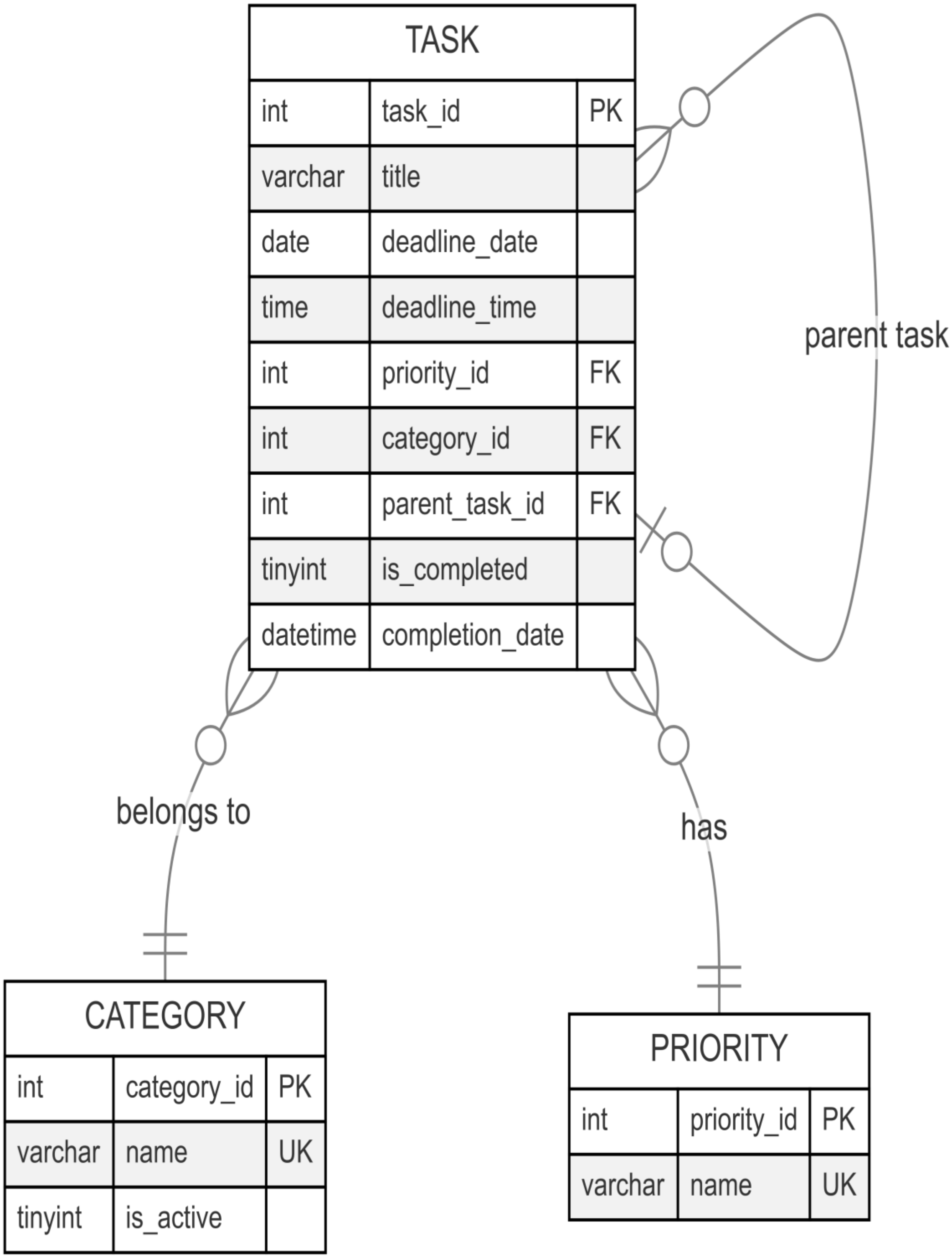
1. Entity-Relationship Diagram (ERD)

- **Tasks ↔ Categories**
- **Type:** Many-to-One
- **Description:** Each task belongs to one category, and a category can have multiple tasks.
- **Foreign Key:** Tasks.category_id references Categories.category_id.
- **Tasks ↔ Priorities**
- **Type:** Many-to-One
- **Description:** Each task has one priority level, and a priority can be assigned to multiple tasks.
- **Foreign Key:** Tasks.priority_id references Priorities.priority_id.
- **Tasks ↔ Tasks (Self-Referencing)**
- **Type:** One-to-Many
- **Description:** A task can have multiple subtasks (children), and each subtask has one parent task.
- **Foreign Key:** Tasks.parent_task_id references Tasks.task_id.



TASK		
int	task_id	PK
varchar	title	
date	deadline_date	
time	deadline_time	
int	priority_id	FK
int	category_id	FK
int	parent_task_id	FK
tinyint	is_completed	
datetime	completion_date	

CATEGORY		
int	category_id	PK
varchar	name	UK
tinyint	is_active	

PRIORITY		
int	priority_id	PK
varchar	name	UK

2. Relationships and Interactions

a. Tasks and Categories

- **Relationship:** Each task is associated with one category (category_id), establishing a **Many-to-One** relationship.
- **Behavior:**
    - **Insertion:** When adding a new task, a valid category_id must be provided.
    - **Deletion:** Deleting a category should consider whether to cascade delete associated tasks or restrict deletion if tasks exist. **Recommendation:** Use ON DELETE RESTRICT to prevent accidental deletion of categories that are in use.
    - **Update:** Updating a category’s details (like name) will reflect in all associated tasks.

b. Tasks and Priorities

- **Relationship:** Each task is assigned one priority level (priority_id), forming a **Many-to-One** relationship.
- **Behavior:**
    - **Insertion:** When creating a task, a valid priority_id must be specified.
    - **Deletion:** Like categories, deleting a priority should handle existing tasks. **Recommendation:** Use ON DELETE RESTRICT to avoid orphaned tasks.
    - **Update:** Changing a priority’s name will automatically update it across all related tasks.

c. Tasks and Subtasks (Self-Referencing)

- **Relationship:** A task can have multiple subtasks, creating a **One-to-Many** self-referencing relationship.
- **Behavior:**
  - **Insertion:** When adding a subtask, the parent_task_id must reference an existing task_id.
  - **Deletion:** Deleting a parent task can lead to cascading deletions of its subtasks or restrict deletion if subtasks exist. **Recommendation:** Use ON DELETE CASCADE if subtasks should be automatically removed when a parent task is deleted, or ON DELETE SET NULL to retain subtasks without a parent.
  - **Update:** Changing a parent task's details (like is_completed) might need to propagate to subtasks, depending on business logic.

d. Categories and Priorities

- **Relationship:** These tables are independent of each other.
- **Behavior:**
  - **Categories:** Manage the classification of tasks.
  - **Priorities:** Define the urgency or importance of tasks.

3. Analysis of Database Design

a. Normalization

The current design adheres to the principles of **Third Normal Form (3NF)**:

1. **First Normal Form (1NF):** All tables have atomic (indivisible) values, and each field contains only a single value.
2. **Second Normal Form (2NF):** All non-key attributes are fully functional dependent on the primary key.

3. **Third Normal Form (3NF):** There are no transitive dependencies; non-key attributes depend only on the primary key.

b. Referential Integrity

- **Foreign Keys:** Properly established to maintain consistency between related tables.
- **Constraints:**
- **Primary Keys:** Ensure each record is uniquely identifiable.
- **Unique Constraints:** Enforce uniqueness on name fields in Categories and Priorities to prevent duplicates.
- **Not Null Constraints:** Enforce essential fields to be filled, ensuring data completeness.

c. Data Types and Constraints

- **Data Types:** Appropriately chosen to match the nature of the data (e.g., VARCHAR for names and titles, DATE and TIME for deadlines, INT for identifiers).
- **Constraints:**
- is_completed **and** is_active**:** Use TINYINT(1) as boolean flags to represent binary states efficiently.
- completion_date**:** Nullable to allow tasks to be incomplete without a completion timestamp.

d. Indexing

- **Primary Keys:** Automatically indexed for efficient retrieval.

- **Foreign Keys (**priority_id**,** category_id**,** parent_task_id**):** Indexed (MUL indicates multiple indexes), facilitating quick joins and lookups.
- **Unique Constraints on** name **Fields:** Implicitly indexed to enforce uniqueness and speed up searches.

e. Self-Referencing in Tasks

- **Use Case:** Enables the creation of hierarchical task structures, allowing tasks to have nested subtasks.