Факультет программной инженерии и компьютерной техники

Информационные системы и базы данных

Лабораторная работа №2

Преподаватель: Николаев Владимир Вячеславович
Выполнил: Ле Чонг Дат
Группа: P33302

2023 г.

1 Задание

Для отношений, полученных при построении предметной области из лабораторной работы #1, выполните следующие действия:

- опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
- приведите отношения в 3NF (как минимум). Постройте схему на основе полученных отношений;
- опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе 3NF;
- преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF;
- какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

2 Инфологическая модель

В этом разделе представлена инфологическая модель проекта.

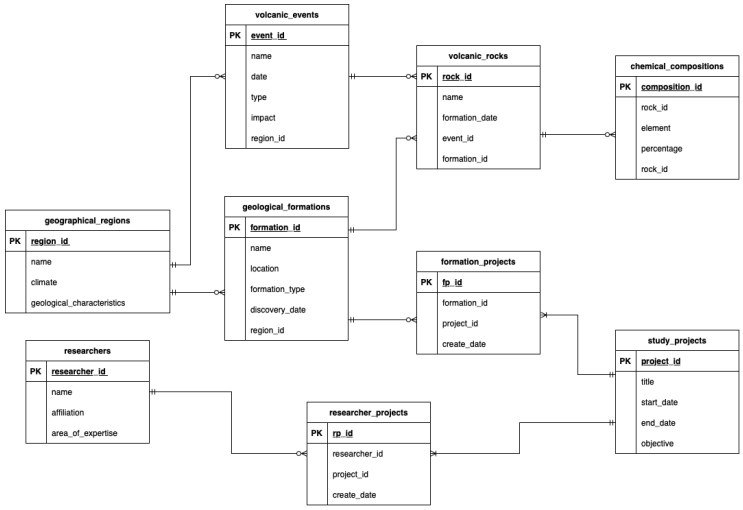

Рис. 1: Инфологическая модель

3 Даталогическая модель

В этом разделе представлена даталогическая модель.

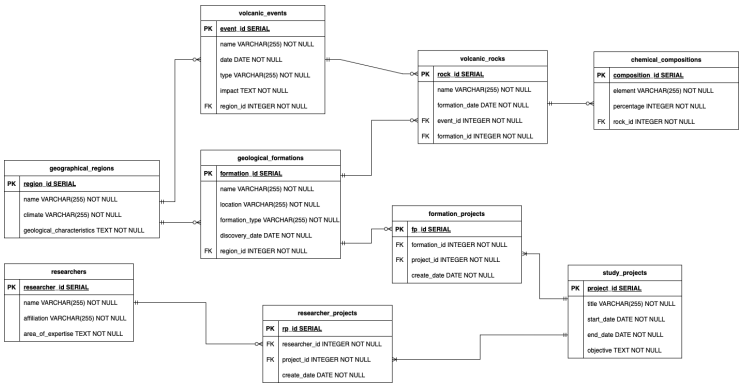

Рис. 2: Даталогическая модель

4 Обработка задач

Task 1: Functional Dependencies

For each table in The schema from Laboratory Work #1, we identify the following functional dependencies:

- **geographical_regions:**
  - region_id → name, climate, geological_characteristics
- **volcanic_events:**
  - event_id → name, date, type, impact, region_id
- **geological_formations:**
  - formation_id → name, location, formation_type, discovery_date, region_id
- **volcanic_rocks:**
  - rock_id → name, formation_date, event_id, formation_id
- **chemical_compositions:**
  - composition_id → rock_id, element, percentage
- **researchers:**
  - researcher_id → name, affiliation, area_of_expertise
- **study_projects:**
  - project_id → title, start_date, end_date, objective
- **formation_projects:**
  - fp_id → formation_id, project_id, create_date
  - formation_id, project_id → create_date

- **researcher_projects:**
  - rp_id → researcher_id, project_id, create_date
  - researcher_id, project_id → create_date

Task 2: Normalization to 1NF, 2NF, and 3NF

1NF (First Normal Form)

Characteristics of 1NF:

- The values in each column of a table must be atomic.
- Each column must have a unique name.
- The order in which data is stored does not matter.

Proof of 1NF Compliance:

- All values in The database schema are atomic.
- Each column in The tables has a unique name.
- The order of rows and columns is irrelevant to the database's functionality.

2NF (Second Normal Form)

Characteristics of 2NF:

- The table is in 1NF.
- There are no partial dependencies of any column on the primary key.

Proof of 2NF Compliance:

- Tables with a single-column primary key inherently meet 2NF.
- Association tables with composite primary keys do not have partial dependencies, as each key component is needed to uniquely identify records.

3NF (Third Normal Form)

Characteristics of 3NF:

- The table is in 2NF.
- There are no transitive dependencies for non-prime attributes on the primary key.

Proof of 3NF Compliance:

- Non-key attributes in each table are functionally dependent only on the primary key.
- There are no transitive dependencies in any of the tables.

The database schema from Laboratory Work #1 meets the requirements of 1NF, 2NF, and 3NF, as it has properly structured attributes with no partial or transitive dependencies violating these normalization rules.

Task 3: Changes in Functional Dependencies after Transforming to 3NF

As discussed in the analysis for Task 2, our schema already adheres to 3NF since it does not have any transitive dependencies. Each non-key attribute in our schema is functionally dependent only on the primary key of its respective table, and there are no cases where a non-key attribute depends on another non-key attribute. This indicates that our original schema did not undergo significant changes in terms of functional dependencies while ensuring 3NF compliance.

Task 4: Transformation to BCNF

Characteristics of BCNF:

- A relation is in 3NF.
- For every non-trivial functional dependency $X \rightarrow Y$, X is a superkey.

Proof of BCNF Compliance: Our schema is analyzed for each table to ensure it meets BCNF criteria:

- **geographical_regions, volcanic_events, geological_formations, volcanic_rocks, chemical_compositions, researchers, study_projects:** In each of these tables, the primary key is the only determinant, and it is a superkey. Therefore, these tables are in BCNF.
- **formation_projects and researcher_projects:** These association tables have composite keys that are the only determinants and are superkeys. Hence, these tables are also in BCNF.

In conclusion, all tables in our schema meet the criteria for BCNF. There are no partial dependencies or transitive dependencies, and every determinant in the functional dependencies is a superkey, fulfilling the requirements for BCNF.

Task 5: Denormalization Considerations

Potential Denormalizations:

- **Denormalization of 'geological_formations' and 'geographical_regions':**
  - Combine 'geological_formations' and 'geographical_regions' into a single table.
  - Justification: If geological formations are typically queried along with their geographical region data, combining these tables can reduce the need for JOIN operations, improving query performance.
- **Denormalization of 'volcanic_events' and 'volcanic_rocks':**
  - Add key attributes of 'volcanic_events' (like 'name', 'type') to the 'volcanic_rocks' table.
  - Justification: If there is frequent access to volcanic rock data along with associated volcanic event details, this denormalization can optimize query performance by avoiding JOINs.

- **Adding Redundant Data to 'researcher_projects':**
  - Include a summary or key details of the 'study_projects' directly in the 'researcher_projects' table.
  - Justification: For use cases where project details are often required when querying researcher projects, this can speed up data retrieval.
- **Precomputed Aggregates:**
  - For tables like 'chemical_compositions', consider adding precomputed aggregate information, such as the average percentage of elements.
  - Justification: Useful for analytical queries where aggregate information is frequently needed. This avoids the overhead of calculating aggregates on the fly.

Considerations for Denormalization:

- **Performance vs. Data Integrity:** Denormalization can improve query performance but at the cost of data integrity and increased data redundancy. It can lead to anomalies in INSERT, UPDATE, and DELETE operations.
- **Storage Costs:** More storage might be required due to redundant data.
- **Maintenance:** Denormalized data requires careful maintenance to ensure consistency.

In summary, the decision to denormalize should be based on specific use cases, query performance requirements, and the acceptable trade-off between data integrity and performance. Each potential denormalization listed above offers performance benefits for certain types of queries but should be implemented with caution, keeping in mind the potential drawbacks.