Database Schema Documentation

Entity Relationship Diagram

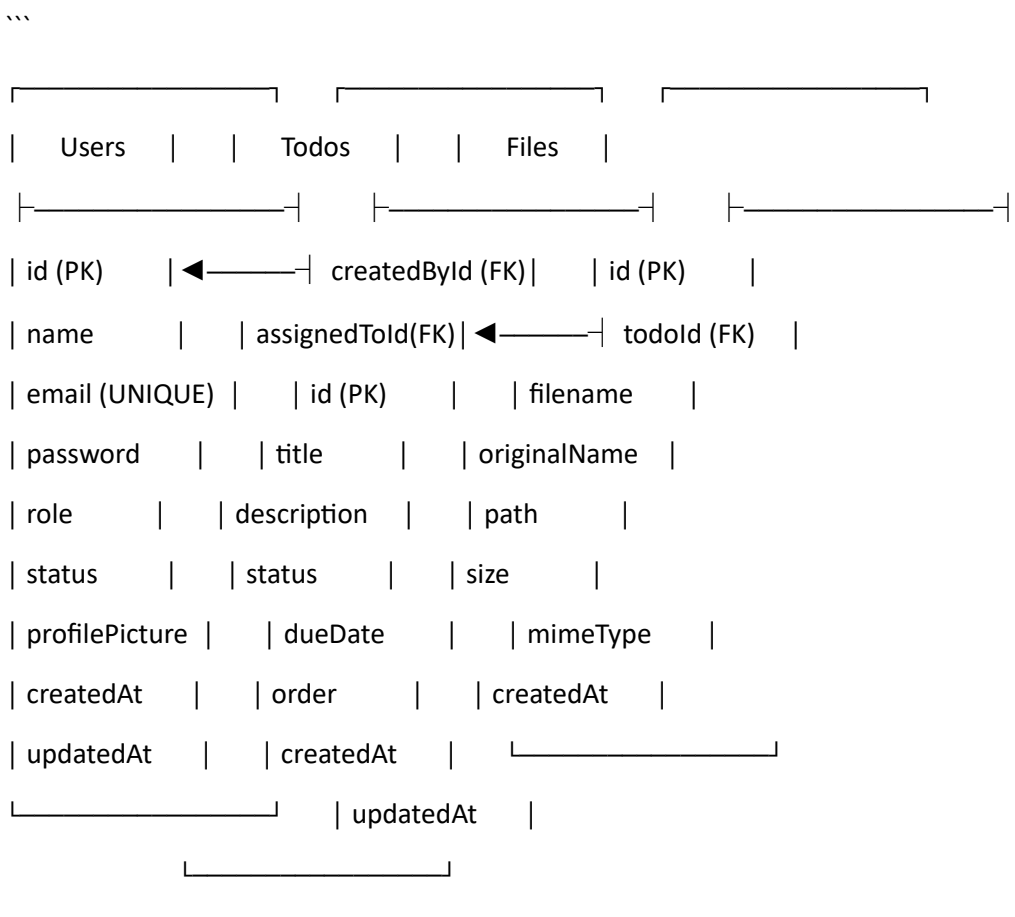

Table Definitions

Users Table

The Users table stores all user account information and authentication data.

Table Name: `users`

Column	Type	Constraints	Description

| id | UUID | PRIMARY KEY | Unique identifier for each user |

| name | VARCHAR(255) | NOT NULL | User's full name |

| email | VARCHAR(255) | NOT NULL, UNIQUE | User's email address (used for login) |

| password | VARCHAR(255) | NOT NULL | Hashed password using bcrypt |

| role | ENUM | NOT NULL, DEFAULT 'USER' | User role: 'USER' or 'ADMIN' |

| status | ENUM | NOT NULL, DEFAULT 'ACTIVE' | Account status: 'ACTIVE' or 'DISABLED' |

| profilePicture | VARCHAR(500) | NULL | Path to user's profile picture |

| createdAt | TIMESTAMP | NOT NULL, DEFAULT NOW() | Account creation timestamp |

| updatedAt | TIMESTAMP | NOT NULL, DEFAULT NOW() | Last update timestamp |

Indexes:

- Primary key on `id`
- Unique index on `email`
- Index on `status` for filtering active users
- Index on `role` for admin queries

Constraints:

- Email must be valid format
- Password must be at least 6 characters (enforced at application level)
- Role must be either 'USER' or 'ADMIN'
- Status must be either 'ACTIVE' or 'DISABLED'

Todos Table

The Todos table stores all todo items with their metadata and relationships.

Table Name: `todos`

| Column | Type | Constraints | Description |

|--------|------|-------------|-------------|

| id | UUID | PRIMARY KEY | Unique identifier for each todo |

| title | VARCHAR(255) | NOT NULL | Todo title/summary |

| description | TEXT | NULL | Detailed description of the todo |

| status | ENUM | NOT NULL, DEFAULT 'PENDING' | Todo status: 'PENDING', 'IN_PROGRESS', 'COMPLETED' |

| dueDate | TIMESTAMP | NULL | Optional due date for the todo |

| order | INTEGER | NOT NULL, DEFAULT 0 | Display order for sorting todos |

| createdAt | TIMESTAMP | NOT NULL, DEFAULT NOW() | Todo creation timestamp |

| updatedAt | TIMESTAMP | NOT NULL, DEFAULT NOW() | Last update timestamp |

| createdById | UUID | NOT NULL, FOREIGN KEY | Reference to user who created the todo |

| assignedToId | UUID | NOT NULL, FOREIGN KEY | Reference to user assigned to the todo |

Indexes:

- Primary key on `id`
- Index on `createdById` for filtering todos by creator
- Index on `assignedToId` for filtering todos by assignee
- Index on `status` for filtering by status
- Index on `dueDate` for sorting by due date
- Index on `order` for sorting todos
- Composite index on `(assignedToId, status)` for efficient filtering

Foreign Key Constraints:

- `createdById` references `users(id)` ON DELETE CASCADE
- `assignedToId` references `users(id)` ON DELETE CASCADE

Constraints:

- Title cannot be empty
- Status must be one of: 'PENDING', 'IN_PROGRESS', 'COMPLETED'
- Order must be non-negative integer
- CreatedById and assignedToId must reference valid users

Files Table

The Files table stores metadata for all files attached to todos.

Table Name: `files`

Column	Type	Constraints	Description
id	UUID	PRIMARY KEY	Unique identifier for each file
filename	VARCHAR(255)	NOT NULL	Generated filename on disk
originalName	VARCHAR(255)	NOT NULL	Original filename from upload
path	VARCHAR(500)	NOT NULL	Full path to file on disk
size	INTEGER	NOT NULL	File size in bytes
mimeType	VARCHAR(100)	NOT NULL	MIME type of the file
createdAt	TIMESTAMP	NOT NULL, DEFAULT NOW()	File upload timestamp
todoId	UUID	NOT NULL, FOREIGN KEY	Reference to associated todo

Indexes:
- Primary key on `id`
- Index on `todoId` for retrieving files by todo
- Index on `mimeType` for filtering by file type
- Index on `createdAt` for sorting by upload date

Foreign Key Constraints:
- `todoId` references `todos(id)` ON DELETE CASCADE

Constraints:
- Filename and originalName cannot be empty
- Size must be positive integer
- Path must be valid file system path
- MimeType must be valid MIME type format

Relationships

User to Todos (One-to-Many)
- **Created Todos**: One user can create multiple todos
- **Assigned Todos**: One user can be assigned multiple todos
- **Self-Assignment**: A user can assign todos to themselves

Todo to Files (One-to-Many)
- One todo can have multiple file attachments
- Each file belongs to exactly one todo
- Files are deleted when their associated todo is deleted

User to Files (Indirect Many-to-Many)
- Users can access files through their created or assigned todos
- File access is controlled by todo permissions

Data Access Patterns

Common Queries

1. **Get todos assigned to a user:**
   ```sql
   SELECT * FROM todos WHERE assignedToId = ? ORDER BY order ASC;
   ```

2. **Get todos created by a user:**
   ```sql
   SELECT * FROM todos WHERE createdById = ? ORDER BY createdAt DESC;
   ```

3. **Get todos with file count:**

```sql
SELECT t.*, COUNT(f.id) as fileCount
FROM todos t
LEFT JOIN files f ON t.id = f.todoId
GROUP BY t.id;
```

4. **Get files for a todo:**

```sql
SELECT * FROM files WHERE todoId = ? ORDER BY createdAt ASC;
```

5. **Get active users for assignment:**

```sql
SELECT id, name, email FROM users WHERE status = 'ACTIVE' ORDER BY name;
```

Performance Considerations

1. **Indexing Strategy:**
   - Composite indexes on frequently queried column combinations
   - Separate indexes on foreign keys for JOIN operations
   - Indexes on enum columns for filtering

2. **Query Optimization:**
   - Use LIMIT and OFFSET for pagination
   - Include only necessary columns in SELECT statements
   - Use JOINs instead of multiple queries when possible

3. **Data Archival:**
   - Consider soft deletes for audit trails

- Archive completed todos older than certain period
- Implement file cleanup for orphaned files

Security Considerations

Data Protection
- Passwords are hashed using bcrypt with salt rounds ≥ 12
- Sensitive data is not logged in application logs
- File paths are validated to prevent directory traversal

Access Control
- Row-level security through application logic
- Users can only access their created or assigned todos
- Admins have full access to all data
- File access is restricted to todo participants

Data Validation
- All inputs are validated at application level
- SQL injection prevention through parameterized queries
- File upload restrictions by type and size
- Email format validation for user accounts

Backup and Recovery

Backup Strategy
- Daily full database backups
- Transaction log backups every 15 minutes
- File system backups for uploaded files
- Cross-region backup replication

Recovery Procedures

- Point-in-time recovery capability
- File restoration from backup storage
- Database consistency checks after recovery
- Application-level data validation post-recovery

Migration History

Version 1.0.0 (Initial Schema)
- Created users, todos, and files tables
- Established foreign key relationships
- Added basic indexes for performance

Future Migrations
- Add notification preferences to users table
- Implement todo categories/tags
- Add file versioning support
- Implement audit logging tables