

NSHURO ARNAUD NELLIGAN

27960

PHASE V

PHASE V — Physical Database Implementation (DDL Scripts)

Objective:

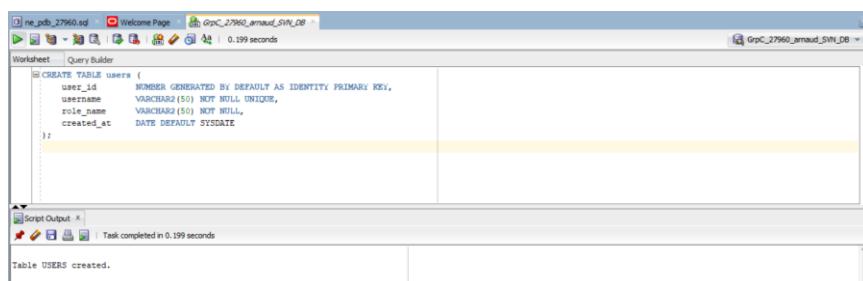
Convert the Phase 3 logical model into a physical Oracle database design using DDL scripts. Implement all entities, constraints, indexes, sequences, triggers (empty stubs), and auditing structures.

1. Physical Tables (DDL)

Below are the **full table creation scripts** based on your Security Violation Notifier project.

❖ USERS Table

Stores registered database users whose actions are being monitored.



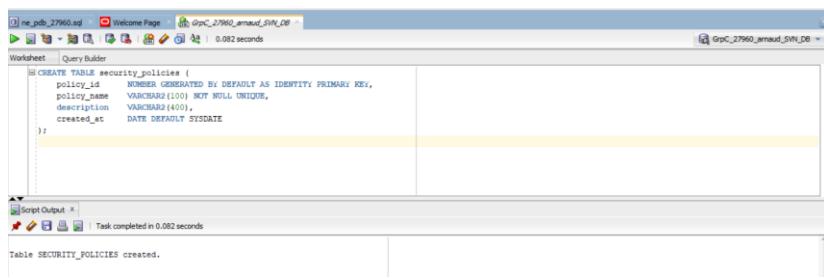
The screenshot shows the Oracle SQL Developer interface with a worksheet titled "Query Builder". The code in the worksheet is:

```
CREATE TABLE users (
    user_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    username VARCHAR2(60) NOT NULL UNIQUE,
    role_name VARCHAR2(60) NOT NULL,
    created_at DATE DEFAULT SYSDATE
);
```

The "Script Output" tab at the bottom shows the message "Table USERS created." and "Task completed in 0.199 seconds".

❖ SECURITY_POLICIES Table

Defines rules describing what is allowed and not allowed.



The screenshot shows the Oracle SQL Developer interface with a worksheet titled "Query Builder". The code in the worksheet is:

```
CREATE TABLE security_policies (
    policy_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    policy_name VARCHAR2(100) NOT NULL UNIQUE,
    description VARCHAR2(400),
    created_at DATE DEFAULT SYSDATE
);
```

The "Script Output" tab at the bottom shows the message "Table SECURITY_POLICIES created." and "Task completed in 0.082 seconds".

❖ OPERATIONS_LOG Table

Stores every monitored action (for auditing and BI).

```

-- Safely drop table if it exists
DROP TABLE operations_log
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE := -942 THEN -- ORA-00942: table or view does not exist
      RAISE;
    END IF;
  /
-- Create the operations_log table
CREATE TABLE operations_log (
  log_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  operation VARCHAR2(50) NOT NULL,
  target_table VARCHAR2(100),
  timestamp DATE DEFAULT SYSDATE,
  details VARCHAR2(400),
  CONSTRAINT fk_log_user FOREIGN KEY (user_id)
    REFERENCES users(user_id)
);

```

Script Output: PL/SQL procedure successfully completed.
Table OPERATIONS_LOG created.

❖ VIOLATIONS Table

Stores only *actual violations* detected by triggers.

```

CREATE TABLE violations (
  violation_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  log_id NUMBER NOT NULL,
  policy_id NUMBER NOT NULL,
  detected_at DATE DEFAULT SYSDATE,
  severity VARCHAR2(20),
  status VARCHAR2(20) DEFAULT 'UNRESOLVED',
  CONSTRAINT fkViolationLog FOREIGN KEY (log_id)
    REFERENCES operations_log(log_id),
  CONSTRAINT fkViolationPolicy FOREIGN KEY (policy_id)
    REFERENCES security_policies(policy_id)
);

```

Script Output: Table VIOLATIONS created.

2. Indexes

```

CREATE INDEX idx_log_user ON operations_log(user_id);
CREATE INDEX idxViolation_policy ON violations(policy_id);
CREATE INDEX idxViolation_log ON violations(log_id);

```

Script Output: Index IDX_LOG_USER created.
Index IDX_VIOLATION_POLICY created.
Index IDX_VIOLATION_LOG created.

3. BI Considerations Integration

- Operations_Log → fact table (high volume)
- Users & Policies → dimension tables

- Violations → fact table with severity metrics
- Timestamps → support time-based analytics (daily, hourly)
- Possible SCD Type-2 on USERS (if roles change)

4. Constraints Summary

- ❖ PKs: All tables have a primary key.-
- ❖ FKs: Strong referential integrity between logs, users, and policies.
- ❖ UNIQUE: usernames, policy names.
- ❖ CHECK: status IN ('UNRESOLVED', 'RESOLVED').

5. Normalization

- ❖ 1NF: No repeating groups or multi-valued attributes.
- ❖ 2NF: All non-key attributes depend on full PK.
- ❖ 3NF: No transitive dependencies