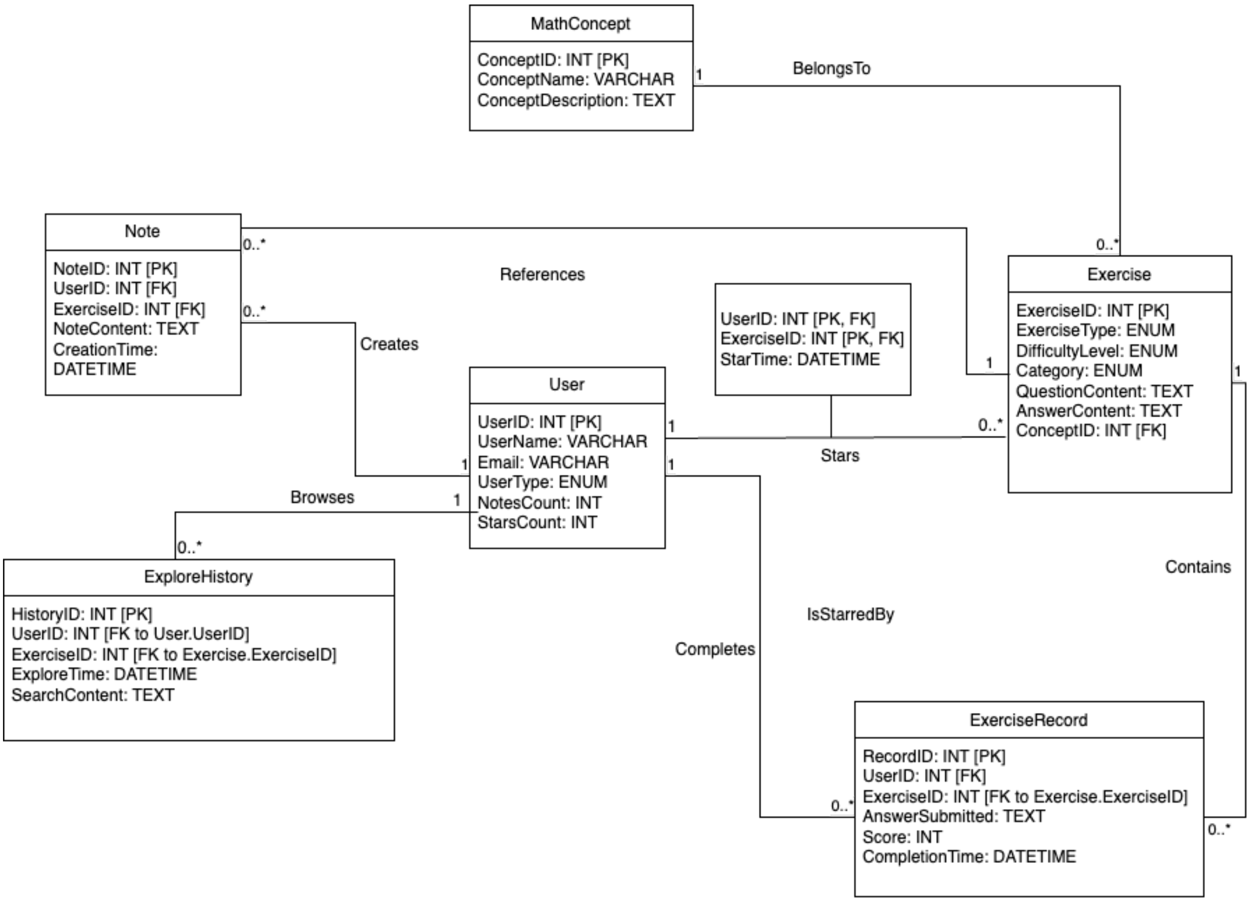
Here is the UML diagram of our database design.



1. Below are the assumptions and explanations for each entity in the mathematics exercise management platform:

User

Assumption: The User entity represents all platform users, including students, teachers, and administrators. Since user information is critical and unique, it is modeled as a separate entity instead of being attributes of other entities.

The User entity stores essential user details like username, email, and user type, as well as the count of notes and stars associated with each user.

MathConcept

Assumption: Math concepts categorize exercises into various domains (e.g., Geometry, Algebra). Each math concept has a distinct name and description, making it necessary to create a separate entity.

This entity enables a many-to-one relationship between exercises and math concepts, facilitating exercise categorization and management.

Exercise

Assumption: Exercises are the core entity of the platform, representing all math problems. Each exercise has attributes such as type, difficulty level, and category, and it is associated with other entities like Note, Star, and ExploreHistory.

Because exercises are linked to multiple entities and have distinct attributes, they are modeled as an independent entity.

Note

Assumption: Notes record a user's thoughts and comments on an exercise. Since notes have unique content and creation time, they are modeled as a separate entity rather than an attribute of the Exercise.

This entity records a 1:N relationship between users and exercises, showing which users have created notes on which exercises.

2.Below is the description of relationships.

User to Note:

Relationship Name: Creates

Cardinality: 1:N (One user can create multiple notes, and each note is created by one user).

User to Star:

Relationship Name: Stars

Cardinality: 1:N (One user can star multiple exercises, but each star entry is associated with one user).

User to ExerciseRecord:

Relationship Name: Completes

Cardinality: 1:N (One user can complete multiple exercises, but each exercise record is linked to one user).

Exercise to MathConcept:

Relationship Name: Belongs to

Cardinality: N:1 (Multiple exercises can belong to one math concept, but each exercise only belongs to one concept).

User to ExploreHistory:

Relationship Name: Browses

Cardinality: 1:N (One user can have multiple browsing history entries, but each entry is linked to one user).

Exercise to Star:

Relationship Name: Is Starred by

Cardinality: M:N (Many users can star many exercises, forming a many-to-many relationship).

Exercise to ExerciseRecord:

Relationship Name: Has

Cardinality: 1:N (One exercise can have multiple exercise records, but each exercise record corresponds to only one exercise).

3.Nomalization

We applied **3NF(Third Normal Form)** to our database.

1.List all Functional Dependencies (FDs):

MathConcept:

ConceptID → ConceptName, ConceptDescription

Note:

NoteID → UserID, ExerciseID, NoteContent, CreationTime

(UserID, ExerciseID) → NoteID

Exercise:

ExerciseID → ExerciseType, DifficultyLevel, Category, QuestionContent, AnswerContent, ConceptID

User:

UserID → UserName, Email, UserType, NotesCount, StarsCount

ExploreHistory:

HistoryID → UserID, ExerciseID, ExploreTime, SearchContent

ExerciseRecord:

RecordID → UserID, ExerciseID, AnswerSubmitted, Score, CompletionTime

(UserID, ExerciseID) → RecordID

2.Calculate Candidate Keys:

MathConcept: ConceptID

Note: NoteID, (UserID, ExerciseID)

Exercise: ExerciseID

User: UserID

ExploreHistory: HistoryID

ExerciseRecord: RecordID, (UserID, ExerciseID)

3.Find Minimal Basis: All functional dependencies can be broken down into their most basic form:

MathConcept:

ConceptID → ConceptName

ConceptID → ConceptDescription

Note:

NoteID → UserID

NoteID → ExerciseID

NoteID → NoteContent

NoteID → CreationTime

(UserID, ExerciseID) → NoteID

Exercise:

ExerciseID → ExerciseType

ExerciseID → DifficultyLevel

ExerciseID → Category

ExerciseID → QuestionContent

ExerciseID → AnswerContent

ExerciseID → ConceptID

User:

UserID → UserName

UserID → Email

UserID → UserType

UserID → NotesCount

UserID → StarsCount

ExploreHistory:

HistoryID → UserID

HistoryID → ExerciseID

HistoryID → ExploreTime

HistoryID → SearchContent

ExerciseRecord:

RecordID → UserID

RecordID → ExerciseID

RecordID → AnswerSubmitted

RecordID → Score

RecordID → CompletionTime

(UserID, ExerciseID) → RecordID

4.Relations (after confirming minimal basis): These functional dependencies confirm that our schema is already in BCNF because:

Each determinant is a candidate key

All non-key attributes are fully dependent on their respective primary keys

No transitive dependencies exist

The schema satisfies 3NF and BCNF requirements without need for further decomposition.

5.Logical design (relational schema)

```
User(
        UserID: INT [PK],
        Username: VARCHAR(50),
        Email: VARCHAR(100),
        UserType: ENUM('Student', 'Teacher', 'Admin'),
        NotesCount: INT,
        StarsCount: INT
)

MathConcept(
        ConceptID: INT [PK],
        ConceptName: VARCHAR(100),
        ConceptDescription: TEXT
)

Exercise(
        ExerciseID: INT [PK],
        ExerciseType: ENUM('Multiple Choice', 'Fill in the Blank', 'Short Answer'),
        DifficultyLevel: ENUM('Very Easy', 'Easy', 'Medium', 'Hard', 'Very Hard'), Category:
        ENUM('Geometry', 'Algebra', 'Calculus', 'Statistics'),
        QuestionContent: TEXT,
        AnswerContent: TEXT,
        ConceptID: INT [FK to MathConcept.ConceptID]
)

Note(
        NoteID: INT [PK],
        UserID: INT [FK to User.UserID],
        ExerciseID: INT [FK to Exercise.ExerciseID],
        NoteContent: TEXT,
        CreationTime: DATETIME
)

Star(
        StarID: INT [PK],
        UserID: INT [FK to User.UserID],
        ExerciseID: INT [FK to Exercise.ExerciseID],
        StarTime: DATETIME
)

ExerciseRecord(
        RecordID: INT [PK],
        UserID: INT [FK to User.UserID],
        ExerciseID: INT [FK to Exercise.ExerciseID],
        AnswerSubmitted: TEXT,
        Score: INT,
        CompletionTime: DATETIME
)

ExploreHistory(
        HistoryID: INT [PK],
        UserID: INT [FK to User.UserID],
        ExerciseID: INT [FK to Exercise.ExerciseID],
        ExploreTime: DATETIME,
        SearchContent: TEXT
)
```