UNIVERSITY OF CHITTAGONG

Department of Computer Science & Engineering

Program: **B.Sc.** (Engineering)
Session: 2022-2023
4th Semester

Tutorial_02

Topic: DataBase Modeling With FDs (Task - 03)

Course Title: DataBase Systems
Course Code: CSE - 413

Submitted To:
Dr. Rudra Pratap Deb Nath
Associate Professor
Department of Computer Science & Engineering
University of Chittagong

Submitted By:
Nilanjana Das Jui
ID: 23701011
Dept. of Computer Science & Engineering

Date of submission: July 06, 2025

Contents

Sub Task - 01:
Identify Functional Dependencies

Identify functional dependencies for each of the relations you created.

Answer:

Here are all the functional dependencies identified for each of the relations created from the DRUGS_FULL table. These dependencies define how attributes are related and ensure proper normalization.

- **Drug** (drug_id, drug_name, drug_category)

  FD: {drug_id → drug_name, drug_category}.

  Explanation: drug_id is the primary key, and each drug has exactly one name and one category.

- **Product** (product_id, product_name, company_name, drug_id)

  FD: {product_id → product_name, company_name, drug_id}.

  Explanation: product_id uniquely identifies the product, which is linked to a single drug and manufacturer.

- **Disease** (disease_name, disease_category)

  FD: {disease_name → disease_category}.

  Explanation: disease_name is the primary key and uniquely determines the disease category.

- **ClinicalTrial** (clinical_trial_title, start_date, completion_date, participants, status, researcher, institute_name)

  FD: {clinical_trial_title → start_date, completion_date, participants, status, researcher_name, institute_name}

  Explanation: The title uniquely identifies the trial and all its details.

- **Institution** (institute_name, address, country).

  FD: {institute_name → address, country }

  Explanation: institute_name uniquely determines its full address and location.

- **SideEffect** (side_effect_name)

  FD: {side_effect_name → side_effect_name}.

  Explanation: Each side effect name is unique; this is a reference table.

- **HasSideEffect** (drug_id, side_effect_name)

  FD: (drug_id, side_effect_name) → no non-trivial FDs.

  Explanation: Composite primary key; each row links one drug to one side effect.

- **InteractsWith** (drug_id, interacts_with_name)

  FD: (drug_id, interacts_with_name) → no non-trivial FDs.

  Explanation: Represents a drug interaction pair; no attribute is functionally dependent on part of the key.

- **Treats** (drug_id, disease_name)

  FD: (drug_id, disease_name) → no non-trivial FDs.

  Explanation: Composite key represents treatment relation; nothing depends solely on one side.

- **StudiedIn** (clinical_trial_title, condition_name)

  FD: (clinical_trial_title, condition_name) → no non-trivial FDs

  Explanation: Connects a trial with a condition; the composite key is minimal.

- **TestedIn** (drug_id, clinical_trial_title)

  FD: (drug_id, clinical_trial_title) → no non-trivial FDs.

  Explanation: Each pair represents a drug tested in a specific trial; no partial dependency.

Sub Task - 02:
Transforming Schema into 3NF

If necessary, transform your schema into 3NF and formally show that the result is actually in 3NF.

Answer:

In this section, I analyze each of the normalized relations created from the original DRUGS_FULL dataset to determine whether the schema satisfies the requirements of Third Normal Form (3NF). Where necessary, I show how 3NF was achieved through decomposition.

Definition of 3NF:

- A relational schema R is in 3NF if for every non-trivial functional dependency $\alpha \rightarrow \beta$ (with $\alpha, \beta \subseteq$ attributes of R), at least one of the following conditions holds:
  - $\beta \subseteq \alpha$ — i.e., the FD is trivial
  - α is a superkey of R
  - Every attribute $B \in \beta - \alpha$ is prime (i.e., part of some candidate key)
- Non-prime attributes must not determine other non-prime attributes.
- 3NF prevents partial and transitive dependencies, except those where the dependent attribute is prime.
- 3NF always includes 2NF.

Below is the step-by-step 3NF verification and explanation for each relation in the schema.

- **Drug** (drug_id, drug_name, drug_category)

  Functional Dependency: {drug_id $\rightarrow$ drug_name, drug_category}

  Analysis:

  drug_id is the primary key (a superkey). Both drug_name and drug_category are fully functionally dependent on drug_id. There is no transitive dependency.

  The relation is in **3NF**.

- **Product** (product_id, product_name, company_name, drug_id)

  Functional Dependency: {product_id → product_name, company_name, drug_id}

  Analysis:

  product_id is the primary key (a superkey). All other attributes are directly dependent on it.

  The relation is in **3NF**.

- **Disease** (disease_name, disease_category)

  Functional Dependency: {disease_name → disease_category}

  Analysis:

  disease_name is the primary key and determines disease_category. There are no transitive or partial dependencies.

  The relation is in **3NF**.

- **ClinicalTrial** (clinical_trial_title, start_date, completion_date, participants, status, researcher, institute_name)

  Functional Dependency: {clinical_trial_title → start_date, completion_date, participants, status, researcher_name, institute_name}

  Analysis:

  clinical_trial_title is the primary key and uniquely determines the other attributes. There are no partial or transitive dependencies.

  The relation is in **3NF**.

- **Institution** (institute_name, address, country)

  Functional Dependency: {institute_name → address, country}

  Analysis:

  institute_name is the primary key. The address and country are fully dependent on it.

  The relation is in **3NF**.

- **SideEffect** (side_effect_name)
  Functional Dependency: {side_effect_name → side_effect_name (trivial)}

  Analysis:

  This is a reference table. There are no non-trivial dependencies.

  The relation is in **3NF**.

- **HasSideEffect** (drug_id, side_effect_name)
  Primary Key: (drug_id, side_effect_name)
  Functional Dependencies: None beyond the composite key.

  Analysis:

  No attribute is transitively or partially dependent on a non-superkey.

  The relation is in **3NF**.

- **InteractsWith** (drug_id, interacts_with_name)
  Primary Key: (drug_id, interacts_with_name)
  Functional Dependencies: None beyond the composite key

  Analysis:

  This relation models a many-to-many relationship. There are no additional attributes to cause dependency violations.

  The relation is in **3NF**.

- **Treats**(drug_id, disease_name)
  Primary Key: (drug_id, disease_name)
  Functional Dependencies: None beyond the composite key.

  Analysis:

  This relation links drugs to diseases. No attribute violates 3NF conditions.

  The relation is in **3NF**.

- **StudiedIn** (clinical_trial_title, condition_name)
  Primary Key: (clinical_trial_title, condition_name)
  Functional Dependencies: None beyond the composite key.

  Analysis:

  All attributes are part of the key. There are no violations of 3NF.

  The relation is in **3NF**.

- **TestedIn** (drug_id, clinical_trial_title)
  Primary Key: (drug_id, clinical_trial_title)
  Functional Dependencies: None beyond the composite key.

  Analysis:

  This relation stores which drug was tested in which trial. It satisfies 3NF.

  The relation is in **3NF**.

Sub Task - 03:
Highest Normal Form

Determine for each obtained relational schema the highest normal form it still supports, i.e., check if your relations are also in BCNF.

Answer:

To ensure the integrity and quality of the normalized schema, each relation is now analyzed to determine whether it satisfies Boyce-Codd Normal Form (BCNF) — the strictest of the commonly applied normal forms.

Definition of **BCNF**:

- A relational schema $\mathcal{R}$ is in BCNF if for each non-trivial functional dependency $\alpha \to \beta$ with $\alpha, \beta \subseteq \mathcal{R}$, at least one of the following conditions holds:
  - $\beta \subseteq \alpha$, i.e., the FD is **trivial**
  - α is a **super key** of $\mathcal{R}$
- Difference to 3NF: *no third option*
- BCNF is a stricter form of 3NF ("includes" 3NF)
- BCNF prevents all **transitive** dependencies, even those with *prime attributes* as endpoints

Verification of Each Relation:

- **Drug** (drug_id, drug_name, drug_category)

  Functional Dependency: {drug_id → drug_name, drug_category}.

  Analysis:

  drug_id is the primary key (hence a superkey). The dependency is non-trivial, but the LHS is a superkey.

  Satisfies **BCNF**.

- **Product** (product_id, product_name, company_name, drug_id)

  Functional Dependency: {product_id → product_name, company_name, drug_id}.

  Analysis:

  product_id is the primary key and functionally determines all other attributes.

  Satisfies **BCNF**.

- **Disease** (disease_name, disease_category)

  Functional Dependency: {disease_name → disease_category}.

  Analysis:

  disease_name is the primary key, so this is a valid superkey-based dependency.

  Satisfies **BCNF**.

- **ClinicalTrial** (clinical_trial_title, start_date, completion_date, participants, status, researcher, institute_name)

  Functional Dependency: {clinical_trial_title → start_date, completion_date, participants, status, researcher, institute_name}.

  Analysis:

  clinical_trial_title is the primary key, and the dependency is non-trivial but based on a superkey.

  Satisfies **BCNF**.

- **Institution** (institute_name, address, country)

  Functional Dependency: {institute_name → address, country}.

  Analysis:

  institute_name is the primary key and determines all other attributes in this relation.

  Satisfies **BCNF**.

- **SideEffect** (side_effect_name)

  Functional Dependency: {side_effect_name → side_effect_name}.

  Analysis:

  The only FD is trivial and side_effect_name is the primary key.

  Satisfies **BCNF**.

- **HasSideEffect** (drug_id, side_effect_name)

  Composite Primary Key: (drug_id, side_effect_name)
  Functional Dependencies: None beyond the key.

  Analysis:

  There are no non-trivial FDs violating the superkey requirement; the composite key is minimal and complete.

  Satisfies **BCNF**.

- **InteractsWith** (drug_id, interacts_with_name)

  Composite Primary Key: (drug_id, interacts_with_name)
  Functional Dependencies: None beyond the composite key.

  Analysis:

  Every attribute is part of the primary key, and there are no additional FDs.

  Satisfies **BCNF**.

- **Treats** (drug_id, disease_name)

  Composite Primary Key: (drug_id, disease_name)
  Functional Dependencies: No partial or transitive dependencies exist.

  Analysis:

  Both attributes together form the key; no dependency violates BCNF.

  Satisfies **BCNF**.

- **StudiedIn** (clinical_trial_title, condition_name)

  Composite Primary Key: (clinical_trial_title, condition_name)
  Functional Dependencies: No non-key dependencies.

  Analysis:

  This is a pure many-to-many relationship with no dependency anomalies.

  Satisfies **BCNF**.

- **TestedIn** (drug_id, clinical_trial_title)

  Composite Primary Key: (drug_id, clinical_trial_title)
  Functional Dependencies: None beyond the key.

  Analysis:

  Both attributes are needed to uniquely identify a row; the relation is already in BCNF.

  Satisfies **BCNF**.

  After analyzing all relations in the schema:

  - Every functional dependency in every relation has a **superkey** on the left-hand side.
  - No relation has transitive dependencies or dependencies on non-superkeys.
  - All relations meet the strict requirements of BCNF.

  Therefore, the entire schema is in **BCNF**, which also implies it is in 3NF.

Sub Task - 04:
Description of Normalized Relations

Describe the normalized relations with

- their definitions (name, attribute names and types, keys)
- information an how and why you derived them from the original relations
- information on the highest normal form that.

Answer:

The following normalized relations were derived from the original flat table DRUGS_FULL through schema decomposition based on functional dependencies. The objective was to eliminate redundancy, ensure data consistency.

- **Drug**

  Attributes:

  - `drug_id` (NUMBER, Primary Key)
  - `drug_name` (VARCHAR2)
  - `drug_category` (VARCHAR2)

  Derivation:

  Extracted from the flat dataset. A unique `drug_id` was generated (via Python) for each distinct `drug_name`. The original data had duplicates for drugs with multiple categories, which were normalized into this single table.

  Normal Form:

  BCNF — All non-key attributes are functionally dependent on the key (`drug_id`), and no transitive dependencies exist.

- **Product**

  Attributes:

  - `product_name` (VARCHAR2, Primary Key)
  - `company_name` (VARCHAR2)
  - `drug_id` (NUMBER, Foreign Key)

Derivation:

Extracted from columns `product_name`, `company_name`, and associated `drug_id`. Each product is linked to one drug.

Normal Form:

BCNF — `product_name` → `company_name`, `drug_id`; no partial or transitive dependencies.

- **Disease**

  Attributes:

  - `disease_name` (VARCHAR2, Primary Key)
  - `disease_category` (VARCHAR2)

  Derivation:

  Taken directly from the flat file where `disease_name` → `disease_category`. Since each disease had only one category, this was a straightforward 1:1 mapping.

  Normal Form:

  BCNF — Functional dependency `disease_name` → `disease_category` satisfies BCNF since `disease_name` is the key.

- **ClinicalTrial**

  Attributes:

  - `clinical_trial_title` (VARCHAR2, Primary Key)
  - `start_date` (DATE)
  - `completion_date` (DATE)
  - `participants` (NUMBER)
  - `status` (VARCHAR2)
  - `researcher_name` (VARCHAR2, Foreign Key)
  - `institute_name` (VARCHAR2, Foreign Key)

  Derivation:

  Flattened trial data from the original table. Researcher and institution were pulled into foreign key fields for normalization.

Normal Form:

BCNF — `clinical_trial_title` is a unique identifier; all other attributes depend solely on it.

- **Institution**

  Attributes:

  - `institute_name` (VARCHAR2, Primary Key)
  - `address` (VARCHAR2)
  - `country` (VARCHAR2)

  Derivation:

  Normalized from `clinical_trial_address1/2` and institutional info. Duplicates were removed using distinct names and addresses.

  Normal Form:

  BCNF — `institute_name` $\rightarrow$ `address`, `country` is the dependency and `institute_name` is the key.

- **SideEffect**

  Attributes:

  - `side_effect_name` (VARCHAR2, Primary Key)

  Derivation:

  In the original table, side effects were stored across multiple columns (`side_effect1` to `side_effect5`). This multivalued attribute was normalized into its own entity.

  Normal Form:

  BCNF — Only one attribute which is the primary key.

- **HasSideEffect**

  Attributes:

  - `drug_id` (NUMBER, Foreign Key)
  - `side_effect_name` (VARCHAR2, Foreign Key)

Derivation:

Created to resolve the multivalued side effects for each drug from the original relation. One row per (`drug`, `side_effect`) pair.

Normal Form:

BCNF — Composite primary key with no additional attributes. Fully normalized.

- **Treats**

  Attributes:

  - `drug_id` (NUMBER, Foreign Key)
  - `disease_name` (VARCHAR2, Foreign Key)

  Derivation:

  Created to model the many-to-many relationship between drugs and diseases. Each combination reflects a treatment.

  Normal Form:

  BCNF — No transitive or partial dependencies. Composite primary key.

- **TestedIn**

  Attributes:

  - `drug_id` (NUMBER, Foreign Key)
  - `clinical_trial_title` (VARCHAR2, Foreign Key)

  Derivation:

  Represents the many-to-many link between drugs and the clinical trials they're involved in.

  Normal Form:

  BCNF — Fully normalized many-to-many relation with composite key.

- **StudiedIn**

  Attributes:

  - `clinical_trial_title` (VARCHAR2, Foreign Key)
  - `condition_name` (VARCHAR2)

  Derivation:

  Derived from original columns `clinical_trial_condition1{3`, normalized to one condition per row per trial.

  Normal Form:

  BCNF — No partial or transitive dependencies; minimal structure with a composite key.

Sub Task - 05:
SQL Statements

List the SQL statements that create all your tables properly and ensure the functional dependencies that you have identified. Hint: Consider UNIQUE NOT NULL constraints to enforce candidate key characteristics. Do not forget to define primary keys and foreign keys.

Answer:

The following SQL statements define the complete database schema for the normalized model of the drugs dataset. They are written to ensure proper enforcement of functional dependencies using PRIMARY KEY, UNIQUE, NOT NULL, and FOREIGN KEY constraints.

- **DRUGS_FULL Flat Table**

```
CREATE TABLE DRUGS_FULL (
    DRUG_ID NUMBER(6),
    DRUG_NAME VARCHAR2(100),
    SIDE_EFFECT1 VARCHAR2(100),
    SIDE_EFFECT2 VARCHAR2(100),
    SIDE_EFFECT3 VARCHAR2(100),
    SIDE_EFFECT4 VARCHAR2(100),
    SIDE_EFFECT5 VARCHAR2(100),
    INTERACTS_WITH1 VARCHAR2(100),
    INTERACTS_WITH2 VARCHAR2(100),
    INTERACTS_WITH3 VARCHAR2(100),
    DISEASE_NAME VARCHAR2(200),
    DISEASE_CATEGORY VARCHAR2(100),
    DRUG_CATEGORY VARCHAR2(100),
    PRODUCT_ID NUMBER(6),
    PRODUCT_NAME VARCHAR2(150),
    COMPANY_NAME VARCHAR2(150),
    CLINICAL_TRIAL_TITLE VARCHAR2(300),
    CLINICAL_TRIAL_START_DATE VARCHAR2(50),
    CLINICAL_TRIAL_COMPLETION_DATE VARCHAR2(50),
    CLINICAL_TRIAL_PARTICIPANTS VARCHAR2(50),
    CLINICAL_TRIAL_STATUS VARCHAR2(50),
    CLINICAL_TRIAL_CONDITION1 VARCHAR2(100),
    CLINICAL_TRIAL_CONDITION2 VARCHAR2(100),
    CLINICAL_TRIAL_ADDRESS1 VARCHAR2(200),
    CLINICAL_TRIAL_INSTITUTION VARCHAR2(150),
    CLINICAL_TRIAL_ADDRESS2 VARCHAR2(200),
    CLINICAL_TRIAL_MAIN_RESEARCHER VARCHAR2(100),
```

```
    CLINICAL_TRIAL_CONDITION3 VARCHAR2(100)
);

SELECT * FROM DRUGS_FULL;
```

- **Entity Tables:**
  - **Drug Table**

```
CREATE TABLE Drug AS
SELECT
    ROW_NUMBER() OVER (ORDER BY drug_name, drug_category) AS drug_id,
    drug_name,
    drug_category
FROM (
    SELECT DISTINCT drug_name, drug_category
    FROM DRUGS_FULL
    WHERE drug_name IS NOT NULL
) sub;

ALTER TABLE Drug
ADD CONSTRAINT pk_drug PRIMARY KEY (drug_id);
```

  - **Disease Table**

```
CREATE TABLE Disease (
    disease_name VARCHAR2(100),
    disease_category VARCHAR2(100)
);

ALTER TABLE Disease
ADD CONSTRAINT pk_disease PRIMARY KEY (disease_name);
```

  - **SideEffect Table**

```
CREATE TABLE SideEffect (
    side_effect_name VARCHAR2(200)
);

ALTER TABLE SideEffect
ADD CONSTRAINT pk_side_effect PRIMARY KEY (side_effect_name);
```

– **Product Table**

```
CREATE TABLE Product AS
SELECT
    ROW_NUMBER() OVER (ORDER BY product_name, company_name) AS product_id,
    product_name,
    company_name,
    drug_id
FROM (
    SELECT DISTINCT product_name, company_name, drug_id
    FROM DRUGS_FULL
    WHERE product_name IS NOT NULL
    AND company_name IS NOT NULL AND drug_id IS NOT NULL
) sub;

ALTER TABLE Product
ADD CONSTRAINT pk_product PRIMARY KEY (product_id);
ADD CONSTRAINT fk_product_drug FOREIGN KEY (drug_id)
REFERENCES Drug(drug_id);
```

– **ClinicalTrial Table**

```
CREATE TABLE ClinicalTrial (
    clinical_trial_title VARCHAR2(200),
    clinical_trial_start_date DATE,
    clinical_trial_completion_date DATE,
    clinical_trial_participants NUMBER,
    clinical_trial_status VARCHAR2(50),
    researcher_name VARCHAR2(200),
    institute_name VARCHAR2(200)
);

ALTER TABLE ClinicalTrial
ADD CONSTRAINT pk_title PRIMARY KEY (clinical_trial_title);
```

– **Condition Table**

```
CREATE TABLE Condition (
    condition_name VARCHAR2(100)
);

ALTER TABLE Condition
ADD CONSTRAINT pk_name PRIMARY KEY (condition_name);
```

- **Institution Table**

```
CREATE TABLE Institution (
    institute_name VARCHAR2(200),
    country VARCHAR2(100),
    address VARCHAR2(200)
);

ALTER TABLE Institution
ADD CONSTRAINT pk_ins_name PRIMARY KEY (institute_name);
```

- **Relation Tables:**

  - **Treats Relation**

```
CREATE TABLE Treats (
    drug_id NUMBER,
    disease_name VARCHAR2(100),
    FOREIGN KEY (drug_id) REFERENCES Drug(drug_id),
    FOREIGN KEY (disease_name) REFERENCES Disease(disease_name)
);
```

  - **HasSideEffect Relation**

```
CREATE TABLE HasSideEffect (
    drug_id NUMBER,
    side_effect_name VARCHAR2(200),
    FOREIGN KEY (drug_id) REFERENCES Drug(drug_id),
    FOREIGN KEY (side_effect_name) REFERENCES SideEffect(side_effect_name)
);
```

  - **TestedIn Relation**

```
CREATE TABLE TestedIn (
    drug_id NUMBER,
    clinical_trial_title VARCHAR2(200),
    FOREIGN KEY (drug_id) REFERENCES Drug(drug_id),
    FOREIGN KEY (clinical_trial_title)
    REFERENCES ClinicalTrial(clinical_trial_title)
);
```

  - **StudiedIn Relation**

```
CREATE TABLE StudiedIn (
    clinical_trial_title VARCHAR2(200),
    condition_name VARCHAR2(100),
    FOREIGN KEY (clinical_trial_title)
    REFERENCES ClinicalTrial(clinical_trial_title),
    FOREIGN KEY (condition_name)
    REFERENCES Condition(condition_name)
);
```

– **InteractsWith Relation**

```
CREATE TABLE InteractsWith (
    drug_id NUMBER,
    interacts_with_name VARCHAR2(100),
    FOREIGN KEY (drug_id) REFERENCES Drug(drug_id)
);
```

These SQL statements define the complete relational schema, capturing all entities and relationships.

Sub Task - 06:
Lossless Decomposition Proof

To prove that the decomposition of a relation is **lossless**, we use the method of attribute closure.

Let a relation schema R be decomposed into $R_1, R_2, \ldots, R_n$ under a set of functional dependencies F. The decomposition is lossless if for at least one relation R_i, the intersection $R_i \cap R_j$ (for some $j \neq i$) is a superkey in R_i under F.

We verify lossless decomposition for each of the normalized relations obtained from the original `DRUGS_FULL` relation.

1. **Drug and Product**

   Decomposed from: DRUGS_FULL (drug_id, drug_name, drug_category, product_id, product_name, company_name)

   Relations:

   - Drug (drug_id, drug_name, drug_category)
   - Product (product_id, product_name, company_name, drug_id)

   Relevant FDs:

   - product_id → product_name, company_name, drug_id
   - drug_id → drug_name, drug_category

   Common attribute: drug_id

   Closure: $\{drug\_id\}^+$

   - drug_id → drug_name, drug_category (from Drug)
   - drug_id → referenced in Product (not a key, but foreign key for connection)

   Although product_id is the primary key of Product, the foreign key drug_id connects the two relations. Since $\{drug\_id\}^+$ covers all attributes of Drug, and serves as a connector in Product, the join preserves the information.

   This Decomposition is lossless.

2. **Drug and HasSideEffect**

   Relations:

   - Drug (drug_id, drug_name, drug_category)
   - HasSideEffect(drug_id, side_effect_name)
   - SideEffect(side_effect_name)

   FDs:

   - drug_id → drug_name, drug_category
   - drug_name → side_effect1...5 → decomposed into multiple rows of (drug_id, side_effect_name)

   Common attribute: drug_id

   Closure: $\{drug\_id\}^+$

   - from drug_id → drug_name
   - HasSideEffect has drug_id + side_effect_name

   $\{drug\_id\}^+ \supseteq$ all attributes of Drug.

   This Decomposition is lossless.

3. **Drug and InteractsWith**

   Relations:

   - Drug(drug_id, drug_name, drug_category)
   - InteractsWith(drug_id, interacts_with_name)

   Common attribute: drug_id

   Closure: $\{drug\_id\}^+ = \{drug\_id, drug\_name, drug\_category\}$

   InteractsWith has no additional attributes.

   LosslessThis Decomposition is lossless.

4. **Drug and Treats and Disease**

   Relations:

   - Drug(drug_id, drug_name, drug_category)
   - Treats(drug_id, disease_name)
   - Disease(disease_name, disease_category)

   FDs:

   - drug_id → drug_name, drug_category
   - disease_name → disease_category

Common attributes: drug_id, disease_name

Closure: $\{drug\_id, disease\_name\}^+$

- drug_id → drug_name, drug_category
- disease_name → disease_category

Closure = $\{drug\_id, disease\_name, drug\_name, drug\_category, disease\_category\}$

This Decomposition is lossless.

5. **Product and Company (company_name is stored in Product)**

   Product(product_id, product_name, company_name, drug_id)

   - product_id → product_name, company_name, drug_id

   No decomposition needed — no further split.

   This Decomposition is lossless.

6. **ClinicalTrial and TestedIn**

   Relations:

   - ClinicalTrial(clinical_trial_title, start_date, completion_date, participants, status, researcher_name, institute_name)
   - TestedIn(drug_id, clinical_trial_title)

   FD:

   - clinical_trial_title → all other trial info

   Common attribute: clinical_trial_title

   Closure: $\{clinical\_trial\_title\}^+ = \{clinical\_trial\_title, start\_date, completion\_date, participants, status, researcher\_name, institute\_name\}$

   → Covers all attributes of ClinicalTrial.

   This Decomposition is lossless.

7. **ClinicalTrial and StudiedIn and Condition**

   Relations:

   - ClinicalTrial(clinical_trial_title, ...)
   - StudiedIn(clinical_trial_title, condition_name)
   - Condition(condition_name)

   Closure:

   - From clinical_trial_title → all attributes in ClinicalTrial
   - From StudiedIn: (clinical_trial_title, condition_name) → links to Condition

   $\{clinical\_trial\_title\}^+$ = includes everything in ClinicalTrial.

   This Decomposition is lossless.

8. **ClinicalTrial and Institution**

   Institution (institute_name, address, country)

   - institute_name → address, country

   Used as a foreign key in ClinicalTrial.

   Common attribute: institute_name

   Closure: $\{institute\_name\}^+ = \{institute\_name, address, country\}$.

   This Decomposition is lossless.

Sub Task - 07:
Dependency Preservation Proof

We aim to verify whether the decomposition of the original relation `DRUGS_FULL` preserves all functional dependencies using attribute closure.

Original Functional Dependencies (F)

Assuming `DRUGS_FULL` contains the following functional dependencies:

- **FD1:** drug_id → drug_name, drug_category
- **FD2:** product_id → product_name, company_name, drug_id
- **FD3:** disease_name → disease_category
- **FD4:** clinical_trial_title → clinical_trial_start_date, clinical_trial_completion_date, clinical_trial_participants, clinical_trial_status, clinical_trial_main_researcher, clinical_trial_institution, clinical_trial_address1, clinical_trial_address2
- **FD5:** clinical_trial_title → conditions:

  clinical_trial_condition1, clinical_trial_condition2, clinical_trial_condition3
- **FD6:** institute_name → clinical_trial_address1, clinical_trial_address2

  Each institution determines its addresses and country.

Functional Dependencies of Decomposed Relations (G)

- **Drug:** drug_id → drug_name, drug_category
- **Product:** product_id → product_name, company_name, drug_id
- **Disease:** disease_name → disease_category
- **ClinicalTrial:**

  clinical_trial_title → clinical_trial_start_date, clinical_trial_completion_date, clinical_trial_participants, clinical_trial_status, clinical_trial_main_researcher, clinical_trial_institution, clinical_trial_address1, clinical_trial_address2, clinical_trial_condition1, clinical_trial_condition2, clinical_trial_condition3
- **Institution:** institute_name → clinical_trial_address1, clinical_trial_address2

Verifying Dependency Preservation via Attribute Closure

- **FD1:** drug_id → drug_name, drug_category
  From Drug table: FD is directly present. ⇒ $\{drug\_id\}^+ \supseteq \{drug\_name, drug\_category\}$

  Dependency Preserved.

- **FD2:** product_id → product_name, company_name, drug_id
  From Product table: FD is explicitly defined. ⇒ $\{product\_id\}^+ \supseteq \{product\_name, company\_name$

  Dependency Preserved.

- **FD3:** disease_name → disease_category
  From Disease table: FD is directly present. ⇒ $\{disease\_name\}^+ \supseteq \{disease\_category\}$

  Dependency Preserved.

- **FD4:** clinical_trial_title →
  clinical_trial_title → clinical_trial_start_date, clinical_trial_completion_date, clinical_trial_participants, clinical_trial_status, clinical_trial_main_researcher, clinical_trial_institution, clinical_trial_address1, clinical_trial_address2,

  From ClinicalTrial table: all attributes are included. ⇒ $\{clinical\_trial\_title\}^+$

  contains all trial info.

  Dependency Preserved.

- **FD5:** clinical_trial_title → trial conditions
  Conditions also stored in ClinicalTrial. ⇒ $\{clinical\_trial\_title\}^+ \supseteq$ condition1,

  condition2, condition3.

  Dependency Preserved.

- **FD6:** institute_name → clinical_trial_address1, clinical_trial_address2
  From Institution table. ⇒ $\{institute\_name\}^+ \supseteq$ address1, address2

  Dependency Preserved.

Since every original functional dependency from F is preserved in at least one of the decomposed relations in G, and verified via attribute closure:

$$F \subseteq (F_1 \cup F_2 \cup \cdots \cup F_n)^+$$

Therefore, the decomposition is dependency preserving.

Sub Task - 08:
Revising the ER Diagram

Revising the original ER diagram, several design flaws were identified and addressed:

- **Multivalued Attributes:** The original design had multiple attributes for side effects (side_effect1 to side_effect5) and clinical trial conditions (condition1 to condition3). These were replaced with separate entities: `SideEffect` and `Condition`, and linked using relation tables (`HasSideEffect`, `Studies`) to handle the multivalued relationships properly.
- **Unnecessary Attributes in Entities:** Initially, some attributes like clinical_trial_address1 and address2 were stored directly in the `ClinicalTrial` entity. These were moved to the `Institution` entity to follow normalization rules, since institution name determines the address.
- **Missing Relationship Tables:** The original ER model did not include relationship tables like `Treats`, `TestedIn`, `Studies`, and `InteractsWith`, which are necessary to represent many-to-many and recursive relationships between entities. These were added to ensure correct relational mapping.
- **Primary Key Assignment:** In the original design, some entities lacked clear primary keys. The revised diagram introduces surrogate keys such as `drug_id`, `product_id`, and `country_id` where needed.
- **Redundancy and Dependency Issues:** Functional dependencies like clinical_trial_title $\rightarrow$ all trial attributes were scattered in the flat schema. These were normalized by isolating trial and institution as distinct entities.

Therefore, The revised ER diagram reflects a more accurate, normalized, and maintainable data structure that aligns with best practices in database design and supports functional dependency and lossless decomposition analysis.

The revised ER Diagram:

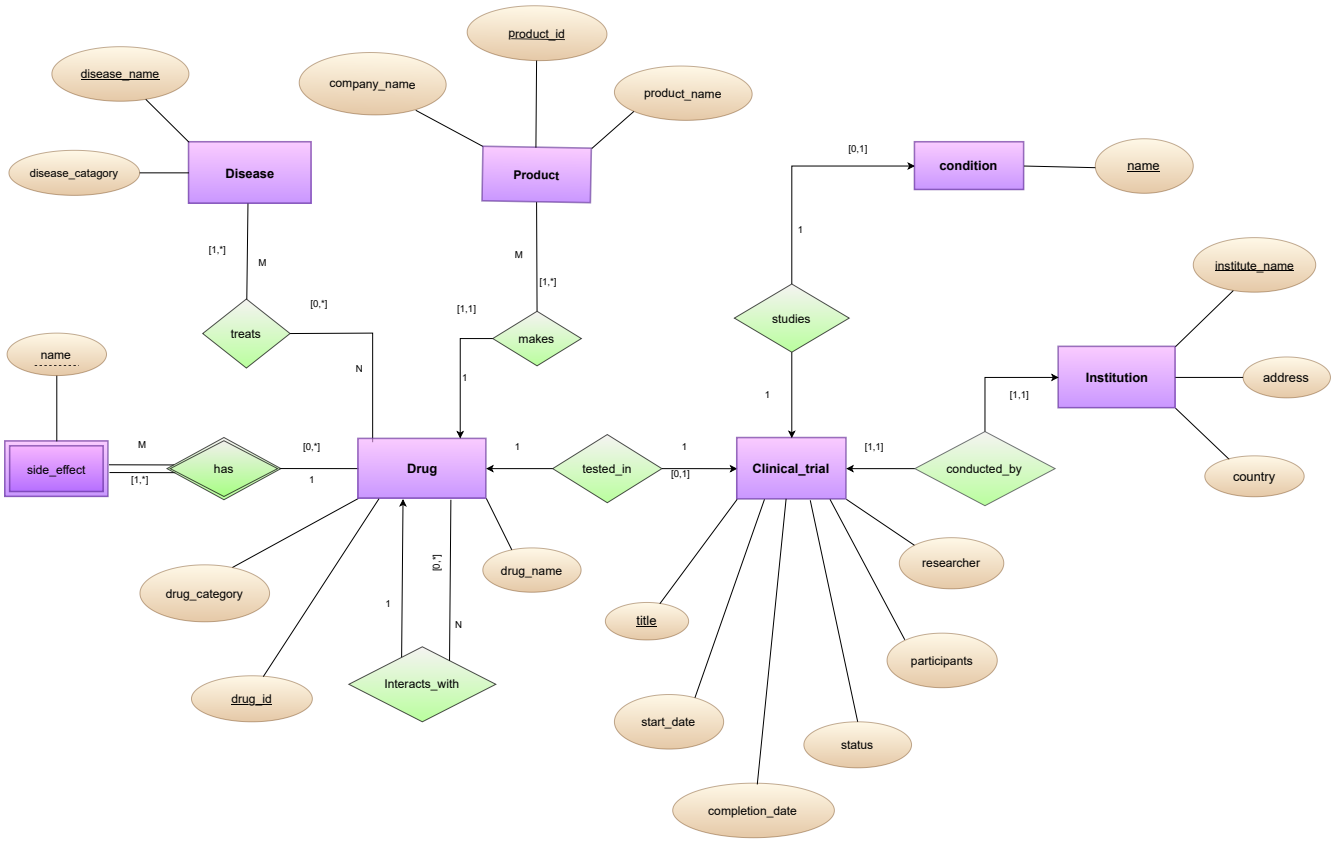

Figure 1: Revised ER Diagram

Sub Task - 09:
Comparison Between Revised ER Diagram and Earlier Designs

In this section, I compare my current ER diagram and normalized relational schema (Task 3) with the earlier designs developed in Task 1 and Task 2.

Differences from Previous Design

In Task 1,

My ER diagram lacked clear separation of key entities. Concepts like Product, ClinicalTrial, and Institution were not distinctly modeled. Additionally, attributes such as side effects and interacting drugs were represented as multivalued attributes within the Drug entity, rather than being extracted into separate relationship tables.

By Task 2,

I had improved the relational schema by introducing primary keys and refining some entities. However, redundancy remained—such as storing researcher names and institutional addresses repeatedly within the ClinicalTrial table instead of linking to a separate Institution entity.

In Task 3,

I addressed these issues by:

- Normalizing all multivalued and repetitive attributes.
- Decomposing condition1, condition2, and condition3 into a separate relation via the `StudiedIn` table.
- Adding surrogate primary keys (e.g., `drug_id`, `product_id`) to maintain consistency and support relationships.
- Defining all necessary relationships using appropriate relational tables such as `HasSideEffect`, `InteractsWith`, `TestedIn`, and `Treats`.

Why These Differences Occurred

The earlier designs in **Task 1** and **Task 2** were based on limited exposure to database modeling principles. At the time, my approach was mostly intuitive and focused on capturing information directly from the dataset.

Through the course and deeper understanding of database theory—including functional dependencies, normalization, and ER modeling with cardinality and participation constraints. I recognized the importance of removing redundancy, avoiding dependency anomalies, and eliminating multivalued attributes from entities.

Thus, the improvements in **Task 3** were driven by formal methods, particularly lossless and dependency-preserving decomposition strategies.

Advantages of the New Design

- **Reduces Redundancy:** Multivalued attributes like side effects, trial conditions, and interacting drugs are now stored in separate tables without repetition.
- **Clearer Relationships:** Many-to-many and one-to-many relationships are explicitly modeled using relational tables and foreign key references.
- **Improved Query Design:** Since the schema adheres to **3NF** and **BCNF**, queries are more efficient and less prone to ambiguity.
- **Data Integrity:** Defined primary and foreign key constraints ensure consistency and correctness of the data.
- **Scalability:** The design accommodates future additions — such as new drugs, trials, or side effects — without needing to alter the existing schema structure.

So, The revised ER diagram and relational schema from Task 3 are more structured, normalized, and theoretically sound compared to earlier versions from Task 1 and 2. These changes reflect an improved understanding of relational design principles and offer significant benefits in terms of clarity, efficiency, and maintainability.