Report

Nhat Nam Nguyen

November 2024

1 Structured Data

1.1 Ex1

```
#!/bin/bash
xml_file= $1
csv_file= $2

if $xml_file = "students.xml" then
    echo "student_name,student_id,student_email,programme,year,
    address,contact,module_id,module_name,module_leader,lecturer1,
    lecturer2,faculty,building,room,exam_mark,coursework1,coursework2,
    coursework3" > $csv_file
    grep -oP '(?<=<student>).*?(?=</student>)'
    $xml_file | while read -r student; do
      student_name=$(echo "$student" | grep -oP '
      (?<=<student_name>).*?(?=</student_name>)')
      student_id=$(echo "$student" | grep -oP
```

```
'(?<=<student_id>).*?(?=</student_id>)')
student_email=$(echo "$student" | grep -oP
'(?<=<student_email>).*?(?=</student_email>)')
programme=$(echo "$student" | grep -oP
'(?<=<programme>).*?(?=</programme>)')
year=$(echo "$student" | grep -oP
'(?<=<year>).*?(?=</year>)')
address=$(echo "$student" | grep -oP
'(?<=<address>).*?(?=</address>)' | sed 's/,//g')
contact=$(echo "$student" | grep -oP
'(?<=<contact>).*?(?=</contact>)')
module_id=$(echo "$student" | grep -oP
'(?<=<module_id>).*?(?=</module_id>)')
module_name=$(echo "$student" | grep -oP
'(?<=<module_name>).*?(?=</module_name>)')
module_leader=$(echo "$student" | grep -oP
'(?<=<module_leader>).*?(?=</module_leader>)')
lecturer1=$(echo "$student" | grep -oP
'(?<=<lecturer1>).*?(?=</lecturer1>)')
lecturer2=$(echo "$student" | grep -oP
'(?<=<lecturer2>).*?(?=</lecturer2>)')
faculty=$(echo "$student" | grep -oP
'(?<=<faculty>).*?(?=</faculty>)')
building=$(echo "$student" | grep -oP
'(?<=<building>).*?(?=</building>)')
room=$(echo "$student" | grep -oP
```

```
        '(?<=<room>).*?(?=</room>)')
        exam_mark=$(echo "$student" | grep -oP
        '(?<=<exam_mark>).*?(?=</exam_mark>)')
        coursework1=$(echo "$student" | grep -oP
        '(?<=<coursework1>).*?(?=</coursework1>)')
        coursework2=$(echo "$student" | grep -oP
        '(?<=<coursework2>).*?(?=</coursework2>)')
        coursework3=$(echo "$student" | grep -oP
        '(?<=<coursework3>).*?(?=</coursework3>)')

        echo "$student_name,$student_id,$student_email,$programme,$year,$ad
    done
else
    echo "faculty,building,room,capacity" > $csv_file
    grep -oP '(?<=<faculty>).*?(?=</faculty>)' $xml_file | while read -r
        faculty=$(echo "$faculty" | grep -oP '(?<=<faculty>).*?(?=</faculty
        building=$(echo "$faculty" | grep -oP '(?<=<building>).*?(?=</buildi
        room=$(echo "$faculty" | grep -oP '(?<=<room>).*?(?=</room>)')
        capacity=$(echo "$faculty" | grep -oP '(?<=<capacity>).*?(?=</capac
        echo "$faculty,$building,$room,$capacity" >> $csv_file
    done

fi
```

Using grep to extract the data in the tags $< faculty >$ *and* $< student >$

1.2 Ex2

```
input = $1
output = $2
cat input | cut -f1 -d | sort >> $output.txt
```

2 Relational Model

2.1 Ex3

Faculty.csv(faculty, building, room, capacity)

Students.csv(student_name, student_id, student_email, programme, year, address, contact, module_id, module_name, module_leader, lecturer1, lecturer2, faculty, building, room, exam_mark, coursework1, coursework2, coursework3)

2.2 Ex4

The minimal set of functional dependencies for Faculty.csv:

{building, room $\rightarrow$ capacity; building, room $\rightarrow$ faculty}

The minimal sets of functional dependencies for Students.csv:

{building, room $\rightarrow$ faculty; student_name, student_id $\rightarrow$ student_email;

student_name, student_id $\rightarrow$ programme; student_name, student_id $\rightarrow$ address;

student_name, student_id $\rightarrow$ contact; module_id $\rightarrow$ module_name, module_leader;

module_id, module_name $\rightarrow$ exam_mark; student_name, student_id $\rightarrow$ year;

module_id, module_name $\rightarrow$ lecturer1; module_id, module_name $\rightarrow$ lecturer2; }

2.3 Ex5

Possible candidate keys for faculty.csv: {building, room}; {faculty, building, room}

Possible candidate keys for faculty.csv: {student_id, module_id}; {student_email

module_id}; {student_id, module_name}; {student_email, module_name}

2.4 Ex6

A primary key for faculty.csv: {building, room}
A primary key for students.csv: {student_id}

3 Normalization

3.1 Ex7

No, the data are not in the first normal form because there are some repeated attributes in the database, specifically lecturer attribute and coursework attribute.
The reason for it to be not in first normal form is Coursework(student_id, module_id, exam_mark, coursework), Lecturer(module_id, lecturer)

3.2 Ex8

By identifying all of the primary keys which is student_id, module_id, module_name decompose, move all of the dependents into a new table

3.3 Ex9

All of the partial-key dependencies are:
$\{(student\_id, module\_id) \rightarrow student\_name, student\_email, programme, year, address, contact\}$

3.4 Ex10

Relations students:
Fields:

- student_id (Primary Key, INTEGER)
- student_name (TEXT)
- student_email (TEXT)
- programme (TEXT)
- year (INTEGER)
- address (TEXT)
- contact (TEXT)

Relation modules(
Fields:

- module_id (Primary Key, TEXT)
- module_name (TEXT)
- module_leader (TEXT)
- faculty (TEXT)
- building (TEXT)
- room (TEXT)

)
Relation modules_lecturers(
Fields:

- module_id (Primary Key, TEXT)
- module_lecturers

)

Relation modules_exammark(

- student_id (Foreign Key, TEXT)
- module_id (Foreign Key, TEXT)
- exam_mark (INTEGER)

)

Relation modules_coursework(

- student_id (Foreign Key, TEXT)
- module_id (Foreign Key, TEXT)
- coursework (INTEGER)

)

3.5 Ex11

I did use an appropriate primary key while decomposing which are student_id and module_id

3.6 Ex12

Transitive dependencies: $module\_id \rightarrow building, room \rightarrow faculty$

3.7 Ex13

rooms_buildings(building, room, faculty); modules(module_id, module_name, module_leader, building, room)

4 Modeling

4.1 Ex14

student_id	student_name	student_email	programme	year	address	contact
INTEGER	TEXT	TEXT	TEXT	INTEGER	TEXT	TEXT

Table 1: Student

module_id	module_name	module_leader	building	room
TEXT	TEXT	TEXT	TEXT	TEXT

Table 2: Module

module_id	lecturer
TEXT	TEXT

Table 3: Lecturer

student_id	module_id	coursework
INTEGER	TEXT	INTEGER

Table 4: student_coursework

student_id	module_id	exam_mark
INTEGER	TEXT	INTEGER

Table 5: student_exam_mark

Student_id	Module_id
INTEGER	TEXT

Table 6: Enrolled

building	room	faculty
TEXT	TEXT	TEXT

Table 7: Building faculty

building	room	capacity
TEXT	TEXT	INTEGER

Table 8: Building capacity

4.2 Ex15

```
.mode csv
.separator ","
.import faculty.csv facultycsv
.import student.csv studentcsv
.ouput ex15.sql
.dump
```

4.3 Ex16

```
CREATE TABLE student(student_id INTEGER PRIMARY KEY NOT NULL UNIQUE,
student_name TEXT NOT NULL, student_email TEXT NOT NULL,
programme TEXT NOT NULL, year INTEGER NOT NULL, address TEXT,
contact TEXT);

INSERT OR IGNORE INTO student SELECT student_id, student_name,
student_email, programme, year, address, contact FROM studentcsv;
```

```
CREATE TABLE enrolled(student_id INTEGER PRIMARY KEY NOT NULL,
module_id TEXT NOT NULL);

INSERT OR IGNORE INTO student SELECT student_id,
module_id FROM studentcsv;

CREATE TABLE module(module_id TEXT PRIMARY KEY NOT NULL UNIQUE,
module_name TEXT NOT NULL UNIQUE, module_leader TEXT NOT NULL,
building TEXT, room TEXT);

INSERT OR IGNORE INTO module SELECT module_id,
module_name, module_leader, building, room FROM studentcsv;

CREATE TABLE lecturer
(module_id PRIMARY KEY NOT NULL, lecturer TEXT UNIQUE);

INSERT OR IGNORE INTO lecturer
SELECT module_id, lecturer1 AS lecturer FROM studentcsv
UNION ALL
SELECT module_id, lecturer2 AS lecturer FROM studentcsv;

CREATE TABLE student_exammark(student_id INTEGER NOT NULL,
module_id TEXT NOT NULL, exam_mark INTEGER);

INSERT OR IGNORE INTO student_exammark SELECT student_id, module_id,
exam_mark FROM studentcsv;
```

```
CREATE TABLE student_coursework(student_id INTEGER NOT NULL,
module_id TEXT NOT NULL, coursework INTEGER);

INSERT OR IGNORE INTO student_coursework
SELECT student_id, module_id, coursework1
AS coursework FROM studentcsv
UNION ALL
SELECT student_id, module_id, coursework2
AS coursework FROM studentcsv
UNION ALL
SELECT student_id, module_id, coursework3
AS coursework FROM studentcsv;

CREATE TABLE building_room(building TEXT NOT NULL,
room TEXT NOT NULL, faculty TEXT);
INSERT OR IGNORE INTO building_room SELECT building,
room, faculty FROM facultycsv;

CREATE TABLE building_capacity(building TEXT NOT NULL,
room TEXT NOT NULL, capacity INTEGER);

INSERT OR IGNORE INTO building_capacity SELECT
building, room, capacity FROM facultycsv;
```

5 Querying

5.1 Ex17

```
SELECT building, SUM(capacity) AS total_capacity
FROM building_capacity
GROUP BY building;
```

5.2 Ex18

```
SELECT student.student_id, student.student_name,
AVG(student_exammark.exam_mark) AS average_mark
FROM student
JOIN student_exammark ON
student.student_id = student_exammark.student_id
WHERE
student.year = 1 AND student.programme = 'Computer Science'
GROUP BY student.student_id, student.student_name
ORDER BY average_mark DESC;
```

5.3 Ex19

```
SELECT module.module_id, module.module_leader, building_room.faculty,
FROM module
JOIN building_room ON module.building = building_room.building
AND module.room = building_room.room
JOIN student_exammark
ON module.module_id = student_exammark.module_id
```

```
    GROUP BY
    module.module_id, module.module_leader, building_room.faculty
    HAVING AVG(student_exammark.exam_mark) = (
        SELECT MAX(average_mark)
        FROM (
            SELECT module.module_id, AVG(student_exammark.exam_mark)
            AS average_mark
            FROM module
            JOIN student_exammark
            ON module.module_id = student_exammark.module_id
            WHERE module.building = building_room.building
            AND module.room = building_room.room
            GROUP BY module.module_id
    )
);
```

5.4 Ex20

```
    SELECT module.module_id, building_capacity.room,
    building_capacity.building
    FROM module
    JOIN building_capacity
    ON module.room = building_capacity.room
    JOIN enrolled
    ON module.module_id = enrolled.module_id
    GROUP BY
    module.module_id, building_capacity.building, building_capacity.room
```

```
HAVING COUNT(enrolled.student_id) > building_capacity.capacity;
```