University of Bari Aldo Moro

Computer Science Department

WHO Disease Monitoring System

Database Systems Project Work

Student:
Mauro Carlucci
830627
m.carlucci66@studenti.uniba.it

Professor:
Paolo Mignone
paolo.mignone@uniba.it

Academic Year 2024/2025

Contents

1 Requirements Collection

Specifics.
"The "WHO" organization has implemented a system for monitoring diseases with detailed descriptions. The system allows WHO storing biological data that could refer to tissues or organs with their description. Specifically, the tissues and organs are described through the donor details, a general condition (control, disease) and, when affected, the specific disease. Furthermore, the system saves the organ/tissue name, textual description, position within the human body, density, if it is required for life (e.g., the brain, the heart and so on). The donors are described through name, surname, date of birth, age, sex. The company has the storage about the diseases and their descriptions: name, discovery date and time, and textual description, whether it is treatable or not, the possible cure. A cure is described with a treatment, a list of drugs to be taken, while the drugs have the name, description, a percentage of success for a specific disease, a list of possible allergies linked to such a drug. For each organ/tissue, the system saves the sequence of treatments with specific drugs and the effects observed. For each positive observed effect, the system could suggest a possible new cure/treatment to be investigated and it suggests possible future works to the list of researchers registered. The system maintains the information of the registered researchers. Specifically, it saves the name, surname, list of publications. A publication is described via a DOI identifier, title, journal, or conference with a degree of quality (top, middle, low)."

2 Requirements Analysis

2.1 Choosing the correct level of abstraction

At line 2, the description of biological data is general, since it may refer either to tissues or organs. Therefore, we assume that a biological data entry is classified as either *tissue* or *organ*, and for each of them the following details are stored: id, name, textual description, position within the human body, density, and whether it is required for life (e.g., brain, heart).

At line 3, the description of tissues and organs refers to donor details. We assume that each donor is described through: name, surname, date of birth, age, sex, and a unique CF identifier.

At line 4, the condition of a tissue/organ is described as general (control or disease). We assume that for each biological data entry there is a condition attribute that can be either *control* or *disease*. In case of *disease*, the biological data entry references a specific disease.

At line 5, the system stores diseases with a general description. We assume that each disease is characterized by: id, name, discovery date, textual description, and whether it is treatable or not.

At line 6, the requirement states that a cure is defined as a treatment. Since no specific distinction is provided between the two concepts, we assume that *cure* and *treatment* refer to the same entity. Each treatment is characterized by: id, name, and a description.

At line 7, the drugs are described in a general manner. Therefore, we assume that each drug is characterized by: id, name, description, and a percentage of success for a specific disease. Drugs may cause allergies.

At line 8, the requirement refers to allergies but does not specify their details. We assume that each allergy is characterized by: id, name, and description.

At line 10, the requirement specifies that positive observed effects may suggest possible new cures or treatments to be investigated. We assume that these *future works* are represented as proposals generated from positive treatments. Each future work is characterized by: id, title, and textual description.

At line 11, the system maintains registered researchers. We assume that each researcher is described by: name, surname, and a unique CF identifier.

At line 12, publications are described in a general way. We assume that each publication is characterized by: DOI identifier, title, publisher (journal or conference), and a quality level (*top*,

middle, *low*).

2.1.1 Output

The "WHO" organization has implemented a system for monitoring diseases with detailed descriptions. The system manages biological data classified as either organs or tissues. Each biological data entry is characterized by: id, name, textual description, position within the human body, density, and whether it is required for life. Each organ or tissue is associated with a donor, characterized by name, surname, date of birth, age, sex, and a CF identifier, and with a condition that can be *control* or *disease*. If the condition is *disease*, the biological data entry references a specific disease.

Each disease is characterized by: id, name, discovery date, description, and treatability. For every disease, one or more treatments may exist. A treatment is described by: id, name, and description, and is composed of one or more drugs. Drugs are characterized by: id, name, description, success percentage, and possible related allergies. Allergies are described by: id, name, and description.

For each organ or tissue, the system records a sequence of treatments. A treatment applied to a disease is characterized by: id, date, effect description, and outcome (positive or negative). When positive effects are observed, the system may generate *future works* to be considered for further investigation. Each future work has an id, a title, and a description.

Researchers are maintained in the system and are characterized by: name, surname, and a CF identifier. Each researcher may be associated with future works and produces publications. A publication is characterized by: DOI identifier, title, publisher (journal or conference), and a quality level (*top*, *middle*, *low*).

2.2 Standardize sentence structure

Main sentence	Sentence transformed
The "WHO" organization has implemented a system for monitoring diseases with detailed descriptions.	The "WHO" organization manages a system for monitoring diseases.
The system manages biological data classified as either organs or tissues. Each biological data entry is characterized by: id, name, textual description, position within the human body, density, and whether it is required for life.	The system manages biological data. For biological data, we handle the id, name, textual description, position, density, and life requirement. Biological data can be classified as either organ or tissue.
Each organ or tissue is associated with a donor, characterized by name, surname, date of birth, age, sex, and a CF identifier, and with a condition that can be *control* or *disease*. If the condition is *disease*, the biological data entry references a specific disease.	Each organ or tissue is associated with a donor. For donor, we handle the name, surname, date of birth, age, sex, and CF identifier. For biological data, we handle the condition that can be {control, disease}. If condition is disease, the biological data references a disease.
Each disease is characterized by: id, name, discovery date, description, and treatability. For every disease, one or more treatments may exist.	For disease, we handle the id, name, discovery date, description, treatability. Each disease may have one or more treatments.
A treatment is described by: id, name, and description, and is composed of one or more drugs. Drugs are characterized by: id, name, description, success percentage, and possible related allergies. Allergies are described by: id, name, and description.	For treatment, we handle the id, name, description. Each treatment is composed of one or more drugs. For drug, we handle the id, name, description, success percentage. For allergy, we handle the id, name, description.
For each organ or tissue, the system records a sequence of treatments. A treatment applied to a disease is characterized by: id, date, effect description, and outcome (positive or negative).	For organ or tissue, the system records a sequence of treatments. For applied treatment, we handle the id, date, effect description, and outcome {positive, negative}.
When positive effects are observed, the system may generate *future works* to be considered for further investigation. Each future work has an id, a title, and a description.	When positive effects are observed, the system generates future works. For future work, we handle the id, title, description.
Researchers are maintained in the system and are characterized by: name, surname, and a CF identifier. Each researcher may be associated with future works and produces publications.	For researcher, we handle the name, surname, CF identifier. Researchers can be associated with future works and produce publications.
A publication is characterized by: DOI identifier, title, publisher (journal or conference), and a quality level (*top*, *middle*, *low*).	For publication, we handle the DOI identifier, title, publisher, and quality level {top, middle, low}.

2.2.1 Output

The "WHO" organization manages a system for monitoring diseases. The system manages biological data. For biological data, we handle the id, name, textual description, position, density, and life requirement. Biological data can be classified as either organ or tissue. Each organ or tissue is associated with a donor. For donor, we handle the name, surname, date of birth, age, sex,

and CF identifier. For biological data, we handle the condition that can be {control, disease}. If condition is disease, the biological data references a disease. For disease, we handle the id, name, discovery date, description, treatability. Each disease may have one or more treatments. For treatment, we handle the id, name, description. Each treatment is composed of one or more drugs. For drug, we handle the id, name, description, success percentage. For allergy, we handle the id, name, description. For organ or tissue, the system records a sequence of treatments. For applied treatment, we handle the id, date, effect description, and outcome {positive, negative}. When positive effects are observed, the system generates future works. For future work, we handle the id, title, description. For researcher, we handle the name, surname, CF identifier. Researchers can be associated with future works and produce publications. For publication, we handle the DOI identifier, title, publisher, and quality level {top, middle, low}.

2.3 Identification of synonyms and homonyms and resolving complex phrases

At a certain point we noticed that in the extract a publication can be associated either with a *journal* or a *conference*. This makes the term ambiguous, therefore we decided to replace both *journal* and *conference* with the unique term *publisher*.

We also observed that the extract uses both the terms *cure* and *treatment*. Since they do not introduce any semantic distinction, we decided to keep only the term *treatment*.

After the identification and resolution of these homonyms we also resolved complex phrases.

2.3.1 Output

The "WHO" organization manages a system for monitoring diseases. The system manages biological data. For biological data, we handle the id, name, textual description, position, density, and life requirement. Biological data can be classified as either organ or tissue.

Each organ or tissue is associated with a donor. For donor, we handle the name, surname, date of birth, age, sex, and CF identifier. For biological data, we handle the condition that can be {control, disease}. If condition is disease, the biological data references a disease.

For disease, we handle the id, name, discovery date, description, treatability. Each disease may have one or more treatments. For treatment, we handle the id, name, description. Each treatment is composed of one or more drugs. For drug, we handle the id, name, description, success percentage. For allergy, we handle the id, name, description.

For organ or tissue, the system records a sequence of treatments. For applied treatment, we handle the id, date, effect description, and outcome {positive, negative}. When positive effects are observed, the system generates future works. For future work, we handle the id, title, description.

For researcher, we handle the name, surname, CF identifier. Researchers can be associated with future works and produce publications. For publication, we handle the DOI identifier, title, publisher, and quality level {top, middle, low}.

2.4 Glossary of Terms

Concept	Description	Synonym	Link
Biological Data	Represents an organ or tissue, characterized by structural and descriptive attributes	Organ, Tissue	Donor, Condition, Disease
Donor	Person associated with an organ or tissue, providing personal and identification data	–	Biological Data
Condition	State of biological data, which can be {control, disease}	–	Biological Data, Disease
Disease	Medical condition associated with biological data, characterized by identification and description	–	Biological Data, Treatment
Treatment	Medical intervention applied to a disease, can consist of one or more drugs	Cure	Disease, Drug
Drug	A medicine used within a treatment, characterized by success rate and possible allergies	–	Treatment, Allergy
Allergy	Negative reaction related to a drug	–	Drug
Future Work	Research direction generated from positive outcomes of treatments	–	Researcher
Researcher	Person in the system that can be associated with future works and publications	–	Future Work, Publication
Publication	Scientific output produced by a researcher, published in a publisher with quality level	-	Researcher

2.5 Phrases grouped by Concept

Phrases related to Biological Data The system manages biological data. For biological data, we handle the id, name, textual description, position, density, and life requirement. Biological data can be classified as either organ or tissue. For biological data, we handle the condition that can be {control, disease}. If condition is disease, the biological data references a disease. For organ or tissue, the system records a sequence of treatments.

Phrases related to Donor Each organ or tissue is associated with a donor. For donor, we handle the name, surname, date of birth, age, sex, and CF identifier.

Phrases related to Disease For disease, we handle the id, name, discovery date, description, treatability. Each disease may have one or more treatments.

Phrases related to Treatment For treatment, we handle the id, name, description. Each treatment is composed of one or more drugs. For applied treatment, we handle the id, date, effect description, and outcome {positive, negative}.

Phrases related to Drug For drug, we handle the id, name, description, success percentage.

Phrases related to Allergy For allergy, we handle the id, name, description.

Phrases related to Future Work When positive effects are observed, the system generates future works. For future work, we handle the id, title, description.

Phrases related to Researcher For researcher, we handle the name, surname, CF identifier. Researchers can be associated with future works and produce publications.

Phrases related to Publication For publication, we handle the DOI identifier, title, publisher, and quality level {top, middle, low}.

3 Conceptual Design

3.1 Skeleton Schema

From the specifications, we read that biological data can refer to organs or tissues, which could naturally be translated in an ER diagram as a generalization. However, since the rest of the specifications do not imply any difference in attributes or relationships for organs versus tissues, it was decided to use only the single entity Biological Data. The skeleton schema that outlines these entities and their main relationships is reported in Figure 1. We will adopt a mixed strategy to make further refinements, carefully considering each step while analyzing the phrases grouped by concepts. This approach allows us to iteratively refine the schema by balancing both a top-down perspective and a bottom-up perspective.

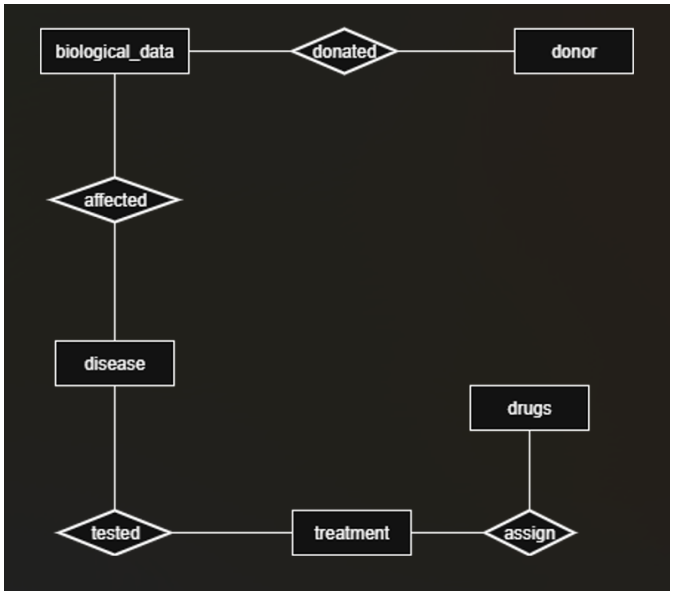

Figure 1: Skeleton Schema representing the main entities of the system

3.2 Conceptual Schema

The conceptual schema has been generated using a mixed strategy that combines both bottom-up and top-down refinements. The resulting model is shown in Figure 2. During the design phase, a crucial modeling choice concerned the relationship between *Disease*, *Treatment*, and *Future Work*. Initially, this was modeled as a ternary relationship, since a *Future Work* can be proposed only as the result of a specific treatment applied to a disease. However, this direct ternary relationship was reified by introducing a new entity: *Experiment*. The *Experiment* entity captures the application of a treatment to a disease for a specific organ or tissue, storing the observed effects and outcomes. This reification provides a clearer representation of the semantics:

- *Disease* and *Treatment* are linked through the *Experiment*.
- Each *Experiment* has attributes such as id, date, effect description, and outcome (positive or negative).

- When the outcome is positive, the system may generate one or more *Future Works* directly linked to the *Experiment*.

This design choice improves extensibility and reduces ambiguity.

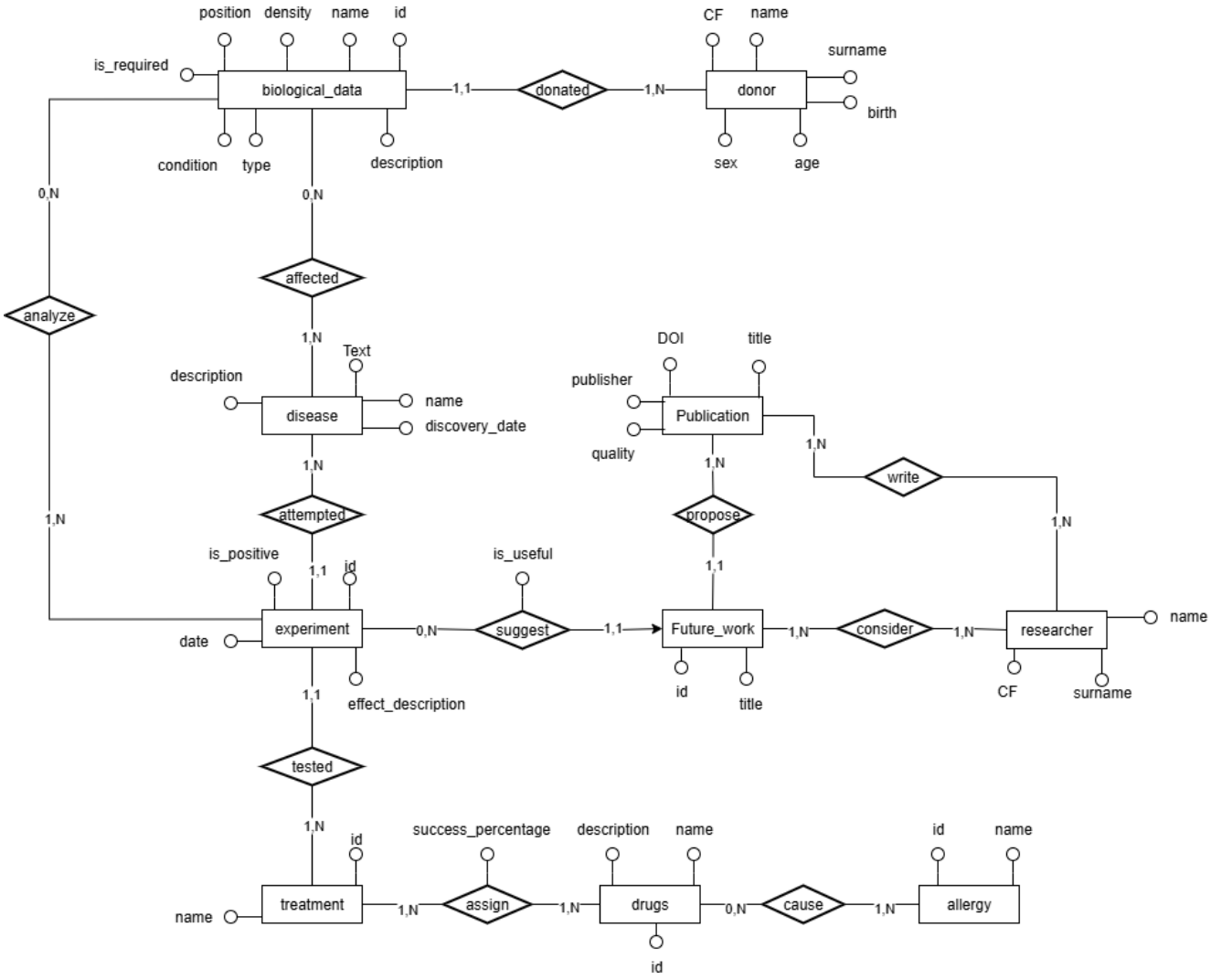

Figure 2: Conceptual Schema of the system after refinements

3.3 Business Rules

The following constraints are not directly representable in the ER diagram.

ID	Rule
BR1	BiologicalData.condition must be one of control, disease.
BR2	BiologicalData.type must be one of organ, tissue.
BR3	BiologicalData.required_for_life must be one of yes, no.
BR4	Experiment.is_positive must be one of yes, no.
BR5	Suggest.is_useful must be one of yes, no.
BR6	Treatment.success_percentage must be an integer in [0, 100].
BR7	Publication.quality must be one of top, middle, low.
BR8	BiologicalData.density must be strictly positive.
BR9	If BiologicalData.condition = disease, then there must exist related Disease via Affected; if condition = control, then no Affected link must exist for that BiologicalData.
BR11	Each FutureWork must be linked to at least one Publication via Propose.
BR12	Experiment.date must be on or after the Disease.discovery_date of the attempted disease.
BR13	Disease.discovery_date must be not in the future (date $\leq$ current date).
BR14	Experiment–sample disease consistency: for any Analyze(BiologicalData, Experiment) and Attempted(Experiment, Disease), there must exist Affected(BiologicalData, Disease). Equivalently, the disease attempted by the experiment must be exactly the disease affecting that BiologicalData. If BiologicalData.condition = control, no such Attempted link is allowed for experiments on that sample.

Table 1: Business Rules Constraints

3.4 Data Dictionary

This section summarizes entities and relationships of the conceptual schema with their main attributes and identifiers. It is intended as a compact reference for the subsequent logical design.

Entity	Description	Attributes	Identifier
Biological Data	Organ or tissue sample managed by the system.	id, name, density, position, description, type (organ|tissue), condition (control|disease), required_for_life (yes|no)	id
Donor	Person associated with a biological sample.	CF, name, surname, birth, age, sex	CF
Disease	Medical condition possibly affecting biological data.	id, name, discovery_date, description	id
Experiment	Application of a treatment to a disease on a specific sample, with observed effects.	id, date, is_positive (yes|no), effect_description	id
Treatment	Medical treatment potentially applied to diseases.	id, name, success_percentage [0..100]	id
Drug	Drug used within treatments.	id, name, description	id
Allergy	Possible allergy linked to drugs.	id, name	id
Future Work	Research direction suggested by positive experiments.	id, title	id
Publication	Scientific output (journal or conference) with a quality level.	DOI, title, publisher, quality (top|middle|low)	DOI
Researcher	Registered researcher producing publications and receiving suggestions.	CF, name, surname	CF

Table 2: Data Dictionary — Entities

Relationship	Description	Entities involved (cardinality)	Attributes
Donated	Associates a biological sample to its donor.	Biological Data (1,1); Donor (1,N)	–
Analyze	Links experiments performed on a specific biological sample.	Biological Data (0,N); Experiment (1,N)	–
Affected	States that a biological sample is affected by a disease.	Biological Data (0,N); Disease (1,N)	–
Attempted	Binds an experiment to the disease it attempts to treat/test.	Disease (1,N); Experiment (1,1)	–
Tested	Binds an experiment to the treatment it applies.	Experiment (1,1); Treatment (1,N)	–
Assign	Many-to-many association between treatments and drugs.	Treatment (1,N); Drug (1,N)	–
Cause	Many-to-many association between drugs and allergies.	Drug (0,N); Allergy (1,N)	–
Suggest	Future works suggested by an experiment.	Experiment (0,N); Future Work (1,1)	is_useful (yes|no)
Propose	Associates future works with related publications.	Future Work (1,1); Publication (1,N)	–
Consider	Delivery of future works to specific researchers.	Future Work (1,N); Researcher (1,N)	–
Write	Authorship relation between researchers and publications.	Publication (1,N); Researcher (1,N)	–

Table 3: Data Dictionary — Relationships

4 Logical Design

In the logical phase, it is essential to consider the operations required to restructure the ER diagram into an optimized system. This process involves analyzing the relationships and entities to ensure that the conceptual model is translated into a more efficient and performant structure.

4.1 Operations

(OP1) Recording an organ or tissue (10 times per day).
(OP2) Print all the organs and tissues below a certain density threshold (once per month).
(OP3) Request information on a specific treatment, its list of drugs, and the possible linked allergies (once per day).
(OP4) Print all the donors with a specific disease affecting only organs/tissues required for life for cui the system provided useful suggestions as future works (once per month).
(OP5) Print all the useful suggestions provided to only researchers with top quality journals published (once per month).

4.2 Table of Volumes and Operations

Considering the requirements, we estimate the following volumes:

- The system manages approximately 10 million organs/tissues.
- There are about 1 million diseases.

- The system maintains 10,000 researchers.
- Each researcher writes on average 50 publications.
- Each publication contains on average 2 future works.
- 1% of the suggestions generated by the system are considered useless.

Concept	Type	Volume	Explanation
Biological Data	E	10,000,000	We assume the system stores around 10 million organs and tissues.
Donor	E	10,000,000	We assume on average each donor donates 1 biological data, thus 10 million donors.
Disease	E	1,000,000	We assume the system tracks about 1 million distinct diseases.
Experiment	E	30,000,000	We assume on average 3 experiments are performed per biological data.
Treatment	E	5,000,000	We assume each disease is linked to 5 treatments on average.
Drug	E	20,000	We assume a pharmaceutical catalog of about 20k drugs.
Allergy	E	5,000	We assume the system stores about 5k known allergies.
Future Work	E	1,000,000	We assume each publication generates on average 2 future works.
Researcher	E	10,000	We assume the system stores 10k researchers.
Publication	E	500,000	We assume 10k researchers each wrote about 50 publications.
Donated	R	10,000,000	Each biological data is linked to exactly one donor.
Affected	R	10,000,000	Each biological data is associated with one disease on average.
Analyze	R	30,000,000	Each biological data generates about 3 experiments.
Attempted	R	30,000,000	Each experiment tests one disease.
Tested	R	30,000,000	Each experiment applies one treatment.
Assign	R	10,000,000	Each treatment is composed of about 2 drugs on average.
Cause	R	40,000	Each Drug has on average 2 possible allergies).
Suggest	R	1,000,000	About 1M future works are suggested by experiments.
Propose	R	1,000,000	Each future work is linked to at least one publication.
Consider	R	3,000,000	Each future work is considered on average by 3 researchers.
Write	R	1.500,000	Each publication is written on average by three researchers.

Table 4: Table of Volumes

Operation	Type	Frequency
OP1: Recording an organ or tissue	I	10 per day
OP2: Print all the organs and tissues below a certain density threshold	R	1 per month
OP3: Request information on a specific treatment, its list of drugs and the possible linked allergies	R	1 per day
OP4: Print all the donors with a specific disease affecting only organs/tissues required for life for cui the system provided useful suggestions as future works	R	1 per month
OP5: Print all the useful suggestions provided to only researchers with top quality journals published	R	1 per month

Table 5: Table of Operations

5 Performance Analysis

5.1 Redundancy Analysis

Redundancy in our conceptual model can arise either from attributes or from structural cycles. In this design:

- Redundant attributes: none. All attributes are functionally dependent on the identifier of their entity, and none can be derived from other stored attributes.
- Structural cycles: two cycles are present and must be kept for semantic reasons; they do not create redundancy because they encode information that cannot be obtained otherwise:
  - Cycle A: FutureWork — Researcher — Publication (links: `consider`, `write`, `propose`). The link `consider` captures the explicit delivery of suggestions to specific researchers and cannot be inferred from authorship or venue.
  - Cycle B: BiologicalData — Disease — Experiment (links: `affected`, `attempted`, `analyze`). The link `analyze` binds each *Experiment* to the exact *Biological Data* (sample). Without it, experiments could only be reached from a *Disease*, not from a given sample.

5.2 Access tables for the main operations

For each operation we show a table with columns: *Concept | Type (E/R) | Access (R/W) | Rows (per execution).* After each table a short calculation note explains the numbers used.

OP1 — Recording an organ or tissue (10 times per day)

Concept	Type	Access	Rows
Biological Data	E	W	1
Donor	E	R	1
Donated (relation)	R	W	1

Table 6: Access table for OP1

Calculation note: Insert one BD row, check donor existence (1 read) and insert the linking row (1 write). Total accesses per execution = 3. Frequency = 10/day → 30 accesses/day.

OP2 — Print all organs/tissues below a density threshold (once per month)

Concept	Type	Access	Rows
Biological Data	E	R	10,000,000

Table 7: Access table for OP2

Calculation note: the query must scan all BD rows (10M).

OP3 — Request information on a specific treatment, its list of drugs and possible linked allergies (once per day)

Concept	Type	Access	Rows
Treatment	E	R	1
Assign	R	R	2
Drug	E	R	2
Cause	R	R	4
Allergy	E	R	4

Table 8: Access table for OP3

Calculation note:

- Each treatment assigns on average 2 drugs
- Each drug causes on average 2 allergies
- Therefore for 2 drugs we estimate 4 possible allergies
- Total estimated reads $\approx 1 + 2 + 2 + 4 + 4 = 13$ rows per execution.

OP4 — Print donors with a specific disease on required-for-life biological data that led to useful future works (once per month)

Concept	Type	Access	Rows
Disease	E	R	1
Affected	R	R	10
Biological Data	E	R	10
Analyze	R	R	3
Attempted	R	R	3
Suggest	R	R	1
Future Work	E	R	1
Donated	R	R	1
Donor	E	R	1

Table 9: Access table for OP4

Calculation note:

- Affected edges per disease: $10\,000\,000/1\,000\,000 = 10$ BD on average for a specific disease.
- Assumption: only 10% of BD are required for life $\Rightarrow$ among those 10 BD, about 1 qualifies for the next steps.

- Experiments per BD: $30\,000\,000/10\,000\,000 = 3$ (via `analyze`).
- Assumption: for an affected BD, its experiments attempt the same disease (so `attempted` rows ≈ 3 here).
- Useful suggestions: we assume 99% of future works are useful $\Rightarrow$ we count 1 FW for the qualifying path.
- Total estimated reads along one qualifying path $\approx 1 + 10 + 10 + 3 + 3 + 1 + 1 + 1 + 1 = 31$ rows per execution.

OP5 — Print useful suggestions sent to researchers with top-quality publications (once per month)

Concept	Type	Access	Rows
Publication	E	R	500,000
Write	R	R	450,000
Researcher	E	R	3,000
Consider	R	R	900,000
Future Work	E	R	300,000

Table 10: Access table for OP5

Calculation note:

- Publications: full scan 500,000; keep top (assumption: 30% $\Rightarrow$ 150,000).
- Write: $150{,}000 \times 3 = 450{,}000$ authorships.
- Researchers: distinct authors $\Rightarrow$ 3,000 eligible.
- Consider: share for eligible researchers $= 3{,}000{,}000 \times (3{,}000/10{,}000) = 900{,}000$ pairs.
- Future Work: each FW is considered on average by 3 researchers.

5.3 Object-Relational Considerations

Based on the UML, we will use Oracle object-relational features with a light, consistent approach:

- **Typed tables and hierarchy.** Define an abstract `person_typ` with common attributes; create `donor_typ` and `researcher_typ` UNDER `person_typ`; store them in typed tables to reuse identity and constraints.
- **One typed table per core entity.** For `biological_data_typ`, `disease_typ`, `experiment_typ`, `treatment_typ`, `drugs_typ`, `allergy_typ`, `future_work_typ`, and `publication_typ` we create corresponding typed tables; natural identifiers (e.g., DOI, CF) remain attributes.
- **M:N as association types.** Model `assign_typ`, `cause_typ`, `writes_typ`, and `consider_typ` as separate typed tables with REFs to endpoints.
- **1:N via REFs.** Place REFs on the N-side: each `experiment_typ` references its `disease_typ` and `treatment_typ`; each `future_work_typ` references its `experiment_typ` and `publication_typ`.

5.4 Translation into Logical Schema (UML)

The following UML class diagram summarizes the object types, inheritance, and associations used in the logical mapping.

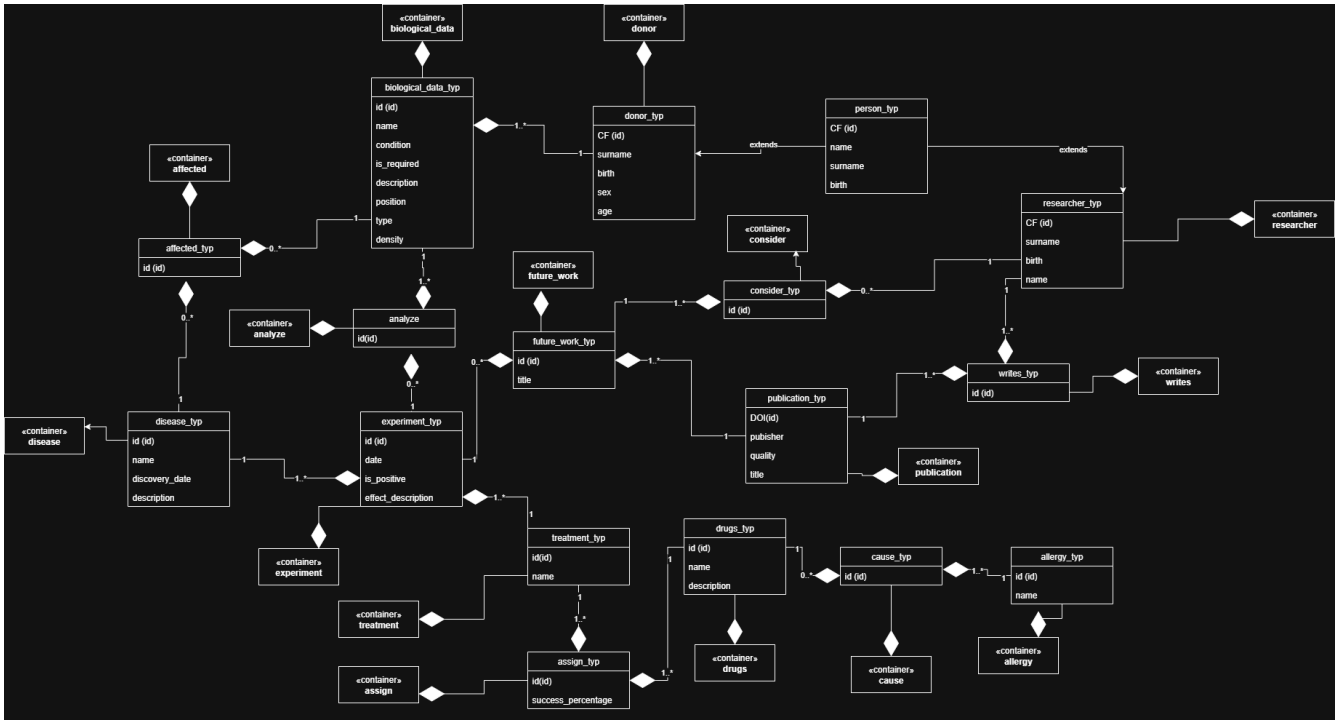

Figure 3: UML class diagram for the object–relational logical schema

6 Physical Design

In this section, we discuss the physical design of our system, where the types defined in the logical and UML schemas are implemented in Oracle using object types, object tables, and association tables. This step translates the abstract entities and relationships into concrete physical structures that support persistence and efficient querying.

6.1 Person hierarchy

PERSON_TYP, DONOR_TYP, RESEARCHER_TYP

```
REATE OR REPLACE TYPE person_typ AS OBJECT (
  CF        CHAR(16),
  name      VARCHAR2(100),
  surname   VARCHAR2(100),
  birth     DATE
) NOT FINAL;
/

CREATE OR REPLACE TYPE donor_typ UNDER person_typ (
  sex CHAR(1),
  age NUMBER(3)
);
/
```

```
CREATE OR REPLACE TYPE researcher_typ UNDER person_typ ();
/

CREATE TABLE donors_tab OF donor_typ (
  CF PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;

CREATE TABLE researchers_tab OF researcher_typ (
  CF PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

We define a general type `person_typ` as the supertype, capturing the attributes common to all persons. Two subtypes, `donor_typ` and `researcher_typ`, extend this base type with their specific details. Typed tables `donors_tab` and `researchers_tab` store the instances, with primary keys defined on the unique identifier CF.

6.2 Entities

DISEASE_TYP

```
CREATE OR REPLACE TYPE disease_typ AS OBJECT (
  id               NUMBER,
  name             VARCHAR2(200),
  discovery_date   DATE,
  description      CLOB
);
/

CREATE TABLE disease_tab OF disease_typ (
  id PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Diseases are represented as a standalone type with a corresponding object table.

BIOLOGICAL_DATA_TYP

```
CREATE OR REPLACE TYPE biological_data_typ AS OBJECT (
  id             NUMBER,
  name           VARCHAR2(200),
  condition      VARCHAR2(200),
  is_required    CHAR(1),
  description    CLOB,
  position       VARCHAR2(100),
  data_type      VARCHAR2(100),
  density        NUMBER,
  donor_ref      REF donor_typ
);
/

CREATE TABLE biological_data_tab OF biological_data_typ (
  id PRIMARY KEY,
  donor_ref NOT NULL,
  SCOPE FOR (donor_ref) IS donors_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Biological data are modeled as a type containing attributes describing tissues and organs,

with a REF to the donor. The typed table enforces referential integrity via the SCOPE clause.

EXPERIMENT_TYP

```
CREATE OR REPLACE TYPE experiment_typ AS OBJECT (
  id                   NUMBER,
  exper_date           DATE,
  is_positive          CHAR(1),
  effect_description   VARCHAR2(1000),
  disease_ref          REF disease_typ,
  treatment_ref        REF treatment_typ
);
/

CREATE TABLE experiment_tab OF experiment_typ (
  id PRIMARY KEY,
  disease_ref NOT NULL,
  treatment_ref NOT NULL,
  SCOPE FOR (disease_ref) IS disease_tab,
  SCOPE FOR (treatment_ref) IS treatment_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Experiments are linked to both diseases and treatments through REFs.

TREATMENT_TYP

```
CREATE OR REPLACE TYPE treatment_typ AS OBJECT (
  id                  NUMBER,
  name                VARCHAR2(200),
  success_percentage  NUMBER(5,2)
);
/

CREATE TABLE treatment_tab OF treatment_typ (
  id PRIMARY KEY,
  name NOT NULL,
  success_percentage NOT NULL
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Experiments reference the treatment they apply; the treatment stores the overall success_percentage for its drug combination.

DRUGS_TYP

```
CREATE OR REPLACE TYPE drugs_typ AS OBJECT (
  id          NUMBER,
  name        VARCHAR2(200),
  description CLOB
);
/

CREATE TABLE drugs_tab OF drugs_typ (
  id PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

ALLERGY_TYP

```
CREATE OR REPLACE TYPE allergy_typ AS OBJECT (
  id    NUMBER,
  name  VARCHAR2(200)
);
/

CREATE TABLE allergy_tab OF allergy_typ (
  id PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Drugs and allergies are both modeled as object types with dedicated tables, allowing reuse and avoiding redundancy. The relationship between them is modeled separately as an association.

6.3 Publications and Future Works

PUBLICATION_TYP

```
CREATE OR REPLACE TYPE publication_typ AS OBJECT (
  DOI        VARCHAR2(120),
  publisher  VARCHAR2(200),
  quality    VARCHAR2(50),
  title      VARCHAR2(300)
);
/

CREATE TABLE publication_tab OF publication_typ (
  DOI PRIMARY KEY
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

FUTURE_WORK_TYP

```
CREATE OR REPLACE TYPE future_work_typ AS OBJECT (
  id        NUMBER,
  title     VARCHAR2(300),
  exp_ref   REF experiment_typ,
  pub_ref   REF publication_typ
);
/

CREATE TABLE future_work_tab OF future_work_typ (
  id PRIMARY KEY,
  exp_ref NOT NULL,
  pub_ref NOT NULL,
  SCOPE FOR (exp_ref) IS experiment_tab,
  SCOPE FOR (pub_ref) IS publication_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

Publications and future works are implemented as separate object types. Future works hold references to both experiments and publications.

6.4 Association types and tables

Many-to-many relationships from the conceptual schema are mapped to association types and stored in dedicated tables. Each table scopes its `REF`s to the appropriate target tables. Pair-wise

uniqueness is enforced with triggers.

ASSIGN_TYP

```
CREATE OR REPLACE TYPE assign_typ AS OBJECT (
  id             NUMBER,
  treatment_ref  REF treatment_typ,
  drug_ref       REF drugs_typ
);
/

CREATE TABLE assign_tab OF assign_typ (
  id PRIMARY KEY,
  treatment_ref NOT NULL,
  drug_ref NOT NULL,
  SCOPE FOR (treatment_ref) IS treatment_tab,
  SCOPE FOR (drug_ref) IS drugs_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

The `assign_tab` table represents the many-to-many relationship between treatments and drugs.

CAUSE_TYP

```
CREATE OR REPLACE TYPE cause_typ AS OBJECT (
  id           NUMBER,
  drug_ref     REF drugs_typ,
  allergy_ref  REF allergy_typ
);
/

CREATE TABLE cause_tab OF cause_typ (
  id PRIMARY KEY,
  drug_ref NOT NULL,
  allergy_ref NOT NULL,
  SCOPE FOR (drug_ref) IS drugs_tab,
  SCOPE FOR (allergy_ref) IS allergy_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

The `cause_tab` table models the association between drugs and allergies.

ANALYZE_TYP

```
CREATE OR REPLACE TYPE analyze_typ AS OBJECT (
  id        NUMBER,
  bio_ref   REF biological_data_typ,
  exp_ref   REF experiment_typ
);
/

CREATE TABLE analyze_tab OF analyze_typ (
  id PRIMARY KEY,
  bio_ref NOT NULL,
  exp_ref NOT NULL,
  SCOPE FOR (bio_ref) IS biological_data_tab,
  SCOPE FOR (exp_ref) IS experiment_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

The `analyze_tab` table records which biological data have been analyzed in which experiments.

AFFECTED_TYP

```
CREATE OR REPLACE TYPE affected_typ AS OBJECT (
  id           NUMBER,
  bio_ref      REF biological_data_typ,
  disease_ref  REF disease_typ
);
/

CREATE TABLE affected_tab OF affected_typ (
  id PRIMARY KEY,
  bio_ref NOT NULL,
  disease_ref NOT NULL,
  SCOPE FOR (bio_ref) IS biological_data_tab,
  SCOPE FOR (disease_ref) IS disease_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

The `affected_tab` table models the association between biological data and diseases.

WRITES_TYP

```
CREATE OR REPLACE TYPE writes_typ AS OBJECT (
  id               NUMBER,
  publication_ref  REF publication_typ,
  researcher_ref   REF researcher_typ
);
/

CREATE TABLE writes_tab OF writes_typ (
  id PRIMARY KEY,
  publication_ref NOT NULL,
  researcher_ref NOT NULL,
  SCOPE FOR (publication_ref) IS publication_tab,
  SCOPE FOR (researcher_ref) IS researchers_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

CONSIDER_TYP

```
CREATE OR REPLACE TYPE consider_typ AS OBJECT (
  id               NUMBER,
  future_work_ref  REF future_work_typ,
  researcher_ref   REF researcher_typ
);
/

CREATE TABLE consider_tab OF consider_typ (
  id PRIMARY KEY,
  future_work_ref NOT NULL,
  researcher_ref NOT NULL,
  SCOPE FOR (future_work_ref) IS future_work_tab,
  SCOPE FOR (researcher_ref) IS researchers_tab
) OBJECT IDENTIFIER IS PRIMARY KEY;
```

The `writes_tab` and `consider_tab` tables represent the associations between researchers and their publications or future works.

6.5 Check constraints (enumerations and ranges)

We enforce value domains and basic ranges using CHECK constraints directly on the typed tables.

Enumerations and ranges (BR1, BR2, BR3, BR4, BR6, BR7, BR8)

```
- BR1: BiologicalData.condition in {control, disease}
ALTER TABLE biological_data_tab
  ADD CONSTRAINT chk_bd_condition
  CHECK (LOWER(condition) IN ('control','disease'));

-- BR3: BiologicalData.is_required in {Y,N}
ALTER TABLE biological_data_tab
  ADD CONSTRAINT chk_bd_is_required
  CHECK (UPPER(is_required) IN ('Y','N'));

-- BR2: BiologicalData.data_type in {organ, tissue}
ALTER TABLE biological_data_tab
  ADD CONSTRAINT chk_bd_data_type
  CHECK (LOWER(data_type) IN ('organ','tissue'));

-- BR8: BiologicalData.density > 0
ALTER TABLE biological_data_tab
  ADD CONSTRAINT chk_bd_density_pos
  CHECK (density > 0);

-- BR4: Experiment.is_positive in {Y,N}
ALTER TABLE experiment_tab
  ADD CONSTRAINT chk_experiment_is_positive
  CHECK (UPPER(is_positive) IN ('Y','N'));

-- BR6: Treatment.success_percentage integer in [0,100]
ALTER TABLE treatment_tab
  ADD CONSTRAINT chk_treatment_success
  CHECK (success_percentage BETWEEN 0 AND 100 AND success_percentage =
  ↪  TRUNC(success_percentage));

-- BR7: Publication.quality in {top,middle,low}
ALTER TABLE publication_tab
  ADD CONSTRAINT chk_publication_quality
  CHECK (LOWER(quality) IN ('top','middle','low'));
```

These constraints are lightweight, declarative, and push invalid data out early without procedural code.

6.6 Triggers

This subsection documents the triggers used to enforce business rules and cross-entity consistency in the Oracle physical schema. Each code box shows the exact DDL, followed by a short rationale.

Temporal consistency for diseases and experiments (BR12, BR13)

```
- BR12: Experiment.exper_date \>= Disease.discovery_date
CREATE OR REPLACE TRIGGER trg_experiment_date_vs_disease
BEFORE INSERT OR UPDATE OF exper_date, disease_ref ON experiment_tab
FOR EACH ROW
DECLARE
```

```
  v_discovery_date DATE;
BEGIN
  SELECT d.discovery_date INTO v_discovery_date
  FROM disease_tab d
  WHERE REF(d) = :NEW.disease_ref;

  IF :NEW.exper_date < v_discovery_date THEN
    RAISE_APPLICATION_ERROR(-20001,
    ↪  'Experiment date must be on or after the Disease discovery_date');
  END IF;
END;
/

-- BR13: Disease.discovery_date must not be in the future
CREATE OR REPLACE TRIGGER trg_disease_discovery_not_future
BEFORE INSERT OR UPDATE OF discovery_date ON disease_tab
FOR EACH ROW
BEGIN
  IF :NEW.discovery_date > SYSDATE THEN
    RAISE_APPLICATION_ERROR(-20002,
    ↪  'Disease discovery_date cannot be in the future');
  END IF;
END;
/
```

These triggers guarantee temporal sanity: experiments cannot predate the discovery of the disease they attempt; diseases cannot be discovered in the future.

Biological Data condition and Affected consistency (BR9)

```
-- Allow Affected only if BD.condition = 'disease'
CREATE OR REPLACE TRIGGER trg_affected_insert_check_condition
BEFORE INSERT OR UPDATE OF bio_ref ON affected_tab
FOR EACH ROW
DECLARE
  v_condition VARCHAR2(200);
BEGIN
  SELECT b.condition INTO v_condition FROM biological_data_tab b WHERE REF(b) =
  ↪  :NEW.bio_ref;
  IF LOWER(v_condition) <> 'disease' THEN
    RAISE_APPLICATION_ERROR(-20003,
    ↪  'Affected link allowed only for BiologicalData with condition = disease');
  END IF;
END;
/

-- Prevent deleting the last Affected when BD is diseased
CREATE OR REPLACE TRIGGER trg_affected_prevent_delete_last
BEFORE DELETE ON affected_tab
FOR EACH ROW
DECLARE
  v_condition VARCHAR2(200);
  v_cnt NUMBER;
BEGIN
  SELECT b.condition INTO v_condition FROM biological_data_tab b WHERE REF(b) =
  ↪  :OLD.bio_ref;
  IF LOWER(v_condition) = 'disease' THEN
```

```
    SELECT COUNT(*) INTO v_cnt
    FROM affected_tab a
    WHERE a.bio_ref = :OLD.bio_ref
      AND NOT (a.bio_ref = :OLD.bio_ref AND a.disease_ref = :OLD.disease_ref);
    IF v_cnt = 0 THEN
      RAISE_APPLICATION_ERROR(-20004,
      ↪  'Cannot delete the last Affected row for a diseased BiologicalData');
    END IF;
  END IF;
END;
/

-- Forbid setting condition=control if Affected links exist
CREATE OR REPLACE TRIGGER trg_bd_condition_change
BEFORE UPDATE OF condition ON biological_data_tab
FOR EACH ROW
DECLARE
  v_cnt NUMBER;
BEGIN
  IF LOWER(:NEW.condition) = 'control' THEN
    SELECT COUNT(*) INTO v_cnt
    FROM affected_tab a
    WHERE a.bio_ref = (SELECT REF(b) FROM biological_data_tab b WHERE b.id =
    ↪  :NEW.id);
    IF v_cnt > 0 THEN
      RAISE_APPLICATION_ERROR(-20010,
      ↪  'Cannot set condition=control while Affected links exist');
    END IF;
  END IF;
END;
/
```

These three triggers keep `Biological Data` and `Affected` coherent with the stated semantics of control vs diseased samples.

Future Work requires a positive Experiment

```
-- By assumption, only useful Future Works are created; enforce positivity of
↪  the source experiment
CREATE OR REPLACE TRIGGER trg_future_work_requires_positive_exp
BEFORE INSERT OR UPDATE OF exp_ref ON future_work_tab
FOR EACH ROW
DECLARE
  v_is_pos CHAR(1);
BEGIN
  SELECT e.is_positive INTO v_is_pos FROM experiment_tab e WHERE REF(e) =
  ↪  :NEW.exp_ref;
  IF UPPER(v_is_pos) <> 'Y' THEN
    RAISE_APPLICATION_ERROR(-20011,
    ↪  'FutureWork requires a positive Experiment');
  END IF;
END;
/
```

Uniqueness on REF pairs with compound triggers. REF columns do not support direct UNIQUE constraints; moreover, row-level checks would require selecting from the same table being modified, which leads to mutating-table issues during multi-row statements. We therefore

adopted compound triggers that collect the key pairs in the *BEFORE EACH ROW* phase and validate duplicates only *AFTER STATEMENT*.

Compound uniqueness on REF pairs (example: Affected)

```
-- Representative example: uniqueness (bio_ref, disease_ref) on AFFECTED via a
↪  compound trigger
CREATE OR REPLACE TRIGGER trg_aff_uni
FOR INSERT OR UPDATE OF bio_ref, disease_ref ON affected_tab
COMPOUND TRIGGER
  TYPE pair_rec IS RECORD (bio_id NUMBER, disease_id NUMBER);
  TYPE pair_tab IS TABLE OF pair_rec INDEX BY PLS_INTEGER;
  g_pairs pair_tab; g_idx PLS_INTEGER := 0;

  PROCEDURE add_pair(p_b REF biological_data_typ, p_d REF disease_typ) IS
    v_b NUMBER; v_d NUMBER;
  BEGIN
    SELECT b.id INTO v_b FROM biological_data_tab b WHERE REF(b) = p_b;
    SELECT d.id INTO v_d FROM disease_tab d       WHERE REF(d) = p_d;
    g_idx := g_idx + 1; g_pairs(g_idx).bio_id := v_b; g_pairs(g_idx).disease_id
    ↪  := v_d;
  END;

  BEFORE EACH ROW IS
  BEGIN
    IF INSERTING OR UPDATING THEN
      add_pair(:NEW.bio_ref, :NEW.disease_ref);
    END IF;
  END BEFORE EACH ROW;

  AFTER STATEMENT IS
    v_cnt NUMBER;
  BEGIN
    FOR i IN 1..g_idx LOOP
      SELECT COUNT(*) INTO v_cnt
      FROM affected_tab a
      WHERE DEREF(a.bio_ref).id = g_pairs(i).bio_id
        AND DEREF(a.disease_ref).id = g_pairs(i).disease_id;
      IF v_cnt > 1 THEN
        RAISE_APPLICATION_ERROR(-20030, 'Duplicate Affected (bio,disease)');
      END IF;
    END LOOP;
  END AFTER STATEMENT;
END;
/
```

The same pattern applies to the other association tables with REF pairs (`analyze_tab`, `assign_tab`, `cause_tab`, `writes_tab`, `consider_tab`); only the identifiers and the error message differ.

Analyze vs Disease consistency (BR14)

```
-- On Analyze insert/update, BD must be diseased and match Experiment.disease
CREATE OR REPLACE TRIGGER trg_analyze_consistency
BEFORE INSERT OR UPDATE OF bio_ref, exp_ref ON analyze_tab
FOR EACH ROW
DECLARE
```

```
  v_condition VARCHAR2(200);
  v_cnt NUMBER;
BEGIN
  SELECT b.condition INTO v_condition FROM biological_data_tab b WHERE REF(b) =
  ↪  :NEW.bio_ref;
  IF LOWER(v_condition) = 'control' THEN
    RAISE_APPLICATION_ERROR(-20005,
    ↪  'Control BiologicalData cannot be analyzed in experiments attempting a disease');
  END IF;

  SELECT COUNT(*) INTO v_cnt
  FROM affected_tab a
  WHERE a.bio_ref = :NEW.bio_ref
    AND a.disease_ref = (SELECT e.disease_ref FROM experiment_tab e WHERE REF(e)
    ↪  = :NEW.exp_ref);
  IF v_cnt = 0 THEN
    RAISE_APPLICATION_ERROR(-20006,
    ↪  'BiologicalData must be affected by the same disease attempted by the Experiment');
  END IF;
END;
/

-- Prevent removing Affected if Analyze rows rely on it
CREATE OR REPLACE TRIGGER trg_affected_block_if_analyze_exists
BEFORE DELETE ON affected_tab
FOR EACH ROW
DECLARE
  v_cnt NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_cnt
  FROM analyze_tab z
  WHERE z.bio_ref = :OLD.bio_ref
    AND EXISTS (
      SELECT 1 FROM experiment_tab e
      WHERE REF(e) = z.exp_ref
        AND e.disease_ref = :OLD.disease_ref
    );
  IF v_cnt > 0 THEN
    RAISE_APPLICATION_ERROR(-20007,
    ↪  'Cannot remove Affected: existing Analyze rows require this disease link');
  END IF;
END;
/
```

Together, these triggers ensure that each analyzed sample is affected by the exact disease attempted by the linked experiment and that the dependency is preserved.

—

In conclusion, the physical design translates the conceptual entities into object types and typed tables. Association entities are modeled as separate types and tables to correctly represent many-to-many relationships while avoiding redundancy. This design balances normalization, reuse of objects, and performance considerations, ensuring an efficient mapping between the logical schema and its physical implementation.

6.7 Operations

This subsection documents how the five main operations are implemented at the physical level. We provide one stored procedure per operation. Read-oriented procedures return a `SYS_REFCURSOR`

to stream results; the insert-oriented procedure commits on success following the course examples.

OP1 — Recording an organ or tissue

```
-- Insert a Biological Data (organ/tissue) for a given donor
CREATE OR REPLACE PROCEDURE proc_record_biological_data (
  p_id           IN NUMBER,
  p_name         IN VARCHAR2,
  p_condition    IN VARCHAR2,   -- 'control' | 'disease'
  p_is_required  IN CHAR,       -- 'Y' | 'N'
  p_description  IN CLOB,
  p_position     IN VARCHAR2,
  p_data_type    IN VARCHAR2,   -- 'organ' | 'tissue'
  p_density      IN NUMBER,
  p_donor_cf     IN CHAR
) AS
  v_donor_ref REF donor_typ;
BEGIN
  SELECT REF(d) INTO v_donor_ref FROM donors_tab d WHERE d.CF = p_donor_cf;
  INSERT INTO biological_data_tab
  VALUES (biological_data_typ(p_id, p_name, p_condition, p_is_required,
                              p_description, p_position, p_data_type,
                              p_density, v_donor_ref));
  COMMIT;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20020, 'Donor not found for provided CF');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20021, 'Error in proc_record_biological_data: ' ||
    ↪  SQLERRM);
END;
/
```

OP2 — Organs/tissues below a density threshold

```
-- Return: id, name, data_type, density, donor_cf, is_required, condition
CREATE OR REPLACE PROCEDURE proc_list_bio_below_density (
  p_threshold IN NUMBER,
  p_result    OUT SYS_REFCURSOR
) AS
BEGIN
  OPEN p_result FOR
    SELECT b.id,
           b.name,
           b.data_type,
           b.density,
           DEREF(b.donor_ref).CF AS donor_cf,
           b.is_required,
           b.condition
      FROM biological_data_tab b
     WHERE b.density < p_threshold;
EXCEPTION
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20022, 'Error in proc_list_bio_below_density: ' ||
    ↪  SQLERRM);
END;
/
```

OP3 — Treatment info with drugs and linked allergies

```
-- One row per (drug, allergy) combination, with treatment details
CREATE OR REPLACE PROCEDURE proc_get_treatment_info (
  p_treatment_id IN NUMBER,
  p_result       OUT SYS_REFCURSOR
) AS
BEGIN
  OPEN p_result FOR
    SELECT t.id                    AS treatment_id,
           t.name                  AS treatment_name,
           t.success_percentage,
           d.id                    AS drug_id,
           d.name                  AS drug_name,
           al.id                   AS allergy_id,
           al.name                 AS allergy_name
      FROM treatment_tab t
      LEFT JOIN assign_tab a
             ON a.treatment_ref = REF(t)
      LEFT JOIN drugs_tab d
             ON a.drug_ref = REF(d)
      LEFT JOIN cause_tab c
             ON c.drug_ref = a.drug_ref
      LEFT JOIN allergy_tab al
             ON c.allergy_ref = REF(al)
       WHERE t.id = p_treatment_id;
EXCEPTION
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20023, 'Error in proc_get_treatment_info: ' ||
    ↪  SQLERRM);
END;
/
```

OP4 — Donors with a given disease on life-required BD and useful FWs

```
-- Distinct donors (CF, name, surname) for a disease name, only if BD is
↪  required for life
-- and at least one Future Work exists via Analyze → Experiment
CREATE OR REPLACE PROCEDURE proc_list_donors_required_disease_with_fw (
  p_disease_name IN VARCHAR2,
  p_result       OUT SYS_REFCURSOR
) AS
BEGIN
  OPEN p_result FOR
    SELECT DISTINCT d.CF,
                    d.name,
                    d.surname
      FROM donors_tab d
      JOIN biological_data_tab b
        ON b.donor_ref = REF(d)
      JOIN affected_tab a
        ON a.bio_ref = REF(b)
      JOIN disease_tab dis
        ON a.disease_ref = REF(dis)
       AND LOWER(dis.name) = LOWER(p_disease_name)
     WHERE LOWER(b.condition) = 'disease'
       AND UPPER(b.is_required) = 'Y'
```

```
      AND EXISTS (
            SELECT 1
              FROM analyze_tab z
              JOIN future_work_tab f
                ON f.exp_ref = z.exp_ref
             WHERE z.bio_ref = REF(b)
           );
EXCEPTION
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20024,
    ↪  'Error in proc_list_donors_required_disease_with_fw: ' || SQLERRM);
END;
/
```

OP5 — Useful suggestions sent to researchers with top-quality pubs

```
-- Return: future work id, title, and researcher identity
CREATE OR REPLACE PROCEDURE proc_list_fw_for_top_researchers (
  p_result OUT SYS_REFCURSOR
) AS
BEGIN
  OPEN p_result FOR
    SELECT
      f.id      AS future_work_id,
      f.title   AS future_work_title,
      r.CF      AS researcher_cf,
      r.name    AS researcher_name,
      r.surname AS researcher_surname
    FROM consider_tab c
    JOIN future_work_tab f
      ON c.future_work_ref = REF(f)
    JOIN (
        SELECT DISTINCT w.researcher_ref
        FROM writes_tab w
        JOIN publication_tab p
          ON w.publication_ref = REF(p)
        WHERE p.quality = 'top'
    ) top_researchers
      ON c.researcher_ref = top_researchers.researcher_ref
    JOIN researchers_tab r
      ON c.researcher_ref = REF(r);
EXCEPTION
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20025,
    ↪  'Error in proc_list_fw_for_top_researchers: ' || SQLERRM);
END;
/
```

6.8 Indexes

Qualitative assessment. In this paragraph we will choose specific indexes based on the type of operation to be performed and the way Oracle accesses data.

OP2

The operation retrieves all biological data (organs / tissues) whose density is below a threshold. Since the attribute is numerical and the operation involves a range predicate, a B+Tree index is

the most appropriate choice.

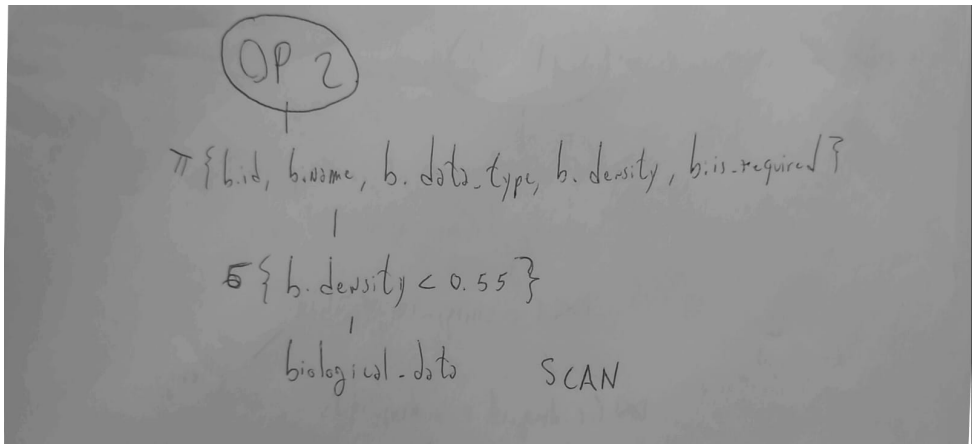

Figure 4: Algebraic tree of OP2

Initial analysis (no index). Autotrace shows a full table scan on `BIOLOGICAL_DATA_TAB`, which is expected because the optimizer has no auxiliary structure to exploit. This implies a cost that grows linearly with the table size.

We then created a B+Tree index:

```c
REATE INDEX idx_bd_density ON biological_data_tab(density);
```

After adding the index. The access path changes to an INDEX RANGE SCAN with Table Access by ROWID.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
SELECT STATEMENT				37	38	127
TABLE ACCESS	BIOLOGICAL_DATA_TAB	BY INDEX ROWID BATCHED	35	37	38	127
INDEX	IDX_BD_DENSITY	RANGE SCAN	35	2	2	17
Access Predicates						
B.DENSITY<.55						

Figure 5: Autotrace of OP2 after adding the index

OP3

The operation retrieves treatment details and associated drugs/allergies. Logically it follows a chain of joins starting from the treatment primary key. Since the entry point is already indexed by the PK, the challenge is optimizing the association tables (`ASSIGN_TAB` and `CAUSE_TAB`).

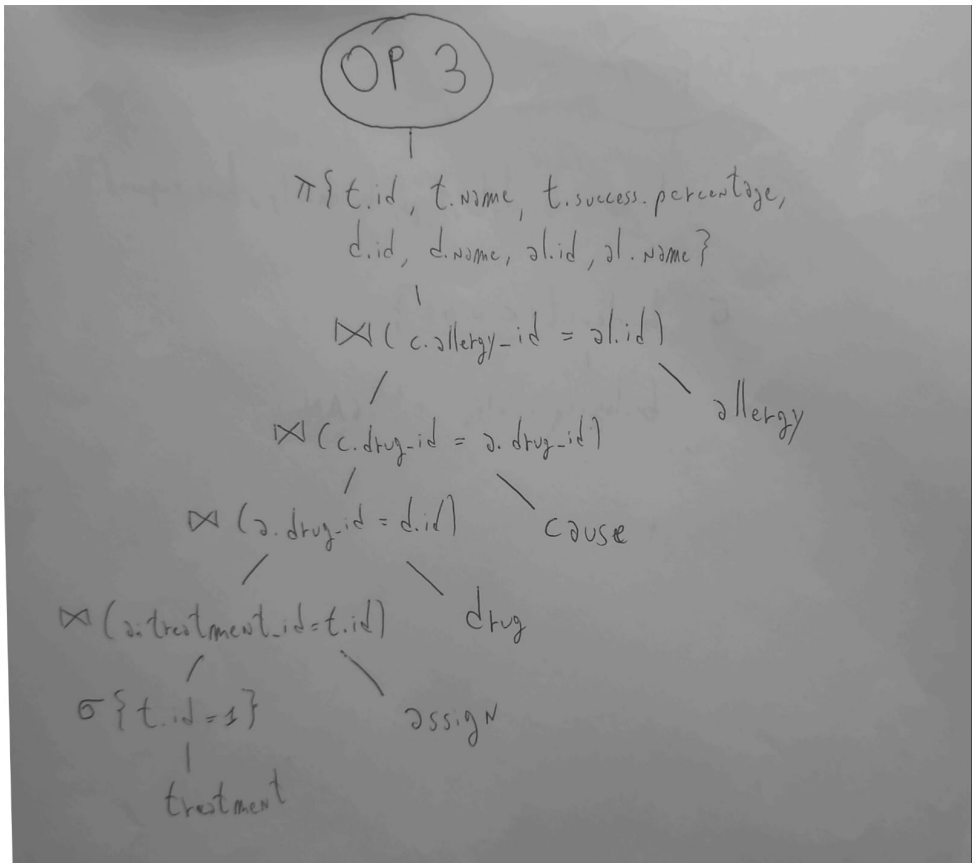

Figure 6: Algebraic tree of OP3

Initial analysis (no auxiliary indexes). Autotrace shows full scans on the association tables. With the current small dataset, this is not too expensive, but it will not scale when the tables grow.

After analyzing the query execution plan, we create B+Tree indexes only on the REF columns that are actually utilized by Oracle's optimizer:

C

```
REATE INDEX idx_assign_treatment_ref ON assign_tab(treatment_ref);
CREATE INDEX idx_cause_drug_ref       ON cause_tab(drug_ref);
```

Rationale for index selection. These two indexes are sufficient because:

- `idx_assign_treatment_ref` — enables efficient access when joining `treatment_tab` to `assign_tab`
- `idx_cause_drug_ref` — enables efficient access when joining from `assign_tab` to `cause_tab`

Additional indexes on `assign_tab(drug_ref)` or `cause_tab(allergy_ref)` would be redundant since:

- The join from `assign_tab` to `drugs_tab` is efficiently served by the primary key index on `drugs_tab`
- The join from `cause_tab` to `allergy_tab` benefits from the primary key index on `allergy_tab`

This minimal indexing strategy reduces storage overhead and maintenance cost while providing optimal query performance.

SELECT STATEMENT				7	12	8529
NESTED LOOPS		OUTER	1	7	12	8529
NESTED LOOPS		OUTER	1	6	10	6279
NESTED LOOPS		OUTER	1	5	8	6206
NESTED LOOPS		OUTER	1	4	6	5207
TABLE ACCESS	TREATMENT_TAB	BY INDEX ROWID	1	2	3	2891
Filter Predicates						
AND						
TRUNC(SUCCESS_PERCENTAGE)>=0						
TRUNC(SUCCESS_PERCENTAGE)<=100						
INDEX	SYS_C009187	UNIQUE SCAN	1	1	2	2036
Access Predicates						
T.ID=6						
TABLE ACCESS	ASSIGN_TAB	BY INDEX ROWID BATCHED	1	2	3	2309
INDEX	IDX_ASSIGN_TREATMENT_REF	RANGE SCAN	1	1	2	2276
Access Predicates						
SYS_NC00006$=ID						
TABLE ACCESS	DRUGS_TAB	BY INDEX ROWID	1	1	2	995
INDEX	SYS_C009182	UNIQUE SCAN	1	0	1	704
Access Predicates						
SYS_NC00007$=ID						
TABLE ACCESS	CAUSE_TAB	BY INDEX ROWID BATCHED	1	1	2	70
INDEX	IDX_CAUSE_DRUG_REF	RANGE SCAN	1	0	1	51
Access Predicates						

Figure 7: Autotrace of OP3 after adding the indexes

OP4

The query selects donors affected by a given disease and linked (through experiments) to future work entries. It therefore traverses multiple association tables.

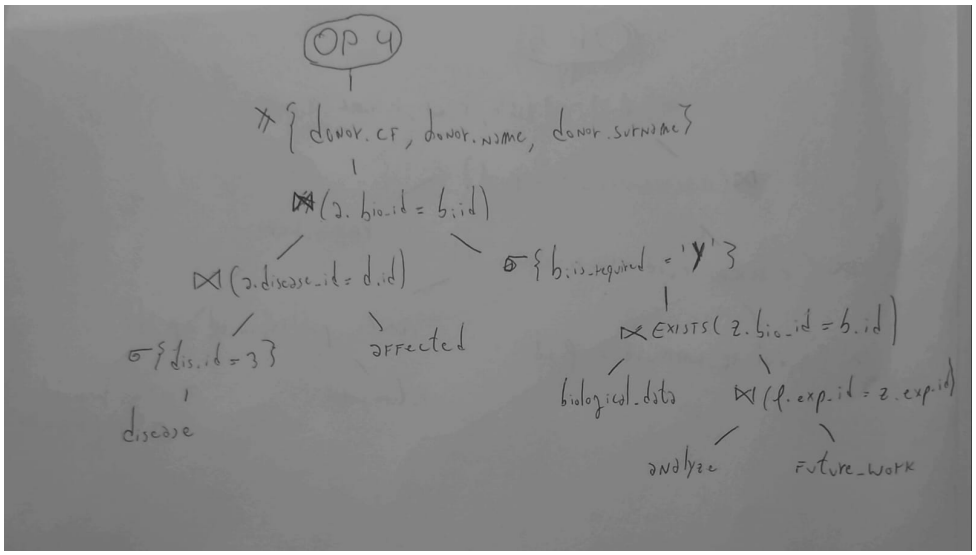

Figure 8: Algebraic tree of OP4

Initial analysis (no index). Autotrace showed several FULL SCANs (on `AFFECTED_TAB`, `ANALYZE_TAB`, `FUTURE_WORK_TAB`). This leads to a high number of logical reads, proportional to the size of these tables.

To optimize, we created narrow B+Tree indexes targeting the selective joins:

C

```
REATE INDEX idx_aff_dis_bio ON affected_tab(disease_ref, bio_ref);
CREATE INDEX idx_analyze_bio ON analyze_tab(bio_ref);
CREATE INDEX idx_fw_exp      ON future_work_tab(exp_ref);
```

Rationale for index selection:

- `idx_aff_dis_bio` — composite index enabling efficient lookup by disease and biological data
- `idx_analyze_bio` — critical for the EXISTS subquery join condition `z.bio_ref = REF(b)`, eliminates FULL SCAN on `analyze_tab` when dataset is sufficiently large

- `idx_fw_exp` — used in the EXISTS subquery to find future works linked to experiments

After adding the indexes. Oracle now exploits index-driven nested loops. The FULL SCAN on `analyze_tab` is eliminated, replaced by index range scans. Logical gets are reduced significantly, and the plan shows efficient index access on all join columns.

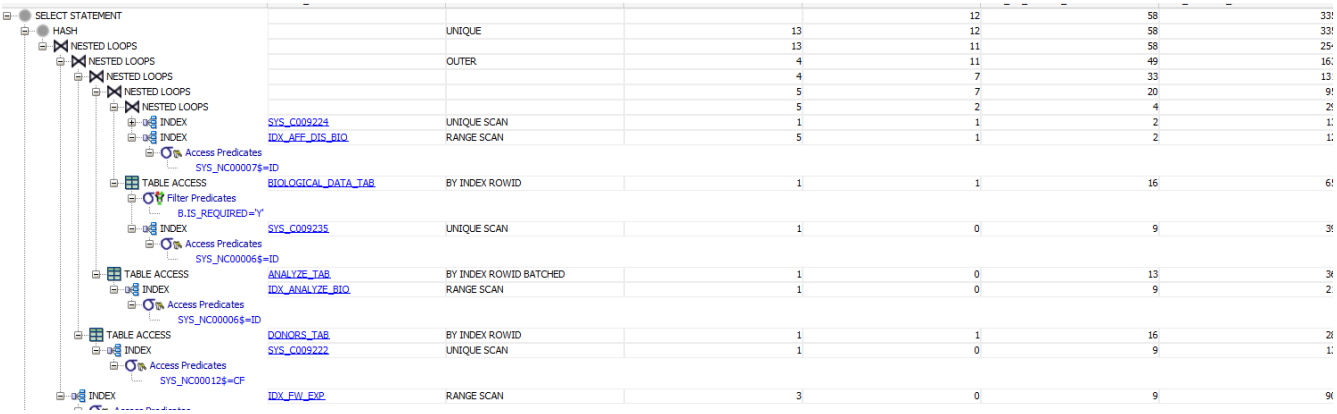

Figure 9: Autotrace of OP4 after adding the indexes

OP5

This operation retrieves all `Future Work` entries considered by researchers with at least one high-quality publication. The query filters on `PUBLICATION_TAB` and then joins `WRITES_TAB` and `CONSIDER_TAB`.

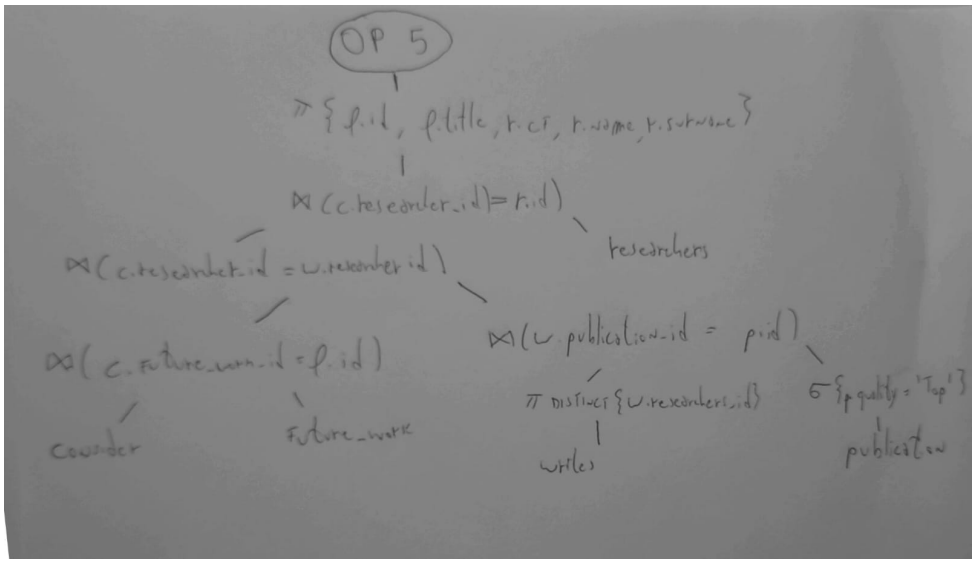

Figure 10: Algebraic tree of OP5

Initial analysis (no index). Autotrace showed full scans on the publication and association tables, combined with hash joins. While functional, this approach scales poorly because each join requires scanning entire tables, regardless of selectivity.

We created B+Tree indexes on the most selective columns to optimize join operations:

C

```
REATE INDEX idx_pub_quality         ON publication_tab(quality);
CREATE INDEX idx_writes_publication_ref  ON writes_tab(publication_ref);
```

```
CREATE INDEX idx_consider_researcher_ref ON consider_tab(researcher_ref);
```

Rationale for index selection:

- `idx_pub_quality` — enables efficient filtering on `quality = 'top'` to find high-quality publications, reducing the workload during the initial scan
- `idx_writes_publication_ref` — used for the join from `publication_tab` to `writes_tab`, avoiding a full table scan on `writes_tab`
- `idx_consider_researcher_ref` — critical for joining the researcher subquery with `consider_tab`, significantly reducing the scan cost on this association table

After adding the indexes. Autotrace confirms that Oracle correctly uses all three indexes:

- `idx_pub_quality` performs an INDEX RANGE SCAN on `publication_tab`, filtering only the top-quality publications
- `idx_writes_publication_ref` performs an INDEX RANGE SCAN on `writes_tab`, efficiently joining publications to their authors
- `idx_consider_researcher_ref` performs an INDEX RANGE SCAN on `consider_tab`, efficiently joining researchers to their assigned future works

These indexes significantly reduce the execution cost by eliminating full table scans on the association tables and enabling selective access to the publication table. The cost-based optimizer leverages these indexes to construct an efficient execution plan with nested loop joins instead of expensive hash joins on full table scans. Note that `researchers_tab` and `future_work_tab` still use FULL TABLE SCAN.

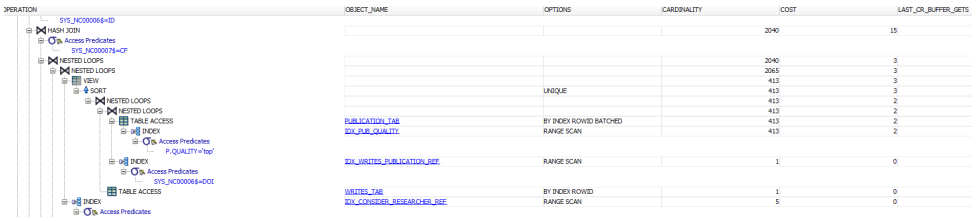

Figure 11: Autotrace of OP5 after adding the indexes

General note on index choice

Hash indexes (or hash clusters) can be advantageous only for *equality* queries on well-distributed, relatively static data. In our context this is not the case: most attributes are queried with ranges or through joins, the data distribution is not uniform, and the system undergoes constant updates (insertion of new experiments, publications, future works, etc.). These factors would increase the cost of hash structures and risk collisions. Moreover, since the operations are executed only once a month, it is not justified to create specialized auxiliary structures with limited usage. For these reasons, B+Tree indexes remain the most appropriate choice, as they provide both flexibility and stability across different query patterns.

7 Web Application

The web application provides a user-friendly interface to interact with the WHO Disease Monitoring System. It allows users to visualize stored data, insert new records, and execute the 5 analytical operations previously described.

7.1 Technology Stack and Architecture

The application is built using:

- **Flask**: Python 3.11 web framework for routing and request handling
- **oracledb**: Oracle database driver
- **Docker**: Containerization for portability and ease of deployment

7.2 Database Interaction

Connection Management The application establishes connections to Oracle Database using the `oracledb` library:

python

```python
def get_db_connection():
    connection = oracledb.connect(
        user=Config.DB_USER,
        password=Config.DB_PASSWORD,
        dsn=Config.get_dsn()
    )
    return connection
```

Data Visualization and Insertion The webapp provides interfaces to:

- **List** all records from each entity table (Biological Data, Donor, Disease, Treatment, Drug, Allergy, Experiment, Future Work, Publication, Researcher)
- **Insert** new records with proper form validation and business rule enforcement
- Manage association tables (Affected, Analyze, Assign, Cause, Consider, Write) with dynamic filtering to prevent constraint violations

For example, the `Analyze` form implements client-side JavaScript filtering to ensure BR14 compliance: when a user selects a biological data sample, the experiment dropdown automatically shows only experiments attempting to treat the same disease affecting that sample.

Analytical Operations with Stored Procedures To maintain business logic entirely within the database layer, all 5 analytical operations are implemented as Oracle **stored procedures**. This approach ensures that complex queries and data processing logic remain in the database, while the application layer handles only presentation and user interaction.

For each operation, custom Oracle object types define the return structure. For example, Operation 2 uses:

sql

```sql
CREATE TYPE op2_result_typ AS OBJECT (
    bio_id NUMBER,
    bio_name VARCHAR2(100),
    density NUMBER,
    position VARCHAR2(200),
    donor_name VARCHAR2(50),
    donor_surname VARCHAR2(50)
);
```

```
CREATE TYPE op2_result_tab AS TABLE OF op2_result_typ;
```

The stored procedure encapsulates the entire query logic:

sql

```
CREATE OR REPLACE FUNCTION func_list_bio_below_density(
    p_threshold NUMBER
) RETURN op2_result_tab PIPELINED IS
BEGIN
    FOR rec IN (
        SELECT b.id, b.name, b.density, b.position,
               DEREF(b.donor).name AS donor_name,
               DEREF(b.donor).surname AS donor_surname
        FROM biological_data_tab b
        WHERE b.density < p_threshold
        ORDER BY b.density
    ) LOOP
        PIPE ROW(op2_result_typ(rec.id, rec.name,
                 rec.density, rec.position,
                 rec.donor_name, rec.donor_surname));
    END LOOP;
    RETURN;
END;
```

The Flask application invokes the stored procedure using standard SQL syntax:

python

```
@app.route('/operations/2', methods=['GET', 'POST'])
def operation_2():
    if request.method == 'POST':
        threshold = float(request.form.get('threshold'))
        cursor.execute("""
            SELECT * FROM TABLE(
                func_list_bio_below_density(:threshold)
            )
        """, {'threshold': threshold})
        results = cursor.fetchall()
        return render_template('operation_2.html',
                               results=results)
    return render_template('operation_2.html')
```

This pattern is replicated for all operations:

- **Operation 2**: `func_list_bio_below_density(threshold)` — Lists biological data below a specified density threshold
- **Operation 3**: `func_get_treatment_info(treatment_id)` — Retrieves treatment details with associated drugs and allergies
- **Operation 4**: `func_list_donors_required_disease_with_fw(disease_id)` — Lists donors with a disease affecting vital organs for which useful future works were suggested
- **Operation 5**: `func_list_fw_for_top_researchers()` — Lists useful future works assigned to researchers with top-quality publications

This architecture ensures:

- All business logic resides in the database layer through procedures
- The application layer remains thin, handling only presentation and routing
- Query optimization and execution plans are managed by Oracle's query optimizer

The `.\stop.ps1` script gracefully shuts down all services and removes the containers.

7.3 User Interface Examples

The following figures illustrate key pages of the web application.

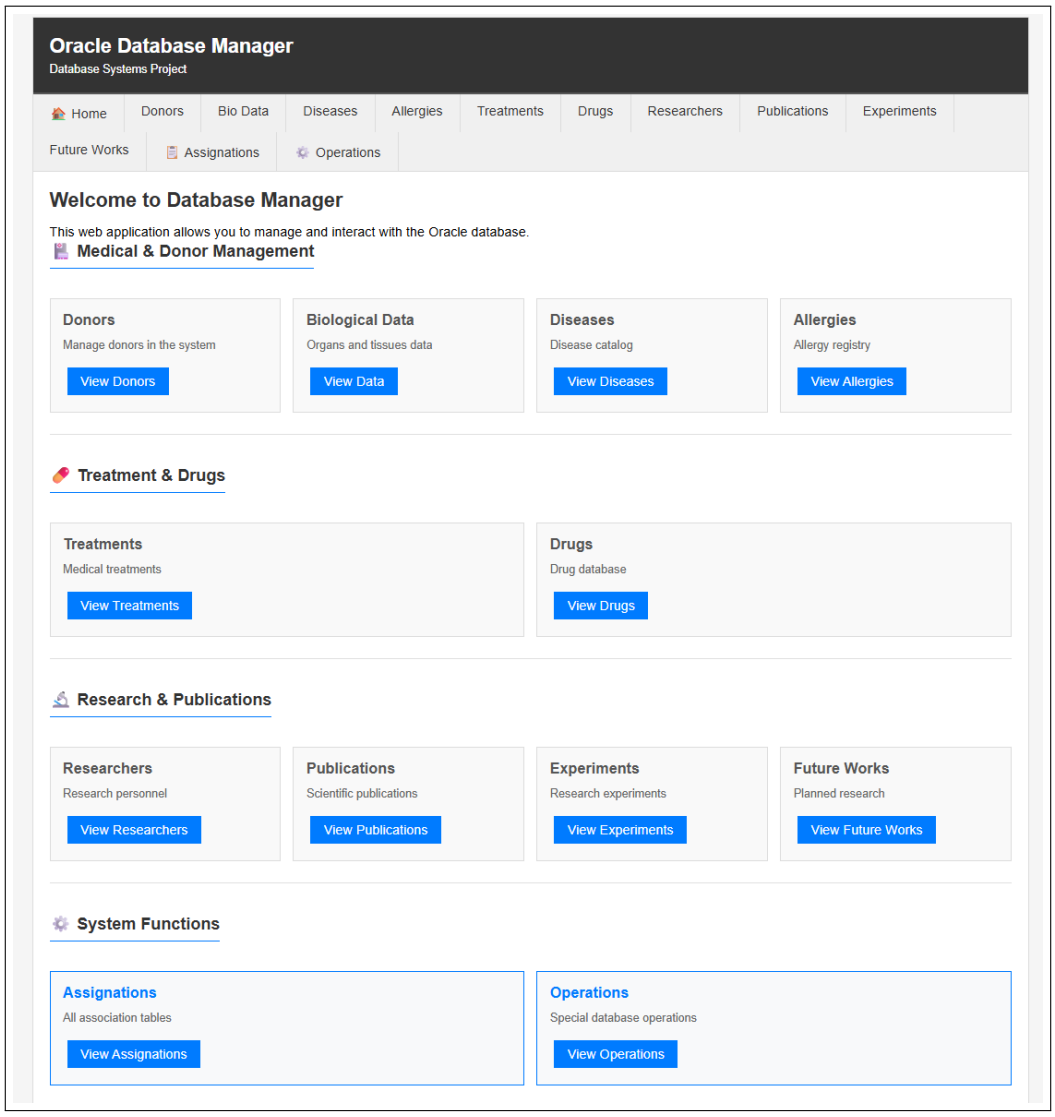

Figure 12: Homepage with navigation menu to all entities and operations

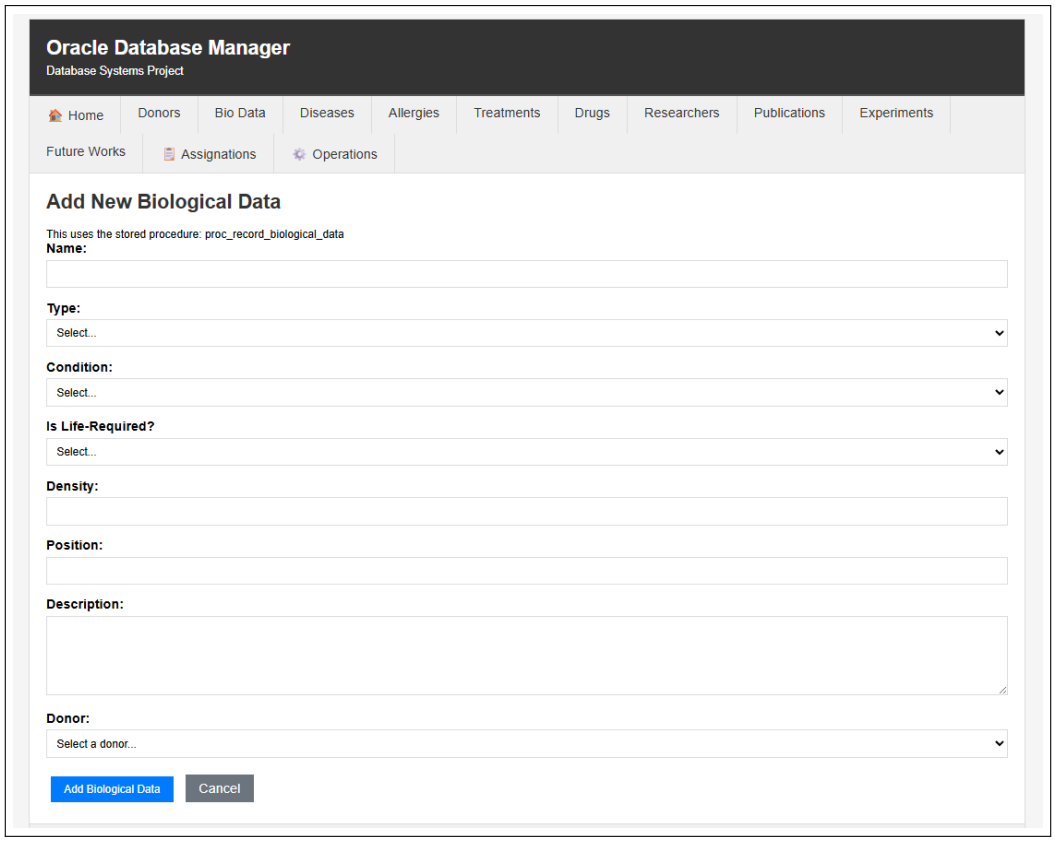

Figure 13: Form for adding new biological data (organ or tissue)

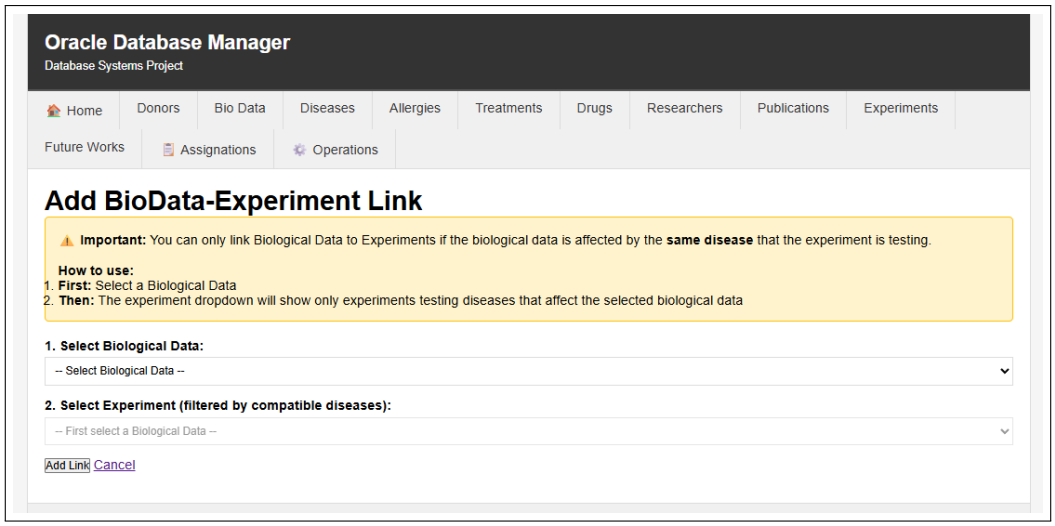

Figure 14: Dynamic filtering in Analyze form: experiment dropdown filtered by biological data disease

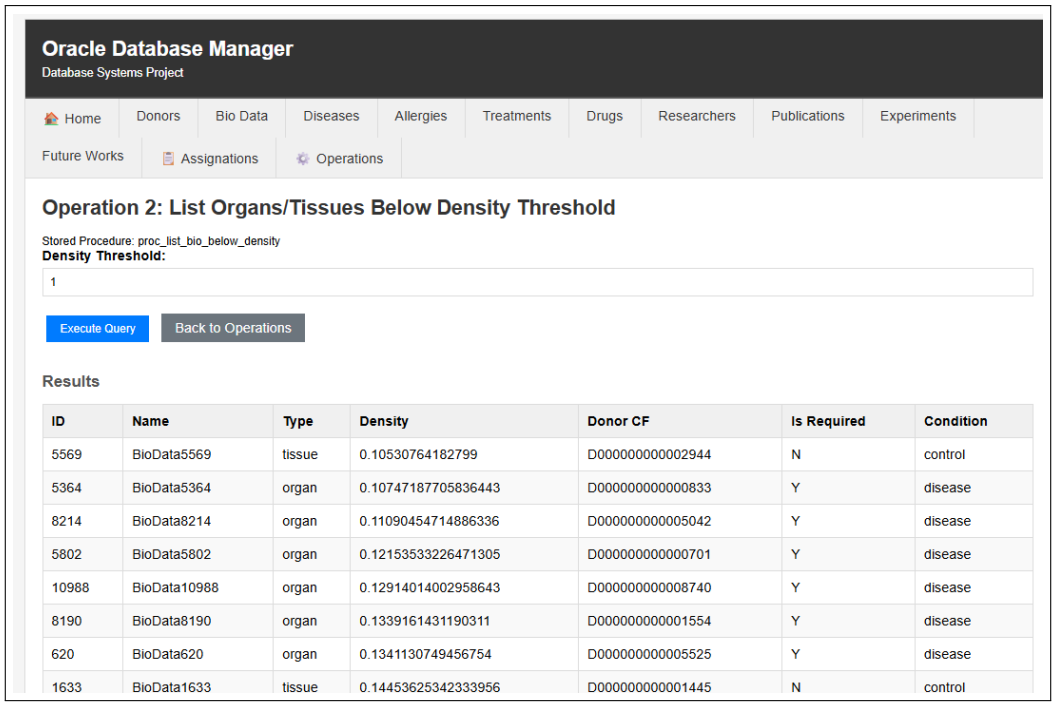

Figure 15: Operation 2: List biological data below density threshold

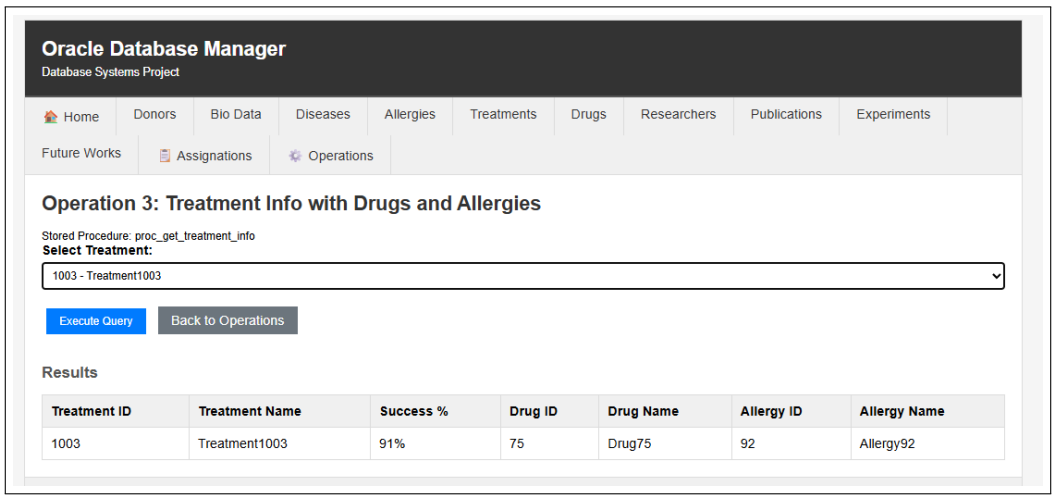

Figure 16: Operation 3: Treatment details with associated drugs and allergies

The web application successfully demonstrates the practical applicability of the designed database system, providing an accessible interface for WHO personnel to manage disease monitoring data effectively.

8 Data Warehouse Design

To support strategic decision-making and analytical processing, we propose a Data Warehouse design for the WHO Disease Monitoring System. This Data Warehouse will enable multidimensional analysis of key performance indicators, facilitating long-term trend analysis and evidence-based research planning.

8.1 Fact and Dimension Analysis

The analysis of facts, measures, and dimensions led to the definition of two key facts: **Disease Research Progress** and **Scientific Publication Output**. The need to analyze disease research progress arises from the goal of evaluating treatment effectiveness and predicting which diseases are closest to finding cures, enabling the WHO to allocate resources efficiently. Scientific publication output, on the other hand, will be analyzed to evaluate research productivity and ensure the organization maintains its reputation for high-quality scientific contributions.

8.1.1 Fact 1: Disease Research Progress

For Disease Research Progress, the dimensions considered will be Time, Disease, Treatment, Biological Data, and Area. The reasons for selecting these dimensions are:

- **Time**: Helps identify research trends and predict future progress based on historical patterns of experiment success rates.
- **Disease**: Allows understanding which diseases are showing research progress and which require additional investment, enabling targeted resource allocation.
- **Treatment**: Identifies which treatments are most effective across different diseases, optimizing drug development strategies.
- **Biological Data**: Provides insights into whether organs or tissues, control or disease samples yield better experimental results, improving sample selection for future experiments.
- **Area**: Enables geographical analysis to study how geographic location impacts disease prevalence, treatment effectiveness, and research outcomes across different regions, continents, and countries.

Measures: numExperiments, successRate.

8.1.2 Fact 2: Scientific Publication Output

For Scientific Publication Output, the dimensions considered will be Time, Researcher, Publication, and Area.

- **Time**: Tracks publication trends over time, helping to identify patterns in research productivity and assess output consistency.
- **Researcher**: Allows us to focus on researchers with lower publication rates or quality scores, identifying causes of underperformance and implementing targeted support programs.
- **Publication**: Helps understand publication quality distribution and publisher preferences, enabling the organization to improve overall publication standards and scientific impact.
- **Area**: Provides geographical insights into research productivity, allowing analysis of which regions produce more high-quality publications and identifying areas requiring additional research support.

Measures: numPublications, avgQualityScore.

8.2 Schema Architecture

The Data Warehouse will be implemented using a **Constellation Schema**. Specifically, this architecture is chosen because multiple dimensions are shared across different facts:

- **Time dimension**: Shared by both facts to enable cross-fact temporal analysis and unified reporting across disease research and publication trends.
- **Area dimension**: Shared by both facts to enable geographical analysis of research impact, disease distribution, treatment effectiveness, and publication productivity across different regions, continents, and countries.

The two fact tables are:

- **ResearchProgress**: Links Time, Disease, Treatment, BiologicalData, and Area dimensions. Measures include numExperiments and successRate. Primary key: Time_FK, Disease_FK, Treatment_FK, BiologicalData_FK, Area_FK.
- **PublicationOutput**: Links Time, Researcher, Publication, and Area dimensions. Measures include numPublications and avgQualityScore. Primary key: Time_FK, Researcher_FK, Publication_FK, Area_FK.

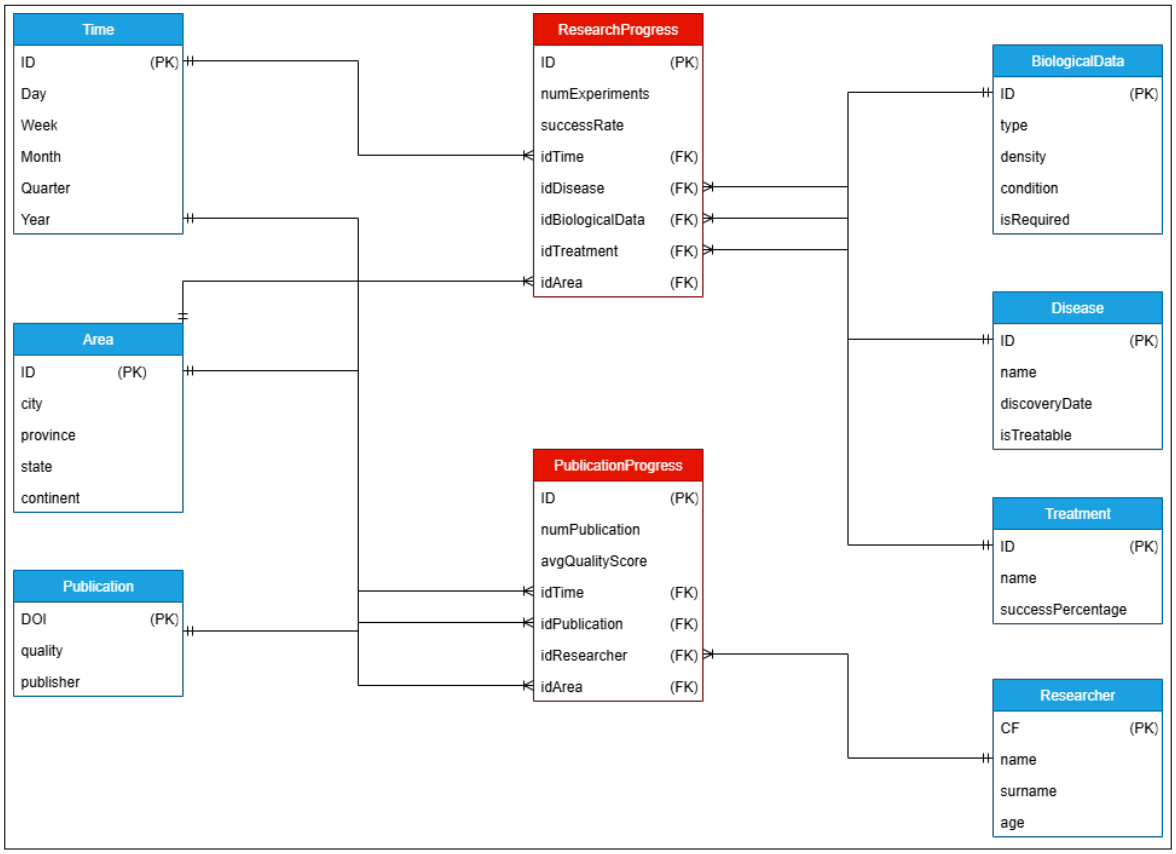

Figure 17: Constellation Schema for WHO Disease Monitoring Data Warehouse

8.3 Summary

The proposed Data Warehouse design provides the WHO with a powerful analytical platform for strategic decision-making. The two facts, Disease Research Progress and Scientific Publication Output, address the organization's core objectives of advancing medical research and maintaining

scientific excellence. The constellation schema architecture ensures efficient data organization while enabling flexible multidimensional analysis across shared dimensions.