**VASAVI COLLEGE OF ENGINEERING**

**DATABASE MANAGEMENT SYSTEMS ASSIGNMENT-I**

**NAME : Pasham Jyoshna**

**ROLL NO : 1602-21-737-023**

**SECTION : IT(A)**

**Proctorial Management System**

Proctorial Management System is a software platform that is designed to manage and streamline the proctoring process for educational institutions. This system provides a centralized platform for managing proctoring tasks such as scheduling, assigning proctors, monitoring exams, and reporting. The goal of this system is to make the proctoring process more efficient and effective while reducing the workload of administrative staff.

The system typically consists of two main components: a back-end database and a front-end user interface. The back-end database is responsible for storing data related to proctoring tasks such as exam schedules, proctor assignments, and exam results. The front-end user interface provides a user-friendly interface for administrators, proctors, and students to manage the proctoring process.

Some of the key features of a Proctorial Management System include:

**Exam scheduling:** The system allows administrators to schedule exams, set up exam rules, and assign proctors to specific exams.

Proctor assignment: The system allows administrators to assign proctors to specific exams based on their availability and qualifications.

Proctor monitoring: The system provides tools for proctors to monitor exams in real-time, including live video feeds, chat, and other communication tools.

**Reporting:** The system generates reports on exam results, proctor performance, and other key metrics.

**Security features:** The system provides robust security features to prevent cheating, including identity verification, plagiarism detection, and remote proctoring.

Overall, a Proctorial Management System can help educational institutions streamline the proctoring process, reduce workload, and improve the overall quality of proctoring.

Manage the database of proctorial system in the college

Managing the database of a Proctorial Management System in a college involves several tasks, including designing and creating the database, populating it with data, maintaining data quality, and ensuring data security. Here are some steps that can be followed to manage the database of a Proctorial Management System in a college:

**Identify the data requirements:** The first step is to identify the data that needs to be stored in the database. This may include information about students, proctors, exam schedules, proctor assignments, and exam results. It's important to define data fields and data types for each type of information.

**Design the database:** Based on the data requirements, the next step is to design the database schema. This involves creating tables, defining relationships between tables, and defining primary and foreign keys. A database management system such as MySQL or Oracle can be used to design and create the database.

**Populate the database:** Once the database schema is created, the next step is to populate it with data. This may involve importing data from existing systems or manually entering data. It's important to ensure that data is entered accurately and consistently to maintain data quality.

**Maintain data quality:** To maintain data quality, it's important to establish processes for data validation and data cleaning. This can involve setting up data validation rules, identifying and fixing data errors, and removing duplicate data.

**Ensure data security:** Finally, it's important to ensure data security by implementing access controls, encryption, and backup and recovery procedures. This can help prevent unauthorized access to data and ensure that data is protected in the event of a system failure or disaster.

Overall, managing the database of a Proctorial Management System in a college requires careful planning, attention to detail, and a focus on data quality and security. By following these steps, colleges can ensure that their Proctorial Management System is effective and efficient in managing the proctoring process.

**Abstract**

A Proctorial Management System is a software platform that is designed to manage and streamline the proctoring process for educational institutions. This system provides a centralized platform for managing proctoring tasks such as scheduling, assigning proctors, monitoring exams, and reporting. The goal of this system is to make the proctoring process more efficient and effective while reducing the workload of administrative staff. With features such as exam scheduling, proctor assignment, monitoring, reporting, and security, a Proctorial Management System can help educational institutions streamline the proctoring process, reduce workload, and improve the overall quality of proctoring.

List of design requirements like number of tables, attributes, domain types etc.

The design requirements for a Proctorial Management System will vary depending on the specific needs of the institution. However, here are some general requirements that may be considered when designing the database:

**Student Table:** This table would typically contain attributes such as student ID, name, email, and course details.

**Proctor Table:** This table would typically contain attributes such as proctor ID, name, contact details, and availability.

**Exam Table:** This table would typically contain attributes such as exam ID, date, time, duration, and course details.

**Exam Rules Table:** This table would typically contain attributes such as allowed resources, exam format, and other exam-specific rules.

**Proctor Assignment Table:** This table would typically contain attributes such as exam ID, proctor ID, and assignment details.

**Exam Results Table:** This table would typically contain attributes such as exam ID, student ID, marks obtained, and any other relevant details.

The domain types for these attributes would depend on the specific needs of the institution. For example, the domain type for student ID may be alphanumeric, while the domain type for exam duration may be numeric.

In addition to these tables, there may be additional tables and attributes required to support specific features such as proctor monitoring, reporting, and security. The design of the database should be based on a thorough analysis of the requirements and should follow best practices for database design and management. ER diagram

Unfortunately, as an AI language model, I don't have the capability to create a visual ER diagram. However, I can give you a brief idea of what an ER diagram for a Proctorial Management System may look like based on the design requirements we discussed earlier.

The ER diagram for a Proctorial Management System may include entities such as Student, Proctor, Exam, Exam Rules, Proctor Assignment, and Exam Results. Relationships between these entities would be defined using cardinality and optionality constraints. For example, a student may take multiple exams, but each exam can be taken by only one student. Similarly, a proctor may be assigned to multiple exams, but each exam can be assigned to only one proctor.

The ER diagram would also include primary and foreign keys for each entity and relationship. For example, the primary key for the Student entity may be the student ID attribute, while the foreign key for the Exam entity may be the student ID attribute.

Overall, the ER diagram would provide a visual representation of the database schema and the relationships between entities, helping to ensure that the database is properly designed and implemented. ER diagram, Mapping cardinalities, constraints i.e., participation, key constraints to be labelled out

Here is a description of the ER diagram, mapping cardinalities, and constraints for a Proctorial Management System based on the design requirements we discussed earlier:

**Entities:**

Student (Attributes: Student ID (PK), Name, Email, Course Details)

Proctor (Attributes: Proctor ID (PK), Name, Contact Details, Availability)

Exam (Attributes: Exam ID (PK), Date, Time, Duration, Course Details)

Exam Rules (Attributes: Rule ID (PK), Allowed Resources, Exam Format, Other Exam-specific Rules)

Proctor Assignment (Attributes: Assignment ID (PK), Exam ID (FK), Proctor ID (FK), Assignment Details)

Exam Results (Attributes: Result ID (PK), Exam ID (FK), Student ID (FK), Marks Obtained, Other Relevant Details)

**Mapping Cardinalities:**

Each Student can take zero or many Exams (1 to Many)

Each Proctor can be assigned to zero or many Exams (1 to Many)

Each Exam has only one set of Exam Rules (1 to 1)

Each Exam can have only one Proctor assigned to it, and each Proctor can be assigned to zero or many Exams (1 to Many)

Each Exam can have zero or many Exam Results (1 to Many)

Each Student can have zero or many Exam Results (1 to Many)

**Constraints:**

he Student ID is the primary key for the Student entity

The Proctor ID is the primary key for the Proctor entity

The Exam ID is the primary key for the Exam entity

The Rule ID is the primary key for the Exam Rules entity

The Assignment ID is the primary key for the Proctor Assignment entity

The Result ID is the primary key for the Exam Results entity

The Exam ID is a foreign key in the Proctor Assignment and Exam Results entities, referencing the Exam entity

The Proctor ID is a foreign key in the Proctor Assignment entity, referencing the Proctor entThe Student ID is a foreign key in the Exam Results entity, referencing the Student entity

These constraints help ensure data integrity and maintain consistency between the various entities in the Proctorial Management System.

**ER-DIAGRAM**

College

Proctor

**Students**

**\***

**Examination**

rules

**Proctor\_assignment**

**Exam\_results**

**DDL-OPERATIONS:**

Logical database design-DDL operations (all the tables syntaxes with outputs)

Here is an example of the logical database design for a Proctorial Management System, including DDL operations for creating the necessary tables:

Student Table

CREATE TABLE student (

student\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE,

course\_details VARCHAR(100)

);

Output: Query OK, 0 rows affected

Proctor Table

CREATE TABLE proctor (

proctor\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

contact\_details VARCHAR(100),

availability VARCHAR(50)

);

Output: Query OK, 0 rows affected

Exam Table

CREATE TABLE exam (

exam\_id INT PRIMARY KEY,

date DATE NOT NULL,

time TIME NOT NULL,

duration INT NOT NULL,

course\_details VARCHAR(100)

);

Output: Query OK, 0 rows affected

Exam Rules Table

CREATE TABLE exam\_rules (

rule\_id INT PRIMARY KEY,

allowed\_resources VARCHAR(100),

exam\_format VARCHAR(50),

exam\_rules VARCHAR(200)

);

Output: Query OK, 0 rows affected

Proctor Assignment Table

CREATE TABLE proctor\_assignment (

assignment\_id INT PRIMARY KEY,

exam\_id INT NOT NULL,

proctor\_id INT NOT NULL,

assignment\_details VARCHAR(200),

FOREIGN KEY (exam\_id) REFERENCES exam(exam\_id),

FOREIGN KEY (proctor\_id) REFERENCES proctor(proctor\_id)

);

Output: Query OK, 0 rows affected

Exam Results Table

CREATE TABLE exam\_results (

result\_id INT PRIMARY KEY,

exam\_id INT NOT NULL,

student\_id INT NOT NULL,

marks\_obtained FLOAT,

other\_details VARCHAR(200),

FOREIGN KEY (exam\_id) REFERENCES exam(exam\_id),

FOREIGN KEY (student\_id) REFERENCES student(student\_id)

);

Output: Query OK, 0 rows affected

These DDL operations would create the necessary tables in the Proctorial Management System with the appropriate attributes, primary keys, and foreign keys. The syntax for creating tables may vary depending on the database management system being used. The output for each query confirms that the table was created successfully.

**Enforcing the constraints like primary, foreign key constraints**

To enforce the constraints in the Proctorial Management System, we can use SQL statements to add the necessary primary and foreign key constraints to the tables. Here's an example:

Enforce primary key constraint on the student table:

ALTER TABLE student

ADD CONSTRAINT pk\_student PRIMARY KEY (student\_id);

Enforce primary key constraint on the proctor table:

ALTER TABLE proctor

ADD CONSTRAINT pk\_proctor PRIMARY KEY (proctor\_id);

Enforce primary key constraint on the exam table:

ALTER TABLE exam

ADD CONSTRAINT pk\_exam PRIMARY KEY (exam\_id);

Enforce primary key constraint on the exam\_rules table:

ALTER TABLE exam\_rules

ADD CONSTRAINT pk\_exam\_rules PRIMARY KEY (rule\_id);

Enforce foreign key constraint on the proctor\_assignment table:

ALTER TABLE proctor\_assignment

ADD CONSTRAINT fk\_exam\_proctor

FOREIGN KEY (exam\_id) REFERENCES exam(exam\_id),

ADD CONSTRAINT fk\_proctor\_exam

FOREIGN KEY (proctor\_id) REFERENCES proctor(proctsssor\_id);

Enforce foreign key constraint on the exam\_results table:

ALTER TABLE exam\_results

ADD CONSTRAINT fk\_exam\_results\_exam

FOREIGN KEY (exam\_id) REFERENCES exam(exam\_id),

ADD CONSTRAINT fk\_exam\_results\_student

FOREIGN KEY (student\_id) REFERENCES student(student\_id);

These statements will enforce the primary key constraintsThese statements will enforce the primary key constraints on the respective tables and add foreign key constraints to ensure referential integrity between the tables. The exact syntax may vary depending on the database management system being used, but the general structure and logic of the statements should be similar.

Here are some examples of Data Manipulation Language (DML) operations for the Proctorial Management System, along with their outputs:

Insert data into the student table:

INSERT INTO student (student\_id, name, email, course\_details)

VALUES (1, 'John Doe', 'johndoe@example.com', 'Computer Science');

INSERT INTO student (student\_id, name, email, course\_details)

VALUES (2, 'Jane Doe', 'janedoe@example.com', 'Mechanical Engineering');

INSERT INTO student (student\_id, name, email, course\_details)

VALUES (3, 'Bob Smith', 'bobsmith@example.com', 'Electrical Engineering');

Output: Query OK, 3 rows affected

Insert data into the proctor table:

INSERT INTO proctor (proctor\_id, name, contact\_details, availability)

VALUES (1, 'Alice Brown', 'alicebrown@example.com', 'Monday to Friday, 9am to 5pm');

INSERT INTO proctor (proctor\_id, name, contact\_details, availability)

VALUES (2, 'Charlie Green', 'charliegreen@example.com', 'Tuesday to Saturday, 10am to 6pm');

Output: Query OK, 2 rows affected

Insert data into the exam table:

INSERT INTO exam (exam\_id, date, time, duration, course\_details)

VALUES (1, '2023-05-15', '10:00:00', 120, 'Computer Science');

INSERT INTO exam (exam\_id, date, time, duration, course\_details)

VALUES (2, '2023-05-17', '14:00:00', 180, 'Mechanical Engineering');

Output: Query OK, 2 rows affected

Insert data into the exam\_rules table:

INSERT INTO exam\_rules (rule\_id, allowed\_resources, exam\_format, exam\_rules)

VALUES (1, 'Calculator, ruler', 'Multiple choice', 'No talking allowed');

Output: Query OK, 1 row affected

Update data in the student table:

UPDATE student

SET course\_details = 'Information Technology'

WHERE student\_id = 3;

Output: Query OK, 1 row affected

These DML operations demonstrate how to add and update data in the various tables of the Proctorial Management System. The output for each query confirms that the operation was successful and indicates how many rows were affected.

Execution ScreenShots:

 