DBMS

Objective

To design and implement a real-world school administration database system, applying ER modeling, normalization, SQL queries, indexing, transactions, recovery, and database security. Students must also prepare a **Low-Level Design (LLD)** section, which explains how the ER diagram translates into SQL tables .

References

LLD & ER→ Relational Mapping

• Mapping from ER Model to Relational Model

https://www.geeksforgeeks.org/dbms/mapping-from-er-model-to-relational-model/

• Convert ER Model to Relational Model

https://www.tutorialspoint.com/dbms/er_model_to_relational_model.htm

• Univ. of Waterloo (slides, PDF) — ER to Relational

https://cs.uwaterloo.ca/~gweddell/cs348/errelational-present.pdf

Normalization & Constraints

• Normal Forms in DBMS

https://www.geeksforgeeks.org/dbms/normal-forms-in-dbms/

• Database Normalization

https://www.tutorialspoint.com/dbms/database_normalization.htm

SQL (DDL, Joins, Subqueries, Views)

• SQL Tutorial

https://www.w3schools.com/sql/

• SQL Tutorial for Data Analysis

https://mode.com/sql-tutorial/

Indexing & Query Optimization

• Indexing in Databases

https://www.geeksforgeeks.org/dbms/indexing-in-databases-set-1/

• PostgreSQL Docs

https://www.postgresql.org/docs/current/sql-explain.html

• MySQL Docs

https://dev.mysql.com/doc/en/explain.html

• Oracle Docs

https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/EXPLAIN

PLAN.html

Transactions, ACID & Recovery

• Transaction Management

https://www.geeksforgeeks.org/dbms/transaction-management/

• DBMS Data Recovery

https://www.tutorialspoint.com/dbms/dbms_data_recovery.htm

Security, & SQL Injection

• Database Security (control methods)

https://www.geeksforgeeks.org/dbms/control-methods-of-database-security/

• OWASP — SQL Injection (overview)

https://owasp.org/www-community/attacks/SQL_Injection

• OWASP — SQL Injection Prevention Cheat Sheet

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Case Study 42: Financial Transaction Record System

Project Description: FinLog is a high-performance centralized ledger designed for regulatory compliance and fraud analysis. Unlike a standard banking app that focuses on user balances, this system acts as the **immutable audit trail** for all financial movements across an institution. It processes millions of records to detect money laundering (AML) patterns and ensures compliance with central bank regulations.

The system handles **high-velocity write operations** and supports complex querying for forensic audits. It tracks metadata such as IP addresses, merchant categories, and device IDs for every transaction. The core challenge is ensuring data immutability—once a transaction is logged, it cannot be altered, only reversed via a "contra-entry."

The system manages key operational entities such as **transactions, accounts, merchants, devices, risk flags, and audit logs**.

- **Transactions** have unique references, timestamps, and types (Credit/Debit).
- **Accounts** are linked to entities (Individuals, Corporations).
- **Merchants** are the receivers of funds with Category Codes (MCC).
- **Devices** record the fingerprint (IP/MAC) used for the transaction.
- **Risk Flags** mark suspicious activities (e.g., High Velocity, Cross-Border).
- **Audit Logs** track who viewed or queried the sensitive financial data.

The system must generate critical insights for the Chief Risk Officer, including:

1. Tracking **suspicious transaction spikes**.
2. Monitoring **merchant chargeback rates**.

3. Identifying **structuring/smurfing** (breaking large cash deposits into small ones).
4. Generating **regulatory compliance reports** (STR/CTR).

FinLog also requires strict data security:

- **Auditor** has read-only access to historical logs.
- **Compliance Officer** manages risk rules and flags.
- **System (API)** performs high-speed inserts.

Design and implement a robust database solution that ensures:

- **Data Integrity** → via normalization and relational constraints.
- **Efficient Query Performance** → using indexing and query optimization.
- **Transaction Safety** → ensuring multi-step operations follow ACID properties.
- **Database Security** → applying role-based access control (RBAC).

Tasks (100 Marks)

1. Low-Level Design (LLD) — 10 Marks

- List all tables with attributes and data types.
- Identify **Primary Keys, Foreign Keys, Unique, and Check constraints**.
- Show relationship mapping (e.g., Transaction → Merchant via merchant_id).
- Write a short note on normalization up to **3NF/BCNF**.

2. SQL Schema & Data Population — 30 Marks

- Implement schema using **DDL**.
- Insert at least **250 realistic records**.

Queries to Implement:

Basic Queries

1. List transactions with value greater than $10,000 (Reporting Threshold).
2. Find transactions initiated from "IP Address 192.168.X.X".
3. Retrieve distinct merchants located in "Cayman Islands".
4. List accounts flagged as "High Risk".
5. Find transactions processed on "Christmas Day".

Joins & Subqueries

6. Display transaction details along with Account Holder and Merchant Name.
7. Find accounts that have transacted with more than 50 distinct merchants.
8. List transactions flagged for "Potential Money Laundering".
9. Identify devices used to access multiple different accounts (Account Takeover risk).
10. List compliance officers who reviewed "Flagged" transactions today.

Aggregation & Reports

11. Top 5 merchants by total transaction volume.
12. Total value of cross-border (International) transactions.
13. Identify the hour of the day with the highest fraud attempts.
14. Average transaction value per Merchant Category Code (MCC).
15. Count of transactions per currency type.

Advanced Queries

16. Generate a structuring report (Accounts with >5 cash deposits just under $10k).
17. List accounts where the total outflow exceeds total inflow by 500% (Bust-out fraud).

18. Identify consecutive transactions made within 1 minute from different countries.
19. Find merchants with a chargeback rate > 1% of total sales.
20. Create a view showing: **Txn ID, Timestamp, Amount, Risk Score, and Status.**

> Requirement: At least 3 queries must be written in two different ways (e.g., subquery vs join).

3. Indexing & Query Optimization — 15 Marks

- Create indexes on transaction_date, account_id, and merchant_id.
- Compare **execution plans** with vs without indexes.
- Rewrite **2 queries** (from the list above) into optimized versions.

4. Transactions & Recovery — 20 Marks

Implement a "Transaction Logging" Transaction with:

1. Validate account status (Rollback if Frozen).
2. Insert transaction record.
3. Update account balance.
4. Run real-time fraud check (Update Risk Score).

Demonstrate:

- **COMMIT, ROLLBACK, SAVEPOINT.**
- **Concurrency conflict:** Two parallel transactions trying to update the same account balance.
- **Deadlock simulation** and resolution.
- Failure & recovery using rollback logs.

5. Database Security — 25 Marks

Define roles:

- **Admin** → Full privileges.
- **Compliance Officer** → UPDATE on Risk Flags.
- **Auditor** → SELECT on Logs.

Tasks:

- Apply **RBAC** with GRANT / REVOKE.
- Demonstrate a **vulnerable SQL Injection** and a **secure fix**.

Deliverables (Documents to be Submitted)

1. LLD Document (3-5 pages):

- Tables with attributes & keys
- Relationship mappings
- Normalization explanation
- Indexing strategy notes
- Transaction pseudo-code
- Security role design

2. Main Report (10-12 pages):

- ER diagram
- SQL code with screenshots
- Query results with explanations
- Execution plans (index vs no index)
- Transaction & recovery demonstrations
- Security role enforcement & SQL injection demo
