Exercise Sheet: Data Normalization and Denormalization

Data, Data Storage, Data Collection

Exercise 1 - Pizzerias

Consider the following schema, where each row represents one product sold by a pizzeria:

Pizzeria(**Name**, **Address**, **ProductSold**, **Price**).

(**a**) Using examples, identify the anomalies and redundancies caused by this schema.

(**a**) **Redundancy:** The address of each pizzeria is repeated for every product it sells. For example, if `Fratelli` sells 12 pizzas, its address appears 12 times.

Update anomaly: If the pizzeria `Fratelli` changes address, the update must be performed on all rows. Missing one row leads to inconsistencies.

Insertion anomaly: It is impossible to insert a new pizzeria that has no products yet: the schema requires a product.

Deletion anomaly: If the last product of a pizzeria is removed, all information about that pizzeria disappears.

Consistency anomaly: When inserting a new product for an existing pizzeria, the address must be typed manually and may differ from the existing ones.

Exercise 2 - Functional Dependencies in a Sample Relation

Consider the relation r over the schema $R(A, B, C, D, E)$ with the following tuples:

A	**B**	**C**	**D**	**E**
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c5	d1	e1

(**a**) Which attributes take only a single value in the dataset? What functional dependencies follow immediately?

(**b**) For each attribute $X \in \{A, B, C, D\}$, examine whether $X \to Y$ holds for any other attribute Y.

(**c**) Check whether any *pairs* of attributes (e.g. AB, AC, AD) determine another attribute. List all FDs of the form $XY \to Z$.

(**d**) From all the above, keep only the *minimal* functional dependencies.

Note that these dependencies hold for this dataset because equal values of the left-hand side always correspond to equal values of the right-hand side. They are not guaranteed to hold in a larger or real-world dataset.

(**a**) Attribute E takes only the value $e1$, so $A \to E$, $B \to E$, and $D \to E$ hold trivially.
(**b**) Since each value of C appears only once, C determines the entire row:

$$C \to A,\ C \to B,\ C \to D,\ C \to E.$$

Attributes A, B, and D do not determine any other attribute except E (for every possibility, there is at least one value in relationship with two values)
(**c**) for $AB \to D$, we look at the pairs (ai, bj) and the associated value of D :

- $(a1, b1) \to d1$
- $(a1, b2) \to d2$
- $(a2, b1) \to d3$
- $(a3, b2) \to d1$

Since every pair of AB corresponds to a unique value of D, the functional dependency $AB \to D$ holds. Similarly, we find the following dependencies the dataset:

$$AD \to B,\ BD \to A.$$

(**d**) Combining all results, the minimal functional dependencies satisfied by this instance are:

- $A \to E$, $B \to E$, $D \to E$ and the dependencies on C grouped as $C \to ABDE$: these are single attribute dependency so minimality is automatic.
- $AB \to D$, $AD \to B$, $BD \to A$: neither attribute alone determines the right-hand side, so the pairs are minimal, e.g; $A \to D$ and $B \to D$ do not hold.

We also do not list FDs with three or more attributes on the left, because none of them can be minimal: every triple contains a pair that we already checked, and if that pair does not determine the right-hand side, adding a third attribute cannot make the dependency minimal.

Exercise 3 - University Course Registration

A university maintains a database of course registrations. The current table is as follows:

StudentID	**StudentName**	**Courses**	**Instructor**
101	Alice	Math, Physics	Dr. Smith, Dr. Johnson
102	Bob	Chemistry, Math	Dr. Brown, Dr. Smith
103	Carol	Biology, Chemistry	Dr. Green, Dr. Brown

(1) First Normal Form (1NF)

(**a**) Identify the issues with the current table. Why is it not in 1NF?
(**b**) Transform the table into 1NF, explain your steps.

(**a**) The table is not in 1NF because the **Courses** and **Instructor** columns contain multiple values (repeating groups).
(**b**) 1NF requires atomic values. The normalized table should look like this:

StudentID	StudentName	Course	Instructor
101	Alice	Math	Dr. Smith
101	Alice	Physics	Dr. Johnson
102	Bob	Chemistry	Dr. Brown
102	Bob	Math	Dr. Smith
103	Carol	Biology	Dr. Green
103	Carol	Chemistry	Dr. Brown

(2) Second Normal Form (2NF)

(**a**) List all superkeys of the 1NF table.
(**b**) Identify the candidate key(s).
(**c**) List the minimal functional dependencies.
(**d**) Is the table in 2NF? If not, explain why and transform it into 2NF.

At the point, we can choose either to either reason based on the values in the table (similar to what we did in the previous exercise), or use our knowledge of the domain to reason based on the name of the attributes. We will here do the later and assume that there might be several students with the same name and that each **Instructor** only teaches one **Course**.

(**a**) Superkeys are sets of attributes that uniquely identifies each row:

- {**StudentID**, **Course**}
- {**StudentID**, **Instructor**}
- {**StudentID**, **StudentName**, **Course**}
- {**StudentID**, **StudentName**, **Instructor**}
- {**StudentID**, **Course**, **Instructor**}
- {**StudentID**, **Course**, **Instructor**}
- {**StudentID**, **Course**, **StudentName**, **Instructor**}

Note that if we had considered that an **Instructor** might teach several **Course**s, then {**StudentID**, **Instructor**} and {**StudentID**, **StudentName**, **Instructor**} are no longer superkeys. Similarly is several **Instructor**s teach in the same **Course**, then {**StudentID**, **Course**} and {**StudentID**, **StudentName**, **Course**} are no longer superkeys.

(**b**) Candidate keys are minimal superkeys:

- {**StudentID**, **Course**}
- {**StudentID**, **Instructor**} (assuming each **Instructor** only teaches one topic.)

(**c**) Minimal functional dependencies:

- StudentID → StudentName
- Course → Instructor
- Instructor → Course

(**d**) The table is **not** in 2NF because there are partial dependencies:

- With respect to the key {**StudentID**, **Course**}: **StudentName** depends only on **StudentID**, and **Instructor** depends only on **Course**.
- With respect to the key {**StudentID**, **Instructor**}: **StudentName** depends only on **StudentID**, and **Course** depends only on **Instructor**.

To achieve 2NF we decompose the table as follows:

StudentID	StudentName
101	Alice
102	Bob
103	Carol

Course	Instructor
Math	Dr. Smith
Physics	Dr. Johnson
Chemistry	Dr. Brown
Biology	Dr. Green

StudentID	Course
101	Math
101	Physics
102	Chemistry
102	Math
103	Biology
103	Chemistry

(3) Third Normal Form (3NF)

Suppose you add a **Department** column to the table. Each **Instructor** belong to one **Department**.

Course	Instructor	Department
Math	Dr. Smith	Mathematics
Physics	Dr. Johnson	Physics
Chemistry	Dr. Brown	Chemistry
Biology	Dr. Green	Biology

(**a**) Is this table in 3NF? Explain your answer. If not, normalize it to 3NF.

(**a**) The table is not in 3NF because **Department** depends on **Instructor**, not directly on the key {**Course**}. To achieve 3NF, we decompose the table so that each non-key attribute depends only on a candidate key:

Course	Instructor
Math	Dr. Smith
Physics	Dr. Johnson
Chemistry	Dr. Brown
Biology	Dr. Green

Instructor	Department
Dr. Smith	Mathematics
Dr. Johnson	Physics
Dr. Brown	Chemistry
Dr. Green	Biology

(4) Fourth Normal Form (4NF)

Now, suppose you add a table for student activities, where each student can join multiple clubs and participate in multiple sports:

StudentID	Club	Sport
101	Chess, Debate	Swimming, Tennis
102	Robotics, Chess	Tennis, Soccer
103	Debate, Music	Swimming, Soccer

(**a**) Transform this table into 3NF.
(**b**) Is the resulting table in 4NF? Explain your answer. If not, normalize it to 4NF.

(**a**) First, 3NF (and 1NF/2NF) requires atomic values. We expand the table as:

StudentID	Club	Sport
101	Chess	Swimming
101	Chess	Tennis
101	Debate	Swimming
101	Debate	Tennis
102	Robotics	Tennis
102	Robotics	Soccer
102	Chess	Tennis
102	Chess	Soccer
103	Debate	Swimming
103	Debate	Soccer
103	Music	Swimming
103	Music	Soccer

At this point, the table satisfies 3NF because all non-key attributes depend on the whole candidate key {**StudentID**, **Club**, **Sport**}.

(**b**) The table is not in 4NF because there are multivalued dependencies: **StudentID** $\twoheadrightarrow$ **Club** and **StudentID** $\twoheadrightarrow$ **Sport**. These are independent of each other. To achieve 4NF, we split the table into two separate tables:

StudentID	Club
101	Chess
101	Debate
102	Robotics
102	Chess
103	Debate
103	Music

StudentID	Sport
101	Swimming
101	Tennis
102	Tennis
102	Soccer
103	Swimming
103	Soccer

(5) Fifth Normal Form (5NF)

Consider the following table representing student projects, where each project involves a team of students, a supervisor, and a topic:

Student	Supervisor	Topic
Alice	Dr. Smith	AI
Alice	Dr. Smith	Data Science
Alice	Dr. Brown	AI
Bob	Dr. Brown	AI
Carol	Dr. Smith	Data Science

(**a**) Is this table in 5NF? Explain your answer. If not, decompose it into 5NF.

(**a**) The table is not in 5NF because it contains a join dependency. The table can be reconstructed from joining the following three tables:

Student	Supervisor
Alice	Dr. Smith
Alice	Dr. Brown
Bob	Dr. Brown
Carol	Dr. Smith

Supervisor	Topic
Dr. Smith	AI
Dr. Smith	Data Science
Dr. Brown	AI

Student	Topic
Alice	AI
Alice	Data Science
Bob	AI
Carol	Data Science

(**b**) To achieve 5NF, we decompose the original table into the three tables above. This eliminates the join dependency and ensures the table is in 5NF.

Exercise 4 - Hospital Patient Records

A hospital database is designed as follows:

- Patients(**PatientID**, **Name**, **Address**, **AdmissionDate**)
- Doctors(**DoctorID**, **Name**, **Specialization**)
- Visits(**VisitID**, **PatientID**, **DoctorID**, **VisitDate**, **Diagnosis**)

(1) Normalization

(**a**) Is this schema normalized? If not, identify the issues and propose a normalized schema.

(**a**) Without the explicit data, we can only deduce the using common knowledge. IN the schema presented, there are no repeating groups, or any partial, transitive, multivalued or join dependency, which indicates compliance with 5NF.

(2) Denormalization

Suppose the hospital imports daily insurance information from an external partner. The provided dataset is a flat file in the form

Insurance(**PatientName**, **Address**, **InsuranceProvider**, **InsurancePlan**, **LastUpdate**)

with no stable patient identifier.

(**a**) Describe a denormalization strategy that allows the hospital database to store the imported data efficiently without reconstructing unique identifiers for every row.

(**a**) The external file does not contain a stable identifier such as **PatientID**. A practical denormalization is to store the imported data as-is in an auxiliary table:

InsuranceRaw(**RecordID**, **PatientName**, **Address**, **InsuranceProvider**, **InsurancePlan**, **LastUpdate**)

The hospital system preserves the external structure instead of normalizing it. This avoids ambiguous matching between external patient names and the internal **Patients** table. Normalization can still happen later through a manual or probabilistic matching

process (not always reliable). Storing the original structure is therefore intentional denormalization to avoid data loss or misidentification.

Hospitals must retain the *exact* doctor specialization that was in effect at the time of each visit, even if the doctor later changes specialization.

(**b**) Show how this requirement forces a denormalization of the normalized schema. Provide a modified version of the Visits table that satisfies the rule, and explain why it is not redundant in this context.

(**b**) A doctor may change specialization, but records must reflect the specialization that applied at the time of the visit. The normalized schema stores specialization only in the Doctors table. But if **Specialization** can change, the historical specialization is not functionally determined by the current doctor data.
A denormalized schema:

Visits(**VisitID**, **PatientID**, **DoctorID**, **VisitDate**, **Diagnosis**, **DoctorSpecializationAtVisit**)

Here, copying the specialization is not redundant: It encodes historical truth that would otherwise be lost. This is a case where domain constraints justify breaking 3NF/BCNF intentionally.

The emergency unit performs thousands of queries per hour to retrieve:

PatientName, DoctorName, Specialization, Diagnosis, VisitDate

for the most recent visit of each patient. The normalized schema requires joining three tables.

(**c**) Propose a denormalized structure that speeds up this workload. Explain the potential anomalies or maintenance costs introduced.

(**c**) For frequent emergency-dashboard queries, a materialized structure can be used:

RecentVisits(**PatientID**, **PatientName**, **DoctorName**, **Specialization**, **Diagnosis**, **LastVisitDate**)

Benefits:

- The dashboard query becomes a direct lookup with no joins.
- The table can be refreshed every few minutes or triggered on insert.

Drawbacks:

- Updates must propagate manually (address updates, specialty changes, doctor name corrections).
- Risk of stale values between refreshes.
- Possible inconsistency if the table RecentVisits is not kept synchronized with the base tables.