CS2102 Project Report

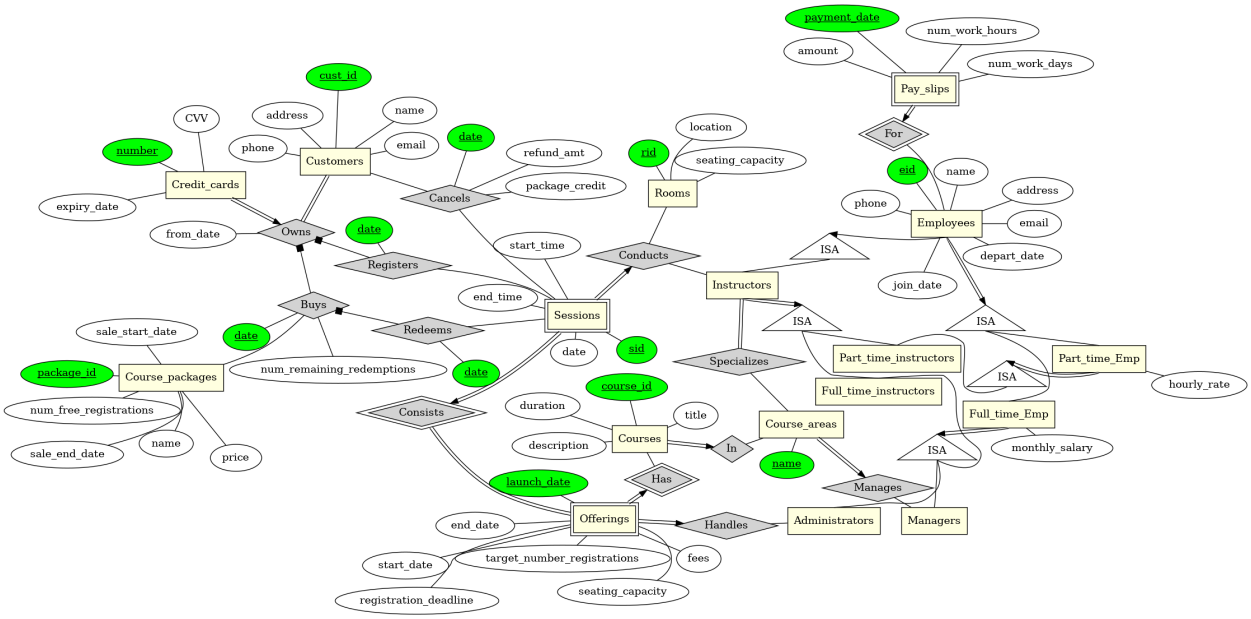
Project Team 72
Lum Wei Boon A0182862X
Zhang Yuanyu A0187219W
Ng Weng Fai A0199890E
Yap Dian Hao A0184679H

Content Page

Project Responsibilities

Project Responsibilities	
Zhang Yuanyu	- Set up team repository with Docker and pgAdmin for easier testing and collaboration for project - Task allocations and goal settings - Handling of functions 9-16 as stipulated in the project requirements - Handling certain triggers - Testing
Lum Wei Boon	- Handling of functions 17-24 as stipulated in the project requirements - Handling certain triggers - Handle tasks related to Offerings, Registrations and Sessions - Testing
Yap Dian Hao	- Handling of functions 1-8, 28 as stipulated in the project requirements - Handling certain triggers - Testing
Ng Weng Fai	- Handling more difficult functions 25-30 as stipulated in the project requirements - Handling certain triggers - Testing

ER Model Justifications and Constraints



Our team has decided to use the suggested ER model provided by the teaching team.

Constraints not captured by ER model

1. The ER diagram is unable to capture the application constraint of registration deadline for a course offering having to be at least 10 days before its start date. This is being captured in the relational schema by the use of a check at schema level.

2. The ER diagram is unable to capture the application constraint of the seating capacity of a course session having the seating capacity of the room where the session is conducted. This is being captured at schema level and the use of triggers.

3. The ER diagram is unable to capture the application constraint of each session having the need to start at or after 9pm, and end at or before 6pm, with no sessions conducted between 12pm to 2pm. This is being captured by the use of a check at schema level.

4. The ER diagram is unable to capture the implied constraint of the date of redemption to be on the same or later date than the date of purchase of course package. This is being captured by the use of a check at schema level.

5. The ER diagram is unable to capture the implied constraint of the course package sale start date to be on the same or later date than the same cours package sale end date. This is being captured by the use of a check at schema level.

Relational Database Schema

Justifications for non-trivial design decisions

Courses duration cannot be more than 4 hours

1. We assumed that every session duration is the same as the course duration it is under.

2. It is stated in the project requirements that a session must start at or after 9am, and end at or before 6pm. Furthermore, no sessions should be conducted between 12pm and 2pm.

3. This means that the course duration cannot exceed 4 hours (since there are 3 hours between 9am to 12pm, and 4 hours between 2pm to 6pm).

On delete cascade on different tables referencing Employees

1. It is stated in the project requirements that an Employee is either full-time or part-time. Furthermore, full-time employees could be classified as a manager, administrator or full-time instructor, while part-time employees are part-time instructors.

2. This means that whenever an employee gets removed from the database, it should cease to exist in other tables referencing it as well.

Model For relationship In Pay-slips Schema

1. We chose to model the weak entity relationship inside Pay_slips schema by setting the employee unique identifier as an attribute in Pay_slips schema, and adding a *NOT NULL* and *ON DELETE CASCADE* foreign key constraint on that attribute, instead of creating a For schema.
2. This ensures that the total participation is enforced as each entry to the Pay_slips schema depends on whether the corresponding employee exists.
3. If we were to include a For schema, an additional trigger has to be implemented to check if every payslip in the Pay_slips table participates in the Employees table.
4. This opposes the For table as a standalone table as total participation constraint should be done on schema level, without the use of triggers.

Model Specializes relationship in Instructors Schema

1. Since Instructors has a total participation constraint (at least one) with the Specializes table, we can either implement it as a separate table or by passing its attributes into the Instructors table.
2. We chose to model the entire relationship in the Instructors schema as this would force each insertion into the Specialization table with simple NOT NULL constraints in the

table schema. This is much easier than using a trigger on the Specialization table to enforce this constraint.

3. Hence our final implementation uses a primary key of (iid, area_name) to enforce the required constraint.

Constraints

1. The relational schema is unable to capture the application constraint of having no two sessions for the same course offering being conducted on the same day and at the same time”. This is captured by the use of a trigger.

2. The relational schema is unable to capture the application constraint of instructors having the need to specialise in the course area before teaching a session from the same course area. This is captured by the use of a trigger.

3. The relational schema is unable to capture the application constraint of customers being able to register for at most one of its sessions before its registration deadline. This is captured by the use of a trigger.

4. The relational schema is unable to capture the application constraint of each customer being able to only have at most one active or partially active package. This is captured by the use of a trigger.

5. The relational schema is unable to capture the application constraint of seating capacity of a course offering to be equal to the sum of the seating capacities of its sessions. This is captured by the use of a trigger.

Triggers

Trigger name:

```
before_insert_buys
```

Usage of trigger:
This trigger is used to enforce the following 2 constraints:

(a) Customers can have at most one active or partially active package.
(b) Customer cannot buy a package outside its selling window

Justification of trigger implementation design:
To enforce the above constraints, the trigger has to be fired whenever there are changes (INSERT) to the Buys table.

This trigger will first check if the buying date lies within the start selling date and end selling date of the course package. If yes, the trigger will loop through each record in the Buys and Redeems Table, and if the customer has any active or partially active package remaining, the buying operation is voided.

Trigger name:

```
before_insert_update_conducts_instrcutor_consec_trigger
```

Usage of trigger:
This trigger is used to prevent instructors from being assigned to conduct back-to-back sessions, and there must be at least an hour of break in between any two sessions the instructor is teaching in a day.
Justification of trigger design:
This trigger is fired whenever there are changes (INSERT / UPDATE) to the Conducts table.

The trigger will loop through each record in the Conducts table, and if there exists a session that shares the same ending time as the new session that is going to be inserted, the INSERT / UPDATE operation will be voided.

Trigger name:

```
before_insert_update_conducts_part_time_30_limit_trigger
```

Usage of trigger:
This trigger is used to prevent part time instructors from being further assigned to teach any courses if their teaching hours in a month exceeds 30.
Justification of trigger design:
This trigger is fired whenever there are changes (INSERT / UPDATE) to the Conducts table.

This trigger will loop through each record in the Conducts table, and if the record that is going to be inserted involves a part time instructor, the aggregate `SUM` function will be used to calculate the part time employee's total teaching hours for the month. If his/her teaching hours, including

the new session duration he is trying to teach for the month, exceeds 30 hours, the INSERT/UPDATE operation will be rejected.

Difficulties Encountered

1. Difficulty on testing on stu's server: Initially our team started by using the team's PSQL path on the school's server, but we soon discovered that it is difficult to collaborate and carry out testing on the school's server. Therefore, we had to run the PSQL on Docker and use PGAdmin to visualize a large number of tables.
2. Trigger dependencies: Different group members may work on different functions and different triggers, and the schemas used in the functions may cause valid operations to be erroneously voided, while members may assume that the bugs exist in the functions when the actual bugs are in the triggers which are implemented by another group member.
3. Multi-fold regression. While developing the functions required of the project, we realised some of our schema design decisions were suboptimal and made implementation of the required functions difficult. In other cases, there were simply erroneous implementation of the schema due to an incorrect understanding of the ER diagram. As such we have to revise the schema multiple times throughout the project causing regression on the previous functions, triggers and procedures we have designed. Overall this slows down our development speed and forces us to do regression testing on functions that have previously "passed" testing.

Lessons Learned

1. Application of lecture notes' contents: After doing the project, we applied PSQL's syntax learned in class including basic and advanced functions such as COALESCE, NATURAL JOIN, CTE, and also procedural PSQL syntax such as FUNCTION, PROCEDURE, and TRIGGERs.
2. Knowledge in advanced PSQL: We had the opportunity to learn advanced PSQL syntax such as arrays, year and month extraction from dates, for loops, and data type conversions. We learned advanced PSQL syntax from the official documentation page, and examples from websites like StackOverflow.
3. ER model: We learned to design an application given its constraints and create an ER model that captures various key and total participation constraints and ISA hierarchies. We learned the importance of spending time designing our ER model before trying to implement our schema in PSQL. This will save us from multiple revisions of both the schema and the ER model.