CS2102 Project

Team members:
Long Zeling, Hazel Tan, Sharmaine Cheong, Ng Jia Xin
Team Number: 66

Project responsibilities

All members participated in the designing of ER Model and Schema. Equal work was also done by each member in populating the database.

For functions, every week, each member would take about 3-4 functions to attempt. We would then come together as a team to discuss and delegate testing.

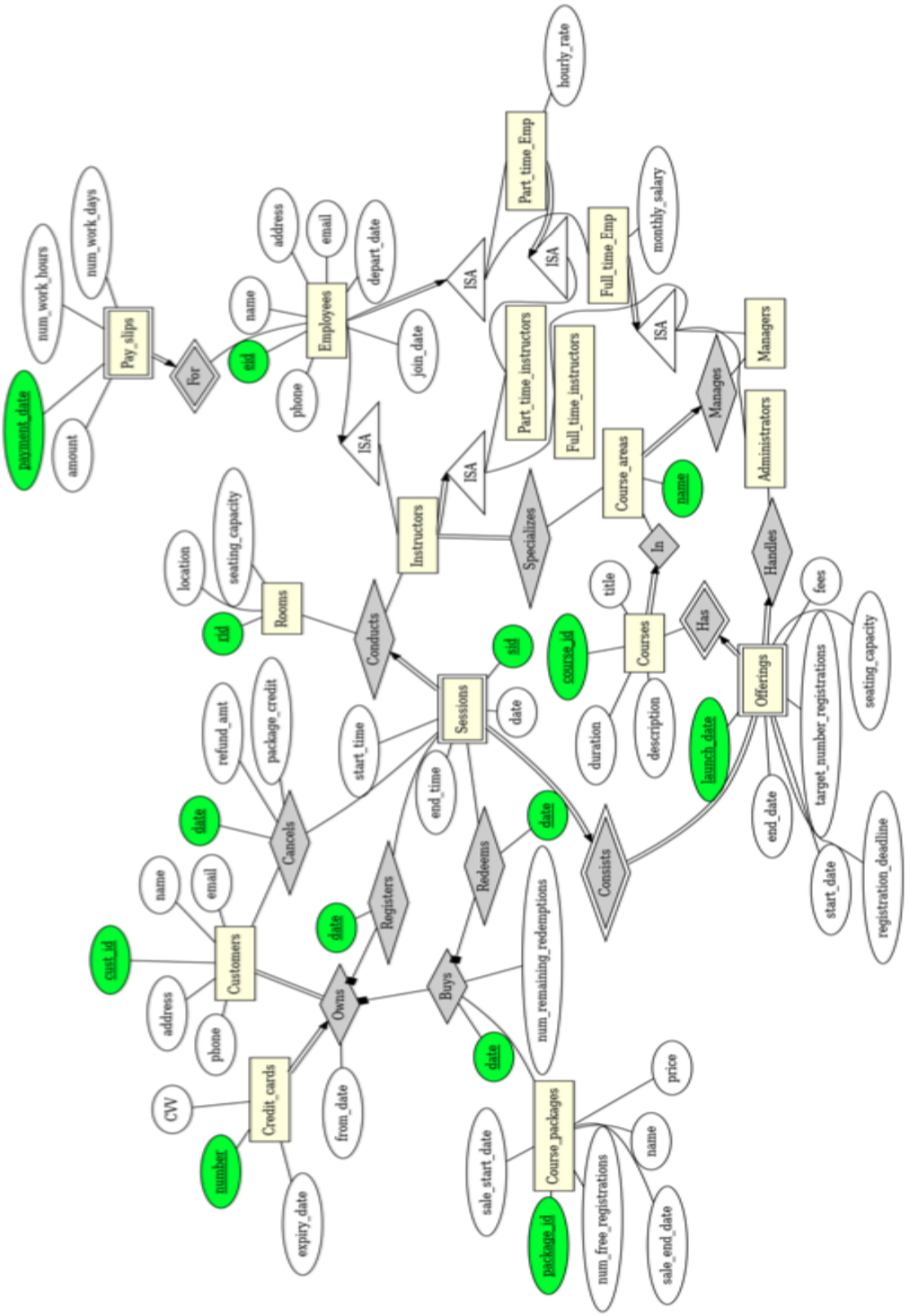
Here is the full table of functions implemented and tested by each member. Functions in brackets indicate that the member attempted the function, but was not involved in implementing and testing.

Zeling	Hazel	Jia Xin	Sharmaine
1 5 6 7 10 11 14 17 19 22 24 (26)	(1) 7 8 9 11 14 15 16 (17) 18 22 24 26 28 30	(1) 2 3 6 12 (13) 14 (15) (21) 22 (23) 25 27	4 7 (8) (11) 13 (18) 20 21 23 29

For triggers, we worked as pairs to discuss and implement.

Jia Xin & Sharmaine	Offerings, Sessions Triggers
Zeling & Hazel	All Employees, Registers, Buys Triggers

ER Data Model



Justification for non-trivial design decisions for ER model based on changes made from previous submitted ER model

Decision: Included a pay_slip table with a pay slip identifier.
Justification: The concept of having pay slip as a transaction to pay salary is clearer than simply having salary for each employee. This way, we are able to have a pay slip identifier for every salary payment.

Decision: Monthly salary attribute is now attached to both full-time and part-time employees.
Justification: Monthly salary is a common attribute.

Decision: Include full-time instructor and part-time instructor tables.
Justification: We thought that it could be easier for us to reference to part time and full time instructors separately when we query from our database. We also decided to make employees ISA-relationship with full-time and part-time employees instead as it was more intuitive and previously, we did not know we could make instructor reference to both full-time and part-time tables.

Decision: Removal of Manages table.
Justification: Having an employee identifier in the offerings table can already capture the concept of a manager managing a particular offering.

Decision: We replaced credit card number as an attribute of payment and redemptions.
Justification: Redemptions and credit cards table cannot have primary keys. Subclass cannot have its own key and has to get from superclass of payment instead.

Decision: Total participation constraint between sessions and rooms.
Justification: Session cannot use multiple lecture rooms.

Decision: Removal of seating capacity as key from rooms.
Justification: Room identifier is sufficient to differentiate between rooms. Additional primary keys can cause problems when room identifiers are the same but seating capacity is different.

Application constraints not captured by ER model

1. We cannot capture the restrictions of the timing of sessions (earliest at 9am, no sessions from 12-2 and latest session ends at 6pm).
2. Each customer can have at most one active or partially active package.
3. An instructor is conducting a session under a course in the correct course area as his specialisation
4. Each part-time instructor must not teach more than 30 hours for each month.
5. Each course offering is managed by the manager of that course area (however, it can be enforced in the schema)

Justification for non-trivial design for schema

Decision: We decided not to add “offering_id” as the course offering identifier.
Justification: “course_id” and “launch_date” is sufficient to identify unique tuple

Decision: Combined “Manage” with “Course_areas”.
Justification: While it would be intuitive to have a “Manage” relation that shows which area is managed by which administrator, it is not necessary as each area must be managed by an administrator. Each area in the “Course_areas” table is unique and the “not null” constraint of eid will make sure each course area is managed by one manager. This can reduce duplicate data in our database.

Decision: Remove the foreign key reference to “Sessions” table on “Cancels” table
Justification: If the session is removed, we still need to keep track of the number of cancels and foreign key references will prevent us from doing so. As such, after removing the foregin key references, we can remove the session from the “Sessions” table and not have any impact on the “Cancels” table.

Application Constraints

The Sessions Relation has several constraints that we sought to address in our trigger. This would include the timing which Sessions are conducted (9am to 6pm, with no sessions between 12-2pm). We also needed to check the constraints of the instructors. This would ensure that no instructors teach multiple sessions concurrently and have a 1 hour break in between. We also needed to address the gap where each part-time instructor must not have taught more than 30 hours per month and specializes in the same area which the course is in.

We also needed to check the Offerings Relation as the current schema is not able to check that the registration deadline is 10 days before the start date and that the offerings had sessions (e.g. we need to prevent an offering with no sessions). We also could not guarantee that the seating capacity of a course offering is equal to the sum of the seating capacity of its sessions.

The current schema was also unable to guarantee that each customer only had one active or partially active package. It could not prevent insertion statements that violated this constraint.

The current schema also could not enforce the ISA Relationship between different classifications of employees. For example, it is possible for an eid to exist in Managers, but not in Employees, which should not be the case.

Triggers

check_employee and similar functions for other relations

Every relation that involves Employees (Employees, Full_Time_Emp, Part_Time_Emp, Managers, Administrators, Instructors, Full_Time_Instructors, Part_Time_Instructors) have a similar trigger.

The purpose of the trigger is to:

1. Ensure that isolated insertion statements will fail while allowing it to work for our function add_employees
2. Ensure that the eid exists in the correct corresponding table

Justification of design:
We decided to make the check_employee trigger deferrable. The main consideration was for our function, add_employee, to work. We concluded that we wanted to first add the eid into the Employees Table. Following that, depending on the information provided in the function, it will be added to the corresponding tables. For example, if the employee being added is a part-time instructor, we need to check that:

- Eid exists in Employees, Instructors and Part_Time_Instructors
- Eid does not exist in Managers, Administrators and Full_Time_Instructors

After the function has finished only then should we check the foreign key constraints. With this in mind, we decided to make this trigger (and check_instructors) deferrable.

The remaining relations have triggers that check for similar conditions but are not deferrable. This is because we felt that there was no need to defer it, since it is the last in the hierarchy to be updated or inserted into.

check_session

The purpose of the trigger is to:

1. Check that no 2 session for the same course offering are conducted on the same day and same time
2. Check that an instructor has an hour break before and after conducting a session
3. Check that the total hours taught by the instructor is less than 30 if part-time
4. No instructor is teaching another course at the same time that the new session is being conducted
5. Allow changes of room id to a different room id, as long as the room is available

Justification of design:
There are more than 1 function that would require adding new sessions for instructors or when updating instructors, we need to check if that instructor can teach the session. Since we need to check for whether an instructor can teach another session multiple times in many functions, we thought that it is more feasible to check as a trigger instead.

Due to the lack of time, we are unable to prevent any independent insert statements from the users.

check_cust_register

The purpose of this trigger is to:

1. Check that the customer does not attempt to register for a course which deadline is already over
2. Check that the customer does not attempt to register for a course that is already full

Justification for the design:
We chose to make this a BEFORE trigger because it would not make sense to allow for insertions or updates to pass. We also raise notices to give users more clarity on the error that they are encountering.

We chose to enforce the constraint that each customer can only register for one course offering for the same course in the schema by making the tuple “course_id”, “launch_date”, “cust_id” unique.

We also chose to check the offering that registered by the customer exists in the “offerings” table before we insert.

Summary

Difficulties encountered	Lessons learned
We realise there is cyclic dependency in check_offering trigger as each offering must have at least 1 session and session reference offering for foreign keys.	We made the check_offering trigger deferrable so that when we add an offering through the add_offering function, we make sure the corresponding sessions are inserted before the trigger is invoked.
Question 7 is very complicated. We think it would be helpful to have a 'table variable' to store some intermediate results.	We found out that we could create a temporary table to store the results and drop the table afterwards. Else, we could also use CREATE OR REPLACE VIEW for other questions.
Dealing with sql datetime type is hard.	We read the documentations carefully and found out that there are many helper functions. We also decided to make all the input text and parse it into a date within the function to overcome datetime styling issues.
Some functions are complicated and hard to test.	Breaking down the function and testing it part by part is helpful in debugging and proper testing.
We couldn't figure out how to get the eid of the inserted tuple for the add_employee function.	We solved this by using "returning" which is especially useful for returning serial id
The employee id is consumed even if the add_employee function call fails.	Move the insertion statements after all checks so that eid is not wasted. However, when foreign key constraint is violated and transaction is aborted, the eid wasted and cannot be helped.
Using input parameter to generate N number of rows for the view_summary_report	We realised that it is not possible to directly input parameter values as part of our query variable. Instead of using the values directly, we had to workaround it by inserting the value in a table which is then extracted to be used in the generate_series function.
Previously, we added sessions before offerings and we checked that if the newly added session falls in between the start date and end date of the offerings. However, it gave us problems when we tried to implement the add_session function.	We removed the checking for within the start date and end date. Instead, when we add a session we update the offering at the same time. This makes the logic more reasonable. A possible use case is that we think that if the course offering is overwhelmingly popular, it is okay to extend the end date of the course offering instead of adding a session in between.
Triggers are tricky. Sometimes, it is hard to differentiate update statements from insert statements. For update_room, we had some problems with session triggers because the same session is being updated and our trigger takes it as an overlap and it would not allow us to update.	We decided to compare OLD and NEW which is specially for UPDATE trigger(the session trigger is an insert/update trigger). This solved our problem.
The function remove_employee checks if the employee can be removed upon fulfilling the	We could have designed the database differently, should not remove instructor from

conditions to leave. For the removal of instructors, although function specification says we would allow the removal of instructors if the departure date is after all of the session start dates, removing instructors from the instructors table would violate foreign key constraints in sessions.	the instructors table even if he/she has departed. Another way is to raise notices with reasons to users so that they can understand why the instructor specified was not removed. I.e. not removed as he is still assigned to some sessions or we could not find other available instructors to replace him.
Commutative law for OR and AND operators can mess up the checking of truth and false.	Should always be specific by putting brackets.