Group-3 Freelancing Mediator

Normalization and Schema Refinement

1. Relations & Schemas:

- *freelancer* (<u>freelancer_id</u>, first_name, middle_name, last_name, date_of_birth, avg_rating, city, country, phone_no, skills,email_id, password)

- *client* (<u>client_id</u>, organization_name, first_name, middle_name, last_name, date_of_birth, city, country, phone_no, email_id, password)

- *project* (<u>project_id</u>, amount, start_date, deadline, project_name, skills,status, project_description)

- *contract* (<u>contract_id</u>, freelancer_id, client_id, project_id, date, time)

- *transaction* (<u>transaction_id</u>, contract_id, amount, date, time, payment_method)

- *reviews* (<u>review_id</u>, contract_id, rating, review)

- *takes* (<u>freelancer_id</u>, <u>project_id</u>)

- *provides* (<u>project_id, </u>client_id)

2. List of update, delete, and insert anomalies for every schema:

- Freelancer table there is no insertion anomalies but for the deletion and the updation if we delete the tuple and its one entry is in the contract or project table then it will create issue that there isn't any freelancer but it had done project or contract. Means where ever table's attribute refer to freelancer tables attribute then it will create deletion and updation anomalies.

- Client table has same update and deletion anomalies having same reason as freelancer.

- Project table also has updation and deletion anomalies because if the delete or update the tuple in the table and same deleted or updated attribute is in the contract or transaction or review table then it will cause issue so it has these type of anomalies.

- Contract,Transaction and Review table has insertion and updation anomalies because if we insert/update the project id in the transaction/review table then it must be in the main project table, same logic for the insert/update of freelancer id and client id will cause insertion/updation anomalies.

- Takes and provide table have the insertion and updation anomalies since if the tuple that is going to be inserted or updated has a new value of project id or client id or freelancer id then if it isn't exist in freelancer or client or project table than it will create inconsistency in the data so it has insertion and updation anomalies.

3. Dependencies:

- *freelancer:*
  - Primary Key: <u>freelancer_id</u>
  - Candidate Key: {freelancer_id}, {email_id}
  - Foreign Key:
  - Functional dependencies:
    - freelancer_id → first_name
    - freelancer_id → middle_name
    - freelancer_id → last_name
    - freelancer_id → date_of_birth
    - freelancer_id → avg_rating
    - freelancer_id → city
    - freelancer_id → email_id
    - freelancer_id → password
    - email_id → freelancer_id
- *client*:
  - Primary Key: <u>client_id</u>
  - Candidate Key: {client_id} , {email_id}
  - Foreign Key:
  - Functional dependencies:
    - client_id → organization_name
    - client_id → first_name
    - client_id → middle_name
    - client_id → last_name
    - client_id → date_of_birth
    - client_id → city
    - client_id → email_id
    - client_id → password
    - email_id → Client_id
- *project*:
  - Primary Key: <u>project_id</u>
  - Candidate Key: {project_id} , {client_id , project_name}

- Foreign Key:
- Functional dependencies:
    - project_id → amount
    - project_id → start_date
    - project_id → client_id
    - project_id → deadline
    - project_id → project_name
    - project_id → status
    - project_id → project_description
    - client_id , project_name → project_id

- *contract*:
    - Primary Key: <u>contract_id</u>
    - Candidate Key: contract_id
    - Foreign Key:
        - transaction.freelancer_id references to freelancer.freelancer_id
        - transaction.client_id references to client.client_id
        - transaction.project_id references to project.project_id
    - Functional dependencies:
        - contract_id → time
        - contract_id → date
- *transaction*:
    - Primary Key: <u>transaction_id</u>
    - Candidate Key: {transaction_id} , {freelancer_id , client_id , project_id}
    - Foreign Key:
        - transaction.freelancer_id references to freelancer.freelancer_id
        - transaction.client_id references to client.client_id
        - transaction.project_id references to project.project_id
    - Functional dependencies:
        - transaction_id → freelancer_id
        - transaction_id → client_id
        - transaction_id → project_id
        - transaction_id → amount
        - transaction_id → time
        - transaction_id → date
        - transaction_id → method
        - project_id → transaction_id
- *reviews*:
    - Primary Key: <u>review_id</u>
    - Candidate key: {review_id} , {freelancer_id , client_id , project_id}
    - Foreign Key:
        - transaction.freelancer_id references to freelancer.freelancer_id
        - transaction.client_id references to client.client_id
        - transaction.project_id references to project.project_id
    - Functional dependencies:

        - review_id → freelancer_id
        - review_id → client_id
        - review_id → project_id
        - review_id → rating
        - review_id → review
        - project_id → freelancer_id
- *takes*:
    - Primary Key: <u>freelancer_id</u>, <u>project_id</u>
    - Candidate Key: {freelancer_id, project_id}
    - Foreign Key:
        - takes.freelancer_id references to freelancer.freelancer_id
        - takes.project_id references to project.project_id
    - Functional dependencies:
- *provides*:
    - Primary Key: <u>project_id</u>
    - Candidate Key: {project_id}
    - Foreign Key:
        - provides.project_id references to project.project_id
        - provides.client_id references to client.client_id
    - Functional dependencies:
        - project_id → client_id
- *project_skills*:
    - Primary Key: <u>project_id</u>, <u>skills</u>
    - Candidate Key: {project_id , skills}
    - Foreign Key:
        - project_skills.project_id references to project.project_id
    - Functional dependencies:
- *freelancer_contact*:
    - Primary Key: <u>freelancer_id</u>, <u>phone_no</u>
    - Candidate Key: {freelancer_id , phone_no}
    - Foreign Key:
        - freelancer_contact.freelancer_id references to freelancer.freelancer_id
    - Functional dependencies:
- *client_contact*:
    - Primary Key: <u>client_id</u>,<u>phone_no</u>
    - Candidate Key: {client_id , phone_no}
    - Foreign Key:
        - client_contact.client_id references to client.freelancer_id
    - Functional dependencies:
- *freelancer_skills*:
    - Primary Key: <u>freelancer_id</u>, <u>skills</u>
    - Candidate Key: {freelancer_id , skills}
    - Foreign Key:
        - freelancer_skills.freelancer_id references to freelancer.freelancer_id

  - Functional dependencies:

4. List of redundancies:

- List of redundancies existing for every schema:
  - In the starting phase of the schema table freelancer and client has the multivalued attribute phone_no and freelancer and project table there was a multivalued attribute skills so those were not in the 1NF.

- Modified schemas:

- *freelancer* (freelancer_id, first_name, middle_name, last_name, date_of_birth, avg_rating, city, country, email_id, password)

- *client* (client_id, organization_name, first_name, middle_name, last_name, date_of_birth, city, country, email_id, password)

- *project* (project_id, amount, start_date, deadline, project_name, status, project_description)

- *contract* (contract_id, freelancer_id, client_id, project_id, date, time)

- *transaction* (transaction_id, contract_id, amount, date, time, payment_method)

- *reviews* (review_id, contract_id, rating, review)

- *takes* (freelancer_id, project_id)

- *provides* (project_id, client_id)

- *project_skills* (project_id, skills)

- *freelancer_phone* (freelancer_id, phone_no)

- *client_phone* (client_id,phone_no)

- *freelancer_skills* (freelancer_id, skills)

5. Normalizing the database up to 1NF:

In the client table and freelancer first contains the mobile number it is the multivalued attribute so it isn’t in 1NF so we made a new table for the client_phone_number and freelancer_phone_number and removed from the freelancer and client table.

Also the Skill is the multivalued attribute for the table freelancer and project so we made another table for the skill and removed skills from freelancer table and project table so now all tables are in 1NF.

6. Normalizing the database up to 2NF:

In all the relations, functional dependencies have No left side is proper subset candidate key and right side is Non Prime attribute so there is no partial dependencies so all tables are in 2NF.

7. List of redundancies existing for the schema in 2NF:

Redundancies after the 2NF is in the client and the freelancer table there is two attribute city and countries and there is the functional dependency city → country assuming there is not any city have same name but have different country.

8. Normalizing the database up to 3NF/BCNF:

Since there is one transitive dependency in the freelancer and client table city → country there is non prime attribute determines another non prime attribute so it isn’t in the 3NF so we broke these to table and make new table of R(<u>city</u>,country) and removed country from the both table or relation. This decomposition is also lossless decomposition because there is a common attribute city and that is primary key of one of the table and also dependency preserving decomposition.
Then, for the BCNF now all the relations functional dependencies have a candidate key on the right side so all are in BCNF and also they are lossless decomposition and dependency preserving.

9. Final list of schemas:

- *freelancer* (<u>freelancer_id</u>, first_name, middle_name, last_name, date_of_birth, avg_rating, city, email_id, password)
- *client* (<u>client_id</u>, organization_name, first_name, middle_name, last_name, date_of_birth, city, email_id, password)
- city_country(<u>city</u>,country)

- *contract* (contract_id, freelancer_id, client_id, project_id, date, time)
- *transaction* (transaction_id, contract_id, amount, date, time, payment_method)
- *reviews* (review_id, contract_id, rating, review)
- *takes* (freelancer_id, project_id)
- *provides* (project_id, client_id)

  → here also then we can merge the relation provides and the project because every project has only one client so one to one and also total participation means that if there is the project then it has to have at least one client but due to one to one it has to have only one client so we can add provides table to project table so new project table schema becomes:

  - *project* (project_id, client_id, amount, start_date, deadline, project_name, status, project_description)

- *project_skills* (project_id, skills)
- *freelancer_contact* (freelancer_id, phone_no)
- *client_contact* (client_id,phone_no)
- *freelancer_skills* (freelancer_id, skills)