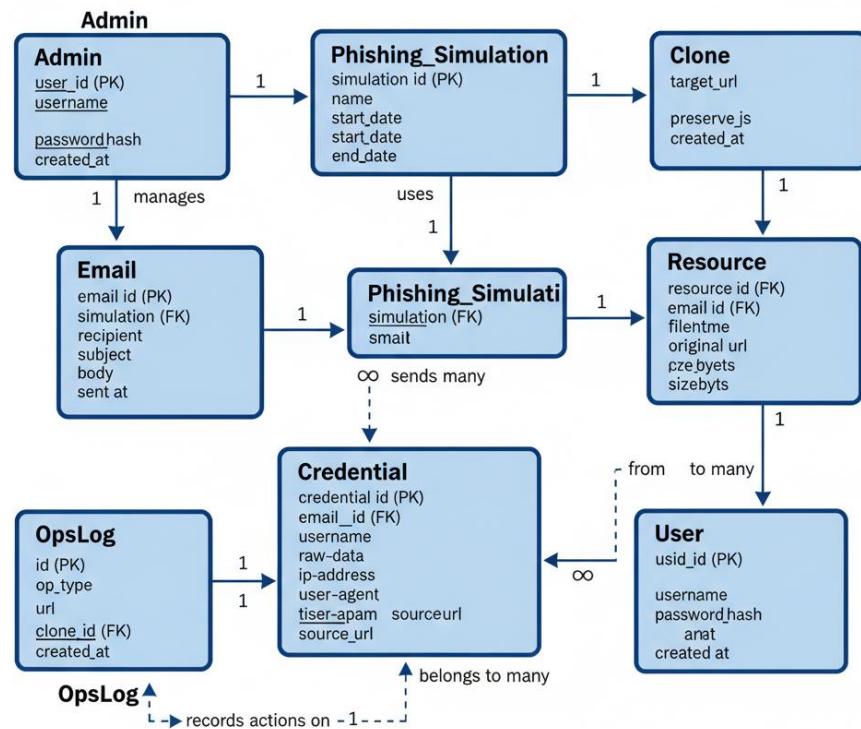


1. ER Diagram



2. Relational Model (SQL)

Based on the entities and the provided schema, here is an analysis and proposal for adding **Foreign Keys (FKs)** to enforce the relationships implied by the previous ER Diagram and the table attributes:

Relational Schema Analysis and Foreign Key Enforcement

The current schema defines the tables and Primary Keys (PKs), but **most Foreign Keys are missing** (except in the resources table), meaning relationships between tables are not enforced by the database.

Here are the proposed modifications to add necessary FK constraints:

1. credentials Table

The credentials table has clone_id, which links it to the clone's table.

SQL

Original Table:

```
CREATE TABLE credentials (
    credential_id TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))),
    clone_id      TEXT, -- Should be FK
    source_url    TEXT,
    username      TEXT,
    password      TEXT,
    raw_data      JSON,
    ip_address    TEXT,
    user_agent    TEXT,
    timestamp     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Proposed Update (Adding FK):

```
CREATE TABLE credentials (
    credential_id TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))),
    clone_id      TEXT,
    source_url    TEXT,
    username      TEXT,
    password      TEXT,
    raw_data      JSON,
    ip_address    TEXT,
    user_agent    TEXT,
    timestamp     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (clone_id) REFERENCES clones(clone_id) ON DELETE SET NULL
);
```

2. ops_log Table

The ops_log table has clone_id, which links it to the clones table.

SQL

Original Table:

```
CREATE TABLE ops_log (
    id      TEXT PRIMARY KEY,
    op_type  TEXT,
    url     TEXT,
    clone_id TEXT, -- Should be FK
    note    TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Proposed Update :

```
CREATE TABLE ops_log (
    id      TEXT PRIMARY KEY,
    op_type  TEXT,
    url     TEXT,
    clone_id TEXT,
    note    TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (clone_id) REFERENCES clones(clone_id) ON DELETE SET NULL
);
```

3. resources Table

The resources table already correctly defines the Foreign Key to clones:

SQL

Current Schema (Correctly defines FK):

```
CREATE TABLE resources (
```

```
resource_id TEXT PRIMARY KEY,  
clone_id TEXT REFERENCES clones(clone_id) ON DELETE CASCADE,  
filename TEXT,  
original_url TEXT,  
size_bytes INTEGER  
);
```

SQL

Users

```
CREATE TABLE users (  
    user_id TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))),  
    username TEXT NOT NULL UNIQUE,  
    password_hash TEXT NOT NULL,  
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP  
);
```

Clones

```
CREATE TABLE clones (  
    clone_id TEXT PRIMARY KEY,  
    target_url TEXT,  
    preserve_js BOOLEAN DEFAULT 0,  
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP  
);
```

Credentials

```
CREATE TABLE credentials (  
    credential_id TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))),  
    clone_id TEXT,  
    source_url TEXT,
```

```
username    TEXT,  
password    TEXT,  
raw_data    JSON,  
ip_address  TEXT,  
user_agent  TEXT,  
timestamp   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
-- FK to Clones  
FOREIGN KEY (clone_id) REFERENCES clones(clone_id) ON DELETE SET NULL  
);
```

Ops Log

```
CREATE TABLE ops_log (  
id        TEXT PRIMARY KEY,  
op_type   TEXT,  
url       TEXT,  
clone_id  TEXT,  
note      TEXT,  
created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
-- FK to Clones  
FOREIGN KEY (clone_id) REFERENCES clones(clone_id) ON DELETE SET NULL  
);
```

3. Pseudo-code (High-level Workflow)

Here is the pseudo-code for the high-level workflow, structured logically and expanding on the brief points provided in the prompt.

1. Application Initialization & Setup

Code snippet

```
FUNCTION INITIALIZE_APPLICATION

    // Configuration and Utilities
    LOAD CONFIGURATION FROM "config.json"

    ENSURE_DIRECTORY_EXISTS("cloned_sites/")
    ENSURE_DIRECTORY_EXISTS("logs/")

    // Database Connection
    CONNECT_TO_DATABASE()

    IF DATABASE_SCHEMA_MISSING THEN
        EXECUTE_SCHEMA_SETUP_SCRIPTS()
    END IF

    // Define Utility Functions
    DEFINE UTILITY is_safe_url(url)
    DEFINE UTILITY ssrf_public_host(url) // Proxy requests
    DEFINE UTILITY download_resource(url, destination_path)
    DEFINE UTILITY sanitize_html(html_content)

END FUNCTION
```

2. Core Phishing Campaign Workflow

2.1. CLONE PAGE

Code snippet

```
FUNCTION CLONE_PAGE(target_url, preserve_js=FALSE) RETURNS clone_id

    // Validation
```

```

IF NOT IS_VALID_URL(target_url) OR NOT is_safe_url(target_url) THEN
    RETURN ERROR "Invalid or Unsafe Target URL"
END IF

// Setup
GENERATE clone_id // e.g., UUID
CREATE DIRECTORY "cloned_sites/" + clone_id

// Fetch and Process
html_content = FETCH_URL(target_url, ssrf_public_host)

// Sanitization & Transformation
transformed_html = NORMALIZE_FORM_ACTIONS(html_content, "/login?clone_id=" + clone_id)
IF NOT preserve_js THEN
    transformed_html = REMOVE_UNSAFE_JAVASCRIPT(transformed_html)
END IF
transformed_html = UPDATE_RESOURCE_PATHS(transformed_html, clone_id)

// Resource Download
resource_list = EXTRACT_RESOURCES(transformed_html)
FOR EACH resource_url IN resource_list DO
    local_path = download_resource(resource_url, "cloned_sites/" + clone_id)
    INSERT_RESOURCE_RECORD(clone_id, local_path, resource_url)
END FOR

// Final Save and Logging
SAVE transformed_html TO "cloned_sites/" + clone_id + "/index.html"
INSERT_CLONE_RECORD(clone_id, target_url, preserve_js)
LOG_OPERATION("CLONE_CREATED", target_url, clone_id, "Page cloned successfully.")

```

```
    RETURN clone_id
```

```
END FUNCTION
```

2.2. CAPTURE CREDENTIALS (POST Handler)

Code snippet

```
FUNCTION HANDLE_CREDENTIAL_POST(request)
```

```
    // Extract Metadata
```

```
    username = request.FORM.username
```

```
    password = request.FORM.password
```

```
    clone_id = request.QUERY.clone_id
```

```
    ip_address = request.METADATA.IP
```

```
    user_agent = request.METADATA.UserAgent
```

```
    raw_data = request.FORM.raw // Full POST body
```

```
    // Store Credential
```

```
    INSERT_CREDENTIAL_RECORD(clone_id, username, password, raw_data, ip_address, user_agent)
```

```
    // User Feedback (Simulated Phishing Notice)
```

```
    REDIRECT_USER_TO("/notice/phishing_simulated") // Or show a static "logged in" page
```

```
END FUNCTION
```

2.3. ANALYZE URL (Phishing Detection Module)

Code snippet

```
FUNCTION ANALYZE_URL_RISK(url) RETURNS risk_score
```

```
    risk_score = 0
```

```
    // Feature Checks
```

```
    IF DOMAIN_AGE(url) < 90 days THEN risk_score += 20 END IF
```

```
    IF NOT HAS_VALID_SSL(url) THEN risk_score += 15 END IF
```

```
    IF CHECK_EXTERNAL_BLACKLISTS(url) THEN risk_score += 50 END IF
```

```

// Heuristic Checks

legit_url = GET_KNOWN_LEGIT_URL(url) // e.g., for google.com

IF LEVENSHTEIN_DISTANCE(url, legit_url) < 3 THEN risk_score += 25 END IF

IF CONTAINS_HIDDEN_FORM_FIELDS(url) THEN risk_score += 10 END IF

LOG_DETECTION(url, risk_score)

RETURN risk_score

END FUNCTION

```

3. User Interface & Administration

3.1. ADMIN UI

Code snippet

```

LOOP ADMIN_SESSION

DISPLAY MENU: (1) New Clone, (2) View Credentials, (3) Analytics, (4) Export, (5) Delete

GET USER_CHOICE

CASE USER_CHOICE OF

1: PROMPT target_url, preserve_js

clone_id = CLONE_PAGE(target_url, preserve_js)

DISPLAY "Clone created with ID: " + clone_id

2: DISPLAY_TABLE_FROM_DB("credentials") // Filterable by clone_id

3: GENERATE_ANALYTICS_REPORT() // E.g., attempts per day, top agents

4: EXPORT_DATA_TO_CSV("credentials", "ops_log")

5: PROMPT item_type (clone/credential/log), item_id

DELETE_FROM_DB(item_type, item_id)

OTHER: DISPLAY "Invalid choice"

END CASE

END LOOP

```

3.2. USER (Phishing Detection Service)

Code snippet

```
LOOP USER_SESSION

// Sign-up/Login

IF NOT USER_LOGGED_IN THEN

    HANDLE_USER_AUTH() // Signup/login against 'users' table

    CONTINUE

END IF


// URL Analysis Interface

PROMPT url_to_check

IF url_to_check IS PROVIDED THEN

    score = ANALYZE_URL_RISK(url_to_check)

    IF score > THRESHOLD THEN

        DISPLAY "WARNING: This URL is highly suspicious. Risk Score: " + score

    ELSE

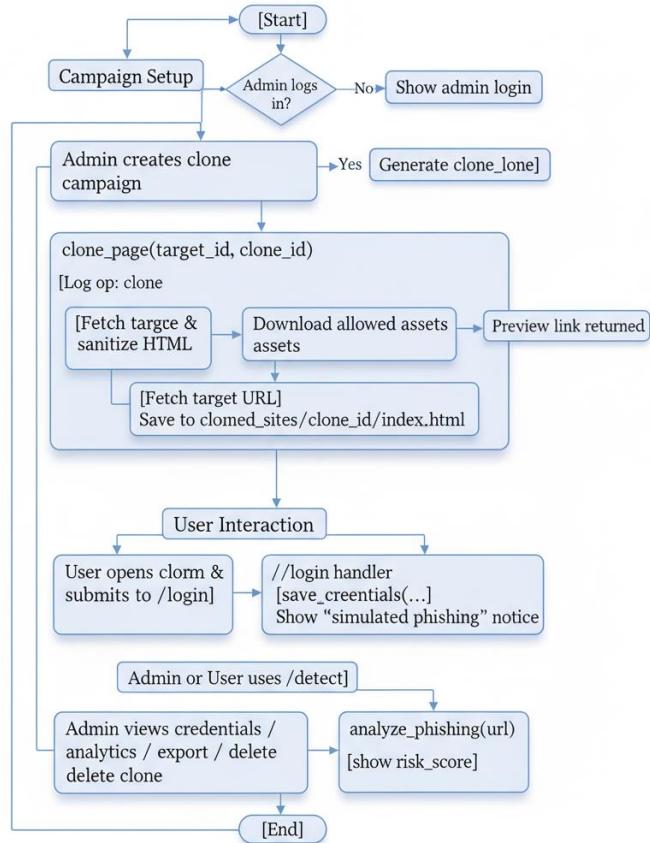
        DISPLAY " URL appears safe (Low Risk Score: " + score + ")"

    END IF

END IF

END LOOP
```

4. Flowchart



Mermaid Flowchart

