Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/40/20230624

Database design for course selection and course grading system

Donghao Liu

School of Information Technology James Madison University,800 S Main St, Harrisonburg, VA, 22807, United States

liu4dx@dukes.jmu.edu

Abstract. In the context of global education, universities and institutions emphasize the importance of proficient academic information management systems. As traditional approaches to course data management become obsolete, there is a growing need to leverage digital transformation for academic management. This paper proposes a novel course database design that centralizes information about subjects, courses, staff, students, buildings, and grades. The database is designed to facilitate the overall academic progress of students by ensuring efficient record keeping, retrieval and updating. The architecture simplifies administrative tasks, underscores the importance of databases in modern educational institutions by providing in-depth student data to support personalized learning, enhance communication among stakeholders, and aid academic research. The conclusion of the experimental test is that a database designer should always ensure they pay attention to avoiding data redundancy and ensuring data diversity when processing data.

Keywords: Database Design, Course Selection, Course Grading System.

1. Introduction

Nowadays, universities and educational institutions around the world are increasingly recognizing the importance of building an efficient academic information management system. The essence of modern education should revolve around acquiring knowledge, managing knowledge and disseminating it [1]. With the continuous development of teaching methods, course management has become a challenging and complex task [2,3,4]. Then the design of course database in this article plays a key role in this aspect. The entire database includes information on subjects, courses, staff and students, all of which contribute to a student's overall academic growth. However, an ongoing challenge is to systematically record, update and retrieve large amounts of data in an efficient and user-friendly manner. With technology integrated into nearly every aspect of education, traditional methods of managing course data have become redundant, if not outright obsolete. Educational institutions, from elementary schools to elite universities, are struggling not only to keep pace with digital transformation, but to harness its power to improve academic management.

The value of educational databases lies in efficient management: the database simplifies administrative tasks for educational institutions. They help to systematically manage and organize large amounts of data, making it easier to access, modify, and update. Personalized Learning: With access to detailed student data, educators can design a more personalized learning experience to meet the needs

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/40/20230624

of individual students. Communication and Collaboration: Databases facilitate better communication between educators, and students. Teachers can provide feedback, and parents can also use feedback to monitor children's progress. Research and Development: Academic databases facilitate scholarly research by providing access to prior research, ensuring that knowledge is not developed in isolation.

This article details the relationship between courses, students, employees, teaching buildings, and grades, and discusses methods for comprehensively storing student information and their corresponding grades. It emphasizes the importance of educational databases for the modern management of educational institutions, highlighting their role in aligning with societal digital transformation trends.

2. Database Design

2.1. Database Requirements

- **Requirements for students**:

Students should be able to browse the list of available courses and select the ones they are interested in. When choosing courses, they should also be able to view the start and end dates for each course. The course selection system should offer search and filter functions, making it easier for students to find courses based on keywords, credits, timings, and other criteria [2]. Detailed information about each course, including the instructor, schedule, and course description, should be accessible to students. Finally, students should have the capability to add chosen courses to their personal course selection list.

- **Requirements for Class:**

Students can view various details of their courses at any time to decide which courses they will choose to take each semester.

- **Requirements for student_Grade:**

Each student's grade for each class will be calculated based on their attendance, homework, and test scores, and the final calculated grade will be their final grade after the class ends.

- **Requirements for Department:**

Every student should choose their own major. Each major corresponds to a department, which has a designated person in charge and a specific teaching building.

- **Requirements for Attendance/Assignments/Exam:**

Each class will record the attendance score of each student, and also record the attendance status of each student, such as presence, lateness, and absence, and each of their grades will be recorded. Each class will record the assignments' scores of each student, and the final assignments will have a percentage of the total grade. Each class will record Exam scores of each student. For example, some classes have 3 exams, namely exam 1, exam 2, and exam 3. And some courses only have 2 exams, one is midterm exam and the other is final exam.

3. Database functions

In response to the above requirements, the following features have been designed for this article

- Update and modify student details, such as date of birth, address, email, phone number, and other related information.
- Add, search, or drop a class, as well as check its start time, end date, corresponding section, and the designated staff.
- View student grades for each class and access detailed breakdowns, including attendance' scores, assignments' scores, and Exam' scores.
- Verify the department chosen by each student and the address of that department.
- Integrate data from courses, student grades, attendance, homework, and exams, allowing a comprehensive view of each student's performance in every class."

DOI: 10.54254/2755-2721/40/20230624

4. ER Diagrams and Explanations

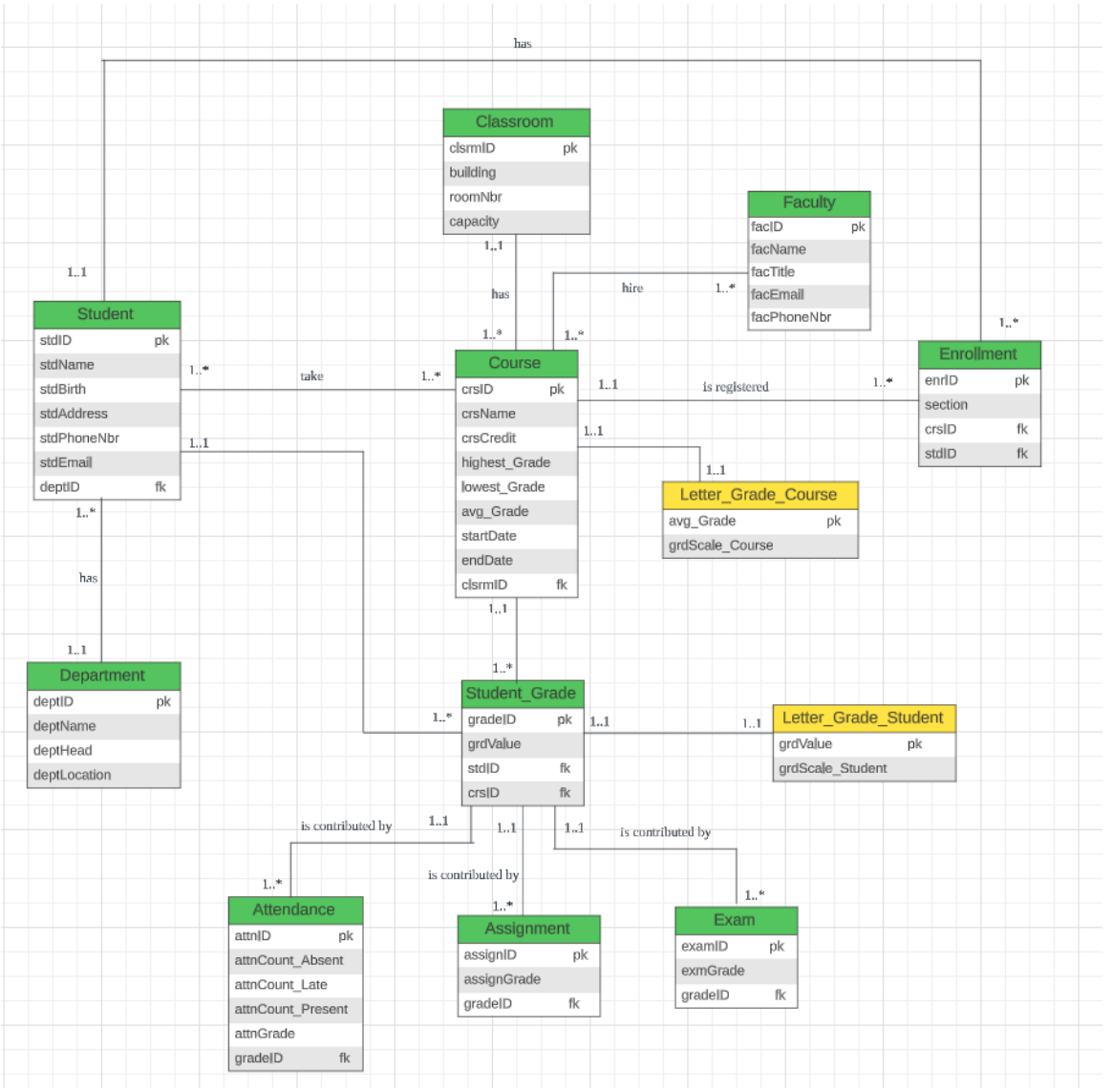

Figure 1. Overall ER diagram.

a) As seen in the relationship between "student" and "department", students are from different departments; each student belongs to only one department, and each department has one head and a specific location on campus (Figure 1).
b) As seen in the relationship between "student" and "course", each student can select more than one course, and each course has a specific name and credit value.
c) As seen in the relationship between "course" and "enrollment", every course has an enrollment, and students can choose different sections of a single course. They have the option to both enroll in and drop courses. All the data can be recorded in the system.
d) As seen in the relationship between "course" and "classroom", every course is assigned a designated classroom located in a specific building with limited capacity.
e) As seen in the relationship between "course" and "faculty", each course is taught by multiple faculty members, including professors, associate professors, TA, etc. The information to be held on each faculty member includes their name, title, email, and phone number.

DOI: 10.54254/2755-2721/40/20230624

f) As seen in the relationship between "student" and "enrollment", one student can have many enrollments since a single student can enroll in multiple courses or sections.
g) As seen in the relationship between "student_grade", "attendance", "assignments", and "exam", we can see that the relationship between "student_grade" and other 3 entities is 1 to many(1..*), since each student's grade is recorded and contains their attendance, assignments and exam scores.

4.1. Entities, attributes and its relational model

When designing the EduHub database, we selected the following 10 entities and their corresponding attributes, as shown in table 1. Next, the relationship between them will be explained in detail in the next section, such as what is a one to one relationship and what is a one to many relationship, and in this ER diagram as shown below that most of the relationships are one to many [4].

Table 1. Entities and the corresponding attributes.

Entities	attributes
Student	(stdID,stdName,stdBirth,stdAddress,stdPhoneNbr,stdEmail) Primary key (stdID) Foreign key deptID Reference Department (deptID)
Department	(deptID, deptName, deptHead, deptLocation) Primary Key (deptID)
Course	(crsID, crsName, crsCredits, highest_Grade, lowest_Grade, avg_Grade, startDate, endDate) Primary Key (crsID) Foreign Key clsrmID Reference Classroom (clsrmID)
Letter_Grade_Course	(avg_Grade, grdScale_Course) Primary Key (avg_Grade)
Enrollment	(enrID, section) Primary Key (enrlID) Foreign Key crsID Reference Course (crsID) Foreign Key stdID Reference Student (stdID)
Classroom	(clsrmID, building, roomNbr, capacity) Primary Key (clsrmID)
Faculty	(facID, facName, facTitle, facEmail, facPhoneNbr) Primary Key (facID)
Student_Grade	(gradeID, grdValue) Primary Key (gradeID) Foreign Key crsID Reference Course (crsID) Foreign Key stdID Reference Student (stdID)
Attendance	(attnID, attnCount_Absent, attnCount_Late, attnCount_Present, attnGrade) Primary Key (attnID) Foreign Key gradeID Reference Student_Grade (gradeID)
Assignment	(assignID, assignGrade) Primary Key (assignID) Foreign Key gradeID Reference Student_Grade (gradeID)
Exam	(examID, examGrade) Primary Key (examID) Foreign Key gradeID Reference Student_Grade (gradeID)
Letter_Grade_Student	(grdValue, grdScale_Student) Primary Key (grdValue)

DOI: 10.54254/2755-2721/40/20230624

Table 1. (continued).

Take	(stdID, crsID) Primary Key(stdID,crsID) Foreign Key stdID References Student (stdID) Foreign Key crsID References Course (crsID)
Hire	(crsID, facID) Primary Key(crsID,facID) Foreign Key crsID References Course (crsID) Foreign Key facID References Faculty (facID)

5. Normalization

When we were designing the EduHub database, and make sure we were creating good entities and its attributes, we noticed that we need to do normalization for "course" entity and "student_grade" entity as seen in Figure 2 and Figure 3.

Normalization in database design is a systematic approach to organizing data to reduce redundancy and ensure data integrity [5]. It should be used when aiming to achieve a clear and efficient database structure, especially when avoiding data anomalies and ensuring referential integrity.

The reason why we decided to make normalization for these two entities is because avg_grade as attributes in "course" entity would determine "grdScale_Course". For example, if the avg_Grade of a course is 93%, it indicates that the grdScale_Course is A- as a letter grade.

And we made normalization for "student_grade" is because grdValue would determine grdScale_student. For example, if a student's final grade (grdValue) is 96%, it would indicate that he gets an A as his letter grade (grdScale_Student).

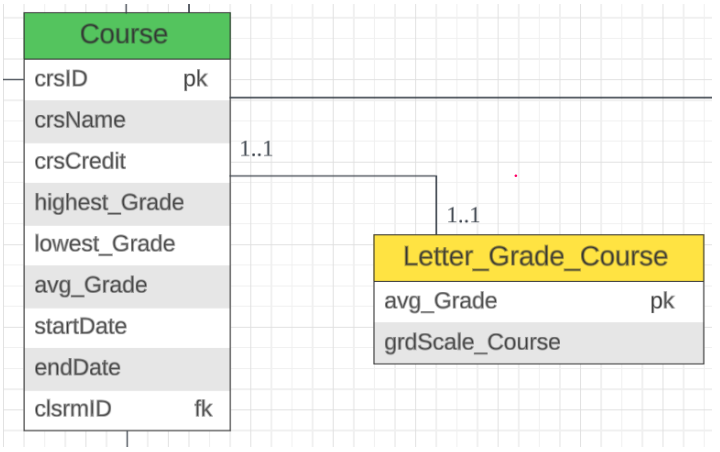

Figure 2. Normalization for "course"

Figure 3. Normalization for "student_grade".

DOI: 10.54254/2755-2721/40/20230624

6. Database Functions

Here is a list of some of the functions used and the goals achieved in the database of this course selection and grade management system.

1. **Data Storage**: Databases provide a systematic and organized way to store large amounts of data, making it easily accessible and manageable [6]. A lot of information is stored in each chart in the database. For example, in the student chart, the following information is stored, student ID, date of birth (in the form of year-month-date), address, phone number, and email address. In fact, according to design requirements, the type and diversity of information can be changed. One of the most basic functions of a database is to store a large amount of data.
2. **Data Retrieval**: One of the primary functions of a database is to allow the retrieval of data for various applications and queries [7]. For example, in the student chart, the student's ID, date of birth, phone number and address, and email address are stored. When we want to query the name of a student, we can query by the student's name, but if some students have the same name, then we need to query by studentID. Querying with studentID is the most accurate, because studentID is unique and cannot be repeated.
3. **Data Update and Deletion**: Databases allow for the modification and deletion of data, ensuring that the information stored remains current and relevant [8-9]. For example, in the student table, if we want to change the email of the student Mike, we need to enter the student table, and then use some sql codes to operate.

7. SQL command for creating database

In this section, this work lists databases as an example to show how to create the database in MySQL platform.

```
CREATE TABLE Department (
  deptID INT PRIMARY KEY,
  deptName VARCHAR(50) NOT NULL,
  deptHead VARCHAR(50),
  deptLocation VARCHAR(50)
);
select * from department
```

Commands 1. Commands for creating “department” table

As shown above, the command “CREATE TABLE DEPARTMENT” is used to create a new table in the database named “Department”. “DeptName” defines a column named “DeptID’ with data type ‘INT’ as integer [10]. The ‘Primary KEY’ constraint uniquely identifies each record in a database table which means that every “deptID’ must be unique and cannot be ‘NULL’. And ‘VARCHAR’ stands for variable character, and it can store up to 50 characters in this case. Select * from department, this sql statement is used to retrieve all the records from the “Department” table.

“Select” is a way of saying select all columns, so it will return every column for every row in the table.

Table 2. Outputs for “department” table.

deptID	deptName	deptHead	deptLocation
1	Computer Science	Dr. Smith	Building A
2	Mathematics	Dr.Johnson	Building B
3	Physics	Dr. Adams	Building C
4	Chemistry	Dr.Williams	Building D
5	Biology	Dr. Brown	Building E
6	History	Dr. Taylor	Building F

DOI: 10.54254/2755-2721/40/20230624

Table 2. (continued).

7	Literature	Dr. Harris	Building G
8	Music	Dr. Wang	Building H
9	Information System	Dr. Green	Building I
10	Language	Dr. Jason	Building J

```
CREATE TABLE Student (
    stdID INT PRIMARY KEY,
    stdName VARCHAR(50) NOT NULL,
    stdBirth DATE,
    stdAddress VARCHAR(100),
    stdPhoneNbr VARCHAR(15),
    stdEmail VARCHAR(50),
    deptID INT,
    FOREIGN KEY (deptID) REFERENCES Department(deptID)
);
```

Commands 2. Commands for creating “student” table

Table 3. Outputs for “student’ table.

stdID	stdName	stdBirth	stdAddress	stdPhoneNbr	stdEmail	deptID
1001	Charlie	1999-07-08	654 Cedar St	5678901234	cha123@gmail.com	1
1002	Diana	1999-09-20	987 Elm St	6789012345	diana111@gmail.com	2
1003	Evan	1995-09-03	135 Fir St	7890123456	evan232@gmail.com	2
1004	Fiona	2002-08-02	246 Grove St	8901234567	fiona434@gmail.com	7
1005	Alice	1998-12-08	321 Birch St	4567890123	alice142@gmail.com	3
1006	Wang	2000-12-01	654 Cedar St	5678901234	wang109@gmail.com	3
1007	Qi	2001-07-03	987 Elm St	6789012345	qi989@gmail.com	4
1008	Mike	1999-04-20	135 Fir St	7890123456	mike2656@gmail.com	4
1009	Nate	2000-06-02	246 Grove St	8901234567	nate5666@gmail.com	5
1010	Vicky	1998-07-02	321 Birch St	4567890123	vicky09090@gmail.com	5
1011	Linda	2000-07-08	654 Cedar St	5678901234	llllinada@gmail.com	6
1012	Lucy	2001-11-12	987 Elm St	6789012345	lucyyyyyy@gmail.com	7
1013	Luna	2002-01-03	135 Fir St	7890123456	lunaluna@gmail.com	8
1014	Mikie	1999-05-18	246 Grove St	8901234567	miiiiikkies@gmail.com	9
1015	Miny	1998-10-15	246 Grove St	8901234567	miny1111@gmail.com	10

```
CREATE TABLE Classroom (
    clsrmID INT PRIMARY KEY,
    building VARCHAR(50),
    roomNbr INT,
    capacity INT
);
```

DOI: 10.54254/2755-2721/40/20230624

```
select * from classroom
```

Commands 3. Commands for creating “Classroom” table

Table 4. Outputs for “classroom” table.

clsrmID	building	roomNbr	capacity
1	A	101	40
2	B	102	50
3	C	201	30
4	D	202	40
5	E	301	50
6	F	302	60
7	G	401	40
8	H	311	50
9	I	300	55
10	J	404	45

8. Example Test for SQL

If we want to list the courses and their classroom, and we also need to see its course name, corresponding building and room number

```
SELECT c.crsID, c.crsName, cls.building, cls.roomNbr
FROM Course c
JOIN Classroom cls ON c.clsrmID = cls.clsrmID;
```

In this case, “c.crsID” is used to retrieve the “crsID” columns from the “course” table. Here, the “Course” table has been aliased as “c” for brevity and to avoid ambiguity. “From Course C” is used to specify the primary table from which you are selecting data. In this example, it is the “Course” table. “JOIN Classroom cls” uses “JOIN” operation, it is instructing the database to combine rows from the “Course” table and the “Classroom” table. And “Classroom” table is given an alias “cls”. “On c.clsrmID=cls.clsrmID” is used to specify the condition for the join, it tells the database how to combine the rows from the “Course” and “Classroom” table.(Table 5)

Table 5. The outputs for courses and their classroom.

crsID	crsName	building	roomNbr
2001	Introduction to Computer Science	A	101
2002	Calculus I	B	102
2003	English Literature	C	201
2004	Physics I	D	202
2005	Introduction to Psychology	E	301
2006	Microeconomics	F	302
2007	Computer Safety	A	101
2008	Calculus II	B	102
2009	American Literature	C	201
2010	Basic Chemistry	D	202

If we want to list the information of the faculty of each course, it should list the name of faculty, the title of faculty, the email of faculty and phone number as well (Table 6).

```
SELECT c.crsID, c.crsName, f.facID, f.facName, f.facTitle, f.facEmail, f.facPhoneNbr
```

DOI: 10.54254/2755-2721/40/20230624

```
FROM Course AS c
JOIN Hire AS h ON c.crsID = h.crsID
JOIN Faculty AS f ON h.facID = f.facID;
```

Table 6. The outputs for the information of the faculty of each course.

crsID	crsName	facID	facName	facTitle	facEmail	facPhoneNbr
2001	Introduction to Computer Science	1	Dr. Smith	Professor	smith@gmail.com	123-456-7890
2001	Introduction to Computer Science	6	Dr. Johnson	Associate Professor	johnson@gmail.com	234-567-8901
2001	Introduction to Computer Science	11	Dr. Taylor	Teacher Assistant	taylor@gmail.com	789-012-3456
2002	Calculus I	2	Dr. Johnson	Associate Professor	johnson@example.com	234-567-8901
2002	Calculus I	7	Prof. Miller	Assistant Professor	miller@gmail.com	345-678-9012
2002	Calculus I	12	Dr. Martinez	Professor	martinez@gmail.com	890-123-4567
2003	English Literature	3	Dr. Williams	Assistant Professor	williams@example.com	345-678-9012
2003	English Literature	8	Dr. Brown	Professor	brown@gmail.com	456-789-0123
2003	English Literature	13	Prof. Thomas	Assistant Professor	thomas@gmail.com	901-234-5678
2004	Physics I	4	Dr. Brown	Lecturer	brown@example.com	456-789-0123
2004	Physics I	9	Prof. Wilson	Assistant Professor	wilson@gmail.com	567-890-1234
2004	Physics I	14	Dr. Harris	Professor	harris@gmail.com	012-345-6789
2005	Introduction to Psychology	5	Dr. Davis	Professor	davis@example.com	567-890-1234
2005	Introduction to Psychology	10	Dr. Anderson	Professor	anderson@gmail.com	678-901-2345
2005	Introduction to Psychology	15	Dr. Mary	Lecturer	maryyy@gmail.com	012-232-6789
2006	Microeconomics	1	Dr. Smith	Professor	smith@gmail.com	123-456-7890
2006	Microeconomics	6	Dr. Johnson	Associate Professor	johnson@gmail.com	234-567-8901
2006	Microeconomics	11	Dr. Taylor	Teacher Assistant	taylor@gmail.com	789-012-3456
2007	Computer Safety	2	Dr. Johnson	Associate Professor	johnson@example.com	234-567-8901
2007	Computer Safety	7	Prof. Miller	Assistant Professor	miller@gmail.com	345-678-9012
2007	Computer Safety	12	Dr. Martinez	Professor	martinez@gmail.com	890-123-4567
2008	Calculus II	3	Dr. Williams	Assistant Professor	williams@example.com	345-678-9012
2008	Calculus II	8	Dr. Brown	Professor	brown@gmail.com	456-789-0123
2008	Calculus II	13	Prof. Thomas	Assistant Professor	thomas@gmail.com	901-234-5678
2009	American Literature	4	Dr. Brown	Lecturer	brown@example.com	456-789-0123
2009	American Literature	9	Prof. Wilson	Assistant Professor	wilson@gmail.com	567-890-1234
2009	American Literature	14	Dr. Harris	Professor	harris@gmail.com	012-345-6789
2010	Basic Chemistry	5	Dr. Davis	Professor	davis@example.com	567-890-1234
2010	Basic Chemistry	10	Dr. Anderson	Professor	anderson@gmail.com	678-901-2345
2010	Basic Chemistry	15	Dr. Mary	Lecturer	maryyy@gmail.com	012-232-6789

If we want to see the courses with average grades above 80

```
SELECT crsID, crsName, avg_Grade
FROM Course
WHERE avg_Grade > 80;
```

In this case, “WHERE avg_grade>80” means that filters the results. The “WHERE” clause is used to filter records. This condition will only retrieve rows from the “Course” table where the value in the “avg_grade” column is greater than 80 (Table 7).

DOI: 10.54254/2755-2721/40/20230624

Table 7. Courses with average grades above 80.

crsID	crsName	avg_Grade
2001	Introduction to Computer Science	85
2005	Introduction to Psychology	83
2007	Computer Safety	85

In summary, effective database queries enable educational institutions to efficiently retrieve and filter critical data. As demonstrated, the aliasing technique helps simplify complex queries, ensuring clarity and avoiding ambiguity. The first query illustrates the advantage of the JOIN operation in combining information from the Courses and Classes tables, providing a clear view of where each course is taught. Finally, the WHERE clause feature is evident in the third query, which enables institutions to set data retrieval criteria, such as identifying high-performing courses based on grade point average. Such targeted queries enable educational entities to make data-driven decisions that leverage optimal resource allocation and enhanced academic management. As the world of education continues to evolve, mastering these database operations remains critical to managing and utilizing the vast amounts of data available. As a database designer, we should be careful and cautious to avoid data duplication and redundancy, while ensuring data integrity.

9. Conclusion

This article mainly discusses the role of a database designer. When designing and generating a database, it is crucial to consider every detail and ensure the correct selection of entities and attributes. This not only determines the structure and efficiency of the database but also impacts the speed of future data queries and operations. Properly defining entities and attributes helps reduce data redundancy and duplication, enhancing the performance and maintainability of the database. Moreover, maintaining data diversity ensures that the database can meet various query needs, rendering the data completer and more accurate.

The article primarily designs a database system for course selection and grade management. This database stores vast amounts of data, including students, courses, staff, teaching buildings, attendance, assignments, exams, and student grades. Its purpose is to collect and tally the scores of every student for each subject. In practice, it can help professors understand the overall performance of their students in their courses, thereby enhancing the quality of teaching. Alternatively, it allows students to clearly understand how their final scores are derived and what they need to do to improve their grades.

In the future, professors can access real-time learning data of students, such as online exam scores and assignment completion rates, allowing them to provide immediate feedback and evaluation. However, with the increase in data volume and growing awareness of personal privacy, database designers must pay greater attention to data security and privacy protection to ensure that the information of students and professors isn't misused or leaked.

References

[1] Connolly, T. M., & Begg, C. E. Database systems: A practical approach to design, implementation and management. *Pearson* 2015, **34**.

[2] MySQL 8.0 Reference Manual: 8.3.2 primary key optimization. MySQL. (n.d.). https://dev.mysql.com/doc/refman/8.0/en/primary-key-optimization.html

[3] P. Kieseberg, S. Schrittwieser, P. Frühwirt and E. Weippl, Analysis of the Internals of MySQL/InnoDB B+ Tree Index Navigation from a Forensic Perspective, International Conference on Software Security and Assurance,2021, 46-51,

[4] Abisoye, O. A., Abisoye, B. O., & Ojonuba, B. E. An Online Outpatient Database System: A Case Study of General Hospital, Minna. Intelligent Information Management, 2016**, 8(4)**, 103–114.

DOI: 10.54254/2755-2721/40/20230624

[5] Al-Hawari, F., Software design patterns for data management features in web-based information systems, *Journal of King Saud University* 2022, **34(10)**:10028-10043
[6] Upadhyay, M.. Second normal form (2NF). GeeksforGeeks. 2023 https://www.geeksforgeeks.org/second-normal-form-2nf/
[7] Guaman, D. Delgado, S. Perez, J., Classifying Model-View-Controller Software Applications Using Self-Organizing Maps, *IEEE Access*, 2021. **9:**45201-45229 2021
[8] Curry, E. Grace, P., 2008, Flexible Self-Management Using the Model-View-Controller Pattern, *IEEE Software*, 2008 **25(3).**
[9] Wen, M.Y. Design and Implementation of Online Ordering System Based on Service. *Journal of Jiamusi Vocational Institute* 2017, 1-10.
[10] Binildas, C. A. Practical microservices architectural patterns: Event-based Java microservices with Spring Boot and Spring Cloud. Apress 2019, **29**.
[11] Jukic N. Vrbsky S. & Nestorov S.. Database systems: introduction to databases and data warehouses. *Prospect Press.* 2017