Educational Resource Distribution Database System Documentation

Github link;

https://github.com/Ivan-Keli/Database-Systems-Jan-April-2025-EC

Group Members

190614-Abdijabbar kanyare

192255-Masud Abdi

145581-Ivan Keli

1. Project Idea Definition

Detailed Explanation

The Educational Resource Distribution Database System addresses the problem of inefficient and inequitable distribution of educational materials, particularly to rural schools in Kenya. Many schools lack essential learning materials such as textbooks, laptops, and other educational resources. This issue is exacerbated by poor tracking systems, leading to uneven distribution, misallocation, and sometimes even loss of resources.

The system provides a centralized database to track the flow of educational materials from suppliers to schools, ensuring transparency and accountability in the distribution process. By monitoring resource allocation, we can identify underserved schools and ensure they receive necessary educational materials.

This project aligns with SDG 4: Quality Education, which aims to "ensure inclusive and equitable quality education and promote lifelong learning opportunities for all." In the Kenyan context, where educational disparities between urban and rural areas remain significant, this system will help bridge the gap by ensuring resources reach schools that need them most.

Scope and Objectives

The scope of the project covers:

- Tracking educational materials (books, laptops, stationery) from suppliers to schools
- Managing information about schools, resources, suppliers, and distributions
- Monitoring distribution status and delivery confirmation
- Generating reports on resource allocation and identifying distribution gaps

Specific objectives include:

1. Create a normalized database structure for storing information about schools, resources, suppliers, and distributions
2. Develop an API for CRUD operations on the database
3. Implement a user interface for managing the distribution process
4. Provide visualization tools for analyzing distribution patterns
5. Enable stakeholders to track resource allocation efficiently

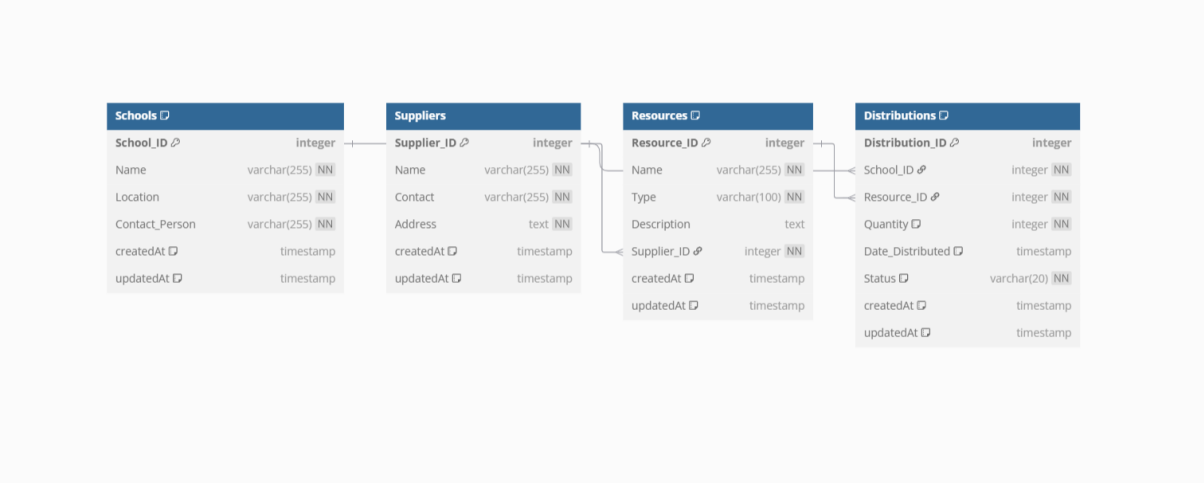
Stakeholders

The main stakeholders include:

1. **Government Education Departments**: Monitor resource allocation across schools, ensuring equitable distribution according to policies and priorities. The system provides them with oversight capabilities and data for planning.
2. **Schools**: Track incoming resources and report on their needs. Schools can use the system to confirm receipt of materials and request additional resources where necessary.
3. **Suppliers**: Manage their inventory and track delivery of educational materials to schools. The system helps suppliers coordinate with schools and education departments.
4. **NGOs/Donors**: Organizations that fund educational resources can use the system to ensure their donations reach the intended beneficiaries and track the impact of their contributions.
5. **Students**: As end beneficiaries, students benefit from improved access to learning materials, enhancing their educational experience.

2. Entity-Relationship Diagram (ERD)

Design



Entities and Attributes

1. **Schools**

- School_ID (Primary Key)
- Name
- Location
- Contact_Person

2. **Resources**
    - Resource_ID (Primary Key)
    - Name
    - Type
    - Description
    - Supplier_ID (Foreign Key)
3. **Suppliers**
    - Supplier_ID (Primary Key)
    - Name
    - Contact
    - Address
4. **Distributions**
    - Distribution_ID (Primary Key)
    - School_ID (Foreign Key)
    - Resource_ID (Foreign Key)
    - Quantity
    - Date_Distributed
    - Status

Relationships

1. **Suppliers to Resources: One-to-Many**
    - A supplier can provide multiple resources
    - Each resource is supplied by exactly one supplier
2. **Schools to Distributions: One-to-Many**

- A school can receive multiple distributions
- Each distribution is received by exactly one school

3. **Resources to Distributions: One-to-Many**
    - A resource can be included in multiple distributions
    - Each distribution includes exactly one type of resource

3. Database Schema

Normalization

The database schema follows normalization principles to eliminate redundancy:

- **First Normal Form (1NF):** All attributes contain atomic values
- **Second Normal Form (2NF):** All non-key attributes are fully dependent on the primary key
- **Third Normal Form (3NF):** No transitive dependencies exist

Tables and Fields

```
-- Schools Table
CREATE TABLE Schools (
   School_ID SERIAL PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
   Location VARCHAR(255) NOT NULL,
   Contact_Person VARCHAR(255) NOT NULL,
   "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Suppliers Table
CREATE TABLE Suppliers (
   Supplier_ID SERIAL PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
```

```
    Contact VARCHAR(255) NOT NULL,
    Address TEXT NOT NULL,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Resources Table
CREATE TABLE Resources (
    Resource_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Type VARCHAR(100) NOT NULL,
    Description TEXT,
    Supplier_ID INTEGER NOT NULL,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Supplier_ID) REFERENCES Suppliers(Supplier_ID)
);

-- Distributions Table
CREATE TABLE Distributions (
    Distribution_ID SERIAL PRIMARY KEY,
    School_ID INTEGER NOT NULL,
    Resource_ID INTEGER NOT NULL,
    Quantity INTEGER NOT NULL CHECK (Quantity > 0),
    Date_Distributed TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (Status IN ('pending',
'in_transit', 'delivered', 'cancelled')),
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (School_ID) REFERENCES Schools(School_ID),
```

```
    FOREIGN KEY (Resource_ID) REFERENCES Resources(Resource_ID)
);
```

Indexes

```
-- Indexes for frequently queried columns
CREATE INDEX idx_distributions_school ON Distributions(School_ID);
CREATE INDEX idx_distributions_resource ON Distributions(Resource_ID);
CREATE INDEX idx_resources_supplier ON Resources(Supplier_ID);
CREATE INDEX idx_schools_location ON Schools(Location);
```

4. SQL Code Implementation

Database Creation

```
-- Create the database
CREATE DATABASE education_db;

-- Connect to the database
\c education_db;

-- Create the tables
CREATE TABLE Schools (
    School_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Location VARCHAR(255) NOT NULL,
    Contact_Person VARCHAR(255) NOT NULL,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Suppliers (
```

```
    Supplier_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Contact VARCHAR(255) NOT NULL,
    Address TEXT NOT NULL,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Resources (
    Resource_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Type VARCHAR(100) NOT NULL,
    Description TEXT,
    Supplier_ID INTEGER NOT NULL,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Supplier_ID) REFERENCES Suppliers(Supplier_ID)
);

CREATE TABLE Distributions (
    Distribution_ID SERIAL PRIMARY KEY,
    School_ID INTEGER NOT NULL,
    Resource_ID INTEGER NOT NULL,
    Quantity INTEGER NOT NULL CHECK (Quantity > 0),
    Date_Distributed TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (Status IN ('pending',
'in_transit', 'delivered', 'cancelled')),
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (School_ID) REFERENCES Schools(School_ID),
```

```
    FOREIGN KEY (Resource_ID) REFERENCES Resources(Resource_ID)
);

-- Create indexes
CREATE INDEX idx_distributions_school ON Distributions(School_ID);
CREATE INDEX idx_distributions_resource ON Distributions(Resource_ID);
CREATE INDEX idx_resources_supplier ON Resources(Supplier_ID);
CREATE INDEX idx_schools_location ON Schools(Location);
```

Data Manipulation

Create (Insert) Operations

```
-- Insert a school
INSERT INTO Schools (Name, Location, Contact_Person)
VALUES ('Central High School', 'Nairobi', 'John Kamau');

-- Insert a supplier
INSERT INTO Suppliers (Name, Contact, Address)
VALUES ('Kenya Publishers Ltd', '+254700123456', 'Industrial Area, Nairobi');

-- Insert a resource
INSERT INTO Resources (Name, Type, Description, Supplier_ID)
VALUES ('Mathematics Textbook Grade 10', 'Book', 'Standard mathematics textbook for grade 10', 1);

-- Insert a distribution
INSERT INTO Distributions (School_ID, Resource_ID, Quantity, Status)
VALUES (1, 1, 100, 'pending');
```

Read (Select) Operations

```
-- Get all schools
```

```
SELECT * FROM Schools;

-- Get a specific school
SELECT * FROM Schools WHERE School_ID = 1;

-- Get all resources by a specific supplier
SELECT * FROM Resources WHERE Supplier_ID = 1;

-- Get all distributions for a specific school
SELECT * FROM Distributions WHERE School_ID = 1;
```

Update Operations

```
-- Update school information
UPDATE Schools
SET Contact_Person = 'Jane Wanjiku'
WHERE School_ID = 1;

-- Update distribution status
UPDATE Distributions
SET Status = 'delivered'
WHERE Distribution_ID = 1;
```

Delete Operations

```
-- Delete a distribution
DELETE FROM Distributions WHERE Distribution_ID = 1;

-- Delete a resource
DELETE FROM Resources WHERE Resource_ID = 1;
```

Advanced Queries

Join Query to Get Distribution Details with School and Resource Names

```
SELECT
    d.Distribution_ID,
    s.Name AS School_Name,
    r.Name AS Resource_Name,
    d.Quantity,
    d.Date_Distributed,
    d.Status
FROM
    Distributions d
JOIN
    Schools s ON d.School_ID = s.School_ID
JOIN
    Resources r ON d.Resource_ID = r.Resource_ID
ORDER BY
    d.Date_Distributed DESC;
```

Aggregate Query to Find Total Resources Distributed by School

```
SELECT
    s.School_ID,
    s.Name AS School_Name,
    s.Location,
    SUM(d.Quantity) AS Total_Resources_Received
FROM
    Schools s
JOIN
    Distributions d ON s.School_ID = d.School_ID
WHERE
```

```
    d.Status = 'delivered'
GROUP BY
    s.School_ID, s.Name, s.Location
ORDER BY
    Total_Resources_Received DESC;
```

Subquery to Find Schools with No Distributions

```
SELECT
    s.School_ID,
    s.Name,
    s.Location
FROM
    Schools s
WHERE
    s.School_ID NOT IN (
        SELECT DISTINCT School_ID
        FROM Distributions
    );
```

Query to Find Resource Distribution by Type

```
SELECT
    r.Type,
    COUNT(d.Distribution_ID) AS Distribution_Count,
    SUM(d.Quantity) AS Total_Quantity
FROM
    Resources r
JOIN
    Distributions d ON r.Resource_ID = d.Resource_ID
GROUP BY
```

```
    r.Type
ORDER BY
    Total_Quantity DESC;
```

Query to Find Distribution Trends over Time

```
SELECT
    DATE_TRUNC('month', d.Date_Distributed) AS Month,
    COUNT(d.Distribution_ID) AS Distribution_Count,
    SUM(d.Quantity) AS Resources_Distributed
FROM
    Distributions d
GROUP BY
    DATE_TRUNC('month', d.Date_Distributed)
ORDER BY
    Month;
```