Khoury Course Registration Web Application

CS5200

Professor Lee

Group 8: Yingjie Huang, Chengcheng Lyu, Wanqing Guo

This project is a web application using Python aiming for three user types: students at SV campus can register courses thru the system, advisors can approve/modify any student’s registration, and admins can view all students’ registration and modify any course information

Phase 1

Problem Statement:

“Khoury Course Registration Web Application” can support web-based course registration at SV campus. There will be three user types in the system with different accessibilities, and each user will need to sign up in order to log in the system and modify profile information.

Student user:

- Register a course by course id or course name
- Add/drop a course

Advisor user:

- View/modify courses registration for each student
- Approve/decline student’s registration
- Send a message to students regarding to any issue on the registration

Admin user:

- Manage the entire system, having access to any data
- Add/modify/remove a course
- Assign a classroom to a course
- View statistics of registration info

Queries:

- Find the total number of students registered per department.
- Find the total number of students registered per school.
- Find the total number of students and all students’ ids registered in a particular course, given the course id/name.
- Find how many courses a student has registered so far given the student id/name.
- Find the total credits a student has registered so far given the student id/name.
- Find the course a student registered by course name
- Find the course a student registered by course id

- Find all the courses assigned to a specific classroom
- Find all the courses during a time period
- Find all the courses taught by a professor
- Find the total number of remote courses
- Find the current available seats for a course
- Find the total number of students in the waitlist for a course
- Find the total number of seats assigned to a course
- Final the waitlist capacity for a course

Assumptions:

1. Tuition for each course and payments will be done on a third party website.
2. Students cannot take more than 3 courses per semester.
3. The minimum credits a student should take per semester are 8, and the maximum credits are 12.
4. A student can only belong to one department and one school.
5. The website will only show 15 courses for presentation purposes. In deployment, the website should include all the courses available.
6. Student will not register same course twice

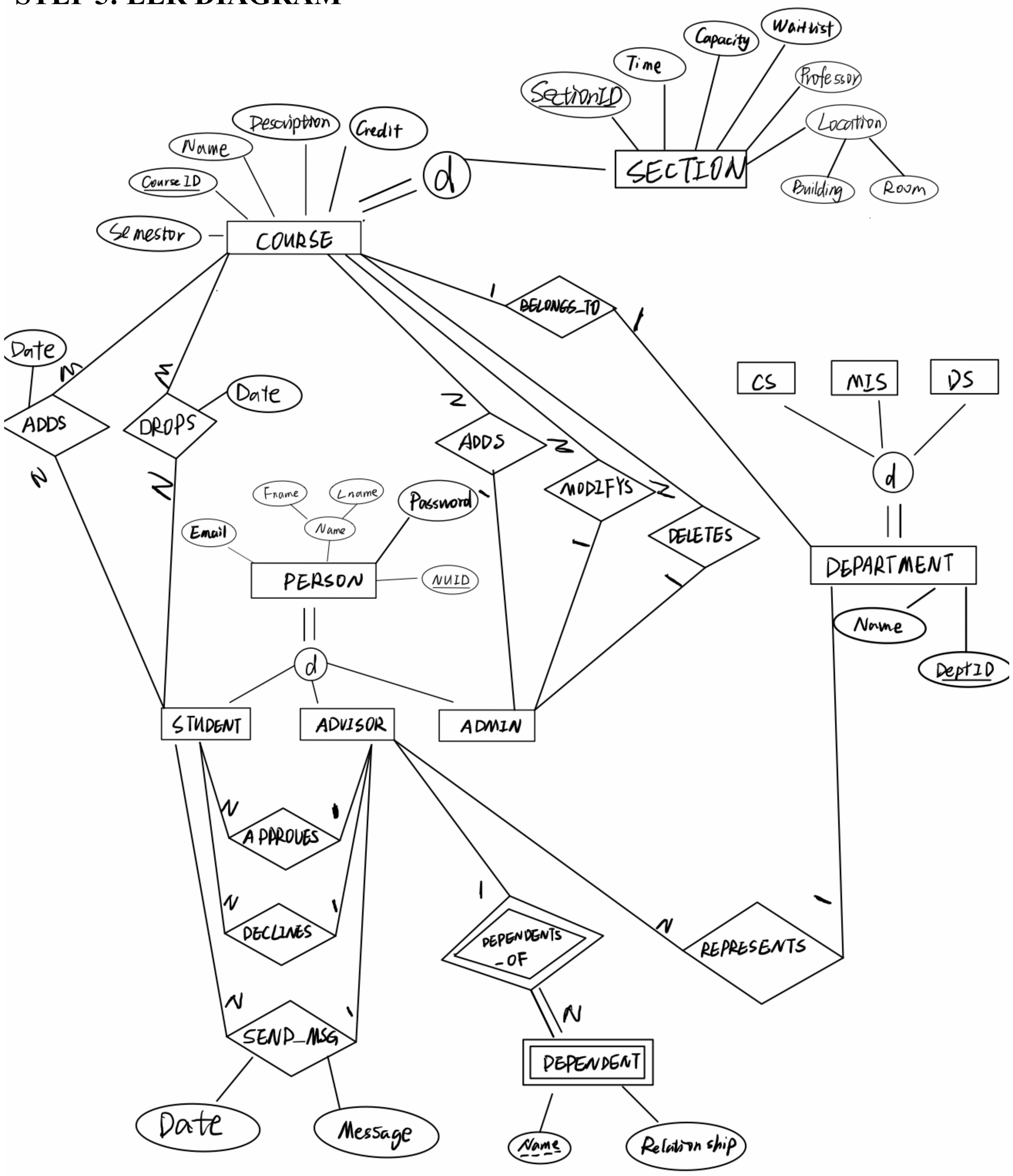
CONCLUSION

In this project, we are designing the web application for three types of users, where they have different relationships with each other. In this project we firstly figure out the required subjects in this application and listed queries that are useful to the ones using the data. In this step we learned how to figure out what we need, and what operations we require in our application. This helps us to think more logically.

Then we worked on the properties of each subject and logic of relationship with others. And we used the EER diagram to represent the relationship of the subjects we need in this project. We created relational models based on EER diagrams. And formulated all the queries with relational algebra. In this step we learned how to use EER to create detailed databases with a smart and efficient technique. And visualize our outlook of the database. And we learned to use relational algebra to get the relational databases.

To get better project output, we also need to figure out the change by operations between different subjects.

STEP 5: EER DIAGRAM



Note: Weak entity is added on purpose.

STEP 6: RELATIONAL SCHEMA DESIGN

COURSE

CourseID	Name	Description	Credit	Semester	Courseno	SectionID	Time	Capacity	Waitlist	Professor	Building	Room	dno	Days

DEPARTMENT

DeptID	Name

CS_DEPARTMENT

DeptID

MIS_DEPARTMENT

DeptID

DS_DEPARTMENT

DeptID

ADDS

SNUID	CourseID	Date

DROPS

SNUID	CourseID	Date

STUDENT

NUID	FName	LName	Email	Password	dno

ADVISOR

NUID	FName	LName	Email	Password	dno

ADMIN

NUID	FName	LName	Email	Password	dno

SEND_MSG

SNUID	ANUID	Date	Message	Status

APPROVE

SNUID	ANUID

DECLINE

SNUID	ANUID

NOTIFICATION

ADVID	SID	Date

REQUEST

RID	COURSEID	NUID	DATES	TYPE	STATUS

ADVISE

AID	ADVID	S_ID

STEP 7: RELATIONAL ALGEBRA FOR QUERIES

1. Find the total number of students registered per department
   RA:
   REGISTER ← ADDS – DROPS
   REG_DEPT ← REGISTER $\bowtie_{\text{CID = CID}}$ COURSE
   RESULT ← $\pi_{\text{dno, COUND\_SNUID}}$ ($\sigma_{\text{dno = given\_department\_id}}$ ($_{\text{DepID}}\mathcal{F}_{\text{COUNT SNUID}}$ (REG_DEPT)))

2. Find the total number of students and all students' ids registered in a particular course, given the course id/name.
   RA:
   $\pi_{\text{SNUID, COUNT\_SNUID}}$ ($\sigma_{\text{CID = given\_course\_id}}$ ($_{\text{CID}}\mathcal{F}_{\text{COUNT SNUID}}$ (ADDS)))

3. Find how many courses a student has registered so far given the student id.
   RA:
   ADD_COURSES ← $_{\text{SNUID}}\mathcal{F}_{\text{COUNT CID}}$ (ADDS)
   DROP_COURSES ← $_{\text{SNUID}}\mathcal{F}_{\text{COUNT CID}}$ (DROPS)
   REGISTER ← ADD_COURSES $\bowtie_{\text{SNUID = SNUID}}$ DROP_COURSES
   RESULT ← $\pi_{\text{(COUNT\_CID\_ADD – COUNT\_CID\_DROP)}}$ ($\sigma_{\text{SNUID = given\_student\_id}}$ (REGISTER))

4. Find the total credits a student has registered so far given the student id.
   RA:
   REGISTER ← ADDS – DROPS
   REG_CREDIT ← REGISTER$\bowtie_{\text{CID = CourseID}}$ COURSE
   RESULT ← $\pi_{\text{SNUID, SUM\_CREDIT}}$ ($\sigma_{\text{SNUID = given\_student\_id}}$ ($_{\text{SNUID}}\mathcal{F}_{\text{SUM Credit}}$ (REG_CREDIT)))

5. Find the course a student dropped by course id
   RA:
   $\sigma_{\text{CID = given\_course\_id AND SNUID = given\_student\_id}}$ (DROPS)

6. Find the course a student registered by course id
   RA:
   $\sigma_{\text{CID = given\_course\_id AND SNUID = given\_student\_id}}$ (ADDS)

7. Find all the courses assigned to a specific classroom
   RA:
   π_{CID} ($\sigma_{\text{room = given\_room}}$ (COURSE))

8. Find all the courses during a time period
   RA:
   π_{CID} ($\sigma_{\text{time= given\_time}}$ (COURSE))

9. Find all the courses taught by a professor
   RA:
   π_{CID} ($\sigma_{\text{professor = given\_name}}$ (COURSE))

10. Find all remote courses
    RA:
    $\pi_{\text{CID}} (\sigma_{\text{building = 'online'}} (\text{COURSE}))$

11. Find the current available seats for a course
    RA:
    $\text{TOTAL\_CAPACITY} \leftarrow {}_{\text{CourseID}} \mathcal{F}_{\text{SUM Capacity}} (\text{COURSE})$
    $\text{TOTAL\_ADD} \leftarrow {}_{\text{CID}} \mathcal{F}_{\text{COUNT SNUID}} (\text{ADDS})$
    $\text{TOTAL\_DROP} \leftarrow {}_{\text{CID}} \mathcal{F}_{\text{COUNT SNUID}} (\text{DROPS})$
    $\text{TOTAL\_REGISTER} \leftarrow \text{TOTAL\_ADD} \bowtie_{\text{CID = CID}} \text{TOTAL\_DROP}$
    $\text{CAP\_REG} \leftarrow \text{TOTAL\_CAPACITY} \bowtie_{\text{CourseID = CID}} \text{TOTAL\_REGISTER}$
    $\text{RESULT} \leftarrow \pi_{\text{(SUM\_CAPACITY – COUNT\_SNUID\_ADD + COUNT\_SNUID\_DROP)}} (\sigma_{\text{CourseID = given\_course\_id}} (\text{CAP\_REG}))$

12. Find the total number of students in the waitlist for a course
    RA:
    $\pi_{\text{Waitlist}} (\sigma_{\text{CourseID = given\_course\_id}} ({}_{\text{CourseID}} \mathcal{F}_{\text{SUM Waitlist}} (\text{COURSE})))$

13. Find the number of students in the waitlist for a given section of a course
    RA:
    $\pi_{\text{Waitlist}} (\sigma_{\text{SectionID = given\_section\_id}} (\text{COURSE}))$

14. Find the total capacity for a course
    RA:
    $\pi_{\text{Capacity}} (\sigma_{\text{CourseID = given\_course\_id}} ({}_{\text{CourseID}} \mathcal{F}_{\text{SUM Capacity}} (\text{COURSE})))$

15. Find all students a given advisor has approved for their course registration
    RA:
    $\pi_{\text{SNUID}} (\sigma_{\text{NUID = given\_advisor\_id}} (\text{APPROVE}))$

CONCLUSION

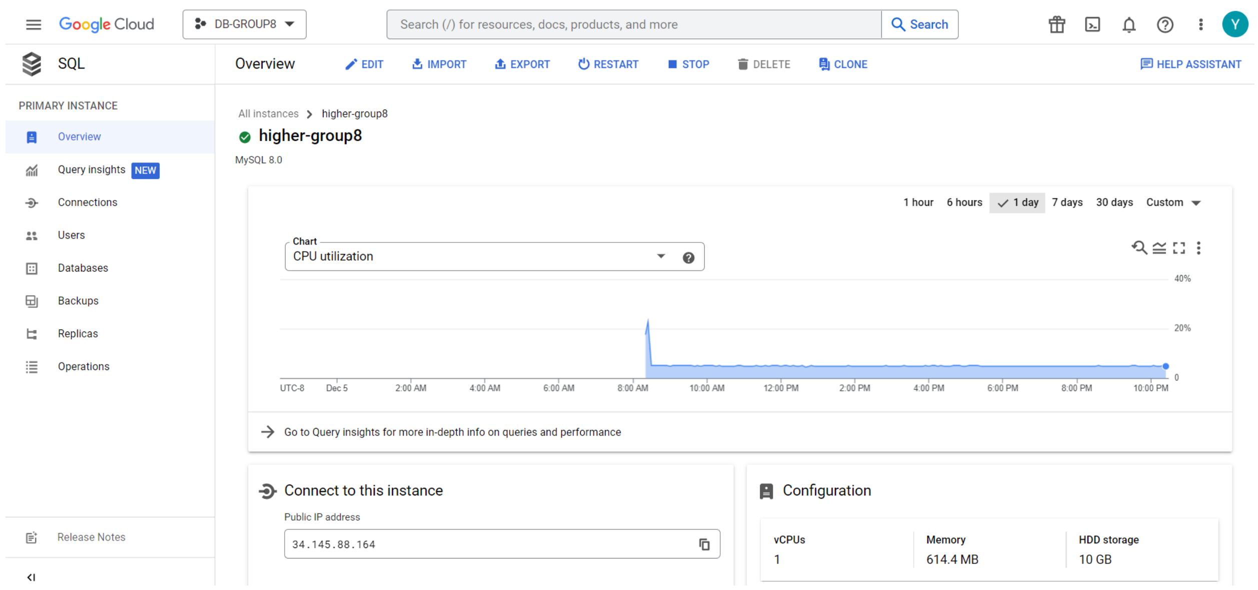
In this project, we are designing the web application for three types of users, where they have different relationships with each other. In this project we firstly figure out the required subjects in this application and listed queries that are useful to the ones using the data. In this step we learned how to figure out what we need, and what operations we require in our application. This helps us to think more logically.

Then we worked on the properties of each subject and logic of relationship with others. And we used the EER diagram to represent the relationship of the subjects we need in this project. We created relational models based on EER diagrams. And formulated all the queries with relational algebra. In this step we learned how to use EER to create detailed databases with a smart and efficient technique. And visualize our outlook of the database. And we learned to use relational algebra to get the relational databases.

To get better project output, we also need to figure out the change by operations between different subjects.

Phase 3

Database Server: DB-GROUP8
Database Instance: higher-group8
Database Name: new



higher-group8
MySQL 8.0

Chart: CPU utilization

Public IP address: 34.145.88.164

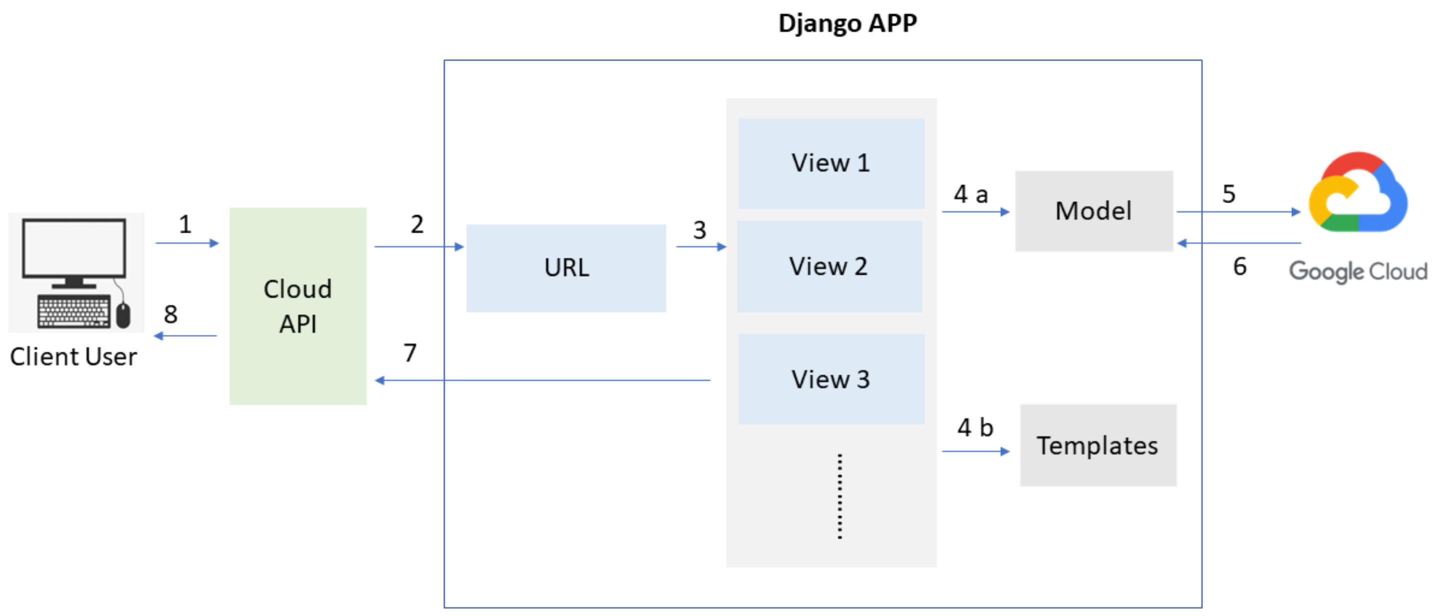
vCPUs	Memory	HDD storage
1	614.4 MB	10 GB

Phase 4

See attached sql files in the zip folder

Phase 5

SYSTEM CONFIGURATION



SOURCE OF APPLICATION

Website: https://db-group8.uw.r.appspot.com/

Github: https://github.khoury.northeastern.edu/yingjhuang/KhouryCourseRegistration

WHAT WE HAVE LEARNED

During this entire project, we have learned a clear process of how to create a web design using MySQL as our databases, and constructing with Django App.

We firstly learned to figure out all elements we needed in our project. And we learned to draw EER models to present the relationships between all tables. Then we learned to formulate all the queries with relational algebra. After listing the relationships between elements, we are able to find our terminated data with given information.

We learned how to build databases on Google Cloud Platform with MySQL. This allows us to make our databases shared with others. Compared to simply building local databases, this is more convenient for group work, allowing all contributors to be updated with new data.

Then we learned to implement the database tables from the normalized set of relations created in the previous phase. Besides getting the necessary data for our project, we also had a chance to think about slightly complex scenarios on our database schema that we learned from class.

In the last phase, we learned how to create a Django App for CRUD with generic class-based views and function views. We have created a virtual environment to run our program and connect the Django App to Mysql using mysqlclient. We used python files to apply queries and get our desired data from databases. And we used HTML to construct web pages to demonstrate our design. All the webpages follow the relationships that we designed. And the webpages respond to us to perform actions to view, add, delete or update our data.

We ran into backward relations in our database during the web application development phase and we had to revise our database structure in order to avoid any further issue. What we could have done better is being more thoughtful when designing the database so that we won't need to spend time revising the database. And if needed, we may design our web page layouts better.