Contents

Project Task

The objective of this team project is for you to apply what you have learned in class to design and develop a database application using PostgreSQL. The project is to be done in teams of four students.

You are to develop a database application for a company to book meeting rooms. Unfortunately, with the current pandemic, there are restrictions on the number of people to be in the meeting room. This project consists of the following five tasks:

1. Design a ER data model for the application. Your ER model should capture as many of the application's constraints as possible.
2. Translate your ER data model into a relational database schema. Your relational schema should enforce as many of the application's constraints as possible.
3. Implement a SQL or PL/pgSQL function/procedure for each of the functionalities listed in Application Functionalities. You should implement appropriate triggers (if necessary) to enforce all the application's constraints. You may use any of the PostgreSQL's features and any SQL or PL/pgSQL constructs beyond what is covered in class.
4. Populate each table in your database with synthetic data. Each table should have at least 10 records. You may use online data generators (*e.g.*, Mockaroo, Generate Data) or write your own scripts to generate the synthetic data.
5. For each table R in your database schema that is not in 3NF, find a dependency-preserving BCNF decomposition of R if it exists; if not, find a 3NF decomposition of R.

Application

Your company is a very large company with many employees and many meeting rooms available. For simplicity, the company is located at a single building.

Unfortunately, with the current pandemic, the meeting rooms cannot be easily used and bookings must be done before any employee can come to office and use the meeting room. Due to interdependence of the components here, you are advised to read everything first before you start to design your ER diagram.

You may want to do the following:

1. Identify all the constraints as clearly as possible.
2. Satisfy as many constraints as possible using ER diagram but you may have to identify constraints that can not be enforced.
    - It's okay not to enforce everything as long as you are clear which are not enforced.
    - You may have to enforce this using triggers.
    - Since you have not learnt triggers yet, you should at least recognize the constraints that are not currently enforced by your ER diagram so that you can enforce it later.
3. Draw the ER diagram and simulate the possibilities of violating the constraints.

Note that the specifications are not formal to mimic a specification that is drafted by people who are not familiar with any database systems. As such, you are to fill in the informal or ambiguous specification with reasonable real-world constraints. Of course, in real-life, you have to ask for clarification about the project but in this project, you are given some flexibility to set your own constraints as long as they are reasonable.

Employee

Each employee is assigned a unique employee ID and a unique email address when they are employed. Additionally, their name and contact number must be recorded along with other employee data.

An employee must belong to a department which is identified by their department ID with their name recorded. There are three different kinds of employees:

1. **Junior**
2. **Senior**
3. **Manager**

Each employee must be one (*and only one*) of the three kinds of employees above. Currently, we do not allow *junior* employee to book any meeting rooms. They have to ask the *senior* employee or *manager* to book the room for them.

Managerial Duty

A manager have additional duties with respect to the meeting room and bookings.

1. Every booking must be approved by a manager from the same department.
2. Every meeting room's capacity can only be set by a manager to conform to social distancing measures.

Health Declaration

Every employee must do a daily health declaration. For simplicity, in the health declaration, they simply have to record their temperature. If the temperature is higher than 37.5 celsius, we say that they are having a fever.

In the case of a fever being detected, all the people who have been in contact with the employee (*i.e.,* in the same meeting room) will have to contacted. We will explain about contact tracing at a later part.

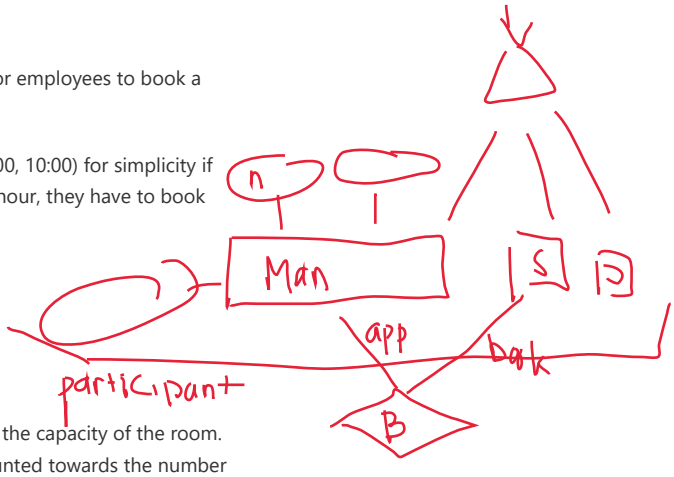
Meeting Rooms

The meeting rooms can be uniquely identified by their floor number and their room number. Unfortunately, due to historical accident, the name of the room may not be unique. This has caused many grievances among the employees but everybody knows that there are many rooms named the "Database Meeting Room" located on different floors.

Besides the name, each meeting room also has their maximum capacity. This capacity may be changed by a manager depending on government regulations.

Booking Procedure

A meeting room can be booked by either senior employees or managers. We do not allow junior employees to book a room.



The booking is based on 1-hour sessions (*e.g.,* from 09:00 (*inclusive*) to 10:00 (*exclusive*), or [09:00, 10:00) for simplicity if you are familiar with the notation). So if an employee needs to book the room for more than 1 hour, they have to book multiple sessions. For obvious reason, each meeting room can only be booked by one group.

The steps for booking is described as follows:

1. A senior employee or a manager books a room by specifying the room and the session.
    - If the room is not available for the given session, no booking can be done.
    - If the employee is having a fever, they cannot book any room.
2. They then add the participants (*i.e.,* employees) to the booked room up to a maximum of the capacity of the room.
    - The employee booking the room is also counted as a participant and should be counted towards the number of people in the room.
    - If any of the participants are having a fever, they cannot be added.
3. A manager from the same department approves the booking.
    - A manager may approve their own booking.
    - A booking that is not approved are immediately deleted to allow for other people to book the room.
    - Once approved, there should be no more changes in the participants and they will definitely come to the meeting room on the stipulated day.

Meeting Room Dynamics

When a meeting room has its capacity changed, any room booking *after the change date* with more participants (**including** the employee who made the booking) will automatically be removed. This is regardless of whether they are approved or not.

The date when the capacity is changed is assumed to be today but it will be part of the input.

Contact Tracing

Due to the pandemic, we have to be vigilant. If an employee is recorded to have a fever at a given day D, a few things must happen:

1. The employee is removed from all future meeting room booking, approved or not.

    - If the employee is the one booking the room, the booking is cancelled, approved or not.
    - This employee cannot book a room until they are no longer having fever.
2. All employees in the same *approved* meeting room from the past 3 (*i.e.,* from day D-3 to day D) days are contacted.
    - These employees are removed from future meeting in the next 7 days (*i.e.,* from day D to day D+7).
    - We say that these employees were in **close contact** with the employee having a fever.

These restrictions are based on the assumptions that once approved, the meeting will occur with all participants attending.

Resignation

When an employee resign, we still want to keep all the *past* records regarding this employee. Otherwise, the contact tracing may be compromised.

However, they are no longer allowed to book or approve any meetings rooms. Additionally, any *future* records (*e.g.,* future meetings) are removed.

Application Functionalities

Your application must support the following functionalities, each of which is to be implemented as a SQL or PL/pgSQL routine (*i.e.,* function or procedure). Some of these routines return a JSON value. PostgreSQL provides many useful functions for JSON data type (e.g., row to JSON, JSON aggregate).

Note that the specifications for the routines are incomplete by design; for example, the data types for the input parameters are not specified and certain parameters are not explicitly defined. For any criterion/process/design issue that is not explicitly stated, you are free to decide on how to address that issue in a reasonable way and justify your decisions in your project report.

Basic

These functionalities are required to populate any data and other administrative works.

1. `add_department`: This routine is used to add a new department. The inputs to the routine should minimally include:
    - Department ID
    - Department ame
2. `remove_department`: This routine is used to remove a department. The inputs to the routine should minimally include:
    - Department ID

   When a department is removed, we assume that all employees belonging to that department have been either (1) transferred to other department or (2) removed.
3. `add_room`: This routine is used to add a new room. The inputs to the routine should minimally include:
    - Floor number
    - Room number
    - Room name
    - Room capacity

   Note that for simplicity, we never remove a room.
4. `change_capacity`: This routine is used to change the capacity of the room. The inputs to the routine should minimally include:
    - Floor number
    - Room number
    - Capacity
    - Date

   The date is assumed to be today but is given as part of the input for simplicity.
5. `add_employee`: This routine is used to add a new employee. The inputs to the routine should minimally include:
    - Employee name
    - Employee contact number
    - Kind (junior, senior or manager)
    - Employee department

   The unique employee ID and email address are automatically generated by the system.
6. `remove_employee`: This routine is used to remove an employee. The inputs to the routine should minimally include:
    - Employee ID

- Date (*i.e.,* the last day of work, assumed to be in the past or today)

Note that all past records should be kept intact while all future records should be removed in accordance with the specification.

Core

These functionalities are related to the core functionality of the system.

1. `search_room`: This routine is used to search for available rooms. The inputs to the routine should minimally include:
   - Capacity
   - Date
   - Start hour
   - End hour

   The routine returns a table containing all meeting rooms that are available from the start hour (*inclusive*) to the end hour (*exclusive*). In other words, [start hour, end hour). Note that the number of hours may be greater than 1 hour and it must be available.

   The table returned should minimally include the following columns:
   - Floor number
   - Room number
   - Department ID
   - Capacity

   The table should be sorted in ascending order of capacity (*i.e.,* we do not want people to hog larger rooms first).
2. `book_room`: This routine is used to book a given room. The inputs to the routine should minimally include:
   - Floor number
   - Room number
   - Date
   - Start hour
   - End hour
   - Employee ID

   The employee ID is the ID of the employee that is booking the room. If the booking is allowed (*see the conditions necessary for this in Application*), the routine will process the booking for people to join and for approval.
3. `unbook_room`: This routine is used to remove booking of a given room. The inputs to the routine should minimally include:
   - Floor number
   - Room number
   - Date
   - Start hour
   - End hour
   - Employee ID

   The employee ID is the ID of the employee that is asking to remove the booking. If this is not the employee doing the booking, the employee is not allowed to remove booking (*the no sabotage rule*). If the booking is already approved, also remove the approval. If there are already employees joining the meeting, also remove them from the respective tables.
4. `join_meeting`: This routine is used to join a booked meeting room. The inputs to the routine should minimally include:
   - Floor number
   - Room number
   - Date
   - Start hour
   - End hour
   - Employee ID

   The employee ID is the ID of the employee that is joining the booked meeting room. If the employee is allowed to join (*see the conditions necessary for this in Application*), the routine will process the join. Since an approved meeting cannot have a change in participants, the employee is not allowed to join an approved meeting.
5. `leave_meeting`: This routine is used to leave a booked meeting room. The inputs to the routine should minimally include:
   - Floor number
   - Room number
   - Date
   - Start hour
   - End hour

- Employee ID

The employee ID is the ID of the employee that is asking to leave the meeting. If this employee is not the meeting in the first place, then do nothing. Otherwise, process the leave. Since an approved meeting cannot have a change in participants, the employee is not allowed to leave an approved meeting.

6. `approve_meeting`: This routine is used to approve a booking. The inputs to the routine should minimally include:
   - Floor number
   - Room number
   - Date
   - Start hour
   - End hour
   - Employee ID

   The employee ID is the ID of the manager that is approving the booking. If the approval is allowed (*see the conditions necessary for this in Application*), the routine will process the approval.

Health

These functionalities are related to the health related aspect of the system.

1. `declare_health`: This routine is used for daily declaration of temperature. The inputs to the routine should minimally include:
   - Employee ID
   - Date
   - Temperature

   There is no need to do contact tracing from this routine. But to be clear, the contact tracing can be done by attaching a trigger on insertion to the table related to this. Then the next routine can be used to process the contact tracing immediately.
2. `contact_tracing`: This routine is used for contact tracing. The inputs to the routine should minimally include:
   - Employee ID

   First, if the employee is not having a fever, then do nothing. Otherwise, perform contact tracing accordingly (*see the necessary steps for this in Application*). The routine returns a table containing all employee ID that are in **close contact** with the given employee ID.
   The table returned should minimally include the following columns:
   - Employee ID

Admin

These functionalities are related to the admnistrative aspect of the system.

1. `non_compliance`: This routine is used to find all employees that do not comply with the daily health declaration (*i.e.,* to snitch). The inputs to the routine should minimally include:
   - Start date
   - End date

   The routine returns a table containing all employee ID that do not declare their temperature at least once from the start date (*inclusive*) to the end date (*inclusive*). In other words, [start date, end date].
   The table returned should minimally include the following columns:
   - Employee ID
   - Number of days

   Number of days is the number of days the employee did not declare their temperature within the given period. The table should be sorted in descending order of number of days.
2. `view_booking_report`: This routine is to be used by employee to find all meeting rooms that are booked by the employee. The inputs to the routine should minimally include:
   - Start date
   - Employee ID

   The routine returns a table containing all meeting rooms that are booked by the given employee as well as its approval status from the given start date onwards.
   The table returned should minimally include the following columns:
   - Floor number
   - Room number
   - Date
   - Start hour
   - Is approved

The table should be sorted in ascending order of date and start hour.

3. `view_future_meeting`: This routine is to be used by employee to find all future meetings this employee is going to have that are already approved. The inputs to the routine should minimally include:
    - Start date
    - Employee ID

    The routine returns a table containing all meetings that are already approved for which this employee is joining from the given start date onwards. Note that the employee need not be the one booking this meeting room.
    The table returned should minimally include the following columns:
    - Floor number
    - Room number
    - Date
    - Start hour

    The table should be sorted in ascending order of date and start hour.
4. `view_manager_report`: This routine is to be used by manager to find all meeting rooms that require approval. The inputs to the routine should minimally include:
    - Start date
    - Employee ID

    If the employee ID does not belong to a manager, the routine returns an empty table. Otherwise, the routine returns a table containing all meeting that are booked but not yet approved from the given start date onwards. Note that the routine should only return all meeting in the room with the same department as the manager.
    The table returned should minimally include the following columns:
    - Floor number
    - Room number
    - Date
    - Start hour
    - Employee ID

    The table should be sorted in ascending order of date and start hour.

Project Deadlines & Deliverables

There are two deadlines for this project:

1. **Week 6**: Saturday, 18 September 2021 (*23:59*)
2. **Week 12**: Saturday, 6 November 2021 (*23:59*)

The project evaluation will be conducted during week 13 (Monday, 8 November 2021 to Friday, 12 November 2021 but possibly may go into Saturday, 12 November 2021). Each team will be allocated 15-20 minutes to demo the project and to answer questions from a project evaluator. The registration for the project evaluation slots will be announced later.

ER Data Model Submissions (5 marks)

The deliverable is due on **Week 6** Saturday, Saturday, 18 September 2021 (*23:59*). Note that the time is 23:59 and it is 23:59:00 since that's the latest time we can put into LumiNUS.

Each team is to upload a pdf file named `teamNN.pdf`, where `NN` is the team number to the LumiNUS file folder belonging to their team.

The submitted PDF file must be at most 4 pages and consists of the following contents:

1. Project team number & names of team members (*on the first page*)
2. ER data model for the application. If your ER model is too large (*i.e.,* spanning more than one page), you may want to include a single-page simplified ER model (*with non-key attributes omitted*) before presenting the detailed ER model.
3. Justification for any non-trivial design decisions made.
4. List down 5 of the application's constraints that are not captured by the proposed ER data model.
    - If there aren't any, we will assume that all constraints are captured.
    - The grading will then be based on this assumption.

For late submissions, one mark will be deducted for each late day up to two late days. Submissions after the second late day will receive zero marks and will not be graded.

We will publish a suggested ER data model on LumiNUS. Your team may consider using the published ER data model or a modified variant of it for your project.

Project Report

Each team is required to submit a project report as part of the project deliverables (due on **Week 12** Saturday, 6 November 2021 (*23:59*)).

The project report (*up to a maximum of 20 pages **including appendix** in pdf format with at least 10-point font size*) should include the following contents:

1. Names and student numbers of all team members and project team number (*on the first page*).
2. A listing of the project responsibilities of each team member.
    - Team members who did not contribute a fair share of the project work will not receive the same marks awarded to the team.
3. The ER data model for your application.
    - If your ER model is too large (*i.e.,* spanning more than one page), you may want to include a single-page simplified ER model (*with non-key attributes omitted*) before presenting the detailed ER model.
    - Provide justification for any non-trivial design decisions made in your ER model.
    - List down 5 of the application's constraints that are not captured by the proposed ER date model.
4. Relational database schema.
    - Provide justification for any non-trivial design decisions made in your relational database schema.
    - List down 5 of the application's constraints that are not captured by the proposed relational schema.
        - *i.e.,* the constraints that are to be enforced using triggers.
5. A description of the three most interesting triggers implemented for the application.
    - Provide the name of the trigger.
    - Explain the usage of the trigger.
    - Justify the design of the trigger implementation.
6. An analysis of normal forms of your relational database schema (*not your ER data model*).
    - For each table R in your database that is not in 3NF:
        - Find a dependency-preserving BCNF decomposition of R if it exists.
        - Otherwise, find a 3NF decomposition of R.
7. A reflection.
    - Summary of any difficulties encountered and lessons learned from the project.

Project Submission (23 marks)

This deliverable is due on **Week 12** Saturday, 6 November 2021 (*23:59*).

- Submit your project deliverables by creating a **local** directory (*i.e.,* in your computer) named `teamNN` (*where `NN` is your team number*) with the following four files:
    1. `report.pdf`: Project report in PDF format.
    2. `schema.sql`: SQL commands to create your application's database schema.
    3. `data.sql`: SQL commands to load data into your application's database.
    4. `proc.sql`: SQL or PL/pgSQL routines of your implementation.
- It is important that your submission files are named as indicated above and that you have tested your code using `psql`.
    - Marks will be deducted if any of your file cannot be run on `psql` in the correct order (*i.e.,* `schema.sql` followed by `data.sql` followed by `proc.sql`).
- Compress the directory into a zip file named `teamNN.zip`.
- Upload the zip file into LumiNUS file folder for your team.

For late submissions, 5 marks will be deducted for each late day up to two late days. Project submitted after the second late day will receive zero marks and will not be graded.

By CS2102 Teaching Team