EduFlex: Educational Management System

Fang Lingqing
e1101634@u.nus.edu
A0268335X

Liu Zhicheng
e1124508@u.nus.edu
A0274762W

Li Hao
e1124758@u.nus.edu
A0275012R

Qiu Yilun
qiuyilun@u.nus.edu
A0296996W

He Zean
hezean@u.nus.edu
A0297000N

An Junwen
junwenan@u.nus.edu
A0305212Y

Abstract

This project introduces an educational management system, EduFlex, that can streamline course-related operations within academic institutions. The work utilizes Oracle Virtual Private Database (VPD), to ensure secure and role-based access to sensitive information. EduFlex supports a variety of roles, including students, teachers, and staff, offering proper access control to functionalities such as course enrollment, material management, grading, and feedback collection. The integration of VPD ensures that access to data is tightly controlled, maintaining both security and privacy. In this report, the application flow, the system's architecture, role-specific permissions, and the implementation of security policies using Oracle VPD are discussed, the report also demonstrates how EduFlex addresses several challenges of managing academic data securely and efficiently.

Keywords

Database Security, Educational Management System, Oracle Database, Virtual Private Database

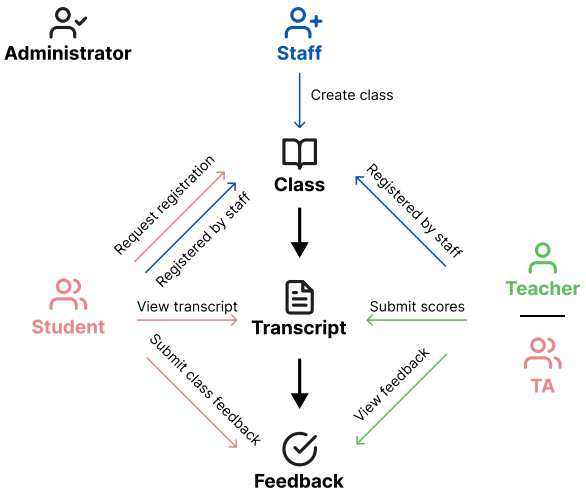

Figure 1: Application Flow

1 Introduction

In the ever-evolving landscape of educational management, the need for efficient, secure, and comprehensive data management systems has become paramount. To address this challenge, we design EduFlex, an educational management system, specifically designed for academic institutions to manage key information across departments, personnel, courses, classes, and academic records. This system facilitates seamless data handling for students, teachers, and administrative staff, ensuring that security, data integrity, and accessibility are maintained at all times. By clearly defining the roles of students, teachers, and staff, the system offers tailored functions for each group. It provides a centralized platform for students to enroll in courses, teachers to manage classes and educational materials, and administrative staff to oversee institutional operations.

In this report, we will first introduce EduFlex, followed by the security features implemented across various tables and roles to ensure each role will only be allowed to access certain tables or information in compliance with the security policies. We propose utilizing Oracle Virtual Private Database (VPD) for this application.

2 Application

EduFlex is an educational management system similar to Canvas, designed to support key operations for various roles involved in managing academic-related information. The overall user flow is illustrated in Figure 1. EduFlex accommodates three primary roles: Student, Teacher, and Staff.

At a high level, staff members are responsible for creating classes and enrolling students and teachers. Students can then select classes to register, subject to approval from staff. When a class concludes, teachers can submit students' grades, which are then added to their transcripts. Students can view their own transcripts once uploaded if their status is available.

After finishing a class, students are given the opportunity to provide feedback on the classes they participated in, and teachers can subsequently review this feedback. The following sections detail the user flow of the application.

2.1 User Flow

The administrator is responsible for initializing all data related to departments, majors, and individuals within the system. In addition to setting up this information at the beginning, the administrator is also responsible for maintaining and updating these records as needed.

Teachers can request approval from the administrative staff in their department to teach a class. Once approved, the class will be created by the staff. It is important to note that classes are differentiated by both course and semester. For instance, CS5322 in Fall 2024 is considered a distinct class from CS5322 in Spring 2024. Teachers are allowed to teach multiple classes within the same semester.

Once a class is created, the staff will register the teacher for that class. Students can then submit enrollment requests, which

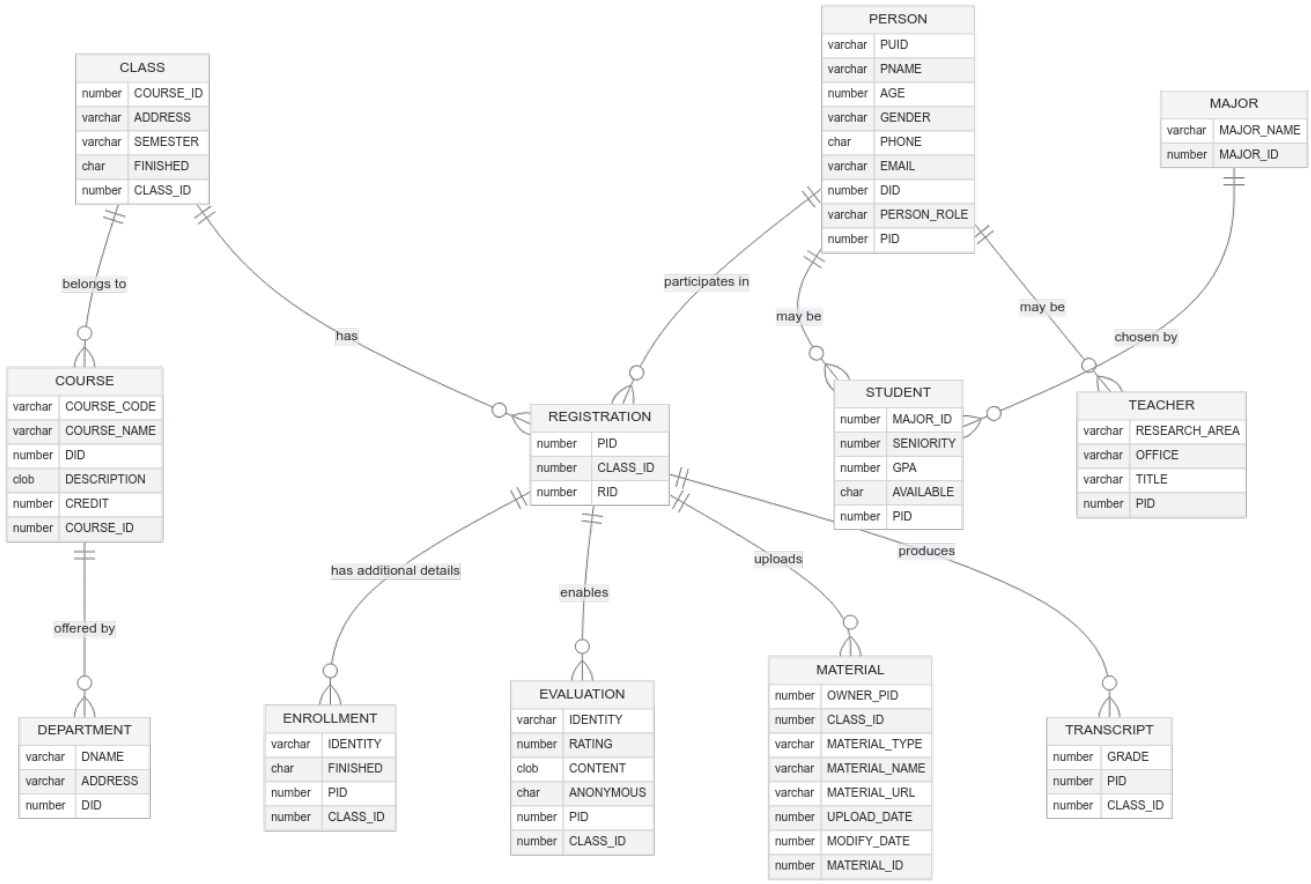

Figure 2: ER Diagram

are reviewed by the department staff. Approval is based on factors such as class capacity and the course's relevance to the student. For example, PhD students are more likely to be enrolled in PhD-level courses compared to Master's or Undergraduate students. Additionally, some mandatory classes, such as a research ethics course, may be automatically registered by staff without requiring a student enrollment request.

Before the class begins, a special student role, Teaching Assistant (TA), may be assigned by department staff. TAs have the same privileges as teachers, but only for the specific class in which they serve as a TA. For example, a student who is a TA for CS5322 in Spring 2024 will have teacher-level privileges in that class but not in others they are enrolled in as a student. A student cannot hold both the student and TA roles for the same class simultaneously.

During the semester, if students find the class unsatisfactory, they can request assistance from their department's staff to help them withdraw. Teachers will also be able to upload class materials to EduFlex for students to view, such as lecture slides and homework. At the end of the semester, teachers can upload each student's scores to their respective transcript. Students can view their final scores if they are available, similar to real-world academic systems.

After the semester concludes, students can submit feedback and evaluations for classes they successfully completed (withdrawn classes are excluded). Feedback is tied to a specific course and semester, helping users distinguish between the teaching quality of different instructors. Students have the option to submit feedback anonymously, and while all feedback is publicly accessible, the identity of the submitter is hidden if anonymity is selected.

Finally, the administrator of EduFlex system has the highest privilege, and he/she is able to modify every relevant table. For example, the administrator can adjust a staff's position from one department to another.

2.2 User Roles

This section outlines the privileges and permissions associated with each user role in EduFlex.

- **Administrator:** The administrator role has the highest privilege among all the roles. It has the permission to view and modify every table.
- **Staff:** The staff role represents administrative personnel in real-world academic settings. Staff members have access to department-specific information, such as class, teacher, and

EduFlex: Educational Management System

student data. For instance, a staff member from the Computer Science department can enroll students in Computer Science courses but cannot manage enrollments for other departments.

- **Teacher:** Teachers have access to the classes they are assigned to teach. However, they cannot create classes on their own; class creation requires approval from the department staff. Similarly, teachers cannot enroll students directly, as student enrollment is also subject to staff approval, reflecting real-world procedures. Within their classes, teachers can upload and modify the class materials and view and modify enrolled students' transcripts but are not permitted to alter class feedback, ensuring the integrity of student evaluations.
- **Student:** Students have the least privileges among the roles. To register for a class, they need approval from the department staff. Once enrolled, students can only view their own transcript information and materials within the class. Additionally, students are allowed to submit feedback for the classes they have completed. In addition to regular students, the **Teaching Assistant** role is a special subset of the student role, granting TAs the same privileges as teachers for the specific class they assist. This role mirrors real-world practices, where TAs support teachers by grading assignments and uploading scores. As such, TAs are given teacher-level permissions within the classes they serve as a TA.

3 ER Diagram and Tables

3.1 ER Diagram

Figure 2 shows the ER (Entity-Relation) Diagram of our application.

3.2 Schema Diagram

Figure 2 shows the Schema Diagram exported from the Datagrip of our application.

3.3 Table Design

In the following sections, we will present our design for the tables in the system, including their fields and relationships with other tables.

3.3.1 `DEPARTMENT` *Table.* This table contains information about various departments within the whole EMS, including a unique department identifier (*DID*), the department name (*DNAME*), and the department address (*ADDRESS*). The *DID* field acts as the primary key, while the *DNAME* field must be both unique and not null.

3.3.2 `PERSON` *Table.* This table stores the personal information of individuals, including a unique identifier (*PID*), a unique user ID (*PUID*), and various personal details such as name (*PNAME*), age (*AGE*), gender (*GENDER*), phone number (*PHONE*), and email address (*EMAIL*). The table enforces constraints to ensure data integrity: each individual must have a unique phone number and email, and the gender must be 'MALE' or 'FEMALE'. Each person also has a field *DID*, which serves as a foreign key linking to the `DEPARTMENT` table, establishing a relationship between individuals and their respective departments. The *PERSON_ROLE* field categorizes individuals as 'STAFF', 'TEACHER', or 'STUDENT', with each role associated with distinct privileges.

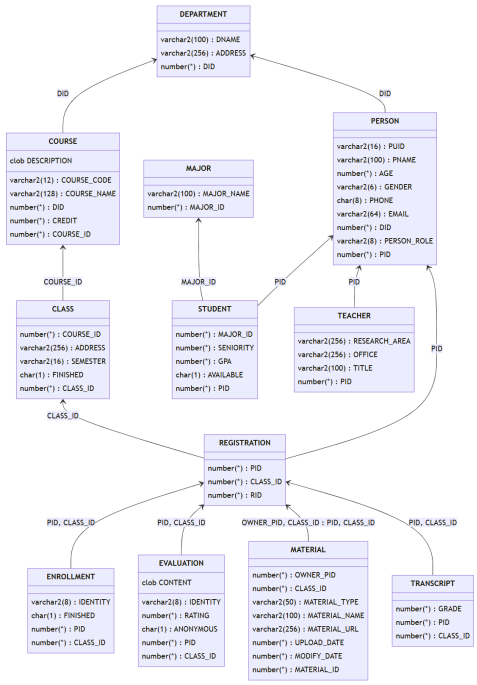

Figure 3: Schema Diagram

3.3.3 `TEACHER` *Table.* This table stores work information about teachers, including a unique identifier (*PID*), which serves as a foreign key linking to the `PERSON` table. It also includes the research area (*RESEARCH_AREA*), office location (*OFFICE*), and academic title (*TITLE*).

3.3.4 `MAJOR` *Table.* This table contains information about different majors, including a unique identifier (*MAJOR_ID*) and a unique major name (*MAJOR_NAME*).

3.3.5 `STUDENT` *Table.* This table holds the academic information about students, including a unique identifier (*PID*), which serves as a foreign key linking to the `PERSON` table, and a major identifier (MAJOR_ID) that links to the `MAJOR` table. It also includes the seniority level (*SENIORITY*), score (*GPA*), and availability status (*AVAILABLE*). The *GPA* is constrained to be between 0 and 100, while the *AVAILABLE* field indicates the student's current availability. If a student is unavailable, they will not be able to access their transcript at that time.

3.3.6 `COURSE` *Table.* This table stores information about academic courses, including a unique identifier (*COURSE_ID*) as the primary key, course code (*COURSE_CODE*), course name (*COURSE_NAME*), and department identifier (*DID*) linking to the `DEPARTMENT` table. It also includes a course description (*DESCRIPTION*) and the credit value (*CREDIT*), indicating the credits awarded for completion.

3.3.7 `CLASS` *Table.* This table contains information about class sessions for specific courses, including a unique identifier (*CLASS_ID*) as the primary key, and a course identifier (*COURSE_ID*) that links to the `COURSE` table. This relationship establishes which course each class session belongs to. The table also includes the class address

(*ADDRESS*), semester (*SEMESTER*), and a status field (*FINISHED*) to indicate whether the class has been completed, with a default value of 'N'.

3.3.8 `REGISTRATION` *Table.* This table serves as the relation table that links students and teachers to their enrolled classes. It uses a composite primary key consisting of the student identifier (*PID*) and the class identifier (*CLASS_ID*), ensuring that each person can only have one registration record for a specific class session.

3.3.9 `MATERIAL` *Table.* This table stores information about educational materials associated with specific classes, including a unique identifier (*MATERIAL_ID*) as the primary key. It features an owner identifier (*OWNER_PID*) and a class identifier (*CLASS_ID*), linking the materials to the respective owners who publish them and the classes they are associated with. The table also includes fields for the type of material (*MATERIAL_TYPE*), the material name (*MATERIAL_NAME*), a URL for accessing the material (*MATERIAL_URL*), and timestamps for upload and modification dates (*UPLOAD_DATE* and *MODIFY_DATE*). A constraint ensures that the *MODIFY_DATE* cannot be earlier than the *UPLOAD_DATE*.

3.3.10 `ENROLLMENT` *Table.* This table records enrollment details for registration records, including the person identifier (*PID*) and a class identifier (*CLASS_ID*), which together form a composite primary key and reference the `REGISTRATION` table. It includes an *IDENTITY* field to specify the individual's role as 'TUTOR', 'STUDENT', or 'TA', and a *FINISHED* field to indicate whether the enrollment is complete, with a default value of 'N'.

3.3.11 `FEEDBACK` *Table.* This table captures feedback for specific classes, including the person identifier (*PID*) for the student and a class identifier (*CLASS_ID*), which together form a composite primary key and reference the `REGISTRATION` table. It features an *IDENTITY* field to indicate the role of the individual providing feedback, a *RATING* field to assess the class on a scale from 1 to 10, and a *CONTENT* field for detailed comments, stored as a CLOB. Additionally, the *ANONYMOUS* field indicates whether the feedback is submitted anonymously, with a default value of 'N'.

3.3.12 `TRANSCRIPT` *Table.* This table records the grades for students in specific classes, including the person identifier (*PID*) for the student and a class identifier (*CLASS_ID*). Together, these fields form a composite primary key and reference the `REGISTRATION` table. The table also includes a *GRADE* field, which is constrained to be between 0 and 100, ensuring valid grade entries.

4 Security Features

4.1 User Session Context Management

We design the `app_ctx_pkg` package, which plays a critical role in user session management within database environments by implementing a robust mechanism for contextual authentication and authorization. Its primary function is to dynamically establish and maintain user contexts, ensuring that users are correctly authenticated and assigned the necessary roles and permissions based on their identities.

There are three main procedures defined within the package: `set_ctx_pkg`, `set_reg_auth`, and `clear_reg_auth`.

- `set_ctx_pkg`: We first design the `app_ctx` to act as the user's session context. This procedure will set up the user ID, role, and department in the user's session context, based on the login credentials. It ensures that the session is configured with the correct access privileges for the logged-in user.
- `set_reg_auth`: This procedure grants authorization for accessing the `REGISTRATION` table by setting the `reg_auth` context, allowing users to view the table. It is designed to be invoked exclusively within VPD policy functions, enabling these functions to select data from the `REGISTRATION` table.
- `clear_reg_auth`: This procedure revokes registration authorization by clearing the `reg_auth` context, ensuring that users no longer have permission to perform registration tasks. It is invoked after the data from the `REGISTRATION` table has been retrieved.

4.2 Security Policies

Based on the tables and roles defined for the system, we have implemented various VPD policies to ensure comprehensive protection. In the following sections, we will introduce the VPD policies for each table individually.

4.2.1 `DEPARTMENT` *and* `MAJOR` *Tables.* These two tables are maintained exclusively by the administrator. The administrator can view the data within these tables and perform insert, delete, and update operations. Users with other roles can query and access all the information contained in these two tables.

4.2.2 `PERSON` *Table.* This table contains several sensitive fields, necessitating the establishment of VPD policies for different columns and user roles. However, since only the administrator can modify the data within this table, we will only focus on the VPD policies on select statements.

- The `select_person_email` function defines the access permissions for the *EMAIL* field in the `PERSON` table. As a means of communication, the email addresses of staff and teachers provide students with a way to reach out to them. Therefore, we have set the email addresses of staff and teachers to be accessible to everyone. Additionally, both staff and teachers can view the email addresses of all students to facilitate communication. However, students may not wish to disclose their email addresses to all their classmates. To address this, a student's email will be shared only with those classmates who are enrolled in the same course, allowing for easier contact. Of course, each student can always access their own email address.
- The `select_person_age` function defines the access permissions for the *AGE* field in the `PERSON` table. Since age is considered relatively private information, we have restricted access so that both teachers and students can only view their own ages. For staff, who have management authority over their respective departments, we have allowed them to access the ages of all individuals within their department.
- The `select_person_phone` function defines the access permissions for the *PHONE* field in the `PERSON` table. While both phone numbers and email addresses serve as contact information, phone numbers are generally considered more

private. Although staff and teachers can view everyone's phone numbers, students do not have direct access to the phone numbers of staff and teachers, nor can they access the phone numbers of classmates enrolled in the same course.

4.2.3 `STUDENT` *Table.* This table stores the academic information of a student, such as GPA and availability. As such, students have access only to their own records. However, staff in the same department have the authority to modify the enrollment status. Therefore, we have designed separate VPD policy functions for querying and updating this information.

- The `select_student_gpa` function defines the access permissions for the *GPA* field in the `STUDENT` table. As GPA is one of the most sensitive pieces of academic information for students, they can only view their own GPA if their availability is granted. However, teachers and staff still have access to students' GPAs within their departments.
- The `select_student_available` function defines the query permissions for the `AVAILABLE` field. This field is used to indicate the current status of a student, with "N" denoting a student under academic warning. As such, students can only view their own status, while staff and teachers still have access to the status of students within their department.
- The `update_student` function defines the update permissions for the `STUDENT` table. In addition to the administrator, staff are granted permission to modify the academic information of students within their own department.

4.2.4 `TEACHER` *Table.* The `TEACHER` table stores teachers' work-related information, which is publicly visible to everyone.

- The `update_teacher` function defines the update permissions for the `TEACHER` table. Teachers are only allowed to modify their own work information, while staff have the authority to update the work information of all teachers within their department to provide necessary support.

4.2.5 `COURSE` *and* `CLASS` *Tables.* These two tables are primarily maintained by the administrator and the staff of their respective departments. All roles in the system can query all information in these tables but do not have permission to modify the data. Staff, however, have the authority to insert, update, and delete course and class information for their department.

- The `insert_course`, `delete_course`, and `update_course` functions define the permissions for operations on the `COURSE` table. The administrator has full permissions for all operations, while teachers and students do not have any operational rights. Staff, however, are allowed to manage course data for their respective departments.
- The `insert_class`, `delete_class`, and `update_class` functions define the permissions for operations on the `COURSE` table. The administrator has full permissions for all operations, while teachers and students do not have any operational rights. Staff, however, are permitted to manage class data for their respective departments. Therefore, the functions first need to verify whether the course associated with the class belongs to the staff's department before granting permission.

4.2.6 `REGISTRATION` *Table.* This table plays a crucial role in establishing the relationship between individuals and their enrolled classes. Typically, only the administrator has access to the data in this table, as users prefer to keep their class enrollment information private. However, since we need to retrieve data from this table for other Virtual Private Database (VPD) policies, we have designed two procedures that act as a safeguard for the table, as mentioned in Section 4.1.

Additionally, students have the ability to add their own data to the `REGISTRATION` table. This operation simulates a student attempting to register for a class. The related data will only be added to the `ENROLLMENT` table once it has been confirmed by the department staff.

- The `select_registration` function defines the access permission for the `REGISTRATION` table. When access to the data in the `REGISTRATION` table is required for other defined VPD policy functions, we can invoke the procedure `set_reg_auth` to obtain the necessary authorization. In the function, we first verify the authorization. If the authorization is granted, the function will ensure that they can view the complete data during that session. After this, the function will revoke access for the user. Users other than the administrator will not be able to access the data in the `REGISTRATION` table directly, thereby enhancing security.
- The `insert_registration` function defines the insert permissions for the `REGISTRATION` table. Staff can insert registration records for individuals within their department for courses offered by that department. Students are allowed to register themselves for classes that have not yet finished. However, teachers do not have permission to insert records, as the staff will handle this on their behalf.
- The `delete_registration` function defines the delete permissions for the `REGISTRATION` table. The permissions for staff are the same as those in `insert_registration` function, allowing them to remove registration records for individuals within their department. However, students and teachers do not have permission to delete records from the `REGISTRATION` table.

4.2.7 `ENROLLMENT` *Table.* The `ENROLLMENT` table stores the role of the registrant for each registration record and whether the registration is in a finished state. We have designed different VPD policy functions for querying and operating this table to ensure proper access control.

- The `select_enrollment` function defines the query permissions for the `ENROLLMENT` table. Staff can view the enrollment information for all individuals and classes within their department. Teachers and students, on the other hand, can access the enrollment records of others in the same class, allowing them to communicate with their classmates or colleagues.
- The three functions for operating the `ENROLLMENT` table, which includes `insert_enrollment`, `delete_enrollment`, and `update_enrollment`, define the permission for inserting, deleting, and updating records. While the administrator has unrestricted access, staff can insert, delete, and update enrollment information for individuals and classes within

their department. Teachers and students, however, do not have any permission to perform these operations.

4.2.8 `MATERIAL` *Table.* The `MATERIAL` table stores class-related learning materials, such as PPTs and assignments. Students enrolled in the class have permission to view these materials, but regular students cannot upload or update them. Teachers and TAs for the class can upload, update, and delete materials. Department staff can delete materials to manage storage or perform cleanup, but they are not permitted to upload or update the content.

- The `select_material` function defines the query permissions for the `MATERIAL` table. The administrator has unrestricted access to all material records. Staff can view all materials for classes within their department, while teachers and students are able to access materials for the classes they are registered in.
- The `insert_material` and `update_material` functions define the permissions for inserting and updating records in the `MATERIAL` table. The administrator has unrestricted access, but staff cannot insert or update records in the `MATERIAL` table. Teachers can insert and update materials for the classes they are registered to teach. Students, however, only have these permissions if they are registered as a TA for the class. Regular students do not have any permission to perform these operations.
- The `delete_material` function defines the deletion permissions for the `MATERIAL` table. The administrator has unrestricted access, while staff are allowed to delete materials for classes within their department. Teachers and TAs have the ability to delete materials for the classes they are registered to. However, regular students do not have permission to delete any materials.

4.2.9 `FEEDBACK` *Table.* The `FEEDBACK` table stores students' feedback on the classes they have finished, including ratings and comments. This information is made public to help other students when selecting courses, though students have the option to submit feedback anonymously. Staff and teachers cannot insert or update `feedback` data, but staff are allowed to delete inappropriate feedback, such as offensive language, for the classes within their department.

- The `select_feedback_pid` function defines the query permissions for the *PID* field in the `FEEDBACK` table. Since students have the option to submit feedback anonymously, their *PID* should not be visible to others if they choose to remain anonymous. However, the administrator retains the ability to query all information to identify students who may have submitted inappropriate, even illegal, feedback, ensuring that any misconduct can be addressed appropriately.
- The `insert_feedback` function defines the insertion permissions for the `FEEDBACK` table. Neither staff nor teachers have the ability to insert feedback. Only students can submit feedback, and they are permitted to do so only if their registration record for this class is marked as finished.
- The `delete_feedback` function defines the deletion permissions for the `FEEDBACK` table. Staff can delete feedback for classes within their department, while teachers do not have any operational permissions regarding the `FEEDBACK` table. Students, however, have the ability to delete their own submitted feedback.
- The `update_feedback` function defines the update permissions for the `FEEDBACK` table. Only the administrator has the authority to update any data within the `FEEDBACK` table. Staff and teachers do not have any update permissions, while students are only allowed to update their own submitted feedback.

4.2.10 `TRANSCRIPT` *Table.* The `TRANSCRIPT` table corresponds to students' grades for specific classes, representing highly sensitive academic information. As such, students can only access their own transcripts. However, to facilitate the grading process, teachers may require TAs to assist with recording grades. Therefore, both teachers and TAs should have the ability to insert, update, and delete transcript records for the students enrolled in the classes they are associated with.

- The `select_transcript` function defines the query permissions for the `TRANSCRIPT` table. Staff can access transcripts for all students in the classes within their department, while teachers can retrieve the transcripts for the students in the classes they teach. When students are not TAs, they can only view their own transcripts, provided their status is available. However, if a student is serving as a TA, they have the same access as teachers and can view the transcripts for all students enrolled in the classes they assist with.
- The `insert_transcript` and `update_transcript` functions generally define the insertion and update permissions for the `TRANSCRIPT` table. Administrators have unrestricted access to insert and update any records. Staff, however, do not have such permission. Teachers and TAs for a particular class are granted permission to insert and update the transcripts for the students enrolled in that class, allowing them to record and modify grades as necessary.
- The `delete_transcript` function defines the deletion permissions for the `TRANSCRIPT` table. Staff have the ability to delete transcripts for students within their department. Additionally, teachers and TAs for a particular class can delete the transcripts of the students enrolled in that class.

5 Implementation

All of our SQL codes were written using DataGrip.

We first designed the `APP_ADMIN` database user as the administrator for the entire system. This user holds advanced privileges, such as creating tables, contexts, database connections, etc. During the development process, we primarily used this account for testing, and all tables were created under its schema.

When designing the application, we initially focused on the structure of the tables and the required fields. As we refined the security measures, we continuously optimized and updated table structures and the fields. After that, we started by executing all the `CREATE TABLE` statements and added primary keys, foreign keys, and other necessary constraints.

Next, we developed a trigger that is called during user login, defining a user session context for the entire application.

EduFlex: Educational Management System

Following this, we moved on to the design of the database security policy functions. While designing the tables, we had already formed some initial ideas regarding security policies. We built upon these ideas, improving them and writing return statements for the functions to do security controls. These statements should query other related tables to implement security controls.

Since different roles in the system require different access levels, we retrieved the currently logged-in user's role from the user session context. We used `if` branches to differentiate between roles and return the appropriate query statements for each role. Additionally, the user session context provided us with the user's ID and department information, further guiding us in designing access permissions. When permission design requires querying data from the registration table, the function will invoke the statement `app_ctx_pkg.set_reg_auth()` to obtain temporary access for querying the registration table. This mechanism ensures that the appropriate authorization is in place while maintaining the security protocols of the system.

Here, we take the implementation of the `select_student_gpa` function as an example. The complete code is shown below:

```
CREATE OR REPLACE FUNCTION select_student_gpa(
    obj_schema IN VARCHAR2,
    obj_name IN VARCHAR2
) RETURN VARCHAR2 IS
    v_pid     INTEGER;
    v_dept_id INTEGER;
BEGIN
    IF SYS_CONTEXT('userenv','isdba') = 'TRUE' OR
       SYS_CONTEXT('userenv','session_user') = 'APP_ADMIN'
       ↪  THEN
        RETURN '';
    ELSIF SYS_CONTEXT('app_ctx','user_role') IN ('STAFF',
    ↪  'TEACHER') THEN
        v_dept_id := SYS_CONTEXT('app_ctx','user_dept_id');
        RETURN 'PID IN (SELECT PID FROM APP_ADMIN.PERSON
        ↪  WHERE DID = '  v_dept_id  ')';
    ELSIF SYS_CONTEXT('app_ctx','user_role') = 'STUDENT' THEN
        v_pid := SYS_CONTEXT('app_ctx','user_id');
        RETURN 'PID = '  v_pid  ' AND AVAILABLE = ''Y''';
    ELSE
        RETURN '1=0';
    END IF;
END;
```

The function header for all VPD security policy functions is fixed. this function starts by checking whether the current user is an administrator. If they are, no permission constraints are needed, and the function simply returns an empty string. For users with the role of staff or teacher, access is restricted to the *GPA* of students within the same department. The function will utilize the department information to query the `PERSON` table and retrieve the *PID* of all students in that department. Students, on the other hand, are permitted to view only their own *GPA*. They must provide their *PID* and also confirm that their academic status is active by checking the *AVAILABLE* field.

After implementing all VPD policy functions, we add these policies to the system, using `select_student_gpa` as an example:

```
DBMS_RLS.ADD_POLICY(
    object_schema => 'APP_ADMIN',
    object_name => 'STUDENT',
    policy_name => 'select_student_gpa_policy',
    function_schema => 'APP_ADMIN',
    policy_function => 'select_student_gpa',
    statement_types => 'SELECT',
    sec_relevant_cols => 'GPA',
    sec_relevant_cols_opt => dbms_rls.ALL_ROWS
);
```

Here, we need to specify the type of statement for this policy function, which is used for SELECT operations, specifically concerning the *GPA* field. Other variables are primarily determined by the name of the policy function.

Then we added all VPD policy functions to the system in a similar manner to the previous implementation, adjusting the variables according to the specific functions and tables.

Finally, we generated a set of test cases based on the application's use cases. Some of these test cases were created directly using scripts, while others were manually designed to better demonstrate the system's security features during the demo. This approach ensures a comprehensive evaluation of the system's functionality and security features.

6 Conclusion

The EduFlexEducational Management System stands out in the realm of academic data management by providing a comprehensive and tailored solution to the complex needs of educational institutions. By applying Oracle's Virtual Private Database (VPD), EduFlexprovides a robust, secure, and flexible platform for managing critical academic information. Applying security policies directly at the database level enables the system to significantly enhance the overall security, ensuring that sensitive data is protected at its core. Additionally, the integration of user session context allows developers to implement proper security policies closest to the data, enabling more finer-grained access control. This approach not only minimizes the risk of data leak due to security vulnerabilities that may arise when access control is handled at the application layer. The system's key strengths lie in its enhanced security features and user-focused design. These elements combine to create a system that not only protects sensitive data but also enhances the efficiency of academic administration.

The development procedure of EduFlexunderscores the importance of careful planning, iterative design, and the striking the balance between security and usability in creating an effective database system. In working on this project, all team members have strengthened their technical skills in database design, SQL programming, and the understanding of Oracle VPD.

The current version of EduFlexlays a robust foundation, also demonstrates a forward-thinking approach to both data security and administrative efficiency. The system's architecture is designed to be scaleable, allowing for future iterations to easily incorporate new features without compromising the core security principles. This positions EduFlexas a powerful tool for educational data management, combining strong security measures with the flexibility

and efficiency needed to meet the evolving demands of modern academic administration. While the current version of EduFlex is robust, there are several areas for future improvement. One possible area is the integration of salary-related tables and functions, which would further expand the system's capabilities. Additionally, exploring advanced analytics and reporting features could enhance data-driven decision-making for educational institutions. These enhancements will further strengthen EduFlex's position as a comprehensive and adaptable educational management system.

7 Contribution

In this section, we will outline the key contributions of our team to this project:

Team member: Fang Lingqing

- Project planning and management.
- Data model and VPD security policies design.
- Report Writing.

Team member: Liu Zhicheng

- Data model and VPD security policies design.
- VPD policy function implementation.
- Report Writing.

Team member: Li Hao

- Data model and VPD security policies design.
- VPD policy function implementation.
- Report Writing.

Team member: Qiu Yilun

- Application and user flow design.
- Data model and VPD security policies design.
- VPD policy function implementation.

Team member: He Zean

- Data model and VPD security policies design.
- Table design and creation.
- Test data design and generation.

Team member: An Junwen

- Application and user flow design.
- Data model and VPD security policies design.
- Report writing.

The contribution ratio of each member in the project is equal.