4NF all nontrivial MVD's are eliminated, as are all FD's that violate BCNF.

The process of normalizing a relation involving the nontrivial MVDs that is not in 4NF consists of decomposing it so that each MVD is represented by a separate relation where it becomes a trivial MVD.

**Example**

Some members of an online Movie Club community were asked to fill a survey and define the genre of 3 movies based on their narrative elements. They could use specific genre tags and each movie could have up to 4 tags. The table below presents the results.

*Table 1 Results relation; members had to define genre of 3 movies.*

| Name | Movies | Genre |
|---|---|---|
| Theodoros | Prisoners [2013] | Drama |
| Theodoros | Prisoners [2013] | Mystery |

| Name | Movies | Genre |
|---|---|---|
| Theodoros | Dark Waters [2019] | Drama |
| Theodoros | Enough Said [2013] | Comedy |
| John | Prisoners [2013] | Mystery |
| John | Dark Waters [2019] | Drama |
| John | Dark Waters [2019] | Crime |
| John | Enough Said [2013] | Comedy |
| Eleni | Prisoners [2013] | Thriller |
| Eleni | Dark Waters [2019] | Biography |
| Eleni | Enough Said [2013] | Romance |
| Eleni | Enough Said [2013] | Comedy |
| Eleni | Enough Said [2013] | Drama |

**Explaination**

The table is in 1NF. We can observe that all attributes are indepedent. The 3 members rated some films and each film was defined with one or more tags. There are two many-to-many relationships which are independent.

Because the Movies are attached to the Name and the Genre are also attached to the Name, since they are based on people personal opinion, these are independent of each other. That means this database design has a multivalued dependency. If we were to add a new movie to be defined, then we would have to add at least one tag for each person, so at least 3 tuples to be inserted. Similarly, if we were to add one more person to define each movie, we should insert at least 3 tuples.

## Task 2.3

There are two MVD in our relation, SSN with LANGUAGES and SSN with COMP_SKILLS.

As mentioned in Task 2.1, the Employee relation is in **1NF**. In order to bring it in the 2NF we need to get rid of the any Partial Dependancy. We can see though that SURNAME, ADDRESS, SALARY and DEPARTMENT only depend on the SSN. Hence, we have PD for four attributes, since they depend only to a part of the PK. A PK is a chosen Candidate Key which consist of the minimal set of attributes (at least one) that uniquely identifies any tuple (which also called Superkey).

To normalize our relation in a set of **2NF** relations, we need to remove the Partial Dependency from our relation Employee and move SURNAME, ADDRESS, SALARY and DEPARTMENT to a new relation. We create two new relations EmployeeInfo and Skills.

**EmployeeInfo**(SSN, SURNAME, ADDRESS, SALARY, DEPARTMENT)

**Skills**(SSN, LANGUAGES, COMP_SKILLS)

- SSN in relation Skills is a FK referencing to SSN in EmployeesInfo.

Our relations are in **2NF** and there is not Transitive Dependency, which happens when a non-prime attribute which on other non-prime attributes rather than depending upon the prime attributes or primary key. So, it fulfills all the criteria to be in the **3NF**.

Similarly, out relations already fulfill all the criteria to be in **BCNF** since it's on the **3NF** and for any dependency A → B, A should is a super key. As mention in Task 2.2, some relations have constraints that cannot be specified as functional dependencies and therefore Boyce-Codd Normal (BCNF) is not violated.

We solve that problem with **4NF** which requires that a relation should already be on BCNF and it does not have any MVD. To remove our MVD we have to split our two MVD from the Employee_Skills relation to two new relation. The final results after of our relations are in 4NF is presented below.

*Table 2 EmployeeInfo relation.*

| **SSN** | SURNAME | ADDRESS | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | MCDONALD | 35 St. Vincent | 51,000 | Computing |
| 2 | BROOKS | 4 Hyndland Rd | 55,000 | Engineering |
| 3 | YOUNG | 1 University Ln | 50,000 | Computing |

*Table 4 Languages relation.*

| **SSN** | **LANGUAGES** |
|---|---|
| 1 | Dutch |
| 1 | English |
| 1 | Greek |
| 2 | French |
| 2 | English |
| 3 | English |
| 3 | Greek |

*Table 3 CompSkills relation.*

| **SSN** | **COMP_SKILLS** |
|---|---|
| 1 | Programming |
| 1 | Data Analysis |
| 2 | Programming |
| 2 | Data Analysis |
| 3 | Data Analysis |

- The PK are underlined
- SSN in relation Languages is a FK referencing to SSN in EmployeeInfo.
- SSN in relation CompSkills is a FK referencing to SSN in EmployeeInfo.

## Task 2.4

**//M1**

Number of tuples: We have to INSERT 1 tuples.

Explanation: After the normalization of our first relation, we just need to insert 1 tuple in Languages relation.

**//M2**

Number of tuples: We have to UPDATE 1 tuples.

Explanation: We have to update ADDRESS for SSN 2 in EmployeeInfo relation.

**//M3**

Number of tuples: We have to INSERT 1 tuples.

Explanation: We have to insert 1 tuple in the CompSkills relation for SSN 2.

**//M4**

Number of tuples: We have to INSERT 500 tuples.

Explanation: We insert 100 new tuples on the Employee relation, 200 tuples in the Language relation (two for each new employee) and 300 new tuples in CompSkills relation (3 skills per new employee).

**//M5**

Number of tuples: We have to delete 1 tuple.

Explanation: We delete 1 tuple from CompSkills relation for SSN1 and Programming.

## Task 2.5

**//SQL1**

```
SELECT ssn, COUNT(distinct comp_skills) as n_comp_skills
FROM compskills
GROUP BY ssn;
```

Modification: After normalization of the relation to 4NF, we change FROM statement focus to compskills relation.

**//SQL2**

```
SELECT ssn, COUNT(distinct language) as languages_count
FROM languages
WHERE SSN=1
GROUP BY ssn;
```

Modification: After normalization of the relation to 4NF, we change FROM statement focus to language relation.

**//SQL3**

```
SELECT COUNT(distinct ssn) as SQL3
FROM languages
WHERE language = 'Greek'
```

Modification: After normalization of the relation to 4NF, we change FROM statement focus to language relation.

**//SQL4**

```
SELECT COUNT(*) as SQL4
FROM(
    SELECT COUNT(distinct ssn) as CountSkills
    FROM compskills
    GROUP BY ssn
    HAVING COUNT(distinct comp_skills) > 1
```

```
) AS whatever
```

Modification: After normalization of the relation to 4NF, we change FROM statement focus to compskills relation.

**//SQL5**

```
SELECT COUNT(*) as SQL5
FROM(
    SELECT COUNT(distinct E.ssn)
    FROM employeeinfo as E, compskills as CS
    GROUP BY E.ssn, E.salary
    HAVING COUNT(distinct CS.comp_skills) > 1 AND E.salary > 50000
) AS whatever
```

Modification: After normalization of the relation to 4NF, we take our data from two relations with aliases, employeeinfo and compskills.

**//SQL6**

```
SELECT DISTINCT E.ssn, E.surname
FROM employee1 AS E, compskills AS CS, languages AS L
WHERE E.ssn = L.ssn AND E.ssn = CS.ssn AND CS.comp_skills = 'Data Analysis'
GROUP BY E.ssn, E.surname
HAVING COUNT(DISTINCT L.language) > 2
```

Modification: After normalization of the relation to 4NF, we take our data from three relations with aliases, employeeinfo, compskills and language. A Where clause has been added to filter records that fulfill the specified conditions above from the 3 different tables.

**//SQL7**

```
SELECT COUNT(*)
FROM(
    SELECT COUNT(distinct E.ssn)
    FROM employeeinfo AS E, Languages AS L
    WHERE E.SSN = L.SSN AND E.Department IN (
            SELECT department
            FROM employee
            GROUP BY department
            HAVING COUNT(distinct ssn) > 2
    )
    GROUP BY E.ssn, E.salary
    HAVING COUNT(distinct L.language) > 2 AND E.salary > 51000
) AS whatever
```

Modification: After normalization of the relation to 4NF, we take our data from two relations with aliases, EmployeeInfo and and Language. A Where clause now includes the equality of ssn from both relations.