MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

DEPARTMENT
OF
COMPUTER ENGINEERING

CNG 351
Data Management and File Structures
Assignment 3

(6% of actual grade)

DUE DATE: 13 December, Tuesday, 23:55 (Cyprus Time).

PURPOSE

This assignment aims to help you revise the following two topics: Normalisation and also Relational Algebra. Therefore, this assignment has two main parts. In the first part, you are given logical designs of sample relations in a given database and you are asked to normalise them. In the second part, you are given a database and you are asked to answer some queries via relational algebra.

IMPORTANT RULES

1. Please make sure that your solutions are clearly explained.
2. Please make sure that your normalisation process is justified with relevant functional dependencies and explained clearly.
3. When you do normalisation, please show all your steps.
4. Create a PDF file for your solution, and upload only one PDF file to ODTUClass, one team member is enough to upload the solution.
5. Assignments will be completed by a team of two people that was formed for the previous assignment. if there is a problem with your partner, please let us know.
6. Please submit a report that includes a cover page with the team details including their names/surnames and also student IDs.

GRADING

This assignment has two parts and the overall grading will be as follows:

1. Part 1: Use Case 1: 20 points;
2. Part 1: Use Case 2: 25 points;
3. Part 2: 50 points;
4. Report: 5 points. A good report means type written, complete (every section fulfilled), with clear explanations in English (where relevant), and submitted via ODTUCLASS as one combined PDF document. The first page of the report must be a title page which should clearly state team details and assignment number. Each use case should also include Assumptions clearly written. The footer of all subsequent pages should be numbered in the format x of y (eg 2 of 6), etc.

PART 1: NORMALISATION

In this part, you are given two logical database designs and you are asked to normalise them. Please, check the following against the different normalization rules (i.e., 1NF, 2NF, 3NF, BCNF). Fix the schema if any of the normalization rules is violated.

Use Case 1 – MENTOR (Metu EveNT OrganiseR) (20 pts)

Event organisation is a complex task; many events occur at METU NCC. You have to keep track of the events, the organiser details, the sponsors, etc. To ease the process of maintaining such details, you received funding to create an application for METU NCC. The following relation is created by their database designer. You need to show the steps required to normalise this relation to 3NF.

Event(eventID, eventDate, eventTime, eventLocation, eventManagerSNN, eventOrganizerSNN, eventOrganizerPhoneNo, eventOrganizerName, eventOrganizerDOB, eventOrganizerGender, eventOrganizerMail, eventOrganizerAddress, crewSSN, crewName, crewRole, mainSponsorID, sponsorName, sponsorAmount, artistName, artistEmail, artistPrice.)

FD_1 :{eventID}– >{eventDate, eventTime, eventLocation}

FD_2 :{eventOrganizerSNN}– >{eventOrganizerName, eventOrganizerDOB, eventOrganizerPhoneNo, eventOrganizerGender, eventOrganizerMail, eventOrganizerAddress}

FD_3 :{mainSponsorID}– >{sponsorName, sponsorAmount}

FD_4 :{eventID}– >{eventManagerSSN, eventOrganiserSSN, mainSponsorID}

FD_5 :{crewSSN}– >{crewName, crewRole}

FD_6 :{artistEmail}– >{artistName, crewSSN}

FD_7 :{eventID, artistEmail}– >{artistPrice}

Use Case 2 – NCCCloud (25 pts)

NCCCloud is a recent established company aiming to provide cheap cloud gaming solutions in Northern Cyprus. For this purpose, you have been hired to design their database system for the company. Table **??** gives the data in one of the relations created. For this relation, you must check the data and create functional dependencies. Once you create functional dependencies, you need to normalise the relation according to 3NF and BCNF. Please clearly explain your functional dependencies first and

then show the steps required to normalise this relation. Please note the following about the attributes:

username: Every user in the system has a unique user name. With a user name, the system stores the user's date of birth and a subscription ID.

dob: Each user has a date of birth stored in the system. Date of birth is mainly in the format of month/day/year.

library_connection_token: A user connects to multiple libraries to play games and for each user and library, the system keeps track of a connectiontoken.

library_name: These are the game libraries where users play games. A user can play many games in many libraries.

game_id: A user can play multiple games stored in the game libraries. For each of these games, the database stores the name of the game which is uniquely identified with a game ID and library name.

game_name: This shows the name of the game that the user played.

subscription_id: In this database, every user has a subscription ID and this ID informs the subscription type so each subscription ID is associated with one type.

subscription_type: A subscription type can be low, mid or high. Each subscription ID's type is stored in the database. Each subscription ID can be associated to one type. Each subscription type has a specific price.

subscription_price: This attribute shows how much is paid for a subscription. Price is associated with the type. For example, Low has a specific price and other types as well.

session_start_date_time: A user can play many games but at a given session start date and time, they can only play a game hosted in a library. Therefore, the username and this session start date and time give us the game ID that the user is playing, the library name that the game stored and the id of the computer allocated for the session.

played_computer_id: This stores the details of the computer ID that is being used to play a game. Therefore, when a user plays at a given session date and time, it is known which machine is used for the game.

played_computer_GPU: This stores the GPU details of computers.

username	dob	library_connection_token	library_name	game_id	game_name	subscription_id	subscription_type	subscription_price	Session_start_date_time	played_computer_id	played_computer_GPU
nheinritz0	6/24/97	044e61f9d672	Steam	212	God of War: Ragnarok	bc86	Low	150	10/10/22 13:50	253	Nvdia GTX 1650
ssolan5	6/24/97	9f9d8e6173ea	Epic Games	222	Cyberpunk 2077	oe22	Low	150	11/8/22 18:36	255	Nvdia GTX 1060
mrzbaum1	3/6/90	074a94021595	Steam	212	God of War: Ragnarok	ty72	Mid	350	4/23/22 9:50	320	Nvdia RTX 3080
nheinritz0	6/24/97	881cd083a7c2	Epic Games	212	Assassin's Creed Valhalla	bc86	Low	150	10/18/22 21:38	253	Nvdia GTX 1650
mrzbaum1	3/6/90	074a94021595	Steam	277	Cyberpunk 2077	ty72	Mid	350	10/14/22 13:50	320	Nvdia RTX 3080
irevie3	5/22/98	1e89613783dd	Steam	561	The Witcher 3: Wild Hunt	hp43	High	450	4/23/22 9:50	325	Nvdia RTX 3090
ssolan5	6/24/97	baff2610bc5a	Steam	386	Don't Starve Together	oe22	Low	150	10/16/22 13:58	254	Nvdia GTX 1650
irevie3	5/22/98	1e89613783dd	Steam	458	We Were Here Together	hp43	High	450	11/13/22 22:10	403	Nvdia RTX 3090

PART 2 – RELATIONAL ALGEBRA

FindJob (50 pts)

FindJob is a social networking site that aims to allow people to connect to other professionals and find a job in North Cyprus. Users and employers can connect through FindJob's social network and build real-world professional relationships. FindJob aims to create groups, publish articles, publish job postings and allow users to create profiles and post photos, and more. Based on the requirements given in the previous assignment, a database was designed and mapped to a logical design as follows. You now need to write some **Relational Algebra** queries to address the following queries. Please note that this is not the full design of the requirements given in the previous step.

Relations

Member (member_id , name, surname, email, username, password, tel, gender, country, works_for_company [FK: Company: company_id])

Company (company_id, companyname, address, companytel)

ConnectionList (member_id [FK: Member: member_id], connection_member_id [FK: Member: member_id], status, connection_date)

Group (group_id, name, description, type, manager_Id[FK:Member: member_id], created_by [FK: Member: member_id], creation_date)

Post (post_id, date, content, title, posted_by[FK:Member: member_id])

GroupMembers (group_id [FK: Group: group_id], member_id [FK: Member: member_id], status, join_date)

Assessment (assessment_id, name, level)

AssessmentTaken (member_id [FK: Member: member_id, assessment_id [FK: Assessment:assessment_id], date, status)

Domains

Member.gender = Male, Female, NonBinary, Unknown
Date is stored in the system as "DD/MM/YYYY", e.g., 01/12/2022
Group.type = Standard, Unlisted
GroupMembers.status = Active, Passive
Assessment.level = Beginner, Expert
AssessmentTaken.status=Pass,Fail

Queries

1. List the id, name, surname and telephone number of all members in the system.
2. List the name and the date of creation of the groups that are “Unlisted” (type) and created by the member with the ID of 1333.
3. List the name, surname, and email address of the Members who created Standard groups after the 1st of January 2022.
4. We would like to display all the posts (listing their title and content) that are made by all the connections of the member called “Yeliz Yesilada”.
5. There is a company called“Curiosity” and this company would like to receive a report that shows the list of employee names (name/surname) that has created a group, the name of that group and also the manager name/surname of that group.
6. There is an assessment in the system named as “Introduction to Phyton”. We would like to list the name/surname of the members and how many times they took this assessment.