Technokrax Data base outline

## Key functionalities:

Content Creation, User interactions, personalized ai tutoring, real time progress tracking, certificate based on time spent learning, highlighting text, saving snippet, writing notes similar to writing on physical textbook

##### Users:

Represent user of the platform

|  |  |
| --- | --- |
| User Id | Primary key, INT |
| Username | VARCHAR |
| Name | Text |
| Email | VARCHAR |
| PasswordHash | VARCHAR |
| ProfilePictureURL | VARCHAR |
| Bio | TEXT |
| DateJoined | DATETIME |
| LastLogin | DATETIME |
|  |  |

##### Content:

|  |  |
| --- | --- |
| ContentID | Primary key, INT |
| UserID | Foreign key, INT |
| ContentType | ENUM: BLOG, Video, images, news, tools |
| Title | VARCHAR |
| Description | Text |
| ContentURL | VARCHAR |
| ThumbnailURL | VARCHAR |
| DatePublished | DATETIME |
| Status | ENUM: Published, draft, archived |
|  |  |

##### Comments:

|  |  |
| --- | --- |
| CommentID | Primary key, INT |
| ContentID | Foreign key, INT |
| UserID | Foreign key, INT |
| CommentText | Text |
| DatePosted | Date Time |
|  |  |
|  |  |

Learning Progress:

|  |  |
| --- | --- |
| ProgressID | Primary key, INT |
| UserID | Foreign key, INT |
| ContentID | Foreign key, INT |
| TimeSpent | INT |
| LastAccessed | Date Time |
| CompletionStatus | Boolean |
|  |  |

Certificates:

|  |  |
| --- | --- |
| CertificatedID | Primary key, INT |
| UserID | Foreign key, INT |
| AwardedDate | Date Time |
| CertificateType | VARCHAR |
| ValidityPeriod | INT |
| TimeSpentLearning | INT |
|  |  |

User Interactions:

|  |  |
| --- | --- |
| Interaction Id | Primary key, INT |
| User ID | Foreign key, INT |
| Target User ID | Foreign key, INT |
| Content ID | TEXT |
| Interaction Type | Enum: Like,Follow, chat |
| Date of Interaction | DateTime |
|  |  |
|  |  |

AI Tutor Sessions:

|  |  |
| --- | --- |
| TutorSessionId | Primary key, INT |
| User ID | Foreign key, INT |
| Session Start Time | DATETIME |
| SessionEndTime | DATETIME |
| TopicsCovered | Text |
| Feedback | Text |
|  |  |

Tags:

|  |  |
| --- | --- |
| TagID | Primary key, INT |
| TagName | VARCHAR |
|  |  |

Content Tags Mapping:

|  |  |
| --- | --- |
| ContentTagMappingID | Primary key, INT |
| UserID | Foreign key, INT |
| TagID | Foreign key, INT |
|  |  |

User Topic Time:

Track the time a user spends on specific topics

|  |  |
| --- | --- |
| UserTopicTimeID | Primary key, INT |
| UserID | Foreign key, INT |
| TopicID | Foreign key, INT |
| TimeSpent | INT |
| LastUpdate | DATETIME |
|  |  |
|  |  |

TOPICs

Store information about topics that can be linked to content and learning progress

|  |  |
| --- | --- |
| TopicID | Primary key, INT |
| TopicName | VARCHAR |
| Description | TEXT |
|  |  |
|  |  |

Heighlights:

Stores information about specific text highlights made by users within content

|  |  |
| --- | --- |
| HeighlightID | Primary key, INT |
| ContentID | Foreign Key, INT |
| UserID | Foreign Key, INT |
| ParaNumber | INT |
| ~~StartLineNumberX~~ | INT |
| Range of highlighted text | INT |
| HighlighText | TEXT |
| DateHighlighted | DATETIME |
|  |  |

Save Snippets:

Store Snippets or excepts saved by users from articles or content.

|  |  |
| --- | --- |
| SnippetID | Primary key, INT |
| ContentID | Foreign Key, INT |
| UserID | Foreign Key, INT |
| ParaNumber | INT |
| ~~StartLineNumber~~ | INT |
| Range of Snipped text | INT |
| SnippetText | TEXT |
| DateSaved | DATETIME |
|  |  |

USERNOTES:

Allow user to write and save notes.

|  |  |
| --- | --- |
| NoteID | Primary key, INT |
| ContentID | Foreign Key, INT |
| UserID | Foreign Key, INT |
| ParaNumber | INT |
| StartLineNumber | INT |
| NoteText | TEXT |
| NoteWrittenDate | DATETIME |
|  |  |

Graph date base

## Graph Database:

Graph database is generally used in the scenario with complex relationships and networked data, such as in our case the connection between users, notes, tags content. Graph databse excel in managing highly connected data and provide efficient querying for complex relationships, which is the current requirement for our platform.

EXAMPLE:

NODES:

USER

NOTE  
CONTENT

TAGS

EDGES:

**CREATED**: Between **User** and **Note** to denote which user created the note.

**TAGGED\_WITH**: Between **Note** and **Tag** to denote which tags are associated with a note.

**REFERENCES**: Between **Note** and **Content** to denote which content a note is related to (if any).

**FOLLOWS**: Between **User** nodes to represent the social connections or interests a user has in other users’ activities.

Use Case Implementation

**Searching Notes by User and Tag**:

When a user searches for notes with specific tags, the graph query starts at the User node, traverses the CREATED relationships to find Note nodes, and then filters those notes by traversing TAGGED\_WITH relationships to only include notes with the specified tags.

Advantages:

**Efficiency**: Graph databases are designed for efficient traversal of connected data, making it much faster to query for notes by tags, especially when compared to traditional relational databases where such queries might involve complex joins and subqueries.

**Flexibility**: Easily add new types of relationships or nodes (e.g., new content types, new user interactions) without significant redesigns of the schema.

**Intuitive Modeling**: The graph model is often more intuitive for representing and querying complex, interconnected data found in social networks, learning platforms, etc.

Implementation Tools:

**Neo4j**: A popular graph database that offers powerful querying capabilities through its Cypher query language.

**Amazon Neptune**: A fully managed graph database service that supports both property graph and RDF models.

**ArangoDB**: A multi-model database that supports graph, document, and key/value data models, suitable for a variety of use cases.

Relationships:

Users to Content: One-to-Many (A user can create multiple content pieces)

Content to Comments: One-to-Many (A content piece can have multiple comments)

Users to Comments: One-to-Many (A user can make multiple comments)

Users to Learning Progress: One-to-Many (A user can have multiple learning records)

Users to Certificates: One-to-Many (A user can earn multiple certificates)

Users to User Interactions: One-to-Many (A user can have multiple interactions)

Users to AI Tutor Sessions: One-to-Many (A user can have multiple tutor sessions)

Content to Content Tags Mapping: One-to-Many (A content piece can have multiple tags)

**UserTopicTime to Topics**: One-to-Many relationship (A topic can have time tracked by multiple users).

**Content to Topics**: Many-to-Many relationship, which could be implemented using a junction table if you want to directly associate content with specific topics.

**Highlights and Snippets to Content**: One-to-Many relationship (A content piece can have multiple highlights and snippets made by users).

**UserNotes**: Can optionally be related to specific content (hence ContentID is nullable) allowing flexibility for notes to be general or content-specific.

Top of Form

Bottom of Form

###### KEY considetation

**Time Tracking for Learning**: The **UserTopicTime** table will be crucial for implementing logic to award certificates based on time spent learning. We will need to implement backend logic to periodically update **TimeSpent** as users engage with content related to specific topics.

**Highlighting and Snippets**: Features for highlighting text and saving snippets will require frontend implementation to allow users to select text and choose actions (highlight/save). The backend will store these actions in the respective tables.

**Notes**: Implementing a note-taking feature will involve creating a user-friendly interface that allows users to jot down notes as they go through content. Consider adding functionalities like tagging notes for better organization.