# **University Magazine Database Documentation**

## **Database Overview**

This PostgreSQL database manages a university magazine submission system with role-based access control. It tracks student contributions across faculties, user roles, comments, and system settings with comprehensive security policies.

## **Table Definitions**

### **1. Users Table**

**Description:** Stores all user accounts with authentication details and role/faculty associations.

CREATE TABLE users (  
 user\_id SERIAL PRIMARY KEY,  
 first\_name VARCHAR(50) NOT NULL,  
 last\_name VARCHAR(50) NOT NULL,  
 email VARCHAR(100) NOT NULL UNIQUE,  
 password VARCHAR(255) NOT NULL,  
 created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 last\_login TIMESTAMP,  
 faculty\_id VARCHAR(10) REFERENCES faculties(faculty\_id),  
 role\_id VARCHAR(10) NOT NULL REFERENCES roles(role\_id),  
 CONSTRAINT valid\_email CHECK (email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')  
);  
  
-- Indexes:  
-- PRIMARY KEY (user\_id)  
-- UNIQUE (email)  
-- Additional index on email (idx\_users\_email)  
-- Foreign keys to faculties and roles tables

**Columns:**

* user\_id: Auto-incrementing primary key.
* first\_name, last\_name: User identification (50 characters max).
* email: Validated email address (100 characters max, unique).
* password: Hashed authentication (255 characters).
* faculty\_id: Foreign key referencing the faculties table.
* role\_id: Foreign key referencing the roles table.

### **2. Roles Table**

**Description:** Defines system roles and permissions.

CREATE TABLE roles (  
 role\_id VARCHAR(10) PRIMARY KEY,  
 role\_name VARCHAR(50) NOT NULL,  
 description TEXT NOT NULL  
);

**Columns:**

* role\_id: Primary key (e.g., 'ADMIN', 'STUDENT').
* role\_name: Human-readable name.
* description: Detailed permissions description.

### **3. Faculties Table**

**Description:** Academic departments and divisions.

CREATE TABLE faculties (  
 faculty\_id VARCHAR(10) PRIMARY KEY,  
 faculty\_name VARCHAR(100) NOT NULL,  
 description TEXT  
);

**Columns:**

* faculty\_id: Primary key.
* faculty\_name: Name of the faculty (100 characters max).
* description: Description of the faculty.

### **4. Submissions Table**

**Description:** Student contributions to the magazine.

CREATE TABLE submissions (  
 submission\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 faculty\_id VARCHAR(10) NOT NULL REFERENCES faculties(faculty\_id),  
 title VARCHAR(200) NOT NULL,  
 description TEXT,  
 file\_path VARCHAR(255) NOT NULL,  
 file\_type VARCHAR(50) NOT NULL,  
 submitted\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 last\_updated TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 status submission\_status NOT NULL DEFAULT 'Submitted',  
 terms\_accepted BOOLEAN NOT NULL DEFAULT false,  
 academic\_year VARCHAR(10) NOT NULL,  
 selected BOOLEAN NOT NULL DEFAULT false  
);  
  
-- Indexes:  
-- PRIMARY KEY (submission\_id)  
-- Indexes on faculty\_id, status, and user\_id  
-- Trigger: update\_submission\_timestamp

**Columns:**

* submission\_id: Auto-incrementing primary key.
* user\_id: Foreign key referencing users table.
* faculty\_id: Foreign key referencing faculties table.
* title: Title of the submission (200 characters max).
* description: Description of the submission.
* file\_path: Path to the uploaded file (255 characters max).
* file\_type: Type of the uploaded file (50 characters max).
* submitted\_at: Timestamp when the submission was created.
* last\_updated: Timestamp when the submission was last updated.
* status: Current status of the submission (using submission\_status enum).
* terms\_accepted: Boolean indicating if terms were accepted.
* academic\_year: Academic year of the submission.
* selected: Boolean indicating if the submission was selected.

**Status Values (enum type):**

* Pending
* Under\_review
* Commented
* Selected
* Rejected
* Submitted

## **Security Implementation**

### **Row-Level Security Policies**

-- Restricts students to their own submissions  
CREATE POLICY student\_submission\_policy ON submissions  
 FOR ALL USING (user\_id = current\_user\_id());  
  
-- Restricts faculty coordinators to their faculty's submissions  
CREATE POLICY faculty\_coordinator\_policy ON submissions  
 FOR ALL USING (faculty\_id::text = current\_user\_faculty\_id()::text);

### **Security Functions**

-- Returns current user's ID  
CREATE FUNCTION current\_user\_id() RETURNS integer  
 LANGUAGE plpgsql STABLE  
 AS $$BEGIN RETURN NULLIF(current\_setting('jwt.claims.user\_id', true), '')::integer; END;$$;  
  
-- Returns current user's faculty ID  
CREATE FUNCTION current\_user\_faculty\_id() RETURNS character varying  
 LANGUAGE plpgsql STABLE  
 AS $$BEGIN RETURN NULLIF(current\_setting('jwt.claims.faculty\_id', true), ''); END;$$;

## **Tracking & Analytics**

### **ActivityLogs Table**

CREATE TABLE activitylogs (  
 log\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER NOT NULL REFERENCES users(user\_id),  
 action\_type VARCHAR(50) NOT NULL,  
 page\_accessed VARCHAR(100),  
 browser\_info VARCHAR(255),  
 ip\_address VARCHAR(45),  
 log\_timestamp TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP  
);

**Columns:**

* log\_id: Auto-incrementing primary key.
* user\_id: Foreign key referencing users table.
* action\_type: Type of action performed.
* page\_accessed: Page accessed by the user.
* browser\_info: Browser information.
* ip\_address: IP address of the user.
* log\_timestamp: Timestamp of the log.

### **Page\_Visits Table**

CREATE TABLE page\_visits (  
 visit\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER REFERENCES users(user\_id) ON DELETE SET NULL,  
 page\_url VARCHAR(255) NOT NULL,  
 visit\_timestamp TIMESTAMP NOT NULL DEFAULT now(),  
 browser\_info VARCHAR(255),  
 ip\_address VARCHAR(45),  
 time\_spent INTEGER  
);

**Columns:**

* visit\_id: Auto-incrementing primary key.
* user\_id: Foreign key referencing users table.
* page\_url: URL of the visited page.
* visit\_timestamp: Timestamp of the visit.
* browser\_info: Browser information.
* ip\_address: IP address of the user.
* time\_spent: Time spent on the page.

## **Database Relationships**

### **Foreign Key Map**

| **Table** | **Column** | **References** | **On Delete** |
| --- | --- | --- | --- |
| comments | submission\_id | submissions(submission\_id) | CASCADE |
| comments | user\_id | users(user\_id) | RESTRICT |
| submissions | user\_id | users(user\_id) | RESTRICT |
| users | faculty\_id | faculties(faculty\_id) | RESTRICT |
| users | role\_id | roles(role\_id) | RESTRICT |

## **Appendix: Complete Object Listing**

### **Tables**

* users
* roles
* faculties
* submissions
* comments
* activitylogs
* page\_visits
* logs
* pagestatistics
* userbrowserstats
* admissions
* Various backup tables

### **Functions**

* current\_user\_id()
* current\_user\_faculty\_id()
* update\_last\_modified()

### **Custom Types**

* submission\_status (enum)

### **Triggers**

* update\_submission\_timestamp

**Note:** The backup tables suggest a migration from integer to string faculty IDs, which appears to be complete.