# National Teachers College

629 J Nepomuceno, Quiapo, Manila, 1001 Metro Manila
Bachelor of Science in Information Technology


Final Project Documentation Database Management w/ Laboratory

***"SDG 5.1: End all forms of discrimination against all women and girls everywhere"***

**A Final Project**
Presented to the Faculty of the
College of Information Technology

In partial fulfillment
of the Course Requirements for the degree
of Bachelor of Science in Information Technology


**Submitted by:**
*Andrada, Jack Zyruz*
*Condino, Zyann Kyle*
*Dela Cruz, Harvey*
*Macabangon, John Carlo*
*Monte, Manuel*

**Instructor:**
*Ms. Justin Louise R. Neypes*


**Date:**
*December 15, 2025*

## Table of Contents

**I. Introduction** 2

1.1. Project Overview & UN SDG Target 2

1.2. Problem Statement 2

**II. Requirements & Analysis** 3

2.1. Functional Requirements and Non-Functional Requirements 3

2.2. Data Requirements 6

2.3. Schema Normalization Analysis 6

**III. Design Specification** 8

3.1. Core DBMS Concepts Used 8

3.2. ER Diagram: 14

3.3. Transaction Flowchart: 19

**IV. Testing and Results** 20

4.1. Test Cases: 20

4.2. ACID Compliance Test 21

**V. Conclusion and Contribution** 25

5.1. Conclusion 25

5.2. Individual Contributions 26

# I. Introduction

### 1.1. Project Overview & UN SDG Target

Women and girls represent half of the world's population and therefore also half of its potential. But gender inequality persists everywhere and stagnates social progress. Sexual violence and exploitation, the unequal division of unpaid care and domestic work, and discrimination in public office, all remain huge barriers. At the rate of progress as of 2023, it would take an estimated 300 years to end child marriage, 286 years to close gaps in legal protection and remove discriminatory laws, 140 years for women to be represented equally in positions of power and leadership in the workplace, and 47 years to achieve equal representation in national parliaments. (Martin, 2025).

This project utilizes a Relational Database Management System (RDBMS) as its primary instrument, centered on **SDG 5.1: Achieving gender equality and empowering all women and girls**. Through a structured and data-driven approach, the project seeks to examine the opportunity disparities between men and women in workplace environments, evaluate how current systems contribute to these gaps, and develop realistic, feasible, and sustainable solutions aligned with the goals of SDG 5.

### 1.2. Problem Statement

By utilizing data and tables in SQL, this project aims to present a comprehensive overview of the disparities between men and women in the workplace. Through systematically organized datasets, we can display information such as employee demographics, qualifications, work roles, and levels of opportunity across different companies. Once the data is gathered, we then perform comparative analysis to clearly identify gaps in treatment, opportunities, and role distribution.

In our case, the system also incorporates a mechanism that allows employees to file complaints regarding gender-based bias or unfair treatment. When a complaint is filed, the system queries a corresponding table that contains predefined consequences and company policies relevant to that issue. This structured and automated approach ensures that complaints are handled consistently, transparently, and fairly.

Through this RDBMS-driven model, the project seeks not only to identify gender disparities but also to propose a systematic, enforceable solution that supports SDG 5 by promoting fairness, accountability, and equal opportunity in the workplace.

# II. Requirements & Analysis

### 2.1. Functional Requirements and Non-Functional Requirements

**Table 1: Functional Requirements (FR)**

| ID | Requirement | Alignment | Compliance |
|---|---|---|---|
| **FR1** | The system must successfully load and store input data into the fully designed relational schema. | DDL/DML | The data inserted in the normalized table is correctly implemented with a proper relationship that connects them. |
| **FR2** | The system must implement the **three minimum mandatory DBMS concepts** correctly. | Core Concepts | The Tables, Views function, and lastly constraints are applied correctly and effectively. |
| **FR3** | System must execute a **transactional operation** (e.g., recording a major event) using a Stored Procedure that explicitly enforces **ACID properties**. | Finals Concepts | Stored Procedures with the usage of BEGIN, COMMIT, ROLLBACK, as well as Trigger in order to validate data to show and maintain ACID Properties. |
| **FR4** | The system must generate the required SDG reports via VIEWS or DQL and display the results clearly in the chosen interface. | DQL/Reporting | Reports show disparities between men and women which connect to SDG 5 and complaints of each gender. |

**Table 2: Non-Functional Requirements (NFR)**

| ID | Requirement | Metric | Compliance |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **NFR1** | **Robustness:** Program/Scripts must handle invalid input/operations gracefully without violating database integrity. | Error messages should be informative. | CHECK and Triggers is the one responsible in blocking invalid input and message indications with clear delivery of error presented. |
| **NFR2** | **Maintainability:** Schema and stored logic code must be modular, well-commented, and adhere to SQL best practices. | Use of comments and logical VIEW definitions. | The scripts as well as the VIEWS are visible,maintained and well presented . |
| **NFR3** | **Performance:** The three required reports (FR4) must execute efficiently. | Reports should return results in under 1 second on standard hardware with test data. | The queries run effectively and as quickly as possible on sample data. |

### 2.2. Data Requirements

The system utilizes a dataset that is precise and complete, which is able to represent both the entities and their attributes clearly. The key tables are employee, company, workplace_complaint and rules. Out of these, the employee table consists of the largest records which are 50 (25 males, 25 females) generated by ChatGPT. With these tables, the system maps gender distribution, opportunity gaps and complaint trends in various workplace settings.

The employee table has 50 records as it helps provide a realistic, manageable, and useful sample size of workplace demographics. The dataset deliberately contains a higher proportion of men in positions of power with lower educational qualifications and a higher proportion of women in positions of power with higher educational qualifications, as seen in organizations that operate in the real world. This balance allows for useful analysis and prevents the creation of fake data that does not reflect workplace inequality norms.

In the same way, there are only 2 records present in the company table which gives a close estimate of the companies used in this case study. As a result, this avoids inflating the data unnecessarily, and ensure logical and factual relationships between employees and their organization

The workplace_complaint table contains 42 records. These records represent real life scenarios. Women statistically file more complaints related to promotion and harassment. These

patterns are similar to gender issues documented in workplaces internationally, reinforcing the need to examine differential treatment in workplaces.

Several rules are enforced to ensure the consistency and validity of the data. For example, the sex attribute only accepts 'M' or 'F'. complaint_type can be either promotion, harassment and termination. The preventative mitigations in place will stop data from being wrongly entered, ensuring complaint records are all valid, clear and analyzable. In a complaint, an employee should exist and a company should exist. The employee and company column of a complaint table will be the candidate for foreign key. This will maintain accuracy and reflection of realism in the above-mentioned context so that the database can assist in the investigation of gender inequality in the workplace in a meaningful manner.

**3. Employee**

| Attribute | Data Type | Description |
|---|---|---|
| Employee_id (PK) | Integer | Unique identifier for the employee |
| Name | String | Full Name of the employee |
| Age | Integer | Age of the employee |
| Academical_Achievement | String | Highest academic level achieved |
| Sex | Char (1) | Gender of the employee (“M” or “F”) |
| Job_Title | String | Job role or position of the employee |
| Date_Hired | Date | The date the employee was hired |
| Department_id (FK) | Integer | Identifier for the department the employee belongs to (Foreign Key). |
| Company_id (FK) | Integer | Identifier for the company the employee works for (Foreign Key). |

**Table 4. Company**

| Attribute Name | Data Type | Description |
|---|---|---|
| Company_id (PK) | Integer | Unique identifier for the company |
| Company_Name | String | Full name of the company |

| | | |
|---|---|---|
| Department | String | Department or branch of the company |
| Industry | String | Industry category of the company |

**Table 5. Department**

| Attribute Name | Data Type | Description |
|---|---|---|
| Department_ID (PK) | Integer | Unique identifier for the department |
| Company_ID (FK) | Integer | Foreign key linking the department to the company |
| Department_Name | String | Name of the department within the company |

**Table 6. Rules**

| Attribute Name | Data Type | Description |
|---|---|---|
| Rules_ID (PK) | Integer | Unique identifier for a rule or policy |
| Policy_Name | String | Name of the workplace rule or guideline |
| Category | String | Category of the policy (e.g., Equality,Benefits) |

**Table 7. Workplace_Complaint**

| Attribute Name | Data Type | Description |
|---|---|---|
| Complaint_ID (PK) | Integer | Unique identifier for the complaint |
| Employee_ID (FK) | Integer | Employee filling the complaint |
| Complaint_Type | String | Types of issues (‘Promotion’, ‘Harassment’, ‘Termination’) |
| Complaint_Date | Date | Date when the complaint was filed |
| Status | String | Current status of the complaint (e.g., Filed, Investigating, Pending, Resolved) |
| Resolution_Notes | String | Notes describing how the complaint was resolved or |

| | | |
|---|---|---|
| | | handled |

### 2.3. Schema Normalization Analysis

The tables presented in our database are strictly ensured or validated that it follows the standard Data normalization which is specifically the Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF).

In the tables that we use that includes employee, company, department, rules, and lastly workplace_complaint uses a single unique ID which is the Primary Key that determines all other attributes. Which clearly indicates that no **Partial Dependencies** exist between tables and no **Transitive Dependencies** occur because all non-key attributes are dependent only on the primary key. This concludes that the database remains normalized, consistent, no redundancy occurred, and lastly efficient for querying and updating.

| Major Entity/Table Name | Primary Key | Functional Dependency | Normalization Level |
|---|---|---|---|
| company | company_id | company_id<br>● All attributes depend only on company_id and no transitive or partial dependencies occur. | BCNF |
| department | department_id | department_id<br>● The department_id which is the primary key is the determinant of all attributes. No dependencies occur. | BCNF |
| employee | employee_id | employee_id<br>● The primary key is the one that determines all of | 3NF/BCNF |

| | | | |
|---|---|---|---|
| | | the details of the employee, which means no transitive or partial dependencies occurred. | |
| rules | rules_id | rules_id<br>● The primary key is the one that all non-key attributes depend on which means no repeated or derived attributes can be seen. | BCNF |
| workplace_complaint | complaint_id | complaint_id<br>● All complaint records are unique ID because complaint_id is its primary key and no partial and transitive dependencies occurred. | 3NF/BCNF |

# III. Design Specification

### 3.1. Core DBMS Concepts Used (The Five):

In order for the data to be accurate, relations to hold correctly, reports to be efficient, and databases to operate reliably, the DBMS concepts which were taught during the Prelim, Midterm, and Finals were applied to the development of the Gender Equality and Workplace Complaint Monitoring System based on UN SDG 5. DBMS concepts let us store data about the employee and complaints in a structured way. They also ensure that valid relationships are preserved, and that the data may be further analyzed to derive meaningful conclusions about such events that occur in the workplace which are gender-based.

In the early stage, the base of the database was created using DDL and DML. DDL was used to create the core tables in the database. The tables created included employee, company, department, rules and workplace_complaint. The attributes and primary keys were also defined. These tables were filled with realistic records for employees, organizations, workplace policies,

and filed complaints using DML. When applied together, they prepare the database to hold data in an organized manner and in structured form.

In the midterm phase, the constraints Primary Key and Foreign Key were deployed to maintain referential integrity among related tables. All nuisance records correspond to an employee and an applicable workplace rule, these restrictions ensure. Because of this, records that are invalid and not connected are prevented and cascading update/delete actions help to maintain consistency each time their referenced data change.

The principles of Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF) were followed to design the database. Each table has a single primary key, and all other attributes are directly dependent on the primary key. This method avoids repetition and does not allow repetition; that is, the non-prime attribute must be dependent on a super key. For example, rather than putting employee terms and conditions or any other filing or documentation in the same table, separate tables are used. Similarly, for keeping complaints, another table is used.

Also, Transaction Control and ACID properties were applied when the database was created and data loading. When operations are enclosed within START TRANSACTION and COMMIT statements the system ensures that all the changes are executed as a single reliable unit. If an error arises, all update work is rolled back to protect the database.

In the end, the implementation of primary keys and foreign keys brought indexing to better the query. The columns that get accessed a lot are indexed like employee_id and rules_id which helps in joining and executing analytical and reporting queries faster. This makes sure there's a prompt indicator of complaint patterns and policy-related problems. So this helps keep track of workplace gender equality.

**Justification and Implementation Details**

In this section, below are the justifications and implementations in a detailed way of the features of the database system. It also shows or highlights the applications of triggers, views, and stored procedures in order to validate data, recording and the completeness as well as reliability of data to be inputted in the database.

1. **TRIGGER**

In our database, the use of TRIGGER in the workplace_complaint table is preferable rather than CHECK constraint to validate the status of the complaint that strictly needs to be either of the four available statuses which are pending, filed, investigating, and resolved. Other than that, invalid input will occur and TRIGGER will take place and error messages will appear.

In addition, if an update happens where the resolve status with a resolution note is changed to either filed, pending, or investigating the resolution note will become null because the status changed from resolved to another. In summary, TRIGGER is the one that ensures the accuracy and consistency are applied in the data and prevents incorrect data entries in the database.

**TRIGGER FOR STATUS COLUMN**

**SQL CODE**

```
DELIMITER $$

CREATE TRIGGER validate_status_on_update
BEFORE UPDATE ON workplace_complaint
FOR EACH ROW
BEGIN
      IF FIND_IN_SET(NEW.status, 'pending,filed,investigating,resolved') = 0 THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'ERROR: Status must be one of "pending, filed,
investigating," or "resolved" only.';
   END IF;

    IF OLD.status = 'resolved' THEN
      IF NEW.status != 'resolved' THEN
         SET NEW.resolution_notes = NULL;
      END IF;
   END IF;
END$$

DELIMITER ;
```

**TRIGGER FOR WORKPLACE_COMPLAINT TABLE**

**SQL CODE**

```
DELIMITER $$

CREATE TRIGGER clear_notes_on_reopen
BEFORE UPDATE ON workplace_complaint
FOR EACH ROW
BEGIN
```

```
        IF OLD.status = 'resolved' THEN
            IF NEW.status != 'resolved' THEN

                    SET NEW.resolution_notes = NULL;
         END IF;
       END IF;
    END$$

    DELIMITER ;
```

## 2. VIEW

The system uses VIEWS to simplify complex queries that involve multiple tables such as workplace_complaint, employee, and rules. Instead of repeatedly writing long SQL statements with joins and conditions, a VIEW stores the query and allows us to retrieve the required information instantly. In this system, VIEWS are used to display organized complaint records including employee details, complaint status, and resolution notes.

This makes it easier to monitor complaints that are pending, investigating, or resolved without rewriting the same query each time. Overall, VIEWS help improve efficiency by making data access faster, cleaner, and more organized, especially when working with complex and frequently used queries.

**VIEW#1 EMPLOYEE_POSITION_LEVELS**

**SQL CODE**

```
CREATE VIEW Employee_Position_Levels AS
SELECT
   e.employee_id,
   e.sex,
   e.academical_achievement;
   CASE
      WHEN e.job_title LIKE '%Manager%' THEN 'High Position'
      WHEN e.job_title LIKE '%Supervisor%' THEN 'High Position'
      WHEN e.job_title LIKE 'Senior %' THEN 'High Position'
      WHEN e.job_title LIKE '%Analyst%' THEN 'High Position'

      WHEN e.job_title LIKE '%Support%' THEN 'Low Position'
      WHEN e.job_title LIKE '%Assistant%' THEN 'Low Position'
      WHEN e.job_title LIKE '%Janitor%' THEN 'Low Position'
      WHEN e.job_title LIKE '%Associate Staff%' THEN 'Low Position'
```

```
        ELSE 'Other/Unclassified'
    END AS position_level
FROM
    employee e;
```

The Employee_Position_Levels view was created to identify whether an employee holds a high or low position based on their job title, while also displaying their sex and academic achievement. This view categorizes employees into position levels using conditional logic, which helps summarize employee roles in a clear and structured way.

In this case, no JOIN statement was used because the data comes from a single table, which is the employee table. There is no need to compare or combine records from other tables. This view is useful because it instantly provides an organized overview of employees with high and low positions. It also minimizes the need to repeatedly rewrite queries and helps reduce errors when accessing employee position data.

**VIEW#2 COMPERHENSIVE_COMPLAINT_POSITION_ANALYSIS**

**SQL CODE**

```
CREATE VIEW Comprehensive_Complaint_Position_Analysis AS
SELECT
    wc.complaint_type,
    e.sex,
    e.academical_achievment,
    CASE
        WHEN e.job_title LIKE '%Manager%' OR e.job_title LIKE '%Supervisor%' OR
e.job_title LIKE 'Senior %' OR e.job_title LIKE '%Analyst%' THEN 'High Position'

        WHEN e.job_title LIKE '%Support%' OR e.job_title LIKE '%Assistant%' OR
e.job_title LIKE '%Janitor%' OR e.job_title LIKE '%Associate Staff%' THEN 'Low
Position'
        ELSE 'Other/Unclassified'
    END AS position_level,
    COUNT(wc.complaint_id) AS Complaint_Count
FROM
    employee e
LEFT JOIN
    workplace_complaint wc ON e.employee_id = wc.employee_id
WHERE
    wc.complaint_id IS NOT NULL
```

```
GROUP BY
    wc.complaint_type,
    e.sex,
    e.academical_achievement,
    position_level
ORDER BY
    Complaint_Count DESC,
    wc.complaint_type;
```

The Comprehensive_Complaint_Position_Analysis view was created to analyze employee complaints by complaint type, sex, academic achievement, and position level. It uses conditional logic to classify employees into high or low positions based on their job titles, then counts the total number of complaints filed under each category. This view uses a LEFT JOIN between the employee and workplace_complaint tables to associate complaints with employee information.

The data is then grouped to compute the total number of complaints per complaint type and position level in an efficient manner. Using a VIEW in this scenario improves efficiency because it automatically performs the necessary joins, filtering, and counting. Instead of repeatedly writing complex SQL queries to analyze complaint data, the VIEW encapsulates the logic and allows users to retrieve summarized results quickly and with fewer errors.

**STORED PROCEDURE**

In the Stored Procedure part, the use of this in our database system, along with the implementation of ACID transactions, ensures that the data in the employee table remains reliable and complete. The procedure handles the insertion or addition of employee data, while the transaction ensures that it follows the ACID properties, which include Atomicity, Consistency, Isolation, and Durability. **Atomicity** ensures that insertions are carried out correctly, and certain procedures must be strictly followed. If any data fails to comply, it will not be recorded, preventing unreliable or incomplete employee data. **Consistency** ensures that the data remains consistent; in other words, it cancels inputs that do not align to the consistency rule, and the transaction or process of recording will roll back to its valid state.

In **Isolation**, each data recording is independent, meaning there is no interference from other transactions that could alter one's data. Lastly, **Durability** guarantees that once data is committed or inserted, it is permanently stored, even in scenarios where the server crashes or the system restarts. The application of Stored Procedures guarantees that employee information is consistently and accurately recorded. In cases of invalid input, an error message will be displayed: "SQL ERROR: Data integrity violation occurred (e.g., Duplicate ID). Transaction Rolled Back."

**SQL CODE**

```
DELIMITER $$

CREATE PROCEDURE Add_New_Employee (
   IN e_employee_id INT,
   IN e_name VARCHAR(255),
   IN e_age INT,
   IN e_academical_achievement VARCHAR(255),
   IN e_sex CHAR(1),
   IN e_job_title VARCHAR(255),
   IN e_date_hired DATE,
   IN p_department_id INT,
   IN p_company_id INT
)
BEGIN
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
   BEGIN
      ROLLBACK;
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SQL ERROR: Data
integrity violation occurred (e.g., Duplicate ID). Transaction rolled back.';
   END;

   IF UPPER(e_sex) NOT IN ('M', 'F') THEN
      SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Validation Error:
Employee sex must be M or F.';
   END IF;

   START TRANSACTION;

   INSERT INTO employee
   (employee_id, name, age, academical_achievement, sex, job_title, date_hired,
department_id, company_id)
   VALUES
   (e_employee_id, e_name, e_age, e_academical_achievement, UPPER(e_sex),
e_job_title, e_date_hired, p_department_id, p_company_id);
```

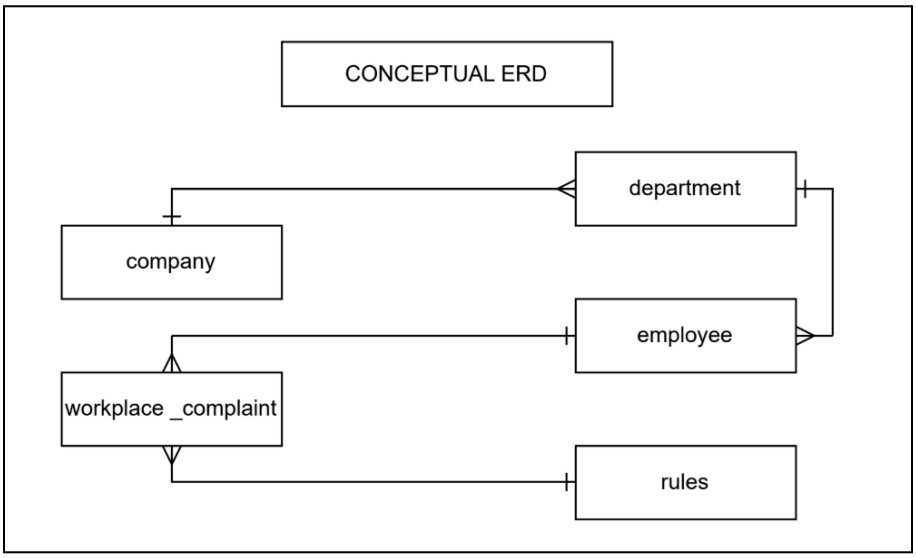
```
    COMMIT;
END $$

DELIMITER ;
```

### 3.2. ER Diagram:

This conceptual ERD that we made presents the main components of our workplace discrimination monitoring system for SDG 5.1 Gender Equality and how they are connected or related to each other.

- **CONCEPTUAL ERD**




*Clear Version of Conceptual ERD Link: CONCEPTUAL ERD.pdf*

**Company, Department, and Employee Tables**

- The Company Table represents the companies that employ workers who can violate the gender-equality practices.
- The Department Table has the group of employees in a company that reflects functional units such as Human Resources, Engineering, Production, and Sales where discrimination may happen.
- The Employee Table represents every individual worker in a department. It is characterized by their employee ID, name, age, academic achievement, sex, job title, date

hired, and their department ID in the lower-level models; at the conceptual level, it simply tells who can experience workplace discrimination.

The crow's foot arrows indicate that employees are associated with a department under a company, meaning that each complaint can be traced or tracked back to both the department and company where the issue occurred.

**Workplace_Complaint Table as the Central SDG 5.1 Gender Equality Entity**

- workplace_complaint table is the central entity that models reported cases of discrimination at work, such as harassment, pay equality, promotion bias, parental rights, and workplace respect, that is directly related to SDG 5.1 Gender Equality goal of ending the discrimination against women and girls in employment.
- The connection from employee to workplace_complaint shows that each complaint is filed by an employee, while an employee may file multiple complaints over time, capturing repeated or multiple incidents.

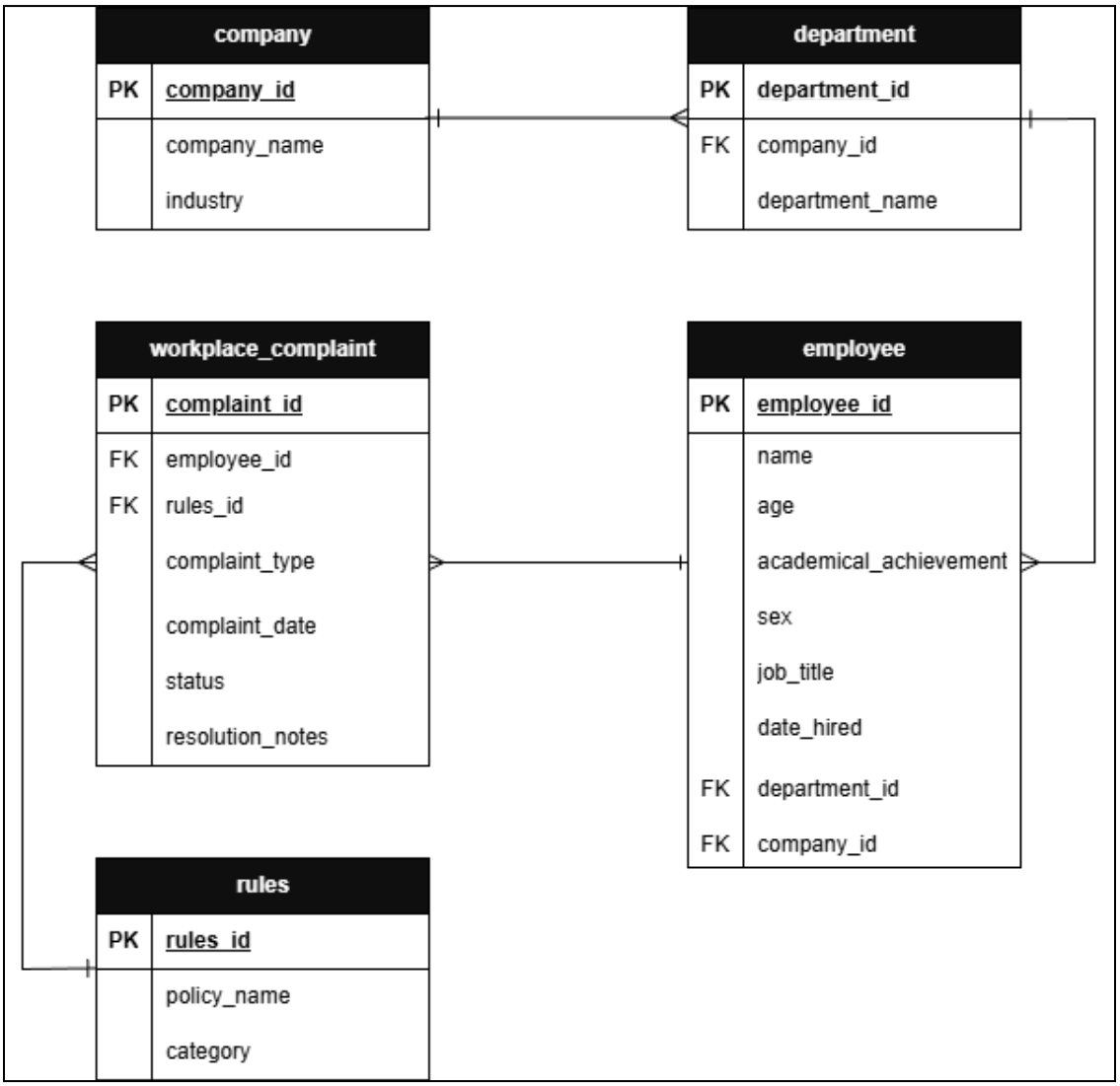
This structure allows the system to record who is affected, where they work, and what type of discriminatory act occurred, providing data for later statistical analysis in the logical and physical models.

**Rules as the Policy and Protection Layer**

- Rules table represent workplace policies and guidelines such as anti-sexual harassment, equal pay for equal work , non-discrimination in promotion, parental leave and care, anti-bullying and respect policies that companies are expectted to implement.
- The connection from workplace_complaint to rules table conceptually shows that each complaint can be evaluated against one or more policies, indicating which protection was allegedly violated.

By including Rules Table in the conceptual model, the system can later support analysis of how well companies comply with gender-equality policies and how often specific rules are violated, which aligns with SDG 5.1 Gender Equality's focus on legal and policy frameworks to eliminate discrimination.

- **LOGICAL ERD**




*Clear Version of Logical ERD Link: LOGICAL ERD.pdf*

Our logical ERD defines the detailed structure of the workplace gender-equality database for SDG 5.1, showing entities, primary keys, foreign keys, and data types that can be directly translated into relational tables.

**Company and Department**

- The company table stores organizations through company_id, company_name, and industry.
- The department table refines this by adding department_id, company_id (FK), and department_name, which creates a one-to-many relationship where each company can have many departments.

**Employee**

- The employee table represents individual workers with employee_id (PK) and attributes such as name, age, academical_achievement, sex, job_title, and date_hired.
- The department_id foreign key links each employee to one department, while a department can contain many employees, allowing analysis of staff composition and discrimination patterns by department and company.

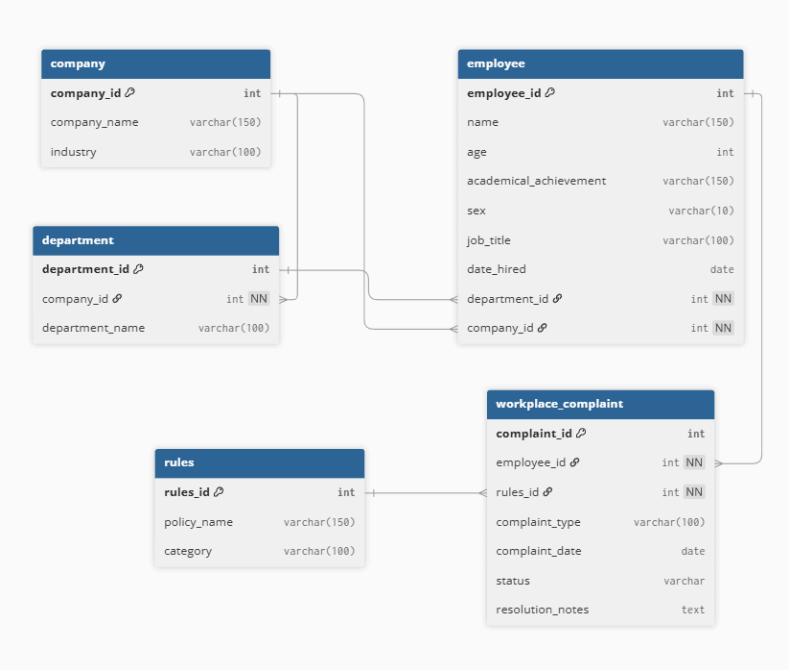
**Rules**

- The rules table stores workplace policies using rules_id (PK), policy_name, and category.
- These records represent anti-discrimination, anti-harassment, and related policies that should protect employees, supporting SDG 5.1's emphasis on legal and policy frameworks against discrimination.

**Workplace_Complaint**

- The workplace_complaint table models reported discrimination cases, with complaint_id (PK), employee_id (FK), rules_id (FK), complaint_type, complaint_date, status, and resolution_notes.
- The foreign key to employee means each complaint is filed by exactly one employee, while the foreign key to rules links the complaint to the policy that may have been violated; together, these links allow the system to track who experienced discrimination, under which company and department they work, and which rule each case involves.

- **PHYSICAL ERD**

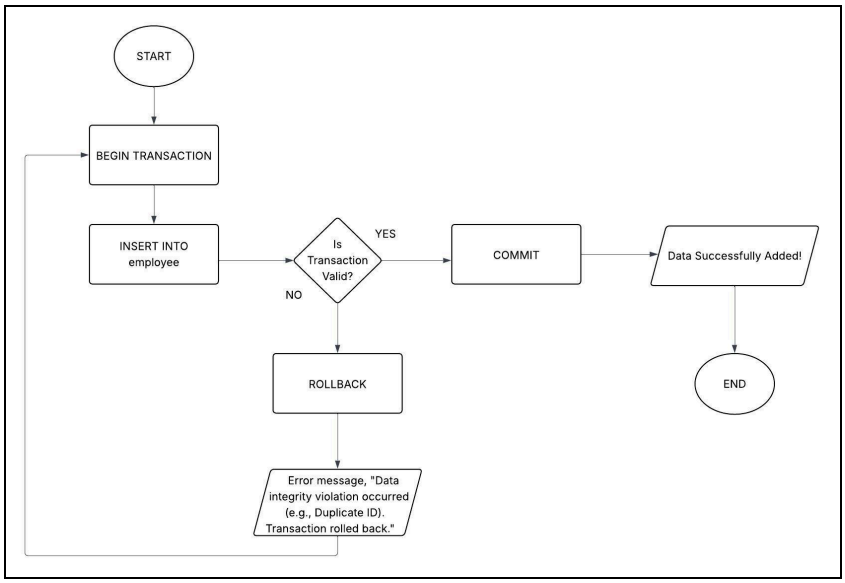



*Clear Version of Physical ERD Link:* *PHYSICAL ERD & CODE.pdf*

The database schema is defined using the domain-specific language (DSL) provided by dbdiagram.io, a modeling syntax designed to represent relational structures in a simplified and readable format. This DSL uses the keyword Table to declare entities, followed by attribute definitions written in the format field_name data_type [constraints]. The physical ERD enforces relational integrity, ensures precise data storage, and provides the foundation for implementing the data views and complaint system.

### 3.3. Transaction Flowchart:

The flowchart below represents the transactional stored procedure for employee data. It demonstrates how the system ensures data integrity by either successfully committing the data when it follows the proper transaction flow or rolling back the transaction if errors occur, such as invalid or duplicate data inputs.




*Clear version of Transaction Flowchart:* *TRANSACTION FLOWCHART*

# IV. Testing and Results

### 4.1. Test Cases:

In this part is where the validation of SDG 5.1 Gender Equality DB trigger function or mechanism, In testing its validation three scenarios was carried out. In the first scenario is where it test the trigger mechanism in enforcing its rule in the database which is designed to test the validity of the data inputted, if it follows the rules design in the trigger and in the first scenario it violates the rule implemented in the sex column that it only takes Male and Female identity and the trigger takes action which error occur and the transaction roll back and no new row was added in employee table. The second scenario demonstrates  a successful insertion where the data inputted is valid sex identity data which results in a new set of employee identity in the employee table. And lastly, is also a confirmation of how trigger ability in maintaining reliability in preventing update in the status of work complaint if it violates the rule design in the trigger where it only accepts status data such as filed, investigating, pending, and resolved. Other than those errors will occur and transactions will roll back and no update will happen.

| | |
|---|---|
| **Project Name** | SDG 5 GENDER EQUALITY DB |
| **Module Name** | Validation of TRIGGER |
| **Created By** | Andrada, Jack Zyruz<br>Condino, Zyann Kyle<br>Dela Cruz, Harvey<br>Macabangon, John Carlo<br>Monte, Manuel |
| **USN** | |
| **Creation Date** | 12-15-2025 |
| **Reviewed By** | |
| **Reviewed Date** | |

| Test Case ID | Test Scenario ID | Test Scenario Description | Test Case | Test Case Description | Preconditions | Post Conditions | Expected Results | Actual Results | Status |
|---|---|---|---|---|---|---|---|---|---|
| TC_DC_001 | TS_DB_TriggerFailure | An attempt in inserting data that violates Trigger Function. | Invalid sex data | 1. Locate and click the Database in phpMyAdmin<br>2. Navigate to SQL Tab<br>3.Proceed in executing the Stored Procedure Code<br>**INVALID SQL CODE:**<br><br>CALL Add_New_Employee(100, 'Juan Dela Cruz', 30, 'Bachelor Degree', 'Z', 'Senior Analyst', '2023-01-15, 3, 1); | User must put a valid sex identification (**'M'** or **'F'**) | 1. After the user successfully in executing the Stored Procedure Code, no new row were inserted.<br>2. The Transaction rolled back. | After the user executed the SQL Procedure, an error message will come front that indicates the failure of adding the invalid data. | Error returns an invalid message, that indicates the transaction rolled back. | PASSED |
| TC_DC_002 | TS_DB_TriggerValid | Succeeding in inserting data with the validation of Trigger in checking data values. | Valid sex data | 1. Locate and click the Database in phpMyAdmin<br>2. Navigate to SQL Tab<br>3.Proceed in executing the Stored Procedure Code<br>**VALID SQL CODE:**<br><br>CALL Add_New_Employee(101, 'Trigger Monte', 24, 'College Degree', 'M', 'Senior Analyst', '2024-04-27, 4, 1'); | User must put a valid sex identification (**'M'** or **'F'**) | 1. A new employee identification will be inserted in employee table.<br>2.Transaction successfully committed. | 1. The Stored Procedure function executes successfully without errors occur.<br>2. With the Database constraint, the CHECK constraint accepts the valid value of sex identification (sex = 'M')<br>3. Transaction successfully commit and a new identification appear in the employee table. | Successfully executed the procedure, and new records were added to the employee table. | PASSED |
| TC_DC_003 | TS_DB_Trigger_Effectiveness | An attempt of a invalid update which test the effectiveness of the trigger function in updating a complaint status | Valid status update | 1. Locate and click the Database in phpMyAdmin<br>2. Navigate to SQL Tab<br>3.Proceed to workplace_complaint table and update the status to "pending," "investigating," "resolved," or "filed."<br>**INVALID SQL CODE:**<br><br>UPDATE workplace_complaint SET status = 'done' WHERE employee_id = 1; | The user must input one of the four available statuses: **'pending'**, **'filed'**, **'investigating'**, or **'resolved'**. | 1. After the user successfully executed the update SQL Code, Trigger will take place.<br>2.Trigger is enable and validates the update status if its valid.<br>3. Trigger validates it as valid and will be commited, a new status of the specific complaint had take place. | Trigger will function, and the expected error message will be displayed: "ERROR: Status must be one of 'pending', 'filed', 'investigating', or 'resolved' only." | The trigger successfully detected that the status was invalid, and the transaction was rolled back. | PASSED |

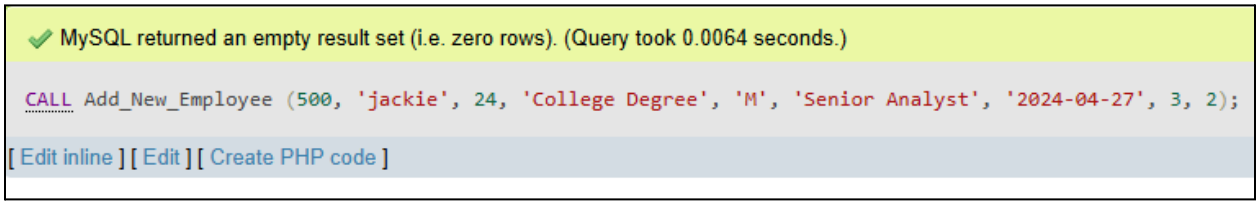
*Clear version of Test Cases Link:* *Test Case for SDG5.1 (TRG DOYS)*

**4.2. ACID Compliance Test:** Demonstrate (via screenshots or CLI output) that the FR3 transactional procedure meets ACID properties (e.g., using a ROLLBACK to prove atomicity).

The ACID Compliance Test uses screenshots output to show that the FR3 transaction satisfies Atomicity, Consistency, Isolation, and Durability. It proves that FR3 is all-or-nothing using ROLLBACK, maintains a valid database state, prevents concurrent conflicts, and makes committed changes permanent, confirming system reliability.

**A. ATOMICITY**

- **COMMIT**



- **ROLLBACK**

**Error**

**SQL query:** Copy

```
CALL Add_New_Employee (500, 'jace', 24, 'College Degree', 'Z', 'Senior Analyst', '2024-04-27', 3, 2);
```

**MySQL said:**

```
#1644 - SQL ERROR: Data integrity violation occurred (e.g., Duplicate ID). Transaction rolled back.
```

## B. CONSISTENCY

### PROCEDURE CONDITION

- **VALID INPUT**

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0064 seconds.)

```
CALL Add_New_Employee (500, 'jackie', 24, 'College Degree', 'M', 'Senior Analyst', '2024-04-27', 3, 2);
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

- **INVALID INPUT**

**Error**

**SQL query:** Copy

```
CALL Add_New_Employee (500, 'jace', 24, 'College Degree', 'Z', 'Senior Analyst', '2024-04-27', 3, 2);
```

**MySQL said:**

```
#1644 - SQL ERROR: Data integrity violation occurred (e.g., Duplicate ID). Transaction rolled back.
```

**SQL Code applying PROCEDURE in employee table**

```
DELIMITER $$

CREATE PROCEDURE Add_New_Employee (
    IN e_employee_id INT,
    IN e_name VARCHAR(255),
    IN e_age INT,
    IN e_academical_achievement VARCHAR(255),
    IN e_sex CHAR(1),
    IN e_job_title VARCHAR(255),
    IN e_date_hired DATE,
    IN p_department_id INT,
    IN p_company_id INT
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SQL ERROR: Data
integrity violation occurred (e.g., Duplicate ID). Transaction rolled back.';
    END;

    IF UPPER(e_sex) NOT IN ('M', 'F') THEN
        SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Validation Error:
Employee sex must be M or F.';
    END IF;

    START TRANSACTION;

    INSERT INTO employee
    (employee_id, name, age, academical_achievement, sex, job_title, date_hired,
department_id, company_id)
    VALUES
    (e_employee_id, e_name, e_age, e_academical_achievement, UPPER(e_sex),
e_job_title, e_date_hired, p_department_id, p_company_id);

    COMMIT;
END $$

DELIMITER ;
```

## C. ISOLATION

- The isolation feature in ACID properties ensures that regardless of other transactions going on in the background, an individual transaction behaves as if it was the only transaction going on.

- Database consistency is maintained as there are no interferences or mixing of changes. Even under conditions of multi-user.

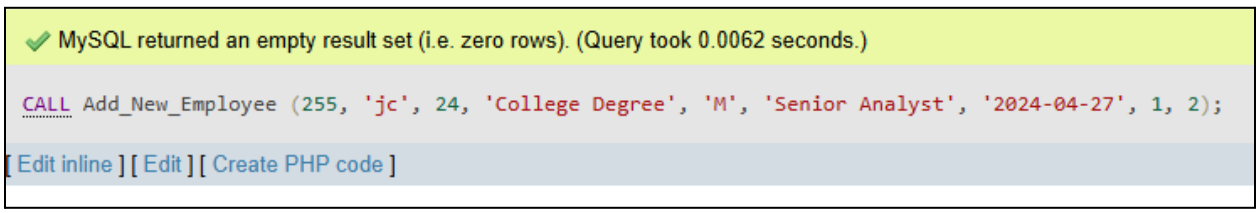
| SQL Code applying PROCEDURE in employee table |
| --- |
| CALL Add_New_Employee (304, 'Trigger Monte', 24, 'College Degree', 'M', 'Senior Analyst', '2024-04-27', 4, 2); |

- Adding an employee record is a single transaction when executing this code so this code ensures Isolation part of ACID property. One transaction runs independently from other transactions that may be inserting, updating or deleting employee information. Thus, this transaction's change does not influence the other transaction, and there is no mixing until the transaction is over.

## D. DURABILITY

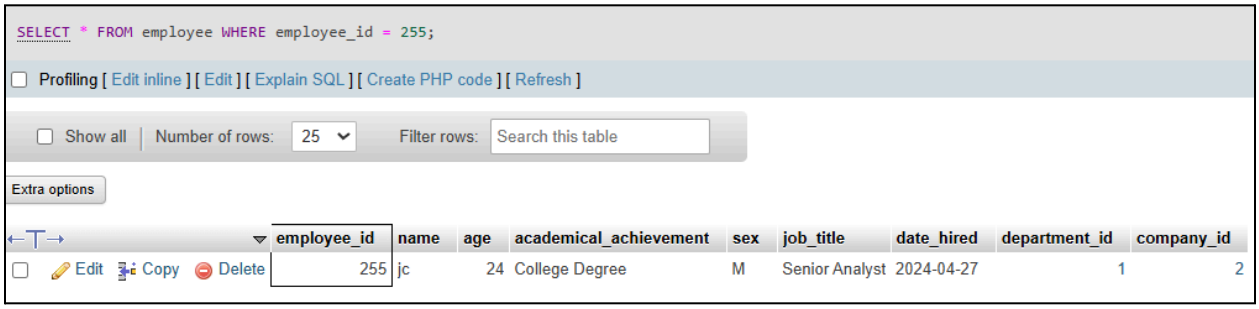
- A successful transaction in durability of ACID Properties is one that is permanently stored in the database. The committed data will not be lost if the system crashes, the database is restarted, or phpMyAdmin is closed and reopened.

## STEP 1: SUCCESSFULLY COMMITTED TRANSACTION




## STEP 2: CLOSE THE phpMyAdmin AND TRY TO OPEN IT AGAIN

**STEP 3: TRY TO FIND THE SQL CODE INSERTED EARLIER**




According to the given screenshot, after executing the CALL Add_New_Employee(...) procedure, the phpMyAdmin was closed and opened once again. When the query SELECT * FROM employee WHERE employee_id = 201; was ran, the inserted record for the employee is still present. By this time you can see that this transaction is permanently stored in the database thus it proves the durability property of acid.

# V. Conclusion and Contribution

## 5.1. Conclusion

In conclusion, This project successfully designed and executed a relational database system that supports SDG 5.1: Ending all forms of discrimination against women and girls with a specific focus on workplace inequality and complaint monitoring. Through the application of core DBMS concepts-including normalized schema design(3NF/BCNF), primary and foreign key constraints, triggers, views, stored procedures, and ACID compliant transactions, we were able to make a system that ensures data integrity, consistency, and reliability.

By centralizing complaints and linking them to specific employees, companies, and workplace rules, the system promotes accountability and prevents discriminatory practices from being overlooked or dismissed. Furthermore, the implementation of transactional control and ACID compliance testing validated the robustness of the system in handling critical operations without data corruption or loss. The analytical views generated by the system provide evidence-based insights that can guide organizations in improving promotion practices, strengthening policy enforcement, and implementing targeted gender-equality interventions.

Overall, our system does not merely record incidents of discrimination but actively supports informed decision-making and long-term monitoring. By integrating technical precision with social responsibility and woman empowerment, the DBMS not only meets academic requirements but also serves as a meaningful solution that can be a practical and sustainable technological tool for advancing gender equality in alignment with SDG 5.1.

### 5.2. Individual Contributions

In this project, each member of our group actively participated in the project, ensuring that all sections reflect both individual contributions and collaborative effort. The specific roles, responsibilities, and tasks completed by each member are systematically documented and presented in the table below.

| Name | Contributions |
|---|---|
| Andrada, Jack Zyruz | ● 2.1. Functional Requirements and Non-Functional Requirements<br>● 2.3. Schema Normalization Analysis<br>● 3.1. Core DBMS Concepts Used<br>● 3.2. ER Diagram<br>● 4.1. Test Cases<br>● 4.2. ACID Compliance Test |
| Condino, Zyann Kyle | ● 3.1. Core DBMS Concepts Used<br>● 3.2. ER Diagram<br>● 3.3. Transaction Flowchart<br>● 4.1. Test Cases<br>● 4.2. ACID Compliance Test<br>● Github |
| Dela Cruz, Harvey | ● 1.1 Project Overview & UN SDG Target<br>● 1.2. Problem Statement<br>● 3.1. Core DBMS Concepts Used<br>● 3.2. ER Diagram<br>● 5.1. Conclusion<br>● PPT |
| Macabangon, John Carlo | ● 3.2. ER Diagram<br>● 3.3. Transaction Flowchart<br>● 4.2. ACID Compliance Test<br>● PPT |
| Monte, Manuel | ● 2.1. Functional Requirements and Non-Functional Requirements<br>● 2.2. Data Requirements<br>● 4.2. ACID Compliance Test |