UCS2404 Database Management Systems

Project Report

Journal / Conference Publication management system

Submitted By

Ram Prasath P (3122 22 5001 103)

Rohith Arumugam S (3122 22 5001 110)

Shreejith Babu G (3122 22 5001 702)

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

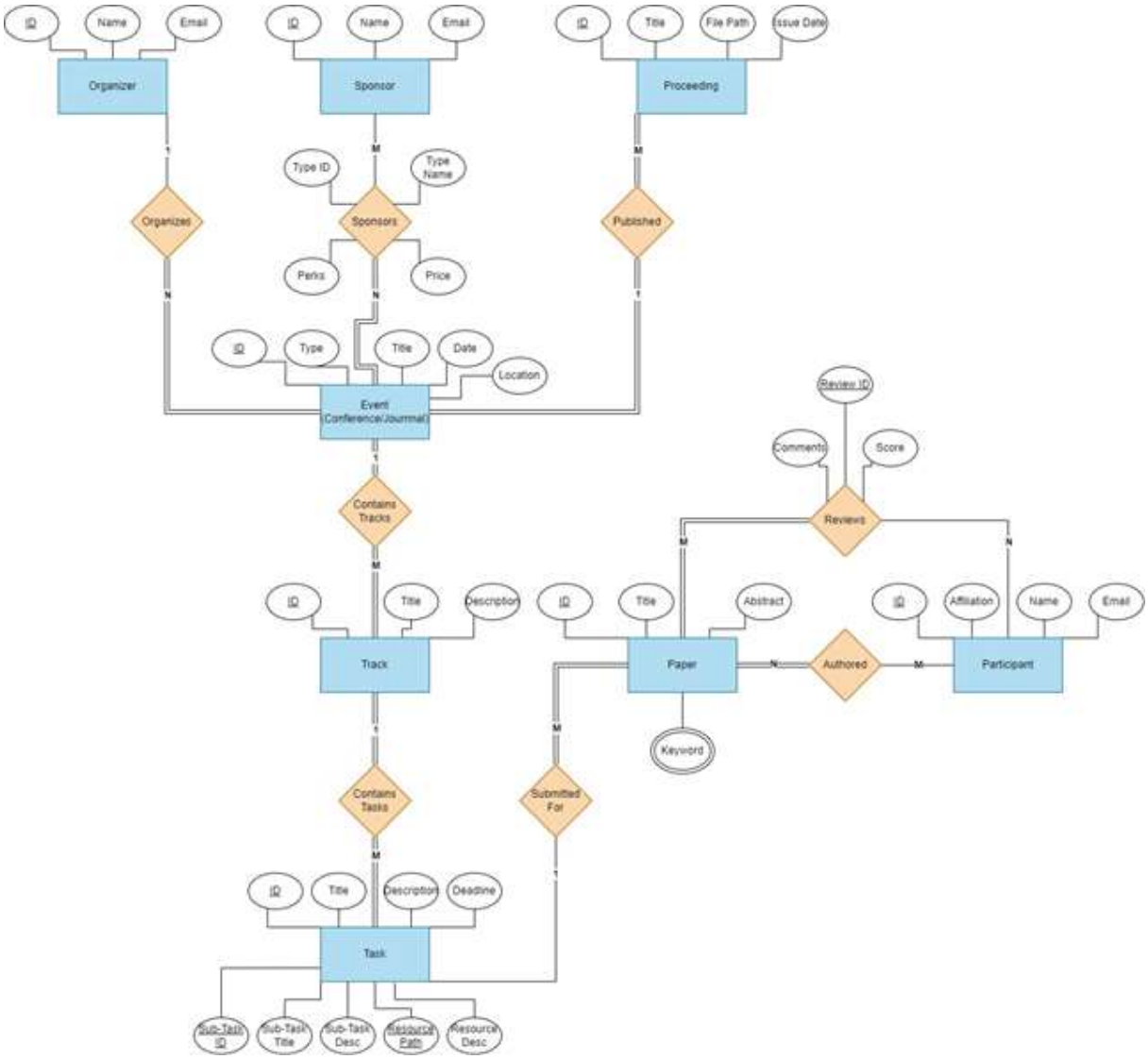
(An Autonomous Institution, Affiliated to Anna University)

Kalavakkam – 603110

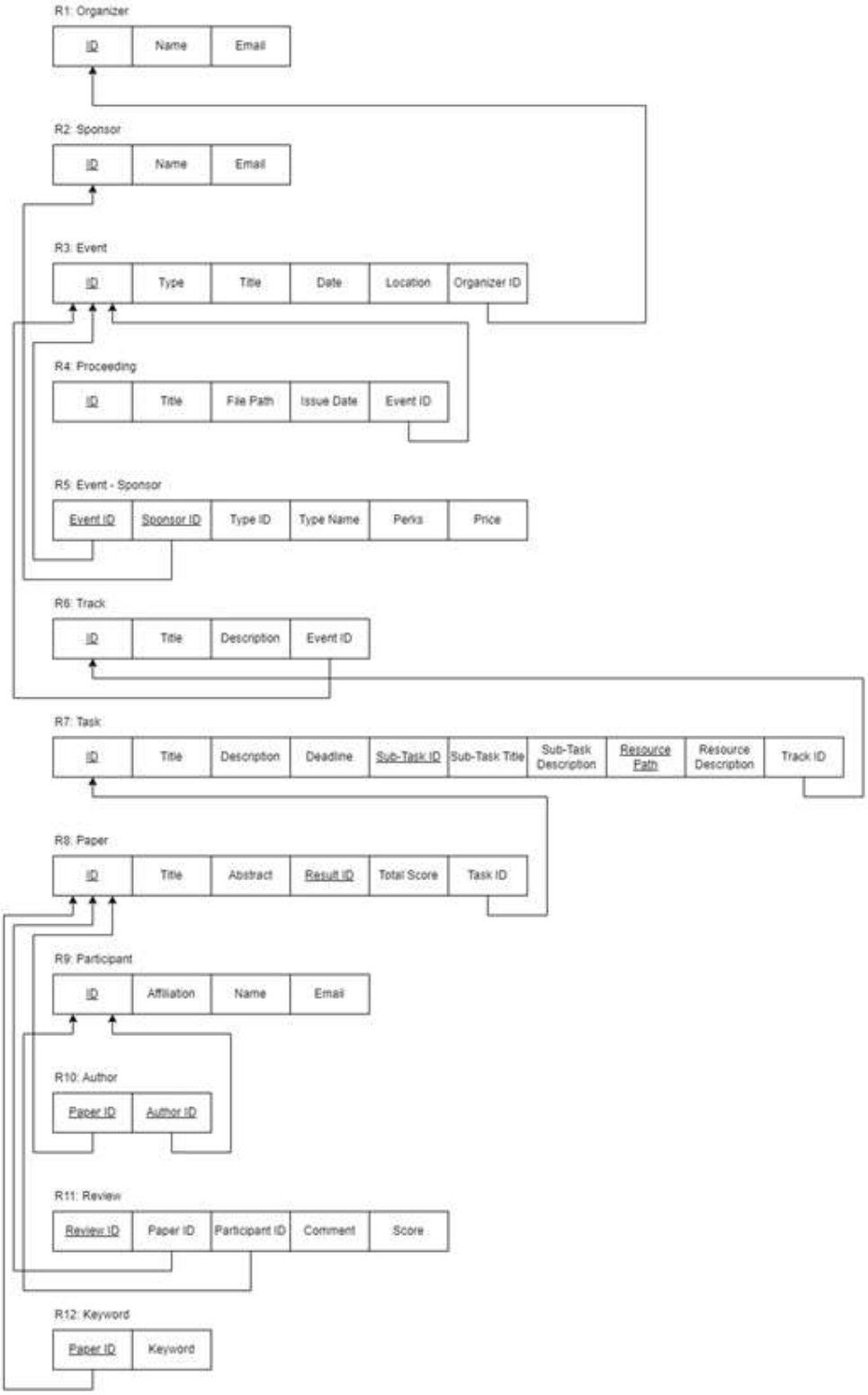
Problem Statement :

Analyse the data for a specific real-time application which uses a Journal Management system .Analyse the data for a specific real-time application which uses a Food Delivery Management system. Analysing the data includes identifying the constraints among the attributes and the existence of dependencies among the attributes. The constraints and dependencies identification could be framed/identified by deeper understanding of the problem specification, interactions/dependencies among various attributes and user-requirements for the specific real-time application. Prepare the documentation showing the list of attributes and identify various functional dependencies among the set of attributes.

ER DIAGRAM :


ID
Name
Email
Organizer
ID
Name
Email
Sponsor
ID
Title
File Path
Issue Date
Proceeding
Type ID
Type Name
Sponsors
Perks
Price
Organizes
Published
ID
Type
Title
Date
Location
Event
Conference/Journal
Contains Tracks
Review ID
Comments
Score
Reviews
ID
Title
Description
Track
ID
Title
Abstract
Paper
Keyword
Authored
ID
Affiliation
Name
Email
Participant
Contains Tasks
Submitted For
ID
Title
Description
Deadline
Task
Sub-Task ID
Sub-Task Title
Sub-Task Desc
Resource Path
Resource Desc

INITIAL REALTIONAL SCHEMA :



R1: Organizer

Attributes:

- A: ID (Primary Key)
- B: Name
- C: Email

Functional Dependencies:

- A -> B
- A -> C

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {A -> B, A -> C}
     - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
   - No extraneous attributes.
3. **Removal of redundant FDs**:
   - Checking redundancy:
     - Trying to remove A -> B:
       - Set of FDs: {A -> C}
       - $\{A\}^+ = \{A, C\}$ (does not include B)
       - Closure is not satisfied.
       - Therefore, A -> B is not redundant.
     - Trying to remove A -> C:
       - Set of FDs: {A -> B}
       - $\{A\}^+ = \{A, B\}$ (does not include C)
       - Closure is not satisfied.
       - Therefore, A -> C is not redundant.

Resulting FDs: {A -> B, A -> C}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Organizer (Divided into Smaller Relations)

Organizer1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C}
- FDs:
  - A → B
  - A → C

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R2: Sponsor

Attributes:

- A: ID (Primary Key)
- B: Name
- C: Email

Functional Dependencies:

- A -> B
- A -> C

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {A -> B, A -> C}
     - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
   - No extraneous attributes.
3. **Removal of redundant FDs**:
   - Checking redundancy:
     - Trying to remove A -> B:
       - Set of FDs: {A -> C}
       - $\{A\}^+$ = {A, C} (does not include B)
       - Closure is not satisfied.
       - Therefore, A -> B is not redundant.
     - Trying to remove A -> C:
       - Set of FDs: {A -> B}
       - $\{A\}^+$ = {A, B} (does not include C)
       - Closure is not satisfied.
       - Therefore, A -> C is not redundant.

Resulting FDs: {A -> B, A -> C,}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C}

NORMALISATION :

Table: Sponsor

Attributes:

- A: ID (Primary Key)
- B: Name
- C: Email

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is `A`.
- The functional dependencies are `A → B` and `A → C`.
- All non-key attributes (`B`, `C`) are fully functionally dependent on the primary key `A`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Sponsor (Divided into Smaller Relations)

Sponsor1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C}
- FDs:
  - A → B
  - A → C

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R3: Event

Attributes:

- A: ID (Primary Key)
- B: Type
- C: Title
- D: Date
- E: Location
- F: Organizer ID (Foreign Key)

Functional Dependencies:

- A -> B
- A -> C
- A -> D
- A -> E
- A -> F

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {A -> B, A -> C, A -> D, A -> E, A -> F}

- No further decomposition needed as the RHS are already single attributes.

2. **Removal of extraneous attributes**:
   - No extraneous attributes.
3. **Removal of redundant FDs**:
   - Checking redundancy:
     - Trying to remove A -> B:
       - Set of FDs: {A -> C, A -> D, A -> E, A -> F}
       - $\{A\}^+$ = {A, C, D, E, F} (does not include B)
       - Closure is not satisfied.
       - Therefore, A -> B is not redundant.
     - Trying to remove A -> C:
       - Set of FDs: {A -> B, A -> D, A -> E, A -> F}
       - $\{A\}^+$ = {A, B, D, E, F} (does not include C)
       - Closure is not satisfied.
       - Therefore, A -> C is not redundant.
     - Trying to remove A -> D:
       - Set of FDs: {A -> B, A -> C, A -> E, A -> F}
       - $\{A\}^+$ = {A, B, C, E, F} (does not include D)
       - Closure is not satisfied.
       - Therefore, A -> D is not redundant.
     - Trying to remove A -> E:
       - Set of FDs: {A -> B, A -> C, A -> D, A -> F}
       - $\{A\}^+$ = {A, B, C, D, F} (does not include E)
       - Closure is not satisfied.
       - Therefore, A -> E is not redundant.
     - Trying to remove A -> F:
       - Set of FDs: {A -> B, A -> C, A -> D, A -> E}
       - $\{A\}^+$ = {A, B, C, D, E} (does not include F)
       - Closure is not satisfied.
       - Therefore, A -> F is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D, A -> E, A -> F}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C, D, E, F}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is `A`.
- The functional dependencies are `A → B`, `A → C`, `A → D`, `A → E`, and `A → F`.
- All non-key attributes (`B`, `C`, `D`, `E`, `F`) are fully functionally dependent on the primary key `A`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C
- A → D
- A → E
- A → F

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Event (Divided into Smaller Relations)

Event1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C, D, E, F}
- FDs:
  - A → B
  - A → C
  - A → D
  - A → E
  - A → F

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R4: Proceeding

Attributes:

- A: ID (Primary Key)
- B: Title
- C: File Path
- D: Issue Date
- E: Event ID (Foreign Key)

Functional Dependencies:

- A -> B
- A -> C
- A -> D
- A -> E

Steps:

1. **Decomposition**:
    - Reduce RHS:
        - Set of FDs: {A -> B, A -> C, A -> D, A -> E}
        - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
    - No extraneous attributes.
3. **Removal of redundant FDs**:
    - Checking redundancy:
        - Trying to remove A -> B:
            - Set of FDs: {A -> C, A -> D, A -> E}
            - $\{A\}^+ = \{A, C, D, E\}$ (does not include B)
            - Closure is not satisfied.
            - Therefore, A -> B is not redundant.
        - Trying to remove A -> C:
            - Set of FDs: {A -> B, A -> D, A -> E}

- $\{A\}^+ = \{A, B, D, E\}$ (does not include C)
- Closure is not satisfied.
- Therefore, A -> C is not redundant.

- Trying to remove A -> D:
  - Set of FDs: {A -> B, A -> C, A -> E}
  - $\{A\}^+ = \{A, B, C, E\}$ (does not include D)
  - Closure is not satisfied.
  - Therefore, A -> D is not redundant.
- Trying to remove A -> E:
  - Set of FDs: {A -> B, A -> C, A -> D}
  - $\{A\}^+ = \{A, B, C, D\}$ (does not include E)
  - Closure is not satisfied.
  - Therefore, A -> E is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D, A -> E}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C, D, E}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is `A`.
- The functional dependencies are `A → B`, `A → C`, `A → D`, and `A → E`.
- All non-key attributes (`B`, `C`, `D`, `E`) are fully functionally dependent on the primary key `A`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C
- A → D
- A → E

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Proceeding (Divided into Smaller Relations)

Proceeding1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C, D, E}
- FDs:
  - A → B
  - A → C
  - A → D
  - A → E

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R5: Event-Sponsor

Attributes:

- A: Event ID
- B: Sponsor ID
- C: Type ID
- D: Type Name
- E: Perks
- F: Price

Functional Dependencies:

- AB -> C
- C -> D, E, F

Steps:

1. **Decomposition**:
    - Reduce RHS:
        - Set of FDs: {AB -> C, C -> D, C -> E, C -> F}
        - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
    - For AB -> C:
        - Taking A as extraneous:
            - B -> C (not logically valid as Sponsor ID alone does not determine Type ID)
        - Taking B as extraneous:
            - A -> C (not logically valid as Event ID alone does not determine Type ID)
        - Therefore, no extraneous attributes in AB -> C.
    - For C -> D:
        - C -> D is already in its minimal form.
    - For C -> E:
        - C -> E is already in its minimal form.
    - For C -> F:
        - C -> F is already in its minimal form.
3. **Removal of redundant FDs**:
    - Checking redundancy:
        - Trying to remove AB -> C:
            - Set of FDs: {C -> D, C -> E, C -> F}
            - $\{C\}^+ = \{C, D, E, F\}$ (does not include AB)
            - Closure is not satisfied.
            - Therefore, AB -> C is not redundant.
        - Trying to remove C -> D:
            - Set of FDs: {AB -> C, C -> E, C -> F}
            - $\{C\}^+ = \{C, E, F\}$ (does not include D)
            - Closure is not satisfied.
            - Therefore, C -> D is not redundant.
        - Trying to remove C -> E:
            - Set of FDs: {AB -> C, C -> D, C -> F}
            - $\{C\}^+ = \{C, D, F\}$ (does not include E)
            - Closure is not satisfied.
            - Therefore, C -> E is not redundant.
        - Trying to remove C -> F:

- Set of FDs: {AB -> C, C -> D, C -> E}
- $\{C\}^+$ = {C, D, E} (does not include F)
- Closure is not satisfied.
- Therefore, C -> F is not redundant.

Resulting FDs: {AB -> C, C -> D, C -> E, C -> F}

Candidate Key: {A, B}

Primary Key: AB

Prime Attributes: {A, B}

Non-Prime Attributes: {C, D, E, F}

NORMALISATION :

Functional Dependencies:

- AB → C
- C → D, E, F

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The composite primary key is `{A, B}`.
- The functional dependencies are `AB → C` and `C → D, E, F`.
- All non-key attributes (`C`, `D`, `E`, `F`) are fully functionally dependent on the composite primary key `{A, B}`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

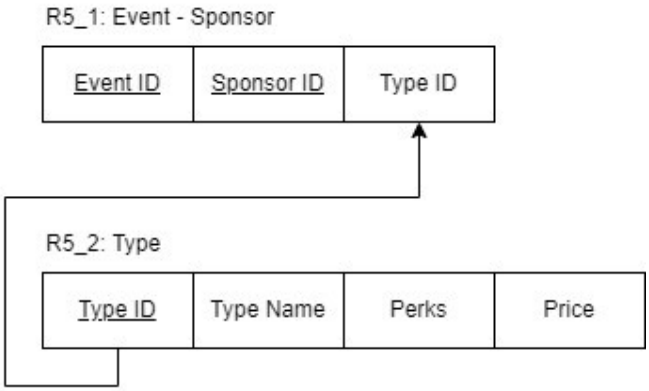
- AB → C
- C → D, E, F

Since C is not a candidate key and it determines non-prime attributes (D, E, F), there is a transitive dependency which violates the 3NF condition.

Normalization to 3NF:

Decomposition:

1. **Event-Sponsor:**
   - Attributes: {A, B, C}
   - Candidate key: {A, B}
   - PA: {A, B}
   - NPA: {C}
   - FDs:
     - AB → C
2. **Type:**
   - Attributes: {C, D, E, F}
   - Candidate key: {C}
   - PA: {C}
   - NPA: {D, E, F}
   - FDs:
     - C → D, E, F



Now, each table is in 3NF.

Checking for BCNF:

- LHS must be a Super-key.

Event-Sponsor:

- Functional Dependency: AB → C
- {A, B} is a composite key and a superkey, so this relation is in BCNF.

Type:

- Functional Dependency: C → D, E, F

- C is a candidate key and a superkey, so this relation is in BCNF.

R6: Track

Attributes:

- A: ID
- B: Title
- C: Description
- D: Event ID

Functional Dependencies:

- A -> B
- A -> D
- A -> C

Steps:

1. **Decomposition**:
    - Reduce RHS:
        - Set of FDs: {A -> B, A -> D, A -> C}
        - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
    - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
    - Checking redundancy:
        - Trying to remove A -> B:
            - Set of FDs: {A -> D, A -> C}
            - $\{A\}^+$ = {A, D, C} (does not include B)
            - Closure is not satisfied.
            - Therefore, A -> B is not redundant.
        - Trying to remove A -> D:
            - Set of FDs: {A -> B, A -> C}
            - $\{A\}^+$ = {A, B, C} (does not include D)
            - Closure is not satisfied.
            - Therefore, A -> D is not redundant.
        - Trying to remove A -> C:
            - Set of FDs: {A -> B, A -> D}

- $\{A\}^+ = \{A, B, D\}$ (does not include C)
- Closure is not satisfied.
- Therefore, A -> C is not redundant.

Resulting FDs: {A -> B, A -> D, A -> C}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C, D}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is `A`.
- The functional dependencies are `A → B`, `A → C`, and `A → D`.
- All non-key attributes (`B`, `C`, `D`) are fully functionally dependent on the primary key `A`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C
- A → D

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Track (Divided into Smaller Relations)

Track1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C, D}
- FDs:
  - A → B
  - A → C
  - A → D

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R7: Task

Attributes:

- A: ID
- B: Title
- C: Description
- D: Deadline
- E: Sub-Task ID
- F: Sub-Task Title
- G: Sub-Task Description
- H: Resource Path
- I: Resource Description
- J: Track ID

Functional Dependencies:

- A -> B, C, D, J
- E -> F, G, H
- H -> I

Steps:

1. **Decomposition**:
   - Reduce RHS:

- Set of FDs: {A -> B, A -> C, A -> D, A -> J, E -> F, E -> G, E -> H, H -> I}

2. **Removal of extraneous attributes**:
   - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
   - Checking redundancy for each FD:
     - Trying to remove A -> B:
       - Set of FDs: {A -> C, A -> D, A -> J, E -> F, E -> G, E -> H, H -> I}
       - $\{A\}^+$ = {A, C, D, J} (does not include B)
       - Closure is not satisfied.
       - Therefore, A -> B is not redundant.
     - Trying to remove A -> C:
       - Set of FDs: {A -> B, A -> D, A -> J, E -> F, E -> G, E -> H, H -> I}
       - $\{A\}^+$ = {A, B, D, J} (does not include C)
       - Closure is not satisfied.
       - Therefore, A -> C is not redundant.
     - Trying to remove A -> D:
       - Set of FDs: {A -> B, A -> C, A -> J, E -> F, E -> G, E -> H, H -> I}
       - $\{A\}^+$ = {A, B, C, J} (does not include D)
       - Closure is not satisfied.
       - Therefore, A -> D is not redundant.
     - Trying to remove A -> J:
       - Set of FDs: {A -> B, A -> C, A -> D, E -> F, E -> G, E -> H, H -> I}
       - $\{A\}^+$ = {A, B, C, D} (does not include J)
       - Closure is not satisfied.
       - Therefore, A -> J is not redundant.
     - Trying to remove E -> F:
       - Set of FDs: {A -> B, A -> C, A -> D, A -> J, E -> G, E -> H, H -> I}
       - $\{E\}^+$ = {E, G, H} (does not include F)
       - Closure is not satisfied.
       - Therefore, E -> F is not redundant.
     - Trying to remove E -> G:
       - Set of FDs: {A -> B, A -> C, A -> D, A -> J, E -> F, E -> H, H -> I}
       - $\{E\}^+$ = {E, F, H} (does not include G)
       - Closure is not satisfied.
       - Therefore, E -> G is not redundant.
     - Trying to remove E -> H:
       - Set of FDs: {A -> B, A -> C, A -> D, A -> J, E -> F, E -> G, H -> I}

- $\{E\}^+ = \{E, F, G\}$ (does not include H)
- Closure is not satisfied.
- Therefore, E -> H is not redundant.

- Trying to remove H -> I:
  - Set of FDs: {A -> B, A -> C, A -> D, A -> J, E -> F, E -> G, E -> H}
  - $\{H\}^+ = \{H\}$ (does not include I)
  - Closure is not satisfied.
  - Therefore, H -> I is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D, A -> J, E -> F, E -> G, E -> H, H -> I}

Candidate Key: {AE}

Primary Key: AE

Prime Attributes: {A, E}

Non-Prime Attributes: {B, C, D, F, G, H, I, J}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The composite primary key is `{A, E}`.
- However, let's first decompose based on the functional dependencies to achieve 2NF.

Decomposition into 2NF:

1. **Task:**
   - Attributes: {A, B, C, D, J}
   - Candidate key: {A}
   - PA: {A}
   - NPA: {B, C, D, J}
   - FDs:
     - A → B, C, D, J
2. **Sub Task:**
   - Attributes: {E, F, G, H, I}
   - Candidate key: {E}
   - PA: {E}

- NPA: {F, G, H, I}
- FDs:
  - E → F, G, H
  - H → I

3. **Task-SubTask:**
   - Attributes: {A, E}
   - Candidate key: {A, E}
   - PA: {A, E}
   - FDs:
     - None (A, E is a composite key and holds the relationship between Task ID and Sub-Task ID)

Now, each table is in 2NF as all non-key attributes are fully functionally dependent on the primary key.

Checking for 3NF:

Task1:

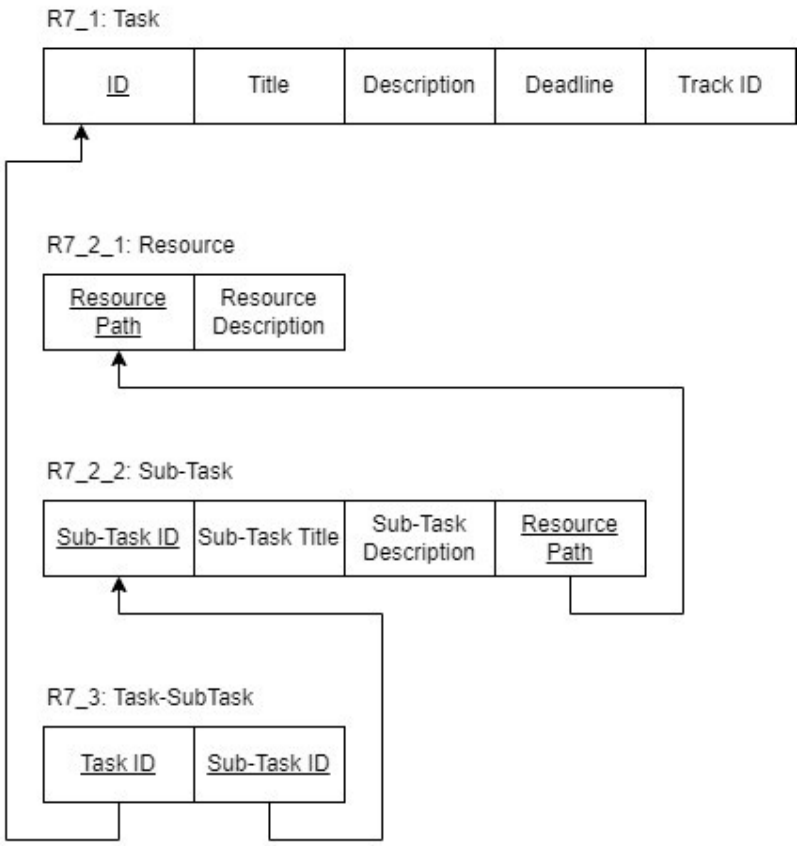
- Functional Dependencies: A → B, C, D, J
- No transitive dependencies exist.
- The table is in 3NF.

Task2 (Decomposed further into Task2a and Task2b for 3NF):

1. **Sub Task:**
   - Attributes: {E, F, G, H}
   - Candidate key: {E}
   - PA: {E}
   - NPA: {F, G, H}
   - FDs:
     - E → F, G, H
2. **Resource:**
   - Attributes: {H, I}
   - Candidate key: {H}
   - PA: {H}
   - NPA: {I}
   - FDs:
     - H → I

Task-SubTask:

- Attributes: {A, E}
- Functional Dependencies: None
- The table is in 3NF.



R8: Paper

Attributes:

- A: ID
- B: Title
- C: Abstract
- D: Result ID
- E: Total Score
- F: Task ID

Functional Dependencies:

- A -> B
- A -> C
- A -> D
- A -> F
- D -> E

Steps:

1. **Decomposition**:
    - Reduce RHS:
        - Set of FDs: {A -> B, A -> C, A -> D, A -> F, D -> E}
2. **Removal of extraneous attributes**:
    - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
    - Checking redundancy:
        - Trying to remove A -> B:
            - Set of FDs: {A -> C, A -> D, A -> F, D -> E}
            - $\{A\}^+$ = {A, C, D, F} (does not include B)
            - Closure is not satisfied.
            - Therefore, A -> B is not redundant.
        - Trying to remove A -> C:
            - Set of FDs: {A -> B, A -> D, A -> F, D -> E}
            - $\{A\}^+$ = {A, B, D, F} (does not include C)
            - Closure is not satisfied.
            - Therefore, A -> C is not redundant.
        - Trying to remove A -> D:
            - Set of FDs: {A -> B, A -> C, A -> F, D -> E}
            - $\{A\}^+$ = {A, B, C, F} (does not include D)
            - Closure is not satisfied.
            - Therefore, A -> D is not redundant.
        - Trying to remove A -> F:
            - Set of FDs: {A -> B, A -> C, A -> D, D -> E}
            - $\{A\}^+$ = {A, B, C, D} (does not include F)
            - Closure is not satisfied.
            - Therefore, A -> F is not redundant.
        - Trying to remove D -> E:
            - Set of FDs: {A -> B, A -> C, A -> D, A -> F}
            - $\{D\}^+$ = {D} (does not include E)
            - Closure is not satisfied.
            - Therefore, D -> E is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D, A -> F, D -> E}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C, D, E, F}

R9: Participant

Attributes:

- A: ID
- B: Affiliation
- C: Name
- D: Email

Functional Dependencies:

- A -> B, C, D

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {A -> B, A -> C, A -> D}
     - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
   - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
   - Checking redundancy:
     - Trying to remove A -> B:
       - Set of FDs: {A -> C, A -> D}
       - $\{A\}^+$ = {A, C, D} (does not include B)
       - Closure is not satisfied.
       - Therefore, A -> B is not redundant.
     - Trying to remove A -> C:
       - Set of FDs: {A -> B, A -> D}
       - $\{A\}^+$ = {A, B, D} (does not include C)
       - Closure is not satisfied.
       - Therefore, A -> C is not redundant.
     - Trying to remove A -> D:
       - Set of FDs: {A -> B, A -> C}
       - $\{A\}^+$ = {A, B, C} (does not include D)
       - Closure is not satisfied.
       - Therefore, A -> D is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D}

Candidate Key: {A}

Primary Key: A **Prime Attributes**: {A}

Non-Prime Attributes: {B, C, D}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is `A`.
- The functional dependencies are `A → B`, `A → C`, and `A → D`.
- All non-key attributes (`B`, `C`, `D`) are fully functionally dependent on the primary key `A`.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B
- A → C
- A → D

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Participant (Divided into Smaller Relations)

Participant1:

- Candidate key: {A}
- PA: {A}
- NPA: {B, C, D}
- FDs:

- A → B
- A → C
- A → D

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R10: Author

Attributes:

- A: Paper ID
- B: Author ID

Functional Dependencies:

- AB -> AB (This means both Paper ID and Author ID together form the composite primary key)

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {AB -> A, AB -> B}
     - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
   - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
   - No redundant FDs.

Resulting FDs: {AB -> A, AB -> B}

Candidate Key: {A, B}

Primary Key: AB **Prime Attributes**: {A, B}

Non-Prime Attributes: {}

R11: Review

Attributes:

- A: Review ID
- B: Paper ID
- C : Participant ID
- D: Comment
- E: Score

Steps:

1. **Decomposition**:
    - Reduce RHS:
        - Set of FDs: {A -> B, A -> C, A -> D, A -> E}
        - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
    - No extraneous attributes.
3. **Removal of redundant FDs**:
    - Checking redundancy:
        - Trying to remove A -> B:
            - Set of FDs: {A -> C, A -> D, A -> E}
            - $\{A\}^+$ = {A, C, D, E} (does not include B)
            - Closure is not satisfied.
            - Therefore, A -> B is not redundant.
        - Trying to remove A -> C:
            - Set of FDs: {A -> B, A -> D, A -> E}
            - $\{A\}^+$ = {A, B, D, E} (does not include C)
            - Closure is not satisfied.
            - Therefore, A -> C is not redundant.
        - Trying to remove A -> D:
            - Set of FDs: {A -> B, A -> C, A -> E}
            - $\{A\}^+$ = {A, B, C, E} (does not include D)
            - Closure is not satisfied.
            - Therefore, A -> D is not redundant.
        - Trying to remove A -> E:
            - Set of FDs: {A -> B, A -> C, A -> D}
            - $\{A\}^+$ = {A, B, C, D} (does not include E)
            - Closure is not satisfied.
            - Therefore, A -> E is not redundant.

Resulting FDs: {A -> B, A -> C, A -> D, A -> E}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B, C, D, E}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The composite primary key is `{A, B}`.
- There are no partial dependencies because there are no non-key attributes.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- AB → AB

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Author (Divided into Smaller Relations)

Author1:

- Candidate key: {A, B}
- PA: {A, B}
- FDs:
    - AB → AB

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

R12: Keyword

Attributes:

- A: Paper ID
- B: Keyword

Functional Dependencies:

- A -> B

Steps:

1. **Decomposition**:
   - Reduce RHS:
     - Set of FDs: {A -> B}
     - No further decomposition needed as the RHS are already single attributes.
2. **Removal of extraneous attributes**:
   - No extraneous attributes in the given set of FDs.
3. **Removal of redundant FDs**:
   - No redundant FDs.

Resulting FDs: {A -> B}

Candidate Key: {A}

Primary Key: A

Prime Attributes: {A}

Non-Prime Attributes: {B}

NORMALISATION :

Checking for 1NF:

- The atomicity of the relation is satisfied.
- The above relation satisfies the 1NF condition.

Checking for 2NF:

- No partial dependency should exist in the relation.
- The primary key is A (assuming that each Paper ID is associated with a unique keyword).
- The functional dependency is A → B.
- All non-key attributes (B) are fully functionally dependent on the primary key A.

The above relation satisfies the 2NF condition.

Checking for 3NF:

- No transitive dependency should exist in the relation.
- NPA → NPA (Non-Prime Attribute does not determine another Non-Prime Attribute)

Given Functional Dependencies:

- A → B

These functional dependencies do not violate the 3NF condition.

The 3NF is satisfied.

Table: Keyword (Divided into Smaller Relations)

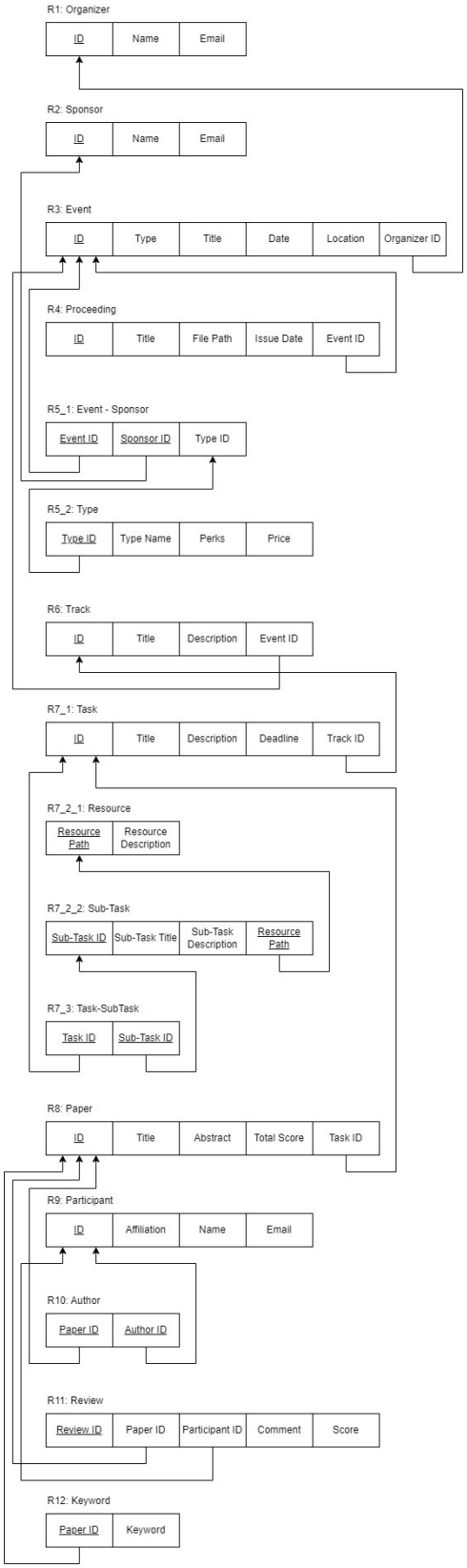
Keyword1:

- Candidate key: {A}
- PA: {A}
- NPA: {B}
- FDs:
    - A → B

Checking for BCNF:

- LHS must be a Super-key.
- The above relation satisfies the BCNF condition.

REALTIONAL SCHEMA AFTER NORMALISATION:



Creating and Populating Tables with Triggers and Constraints:

```
--@ "C:\rohith\college\sem4\dbms\project\database.sql"

-- Drop existing tables with cascade constraints
DROP TABLE Keyword CASCADE CONSTRAINTS;
DROP TABLE Organizer CASCADE CONSTRAINTS;
DROP TABLE Sponsor CASCADE CONSTRAINTS;
DROP TABLE Event_Details CASCADE CONSTRAINTS;
DROP TABLE Event_Sponsor CASCADE CONSTRAINTS;
DROP TABLE Track CASCADE CONSTRAINTS;
DROP TABLE Task CASCADE CONSTRAINTS;
DROP TABLE Task_SubTask CASCADE CONSTRAINTS;
DROP TABLE SubTask CASCADE CONSTRAINTS;
DROP TABLE Task_Resource CASCADE CONSTRAINTS;
DROP TABLE Participant CASCADE CONSTRAINTS;
DROP TABLE Author CASCADE CONSTRAINTS;
DROP TABLE Review CASCADE CONSTRAINTS;
DROP TABLE Proceeding CASCADE CONSTRAINTS;
DROP TABLE Sponsor_Type CASCADE CONSTRAINTS;
DROP TABLE Paper CASCADE CONSTRAINTS;
DROP TRIGGER UpdateTotalScore;

--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------

-- Create Keyword table
CREATE TABLE Keyword (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100)
);

-- Create Organizer table
CREATE TABLE Organizer (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Email VARCHAR2(100),
    Type VARCHAR2(50)
);

-- Create Sponsor table
CREATE TABLE Sponsor (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
```

```
    Email VARCHAR2(100)
);

-- Create Event table
CREATE TABLE Event_Details (
    ID NUMBER PRIMARY KEY,
    Type VARCHAR2(50),
    Title VARCHAR2(200),
    Event_Date DATE,
    Location VARCHAR2(200),
    OrganizerID NUMBER,
    FOREIGN KEY (OrganizerID) REFERENCES Organizer(ID)
);

-- Create Event-Sponsor relationship table
CREATE TABLE Event_Sponsor (
    EventID NUMBER,
    SponsorID NUMBER,
    PRIMARY KEY (EventID, SponsorID),
    FOREIGN KEY (EventID) REFERENCES Event_Details(ID),
    FOREIGN KEY (SponsorID) REFERENCES Sponsor(ID)
);

-- Create Track table
CREATE TABLE Track (
    ID NUMBER PRIMARY KEY,
    Title VARCHAR2(200),
    Description VARCHAR2(500),
    EventID NUMBER,
    FOREIGN KEY (EventID) REFERENCES Event_Details(ID)
);

-- Create Task table
CREATE TABLE Task (
    ID NUMBER PRIMARY KEY,
    Title VARCHAR2(200),
    Deadline DATE,
    TrackID NUMBER,
    FOREIGN KEY (TrackID) REFERENCES Track(ID)
);

-- Create Paper table with TotalScore column
CREATE TABLE Paper (
    ID NUMBER PRIMARY KEY,
    Title VARCHAR2(20),
    Abstract VARCHAR2(30),
    TaskID NUMBER,
    TotalScore NUMBER DEFAULT 0,
```

```
    FOREIGN KEY (TaskID) REFERENCES Task(ID)
);

-- Create Sub-Task table
CREATE TABLE SubTask (
    ID NUMBER PRIMARY KEY,
    Title VARCHAR2(200),
    Description VARCHAR2(500),
    ResourcePath VARCHAR2(300)
);

-- Create Task-SubTask relationship table
CREATE TABLE Task_SubTask (
    TaskID NUMBER,
    SubTaskID NUMBER,
    PRIMARY KEY (TaskID, SubTaskID),
    FOREIGN KEY (TaskID) REFERENCES Task(ID),
    FOREIGN KEY (SubTaskID) REFERENCES SubTask(ID)
);

-- Create Resource table
CREATE TABLE Task_Resource (
    ID NUMBER PRIMARY KEY,
    Description VARCHAR2(500),
    Path VARCHAR2(300)
);

-- Create Participant table
CREATE TABLE Participant (
    ID NUMBER PRIMARY KEY,
    Affiliation VARCHAR2(20),
    Name VARCHAR2(20),
    Email VARCHAR2(20),
    Type VARCHAR2(20)
);

-- Create Author table
CREATE TABLE Author (
    PaperID NUMBER,
    AuthorID NUMBER,
    PRIMARY KEY (PaperID, AuthorID),
    FOREIGN KEY (PaperID) REFERENCES Paper(ID),
    FOREIGN KEY (AuthorID) REFERENCES Participant(ID)
);

-- Create Review table
CREATE TABLE Review (
    ReviewID NUMBER,
```

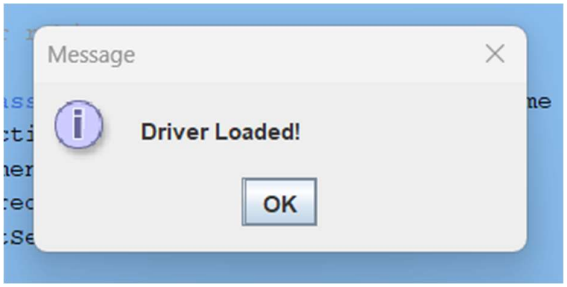
```
    PaperID NUMBER,
    ParticipantID NUMBER,
    Comments VARCHAR2(20),
    Score NUMBER,
    PRIMARY KEY (PaperID, ParticipantID),
    FOREIGN KEY (PaperID) REFERENCES Paper(ID),
    FOREIGN KEY (ParticipantID) REFERENCES Participant(ID)
);

-- Create Proceeding table
CREATE TABLE Proceeding (
    ID NUMBER PRIMARY KEY,
    Title VARCHAR2(200),
    FilePath VARCHAR2(300),
    IssueDate DATE,
    EventID NUMBER,
    FOREIGN KEY (EventID) REFERENCES Event_Details(ID)
);

-- Create Type table
CREATE TABLE Sponsor_Type (
    ID NUMBER PRIMARY KEY,
    TypeName VARCHAR2(100),
    Perks VARCHAR2(200),
    Price NUMBER
);

-- Create trigger to update TotalScore in Paper table
CREATE OR REPLACE TRIGGER UpdateTotalScore
BEFORE INSERT OR UPDATE OR DELETE ON Review
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        -- Update TotalScore when a new review is added
        UPDATE Paper
        SET TotalScore = TotalScore + :NEW.Score
        WHERE ID = :NEW.PaperID;

    ELSIF UPDATING THEN
        -- Adjust TotalScore when an existing review is updated
        UPDATE Paper
        SET TotalScore = TotalScore - :OLD.Score + :NEW.Score
        WHERE ID = :NEW.PaperID;

    ELSIF DELETING THEN
        -- Update TotalScore when a review is deleted
        UPDATE Paper
        SET TotalScore = TotalScore - :OLD.Score
```

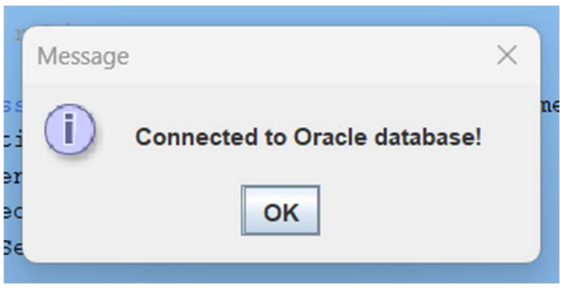
```
        WHERE ID = :OLD.PaperID;
    END IF;
END;
/

--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------

-- Insert values into Keyword table
INSERT INTO Keyword (ID, Name) VALUES (1, 'Artificial Intelligence');
INSERT INTO Keyword (ID, Name) VALUES (2, 'Machine Learning');
INSERT INTO Keyword (ID, Name) VALUES (3, 'Data Science');

-- Insert values into Organizer table
INSERT INTO Organizer (ID, Name, Email, Type) VALUES (1, 'John Doe',
'john.doe@example.com', 'Individual');
INSERT INTO Organizer (ID, Name, Email, Type) VALUES (2, 'Tech Org',
'info@techorg.com', 'Organization');
INSERT INTO Organizer (ID, Name, Email, Type) VALUES (3, 'Event Planner',
'planner@example.com', 'Individual');

-- Insert values into Sponsor table
INSERT INTO Sponsor (ID, Name, Email) VALUES (1, 'Company A',
'contact@companya.com');
INSERT INTO Sponsor (ID, Name, Email) VALUES (2, 'Company B',
'contact@companyb.com');
INSERT INTO Sponsor (ID, Name, Email) VALUES (3, 'Company C',
'contact@companyc.com');

-- Insert values into Event_Details table
INSERT INTO Event_Details (ID, Type, Title, Event_Date, Location, OrganizerID)
VALUES (1, 'Conference', 'Tech Conference 2024', TO_DATE('2024-09-01', 'YYYY-MM-
DD'), 'New York', 1);
INSERT INTO Event_Details (ID, Type, Title, Event_Date, Location, OrganizerID)
VALUES (2, 'Workshop', 'AI Workshop', TO_DATE('2024-10-10', 'YYYY-MM-DD'), 'San
Francisco', 2);
INSERT INTO Event_Details (ID, Type, Title, Event_Date, Location, OrganizerID)
VALUES (3, 'Seminar', 'Data Science Seminar', TO_DATE('2024-11-15', 'YYYY-MM-DD'),
'Los Angeles', 3);

-- Insert values into Event_Sponsor table
INSERT INTO Event_Sponsor (EventID, SponsorID) VALUES (1, 1);
INSERT INTO Event_Sponsor (EventID, SponsorID) VALUES (2, 2);
INSERT INTO Event_Sponsor (EventID, SponsorID) VALUES (3, 3);

-- Insert values into Track table
```

```
INSERT INTO Track (ID, Title, Description, EventID)
VALUES (1, 'AI Track', 'All about Artificial Intelligence', 1);
INSERT INTO Track (ID, Title, Description, EventID)
VALUES (2, 'ML Track', 'Deep dive into Machine Learning', 2);
INSERT INTO Track (ID, Title, Description, EventID)
VALUES (3, 'DS Track', 'Exploring Data Science', 3);

-- Insert values into Task table
INSERT INTO Task (ID, Title, Deadline, TrackID)
VALUES (1, 'Task 1', TO_DATE('2024-08-15', 'YYYY-MM-DD'), 1);
INSERT INTO Task (ID, Title, Deadline, TrackID)
VALUES (2, 'Task 2', TO_DATE('2024-09-25', 'YYYY-MM-DD'), 2);
INSERT INTO Task (ID, Title, Deadline, TrackID)
VALUES (3, 'Task 3', TO_DATE('2024-10-30', 'YYYY-MM-DD'), 3);

-- Insert values into Paper table
INSERT INTO Paper (ID, Title, Abstract, TaskID, TotalScore)
VALUES (1, 'Paper 1', 'Abstract of Paper 1', 1, 0);
INSERT INTO Paper (ID, Title, Abstract, TaskID, TotalScore)
VALUES (2, 'Paper 2', 'Abstract of Paper 2', 2, 0);
INSERT INTO Paper (ID, Title, Abstract, TaskID, TotalScore)
VALUES (3, 'Paper 3', 'Abstract of Paper 3', 3, 0);

-- Insert values into SubTask table
INSERT INTO SubTask (ID, Title, Description, ResourcePath)
VALUES (1, 'SubTask 1', 'Description of SubTask 1', '/path/to/resource1');
INSERT INTO SubTask (ID, Title, Description, ResourcePath)
VALUES (2, 'SubTask 2', 'Description of SubTask 2', '/path/to/resource2');
INSERT INTO SubTask (ID, Title, Description, ResourcePath)
VALUES (3, 'SubTask 3', 'Description of SubTask 3', '/path/to/resource3');

-- Insert values into Task_SubTask table
INSERT INTO Task_SubTask (TaskID, SubTaskID) VALUES (1, 1);
INSERT INTO Task_SubTask (TaskID, SubTaskID) VALUES (2, 2);
INSERT INTO Task_SubTask (TaskID, SubTaskID) VALUES (3, 3);

-- Insert values into Task_Resource table
INSERT INTO Task_Resource (ID, Description, Path)
VALUES (1, 'Resource 1 Description', '/path/to/resource1');
INSERT INTO Task_Resource (ID, Description, Path)
VALUES (2, 'Resource 2 Description', '/path/to/resource2');
INSERT INTO Task_Resource (ID, Description, Path)
VALUES (3, 'Resource 3 Description', '/path/to/resource3');

-- Insert values into Participant table
INSERT INTO Participant (ID, Affiliation, Name, Email, Type)
VALUES (1, 'Univ A', 'Alice', 'alice@ex.com', 'Author');
INSERT INTO Participant (ID, Affiliation, Name, Email, Type)
```

```
VALUES (2, 'Univ B', 'Bob', 'bob@ex.com', 'Reviewer');
INSERT INTO Participant (ID, Affiliation, Name, Email, Type)
VALUES (3, 'Univ C', 'Charlie', 'charlie@ex.com', 'Attendee');

-- Insert values into Author table
INSERT INTO Author (PaperID, AuthorID) VALUES (1, 1);
INSERT INTO Author (PaperID, AuthorID) VALUES (2, 1);
INSERT INTO Author (PaperID, AuthorID) VALUES (3, 1);

-- Insert values into Review table
INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score)
VALUES (1, 1, 2, 'Excellent.', 9);
INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score)
VALUES (2, 2, 3, 'Good.', 8);
INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score)
VALUES (3, 3, 1, 'Splendid.', 7);
INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score)
VALUES (4, 1, 3, 'Insightful.', 8);
INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score)
VALUES (5, 2, 1, 'Great.', 6);


-- Insert values into Proceeding table
INSERT INTO Proceeding (ID, Title, FilePath, IssueDate, EventID)
VALUES (1, 'Proceeding 1', '/path/to/proceeding1', TO_DATE('2024-09-02', 'YYYY-MM-
DD'), 1);
INSERT INTO Proceeding (ID, Title, FilePath, IssueDate, EventID)
VALUES (2, 'Proceeding 2', '/path/to/proceeding2', TO_DATE('2024-10-11', 'YYYY-MM-
DD'), 2);
INSERT INTO Proceeding (ID, Title, FilePath, IssueDate, EventID)
VALUES (3, 'Proceeding 3', '/path/to/proceeding3', TO_DATE('2024-11-16', 'YYYY-MM-
DD'), 3);

-- Insert values into Sponsor_Type table
INSERT INTO Sponsor_Type (ID, TypeName, Perks, Price)
VALUES (1, 'Platinum', 'All-access pass, Logo on materials', 10000);
INSERT INTO Sponsor_Type (ID, TypeName, Perks, Price)
VALUES (2, 'Gold', 'All-access pass', 5000);
INSERT INTO Sponsor_Type (ID, TypeName, Perks, Price)
VALUES (3, 'Silver', 'Logo on materials', 2500);

commit;
```

Loading the Driver and Establishing the Connection:





Main Code to Select the Table:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package com.oracle.paper_journal;
import javax.swing.*;
import java.sql.*;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author rohit
 */
public class review extends javax.swing.JFrame {
    Connection con;
    Statement st;
```

```
    PreparedStatement ps;
    ResultSet rs;

    /**
     * Creates new form review
     */
    public review() {
        initComponents();
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            JOptionPane.showMessageDialog(this, "Driver Loaded!");

            try {
                con =
DriverManager.getConnection("jdbc:oracle:thin:@Rohith:1521:orcla", "scott",
"tiger");
                JOptionPane.showMessageDialog(this, "Connected to Oracle
database!");
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
                JOptionPane.showMessageDialog(this, ex.getMessage());
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jLabel6 = new javax.swing.JLabel();
        PID = new javax.swing.JTextField();
        PaID = new javax.swing.JTextField();
        Comments = new javax.swing.JTextField();
        Score = new javax.swing.JTextField();
        insertButton = new javax.swing.JButton();
        updateButton = new javax.swing.JButton();
        searchButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();
```

```
        clearButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        Title_ = new javax.swing.JLabel();
        Abstract_ = new javax.swing.JLabel();
        Result_ = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        RID = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel6.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
        jLabel6.setText("Review Table");

        PID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PIDActionPerformed(evt);
            }
        });

        PaID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PaIDActionPerformed(evt);
            }
        });

        Comments.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                CommentsActionPerformed(evt);
            }
        });

        Score.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ScoreActionPerformed(evt);
            }
        });

        insertButton.setText("Insert");
        insertButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insertButtonActionPerformed(evt);
            }
        });

        updateButton.setText("Update");
        updateButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateButtonActionPerformed(evt);
```

```
            }
        });

        searchButton.setText("Search");
        searchButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                searchButtonActionPerformed(evt);
            }
        });

        deleteButton.setText("Delete");
        deleteButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deleteButtonActionPerformed(evt);
            }
        });

        clearButton.setText("Clear");
        clearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearButtonActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel1.setText("Paper ID");

        Title_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Title_.setText("Participant ID");

        Abstract_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Abstract_.setText("Comments");

        Result_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Result_.setText("Score");

        jLabel2.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel2.setText("Reviwer ID");

        RID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                RIDActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
```

```
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(91, 91, 91)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(clearButton)
                        .addGap(83, 83, 83))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(insertButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 92, Short.MAX_VALUE)
                        .addComponent(updateButton)
                        .addGap(101, 101, 101)
                        .addComponent(searchButton)))
                .addGap(69, 69, 69))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jLabel6)
                .addGap(203, 203, 203))
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(170, 170, 170)
                        .addComponent(deleteButton))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(112, 112, 112)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.TRAILING)
                            .addComponent(jLabel1)
                            .addComponent(Title_)
                            .addComponent(Abstract_)
                            .addComponent(Result_)
                            .addComponent(jLabel2))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                            .addComponent(PaID,
javax.swing.GroupLayout.DEFAULT_SIZE, 193, Short.MAX_VALUE)
                            .addComponent(Comments)
                            .addComponent(RID)
                            .addComponent(PID,
javax.swing.GroupLayout.Alignment.TRAILING)
```

```
                            .addComponent(Score))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel6)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
, 33, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(RID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(PID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Title_)
                    .addComponent(PaID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Abstract_)
                    .addComponent(Comments, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Result_)
                    .addComponent(Score, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(insertButton)
                    .addComponent(updateButton)
                    .addComponent(searchButton))
                .addGap(18, 18, 18)
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(deleteButton)
                    .addComponent(clearButton))
                .addGap(47, 47, 47))
        );

        pack();
    }// </editor-fold>

    private void PIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void PaIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void CommentsActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void ScoreActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void insertButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "INSERT INTO Review (ReviewID, PaperID, ParticipantID, Comments, Score) VALUES (?, ?, ?, ?, ?)";
            ps = con.prepareStatement(sql);
            ps.setString(1, RID.getText());
            ps.setString(2, PID.getText());
            ps.setString(3, PaID.getText());
            ps.setString(4, Comments.getText());
            ps.setString(5, Score.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Inserted!");
        } catch (SQLException ex) {
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }
```

```
    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Prepare the SQL update statement
            String sql = "UPDATE Review SET PaperID = ?, ParticipantID = ?,
Comments = ?, Score = ? WHERE ReviewID = ?";
            PreparedStatement pstmt = con.prepareStatement(sql);

            // Set the parameters for the prepared statement
            pstmt.setString(1, PID.getText());
            pstmt.setString(2, PaID.getText());
            pstmt.setString(3, Comments.getText());
            pstmt.setString(4, Score.getText());
            pstmt.setString(5, RID.getText());

            // Execute the update statement
            int rowsUpdated = pstmt.executeUpdate();

            // Commit the transaction
            con.commit();

            // Check if the update was successful
            if (rowsUpdated > 0) {
                JOptionPane.showMessageDialog(this, "Record Updated
Successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
```

```
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "SELECT * FROM Review WHERE ReviewID = '" + RID.getText()
+ "'";
            st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            rs = st.executeQuery(sql);

            if (rs.next()) {
                RID.setText(rs.getString(1));
                PID.setText(rs.getString(2));
                PaID.setText(rs.getString(3));
                Comments.setText(rs.getString(4));
                Score.setText(rs.getString(5));
                JOptionPane.showMessageDialog(this, "Record Found!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Delete the record from the Review table
            String deleteReviewSql = "DELETE FROM Review WHERE ReviewID = '" +
RID.getText() + "'";
            st = con.createStatement();
            int rowsDeleted = st.executeUpdate(deleteReviewSql);

            // Commit the transaction
            con.commit();

            if (rowsDeleted > 0) {
```

```
                JOptionPane.showMessageDialog(this, "Record Deleted
Successfully!");
                // Optionally, clear the fields
                RID.setText("");
                PID.setText("");
                PaID.setText("");
                Comments.setText("");
                Score.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        RID.setText("");
        PID.setText("");
        PaID.setText("");
        Comments.setText("");
        Score.setText("");
    }

    private void RIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
```

```
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new review().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel Abstract_;
    private javax.swing.JTextField Comments;
    private javax.swing.JTextField PID;
    private javax.swing.JTextField PaID;
    private javax.swing.JTextField RID;
    private javax.swing.JLabel Result_;
```

```
    private javax.swing.JTextField Score;
    private javax.swing.JLabel Title_;
    private javax.swing.JButton clearButton;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton insertButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton searchButton;
    private javax.swing.JButton updateButton;
    // End of variables declaration
}
```

Selecting the Table:

```
Enter the table to be viewed:
1. Participant
2. Review
3. Paper
```

Code for the Participant Table:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package com.oracle.paper_journal;
import javax.swing.*;
import java.sql.*;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author rohit
 */
public class participant extends javax.swing.JFrame {
    Connection con;
```

```
    Statement st;
    PreparedStatement ps;
    ResultSet rs;
    /**
     * Creates new form participant
     */
    public participant() {
        initComponents();
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            JOptionPane.showMessageDialog(this, "Driver Loaded!");

            try {
                con =
DriverManager.getConnection("jdbc:oracle:thin:@Rohith:1521:orcla", "scott",
"tiger");
                JOptionPane.showMessageDialog(this, "Connected to Oracle
database!");
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
                JOptionPane.showMessageDialog(this, ex.getMessage());
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        clearButton = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        ParID = new javax.swing.JTextField();
        Title_ = new javax.swing.JLabel();
        Abstract_ = new javax.swing.JLabel();
        Result_ = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        Affiliation = new javax.swing.JTextField();
```

```
        Name = new javax.swing.JTextField();
        Email = new javax.swing.JTextField();
        Type = new javax.swing.JTextField();
        insertButton = new javax.swing.JButton();
        updateButton = new javax.swing.JButton();
        searchButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        clearButton.setText("Clear");
        clearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearButtonActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel2.setText("Participant ID");

        jLabel1.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel1.setText("Affiliation");

        ParID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ParIDActionPerformed(evt);
            }
        });

        Title_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Title_.setText("Name");

        Abstract_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Abstract_.setText("Email");

        Result_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Result_.setText("Type");

        jLabel6.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
        jLabel6.setText("Participant Table");

        Affiliation.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                AffiliationActionPerformed(evt);
            }
        });

        Name.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                NameActionPerformed(evt);
            }
        });

        Email.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                EmailActionPerformed(evt);
            }
        });

        Type.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TypeActionPerformed(evt);
            }
        });

        insertButton.setText("Insert");
        insertButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insertButtonActionPerformed(evt);
            }
        });

        updateButton.setText("Update");
        updateButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateButtonActionPerformed(evt);
            }
        });

        searchButton.setText("Search");
        searchButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                searchButtonActionPerformed(evt);
            }
        });

        deleteButton.setText("Delete");
        deleteButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deleteButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
```

```
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(91, 91, 91)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(clearButton)
                        .addGap(83, 83, 83))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(insertButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 92, Short.MAX_VALUE)
                        .addComponent(updateButton)
                        .addGap(101, 101, 101)
                        .addComponent(searchButton)))
                .addGap(69, 69, 69))
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(170, 170, 170)
                        .addComponent(deleteButton))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(112, 112, 112)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.TRAILING)
                            .addComponent(jLabel1)
                            .addComponent(Title_)
                            .addComponent(Abstract_)
                            .addComponent(Result_)
                            .addComponent(jLabel2))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                            .addComponent(Name,
javax.swing.GroupLayout.DEFAULT_SIZE, 193, Short.MAX_VALUE)
                            .addComponent(Email)
                            .addComponent(ParID)
                            .addComponent(Affiliation,
javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(Type))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jLabel6)
                .addGap(169, 169, 169))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel6)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
, 33, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(ParID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(Affiliation,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Title_)
                    .addComponent(Name, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Abstract_)
                    .addComponent(Email, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Result_)
                    .addComponent(Type, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(insertButton)
                    .addComponent(updateButton)
```

```
                    .addComponent(searchButton))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(deleteButton)
                    .addComponent(clearButton))
                .addGap(47, 47, 47))
        );

        pack();
    }// </editor-fold>

    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        ParID.setText("");
        Affiliation.setText("");
        Name.setText("");
        Email.setText("");
        Type.setText("");
    }

    private void ParIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void AffiliationActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void NameActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void EmailActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void TypeActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void insertButtonActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        try {
            String sql = "INSERT INTO Participant (ID, Affiliation, Name, Email,
Type) VALUES (?, ?, ?, ?, ?)";
            ps = con.prepareStatement(sql);
            ps.setString(1, ParID.getText());
            ps.setString(2, Affiliation.getText());
            ps.setString(3, Name.getText());
            ps.setString(4, Email.getText());
            ps.setString(5, Type.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Inserted!");
        } catch (SQLException ex) {
            Logger.getLogger(participant.class.getName()).log(Level.SEVERE, null,
ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Prepare the SQL update statement
            String sql = "UPDATE Participant SET Affiliation = ?, Name = ?, Email =
?, Type = ? WHERE ID = ?";
            PreparedStatement pstmt = con.prepareStatement(sql);

            // Set the parameters for the prepared statement
            pstmt.setString(1, Affiliation.getText());
            pstmt.setString(2, Name.getText());
            pstmt.setString(3, Email.getText());
            pstmt.setString(4, Type.getText());
            pstmt.setString(5, ParID.getText());

            // Execute the update statement
            int rowsUpdated = pstmt.executeUpdate();

            // Commit the transaction
            con.commit();

            // Check if the update was successful
            if (rowsUpdated > 0) {
                JOptionPane.showMessageDialog(this, "Record Updated
Successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
```

```
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(participant.class.getName()).log(Level.SEVERE,
null, rollbackEx);
            }
            Logger.getLogger(participant.class.getName()).log(Level.SEVERE, null,
ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(participant.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
    }

    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "SELECT * FROM Participant WHERE ID = '" + ParID.getText()
+ "'";
            st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            rs = st.executeQuery(sql);

            if (rs.next()) {
                ParID.setText(rs.getString(1));
                Affiliation.setText(rs.getString(2));
                Name.setText(rs.getString(3));
                Email.setText(rs.getString(4));
                Type.setText(rs.getString(5));
                JOptionPane.showMessageDialog(this, "Record Found!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            Logger.getLogger(participant.class.getName()).log(Level.SEVERE, null,
ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }
```

```
    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Delete the record from the Participant table
            String deleteParticipantSql = "DELETE FROM Participant WHERE ID = '" +
ParID.getText() + "'";
            st = con.createStatement();
            int rowsDeleted = st.executeUpdate(deleteParticipantSql);

            // Commit the transaction
            con.commit();

            if (rowsDeleted > 0) {
                JOptionPane.showMessageDialog(this, "Record Deleted
Successfully!");
                // Optionally, clear the fields
                ParID.setText("");
                Affiliation.setText("");
                Name.setText("");
                Email.setText("");
                Type.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(participant.class.getName()).log(Level.SEVERE,
null, rollbackEx);
            }
            Logger.getLogger(participant.class.getName()).log(Level.SEVERE, null,
ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(participant.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
```

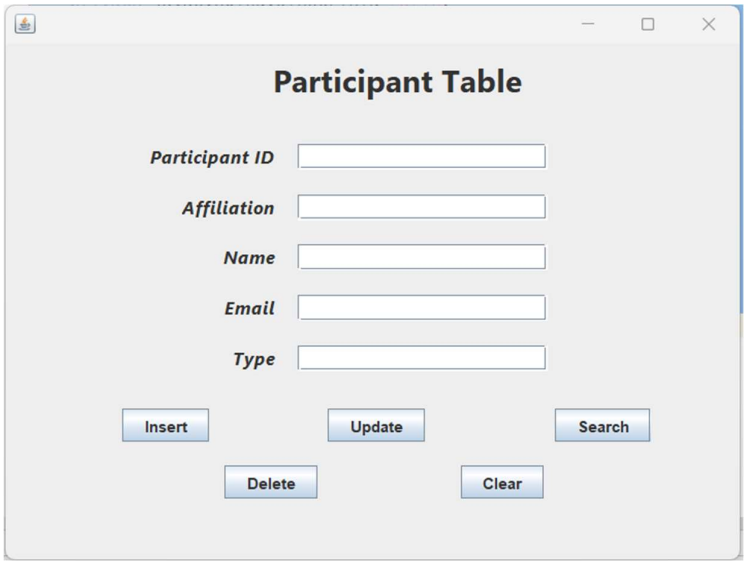
```
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(participant.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(participant.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(participant.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(participant.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new participant().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel Abstract_;
    private javax.swing.JTextField Affiliation;
```

```
    private javax.swing.JTextField Email;
    private javax.swing.JTextField Name;
    private javax.swing.JTextField ParID;
    private javax.swing.JLabel Result_;
    private javax.swing.JLabel Title_;
    private javax.swing.JTextField Type;
    private javax.swing.JButton clearButton;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton insertButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton searchButton;
    private javax.swing.JButton updateButton;
    // End of variables declaration
}
```

Selecting and Working on the Participant Table:

```
Enter the table to be viewed:
1. Participant
2. Review
3. Paper
1
```

Basic UI

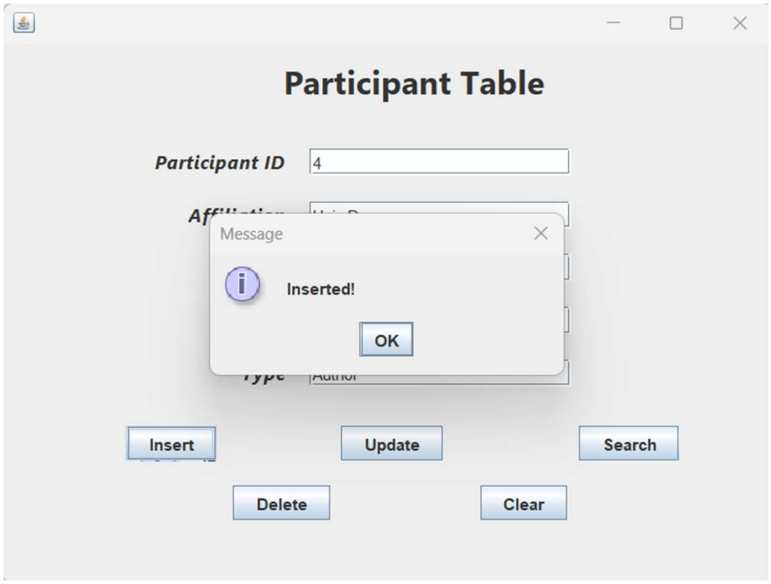


Inserting:

Before Insertion:

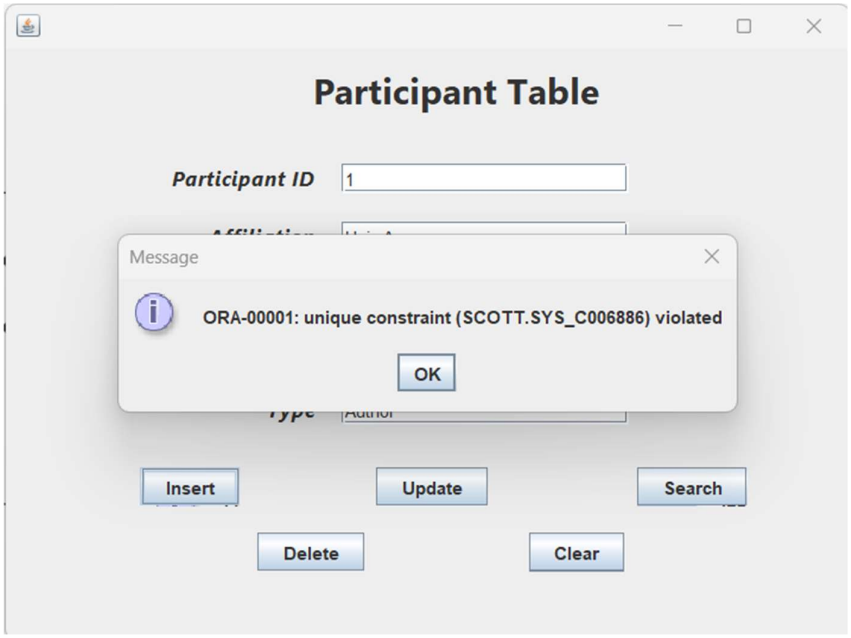
```
        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         1 Univ A               Alice                alice@ex.com
Author

         2 Univ B               Bob                  bob@ex.com
Reviewer

         3 Univ C               Charlie              charlie@ex.com
Attendee
```

After Successful Insertion:




```
        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         4 Univ D               Daisy                daisy@ex.com
Author

         1 Univ A               Alice                alice@ex.com
Author

         2 Univ B               Bob                  bob@ex.com
Reviewer


        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         3 Univ C               Charlie              charlie@ex.com
Attendee
```

Incorrect Insertion:

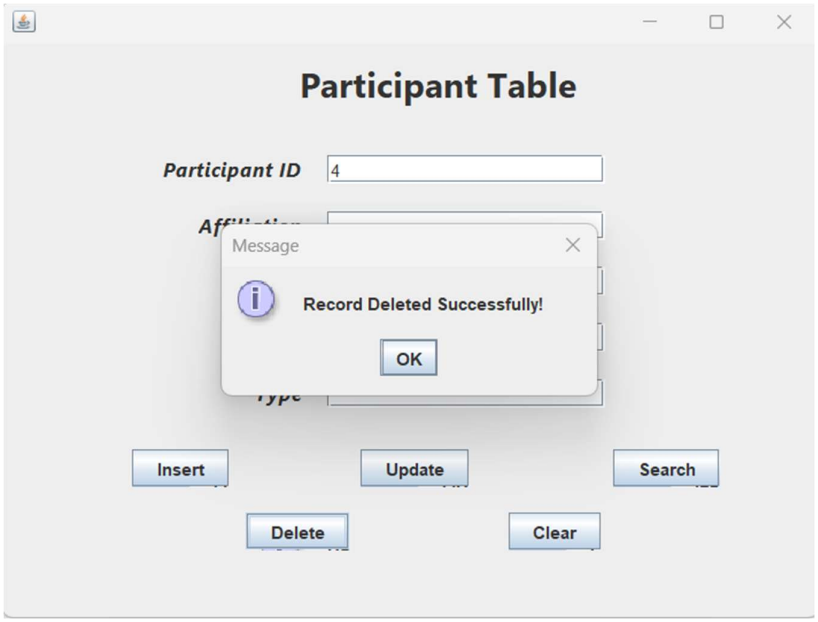


Deleting:

Before Deleting a Record:

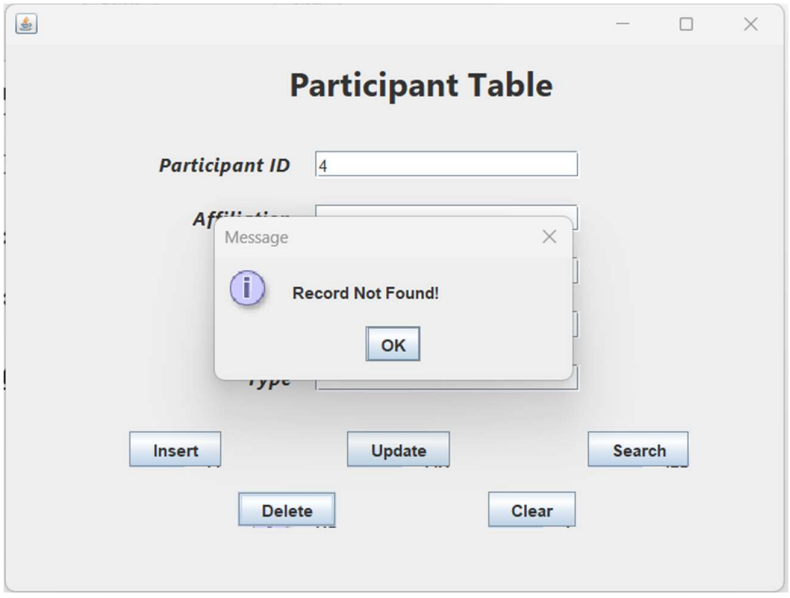
```
        ID AFFILIATION          NAME                EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         4 Univ D               Daisy               daisy@ex.com
Author

         1 Univ A               Alice               alice@ex.com
Author

         2 Univ B               Bob                 bob@ex.com
Reviewer


        ID AFFILIATION          NAME                EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         3 Univ C               Charlie             charlie@ex.com
Attendee
```

After Deleting a Record Successfully:




```
        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         1 Univ A               Alice                alice@ex.com
Author

         2 Univ B               Bob                  bob@ex.com
Reviewer

         3 Univ C               Charlie              charlie@ex.com
Attendee
```

Incorrect Deletion:

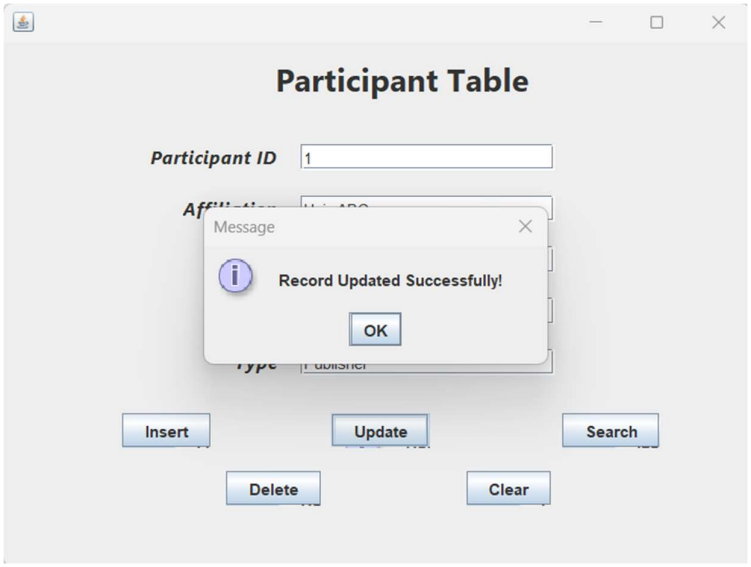


Updating:

Before Updating:

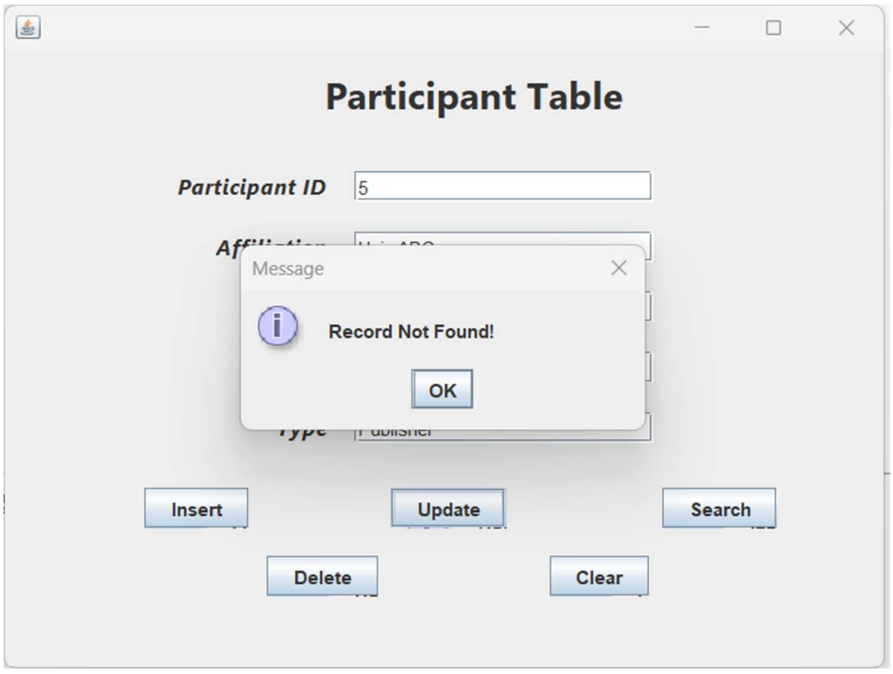
```
        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         1 Univ A               Alice                alice@ex.com
Author

         2 Univ B               Bob                  bob@ex.com
Reviewer

         3 Univ C               Charlie              charlie@ex.com
Attendee
```

After Successful Update:



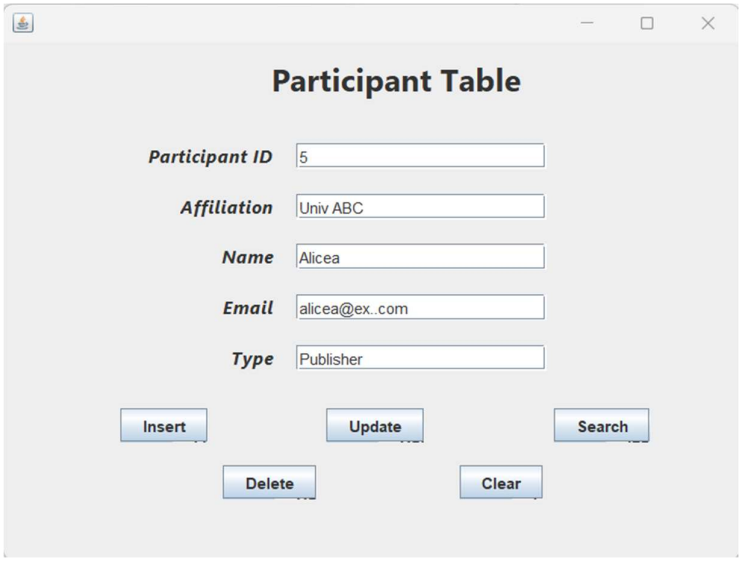

```
        ID AFFILIATION          NAME                 EMAIL
---------- -------------------- -------------------- --------------------
TYPE
--------------------
         1 Univ ABC             Alicea               alicea@ex..com
Publisher

         2 Univ B               Bob                  bob@ex.com
Reviewer

         3 Univ C               Charlie              charlie@ex.com
Attendee
```

Updating incorrectly:

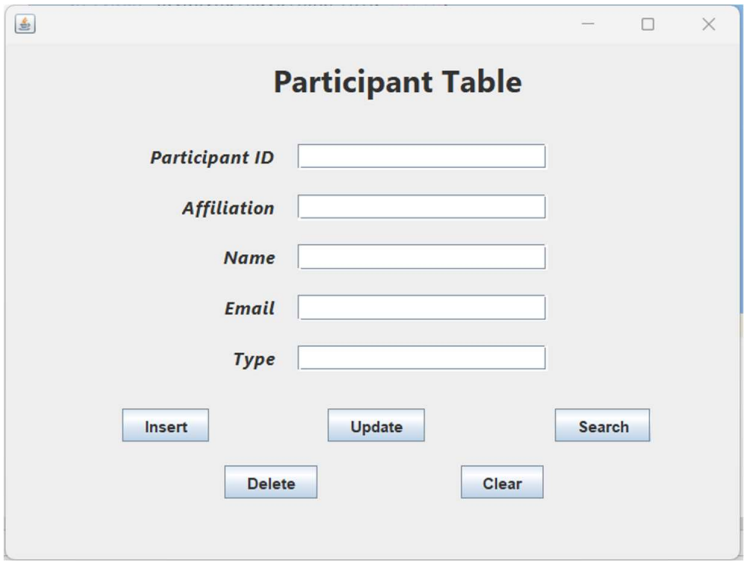


Clear:

Before:

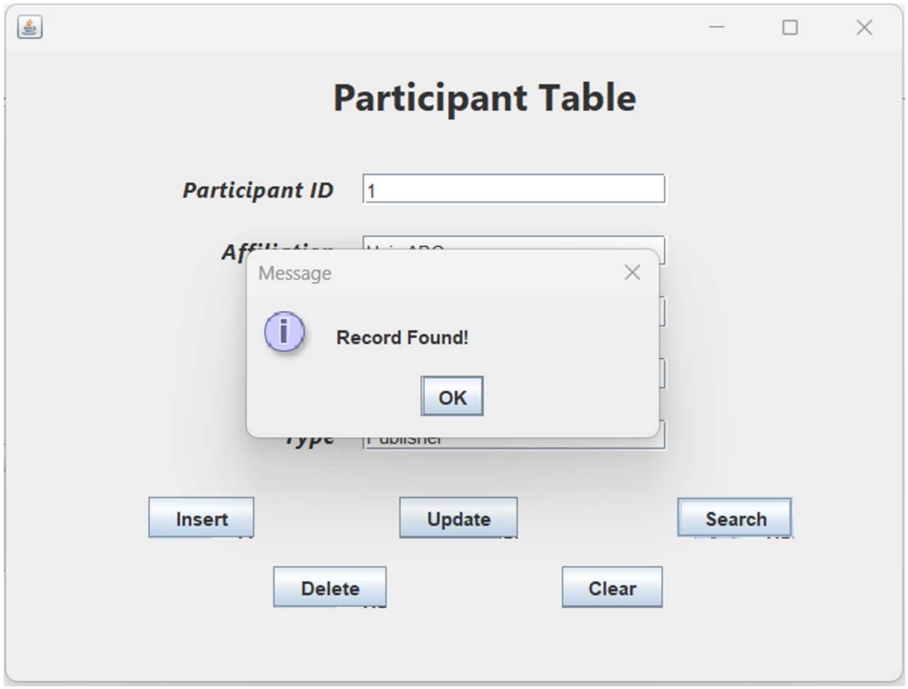


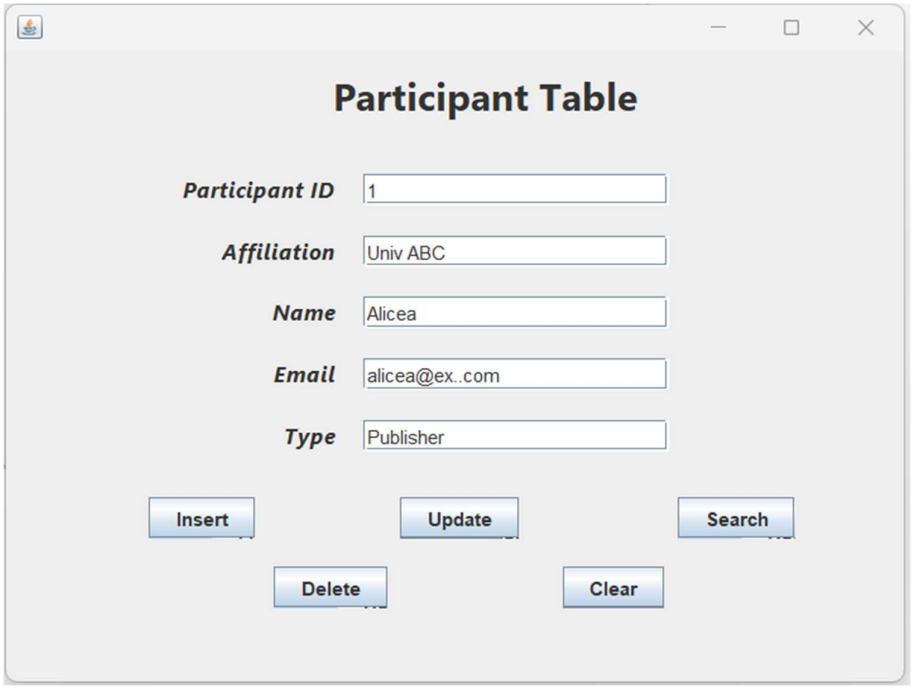
After:



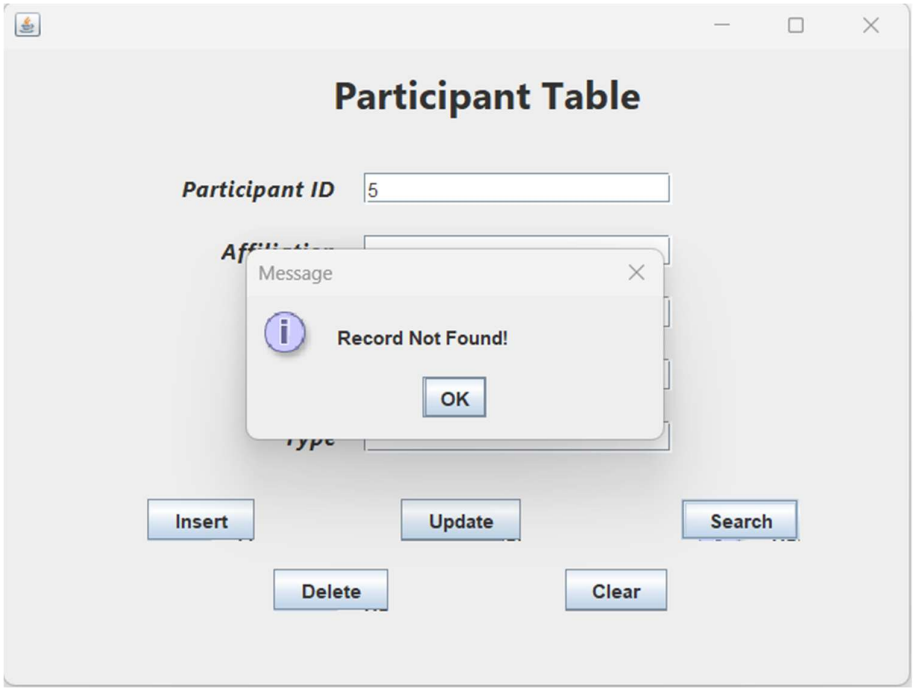
Search:

If Record is Found:





If Record is Not Found:



Code for the Review Table:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package com.oracle.paper_journal;
import javax.swing.*;
import java.sql.*;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author rohit
 */
public class review extends javax.swing.JFrame {
    Connection con;
    Statement st;
    PreparedStatement ps;
    ResultSet rs;

    /**
     * Creates new form review
     */
    public review() {
        initComponents();
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            JOptionPane.showMessageDialog(this, "Driver Loaded!");

            try {
                con =
DriverManager.getConnection("jdbc:oracle:thin:@Rohith:1521:orcla", "scott",
"tiger");
                JOptionPane.showMessageDialog(this, "Connected to Oracle
database!");
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
                JOptionPane.showMessageDialog(this, ex.getMessage());
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
```

```
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jLabel6 = new javax.swing.JLabel();
        PID = new javax.swing.JTextField();
        PaID = new javax.swing.JTextField();
        Comments = new javax.swing.JTextField();
        Score = new javax.swing.JTextField();
        insertButton = new javax.swing.JButton();
        updateButton = new javax.swing.JButton();
        searchButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();
        clearButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        Title_ = new javax.swing.JLabel();
        Abstract_ = new javax.swing.JLabel();
        Result_ = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        RID = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel6.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
        jLabel6.setText("Review Table");

        PID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PIDActionPerformed(evt);
            }
        });

        PaID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PaIDActionPerformed(evt);
            }
        });

        Comments.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            CommentsActionPerformed(evt);
        }
    });

    Score.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ScoreActionPerformed(evt);
        }
    });

    insertButton.setText("Insert");
    insertButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            insertButtonActionPerformed(evt);
        }
    });

    updateButton.setText("Update");
    updateButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            updateButtonActionPerformed(evt);
        }
    });

    searchButton.setText("Search");
    searchButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            searchButtonActionPerformed(evt);
        }
    });

    deleteButton.setText("Delete");
    deleteButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            deleteButtonActionPerformed(evt);
        }
    });

    clearButton.setText("Clear");
    clearButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            clearButtonActionPerformed(evt);
        }
    });

    jLabel1.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
    jLabel1.setText("Paper ID");
```

```
        Title_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Title_.setText("Participant ID");

        Abstract_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Abstract_.setText("Comments");

        Result_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Result_.setText("Score");

        jLabel2.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel2.setText("Reviwer ID");

        RID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                RIDActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(91, 91, 91)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(clearButton)
                        .addGap(83, 83, 83))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(insertButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 92, Short.MAX_VALUE)
                        .addComponent(updateButton)
                        .addGap(101, 101, 101)
                        .addComponent(searchButton)))
                .addGap(69, 69, 69))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jLabel6)
                .addGap(203, 203, 203))
            .addGroup(layout.createSequentialGroup()
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(170, 170, 170)
                        .addComponent(deleteButton))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(112, 112, 112)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.TRAILING)
                            .addComponent(jLabel1)
                            .addComponent(Title_)
                            .addComponent(Abstract_)
                            .addComponent(Result_)
                            .addComponent(jLabel2))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                            .addComponent(PaID,
javax.swing.GroupLayout.DEFAULT_SIZE, 193, Short.MAX_VALUE)
                            .addComponent(Comments)
                            .addComponent(RID)
                            .addComponent(PID,
javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(Score))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel6)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
, 33, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(RID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(PID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Title_)
                    .addComponent(PaID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Abstract_)
                    .addComponent(Comments, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Result_)
                    .addComponent(Score, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(insertButton)
                    .addComponent(updateButton)
                    .addComponent(searchButton))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(deleteButton)
                    .addComponent(clearButton))
                .addGap(47, 47, 47))
        );

        pack();
    }// </editor-fold>

    private void PIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void PaIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void CommentsActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }
```

```
    private void ScoreActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void insertButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "INSERT INTO Review (ReviewID, PaperID, ParticipantID,
Comments, Score) VALUES (?, ?, ?, ?, ?)";
            ps = con.prepareStatement(sql);
            ps.setString(1, RID.getText());
            ps.setString(2, PID.getText());
            ps.setString(3, PaID.getText());
            ps.setString(4, Comments.getText());
            ps.setString(5, Score.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Inserted!");
        } catch (SQLException ex) {
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Prepare the SQL update statement
            String sql = "UPDATE Review SET PaperID = ?, ParticipantID = ?,
Comments = ?, Score = ? WHERE ReviewID = ?";
            PreparedStatement pstmt = con.prepareStatement(sql);

            // Set the parameters for the prepared statement
            pstmt.setString(1, PID.getText());
            pstmt.setString(2, PaID.getText());
            pstmt.setString(3, Comments.getText());
            pstmt.setString(4, Score.getText());
            pstmt.setString(5, RID.getText());

            // Execute the update statement
            int rowsUpdated = pstmt.executeUpdate();

            // Commit the transaction
            con.commit();
```

```
            // Check if the update was successful
            if (rowsUpdated > 0) {
                JOptionPane.showMessageDialog(this, "Record Updated
Successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "SELECT * FROM Review WHERE ReviewID = '" + RID.getText()
+ "'";
            st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            rs = st.executeQuery(sql);

            if (rs.next()) {
                RID.setText(rs.getString(1));
                PID.setText(rs.getString(2));
                PaID.setText(rs.getString(3));
                Comments.setText(rs.getString(4));
                Score.setText(rs.getString(5));
                JOptionPane.showMessageDialog(this, "Record Found!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
```

```
        } catch (SQLException ex) {
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Delete the record from the Review table
            String deleteReviewSql = "DELETE FROM Review WHERE ReviewID = '" +
RID.getText() + "'";
            st = con.createStatement();
            int rowsDeleted = st.executeUpdate(deleteReviewSql);

            // Commit the transaction
            con.commit();

            if (rowsDeleted > 0) {
                JOptionPane.showMessageDialog(this, "Record Deleted
Successfully!");
                // Optionally, clear the fields
                RID.setText("");
                PID.setText("");
                PaID.setText("");
                Comments.setText("");
                Score.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(review.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
```

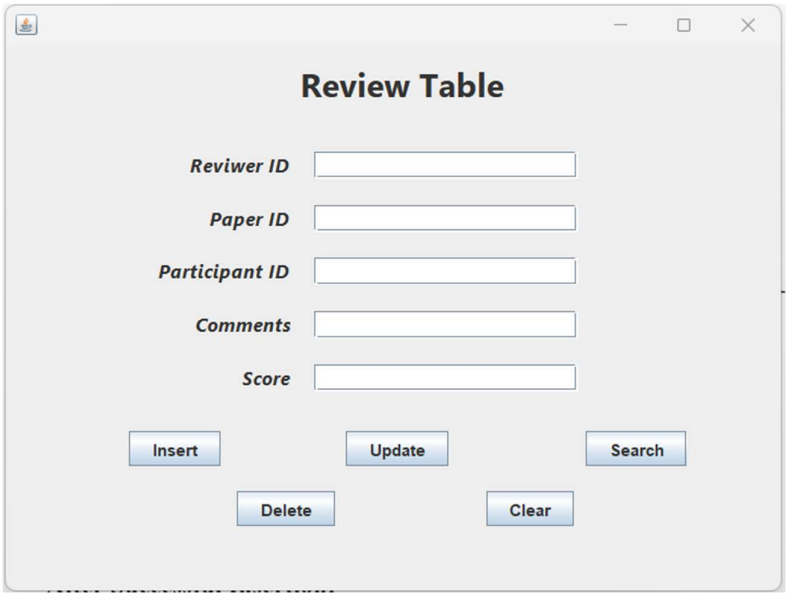
```
                Logger.getLogger(review.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        RID.setText("");
        PID.setText("");
        PaID.setText("");
        Comments.setText("");
        Score.setText("");
    }

    private void RIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(review.class.getName()).log(java.uti
l.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new review().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel Abstract_;
    private javax.swing.JTextField Comments;
    private javax.swing.JTextField PID;
    private javax.swing.JTextField PaID;
    private javax.swing.JTextField RID;
    private javax.swing.JLabel Result_;
    private javax.swing.JTextField Score;
    private javax.swing.JLabel Title_;
    private javax.swing.JButton clearButton;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton insertButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton searchButton;
    private javax.swing.JButton updateButton;
    // End of variables declaration
}
```

Selecting and Working on the Review Table:

```
Enter the table to be viewed:
1. Participant
2. Review
3. Paper
2
```

Basic UI

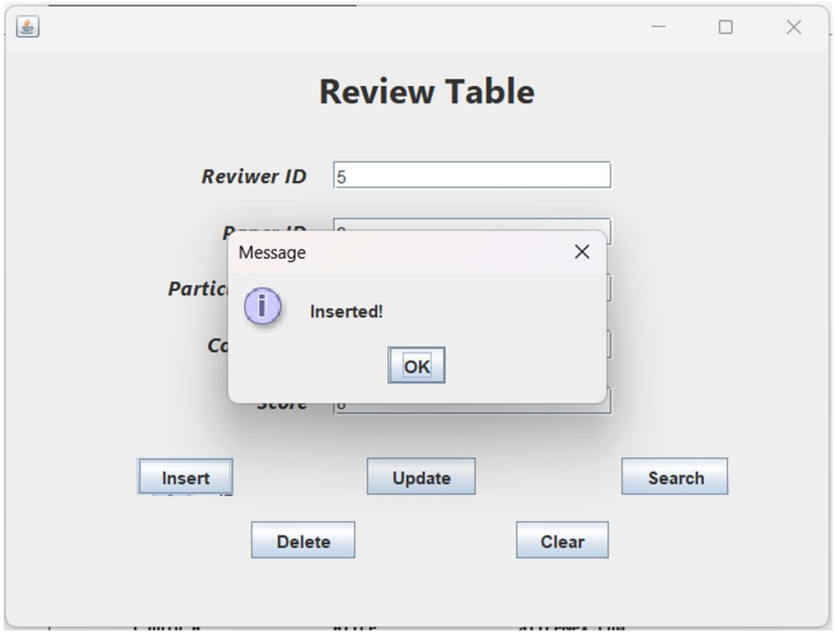


Inserting:

Before Insertion:

REVIEWID	PAPERID	PARTICIPANTID	COMMENTS	SCORE
1	1	2	Excellent.	9
2	2	3	Good.	8
3	3	1	Splendid.	7
4	1	3	Insightful.	8
5	2	1	Great.	6

After Successful Insertion:



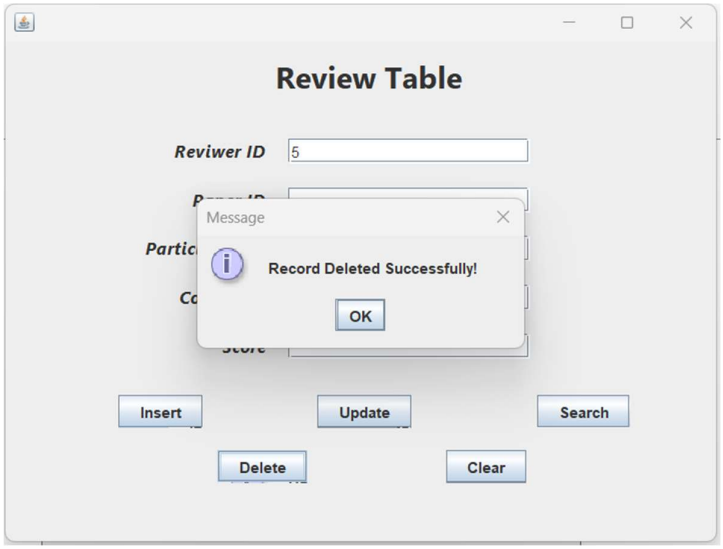
REVIEWID	PAPERID	PARTICIPANTID	COMMENTS	SCORE
1	1	2	Excellent.	9
2	2	3	Good.	8
3	3	1	Splendid.	7
4	1	3	Insightful.	8
5	2	1	Great.	6
5	2	2	Great!	8

Deleting:

Before Deleting a Record:

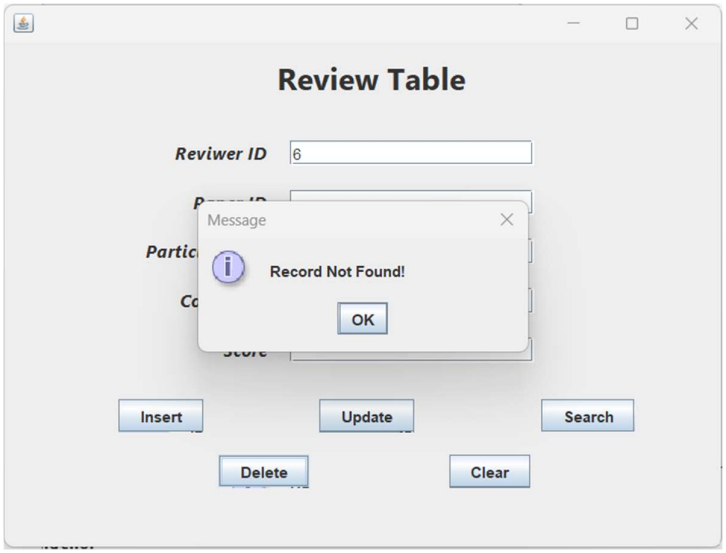
REVIEWID	PAPERID	PARTICIPANTID	COMMENTS	SCORE
1	1	2	Excellent.	9
2	2	3	Good.	8
3	3	1	Splendid.	7
4	1	3	Insightful.	8
5	2	1	Great.	6
5	2	2	Great!	8

After Deleting a Record Successfully:




```
SQL> select * from review;

  REVIEWID    PAPERID PARTICIPANTID COMMENTS                    SCORE
---------- ---------- ------------- -------------------- ----------
         1          1             2 Excellent.                    9
         2          2             3 Good.                         8
         3          3             1 Splendid.                     7
         4          1             3 Insightful.                   8
```

Incorrect Deletion:

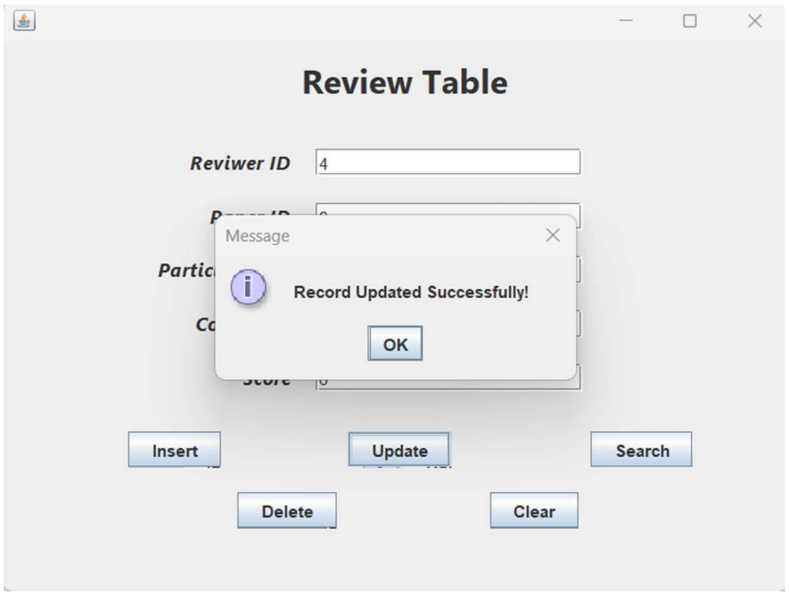


Updating:

Before Updating:

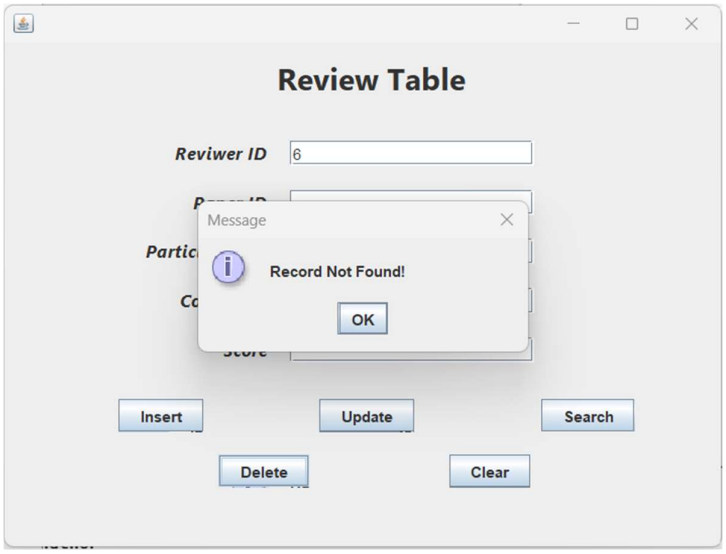
```
SQL> select * from review;

  REVIEWID    PAPERID PARTICIPANTID COMMENTS                    SCORE
---------- ---------- ------------- -------------------- ----------
         1          1             2 Excellent.                    9
         2          2             3 Good.                         8
         3          3             1 Splendid.                     7
         4          1             3 Insightful.                   8
```

After Successful Update:




```
  REVIEWID    PAPERID PARTICIPANTID COMMENTS                    SCORE
---------- ---------- ------------- -------------------- ----------
         1          1             2 Excellent.                    9
         2          2             3 Good.                         8
         3          3             1 Splendid.                     7
         4          2             2 Decently Written              6
```

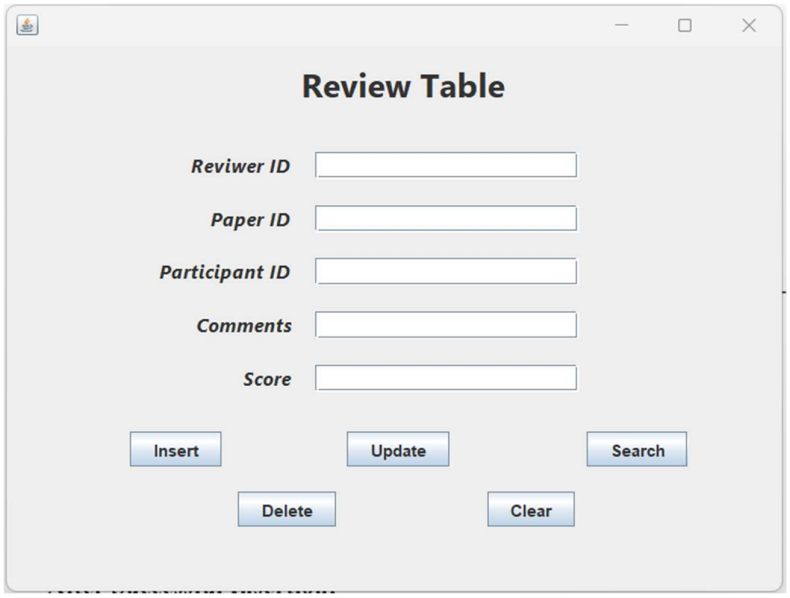
Updating incorrectly:



Clear:

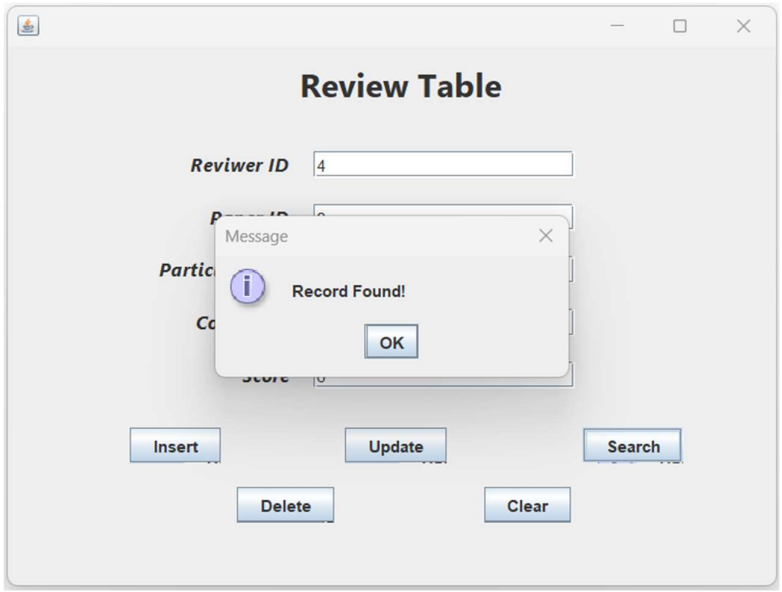
Before:

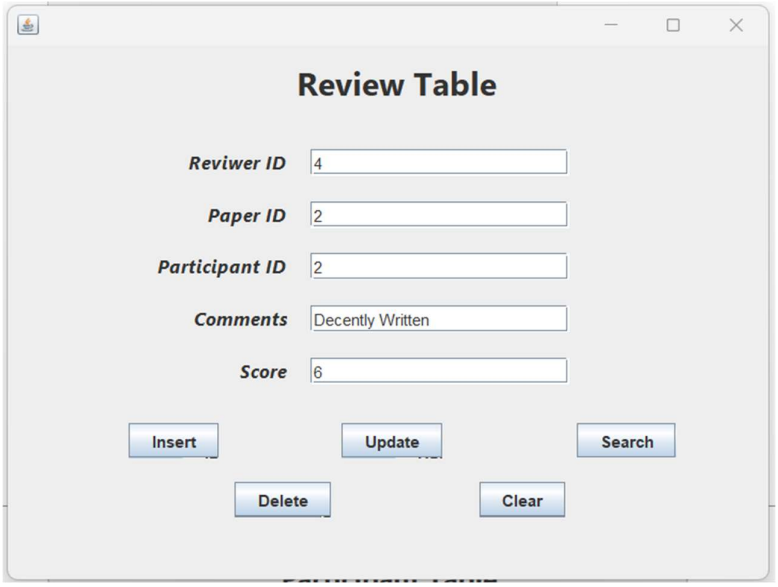
After:



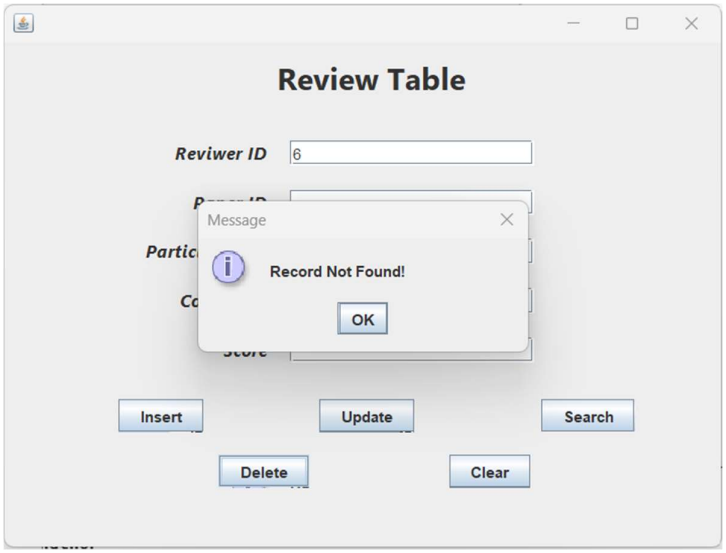
Search:

If Record is Found:





If Record is Not Found:



Code for the Review Table:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package com.oracle.paper_journal;
import javax.swing.*;
import java.sql.*;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;



/**
 *
 * @author rohit
 */


public class paper extends javax.swing.JFrame {
    Connection con;
    Statement st;
    PreparedStatement ps;
    ResultSet rs;

    /**
     * Creates new form paper
     */
    public paper() {
        initComponents();

        try {
            Class.forName("oracle.jdbc.OracleDriver");
            JOptionPane.showMessageDialog(this, "Driver Loaded!");

            try {
                con =
DriverManager.getConnection("jdbc:oracle:thin:@Rohith:1521:orcla", "scott",
"tiger");
                JOptionPane.showMessageDialog(this, "Connected to Oracle
database!");
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
```

```
                JOptionPane.showMessageDialog(this, ex.getMessage());
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }

    }



    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        insertButton = new javax.swing.JButton();
        updateButton = new javax.swing.JButton();
        searchButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();
        clearButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        Title_ = new javax.swing.JLabel();
        Abstract_ = new javax.swing.JLabel();
        Result_ = new javax.swing.JLabel();
        Task_ = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        PID = new javax.swing.JTextField();
        Title = new javax.swing.JTextField();
        Abstract = new javax.swing.JTextField();
        Task = new javax.swing.JTextField();
        TScore = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        insertButton.setText("Insert");
        insertButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insertButtonActionPerformed(evt);
            }
        });

        updateButton.setText("Update");
```

```
        updateButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                updateButtonActionPerformed(evt);
            }
        });

        searchButton.setText("Search");
        searchButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                searchButtonActionPerformed(evt);
            }
        });

        deleteButton.setText("Delete");
        deleteButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deleteButtonActionPerformed(evt);
            }
        });

        clearButton.setText("Clear");
        clearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearButtonActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        jLabel1.setText("Paper ID");

        Title_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Title_.setText("Title");

        Abstract_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Abstract_.setText("Abstract");

        Result_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Result_.setText("Task ID");

        Task_.setFont(new java.awt.Font("Segoe UI", 3, 14)); // NOI18N
        Task_.setText("Total Score");

        jLabel6.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
        jLabel6.setText("Paper Table");

        PID.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PIDActionPerformed(evt);
```

```
            }
        });

        Title.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TitleActionPerformed(evt);
            }
        });

        Abstract.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                AbstractActionPerformed(evt);
            }
        });

        Task.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TaskActionPerformed(evt);
            }
        });

        TScore.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TScoreActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(67, 67, 67)
                .addComponent(insertButton)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
, 104, Short.MAX_VALUE)
                .addComponent(updateButton)
                .addGap(98, 98, 98)
                .addComponent(searchButton)
                .addGap(84, 84, 84))
            .addGroup(layout.createSequentialGroup()
                .addGap(152, 152, 152)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                    .addComponent(deleteButton)
                    .addComponent(Task_)
                    .addComponent(jLabel1)
```

```
                    .addComponent(Title_)
                    .addComponent(Abstract_)
                    .addComponent(Result_))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel6)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                            .addComponent(PID)
                            .addComponent(Title)
                            .addComponent(Abstract)
                            .addComponent(Task)
                            .addComponent(TScore,
javax.swing.GroupLayout.PREFERRED_SIZE, 171,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(clearButton,
javax.swing.GroupLayout.Alignment.TRAILING))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel6)
                .addGap(33, 33, 33)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(PID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Title_)
                    .addComponent(Title, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Abstract_)
                    .addComponent(Abstract, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Result_)
                    .addComponent(Task, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(Task_)
                    .addComponent(TScore, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
, 50, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(insertButton)
                    .addComponent(updateButton)
                    .addComponent(searchButton))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(deleteButton)
                    .addComponent(clearButton))
                .addGap(47, 47, 47))
        );

        pack();
    }// </editor-fold>

    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "SELECT * FROM paper WHERE ID = '" + PID.getText() + "'";
            st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            rs = st.executeQuery(sql);

            if (rs.next()) {
                PID.setText(rs.getString(1));
                Title.setText(rs.getString(2));
                Abstract.setText(rs.getString(3));
                Task.setText(rs.getString(4));
                TScore.setText(rs.getString(5));
                JOptionPane.showMessageDialog(this, "Record Found!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
```

```
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }



    }

    private void PIDActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void insertButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            String sql = "INSERT INTO Paper (ID, Title, Abstract, TaskID,
TotalScore) VALUES (?, ?, ?, ?, ?)";
            ps = con.prepareStatement(sql);
            ps.setString(1, PID.getText());
            ps.setString(2, Title.getText());
            ps.setString(3, Abstract.getText());
            ps.setString(4, Task.getText());
            ps.setString(5, TScore.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Inserted!");
        } catch (SQLException ex) {
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        }
    }

    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Prepare the SQL update statement
            String sql = "UPDATE Paper SET Title = ?, Abstract = ?, TaskID = ?,
TotalScore = ? WHERE ID = ?";
            PreparedStatement pstmt = con.prepareStatement(sql);

            // Set the parameters for the prepared statement
            pstmt.setString(1, Title.getText());
            pstmt.setString(2, Abstract.getText());
            pstmt.setInt(3, Integer.parseInt(Task.getText()));
            pstmt.setInt(4, Integer.parseInt(TScore.getText()));
```

```
            pstmt.setInt(5, Integer.parseInt(PID.getText()));

            // Execute the update statement
            int rowsUpdated = pstmt.executeUpdate();

            // Commit the transaction
            con.commit();

            // Check if the update was successful
            if (rowsUpdated > 0) {
                JOptionPane.showMessageDialog(this, "Record Updated
Successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Record Not Found!");
            }
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        try {
            // Start a transaction
            con.setAutoCommit(false);

            // Delete related records in the Review table
            String deleteReviewSql = "DELETE FROM Review WHERE PaperID = '" +
PID.getText() + "'";
            st = con.createStatement();
            st.executeUpdate(deleteReviewSql);
```

```
            // Delete related records in the Author table
            String deleteAuthorSql = "DELETE FROM Author WHERE PaperID = '" +
PID.getText() + "'";
            st.executeUpdate(deleteAuthorSql);

            // Now delete the record from the Paper table
            String deletePaperSql = "DELETE FROM Paper WHERE ID = '" +
PID.getText() + "'";
            st.executeUpdate(deletePaperSql);

            // Commit the transaction
            con.commit();

            JOptionPane.showMessageDialog(this, "Record Deleted Successfully!");
            // Optionally, clear the fields
            PID.setText("");
            Title.setText("");
            Abstract.setText("");
            Task.setText("");
            TScore.setText("");
        } catch (SQLException ex) {
            try {
                // Rollback in case of an error
                con.rollback();
            } catch (SQLException rollbackEx) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
rollbackEx);
            }
            Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, ex.getMessage());
        } finally {
            try {
                // Restore the default autocommit mode
                con.setAutoCommit(true);
            } catch (SQLException ex) {
                Logger.getLogger(paper.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        PID.setText("");
        Title.setText("");
        Abstract.setText("");
        Task.setText("");
        TScore.setText("");
```

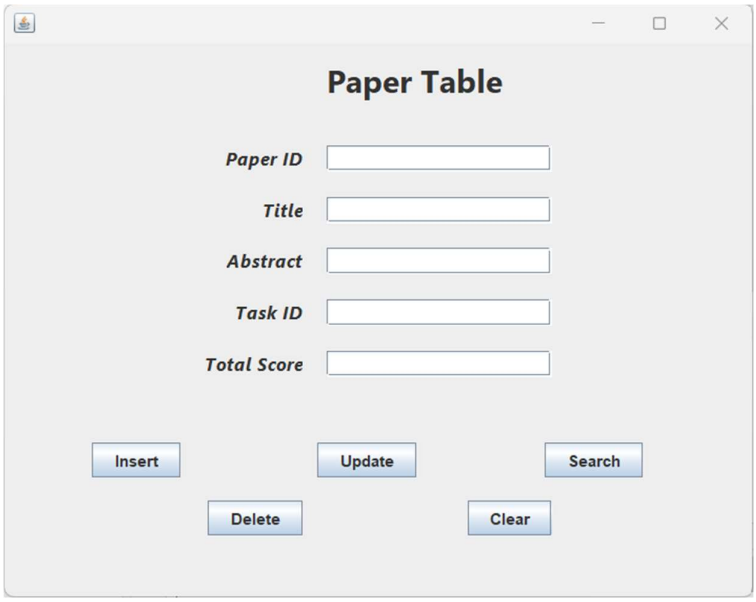
```
    }

    private void TitleActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void AbstractActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void TaskActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void TScoreActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(paper.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(paper.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
```

```
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(paper.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(paper.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new paper().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField Abstract;
    private javax.swing.JLabel Abstract_;
    private javax.swing.JTextField PID;
    private javax.swing.JLabel Result_;
    private javax.swing.JTextField TScore;
    private javax.swing.JTextField Task;
    private javax.swing.JLabel Task_;
    private javax.swing.JTextField Title;
    private javax.swing.JLabel Title_;
    private javax.swing.JButton clearButton;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton insertButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton searchButton;
    private javax.swing.JButton updateButton;
    // End of variables declaration
}
```

Selecting and Working on the Paper Table:

```
Enter the table to be viewed:
1. Participant
2. Review
3. Paper
3
```

Basic UI

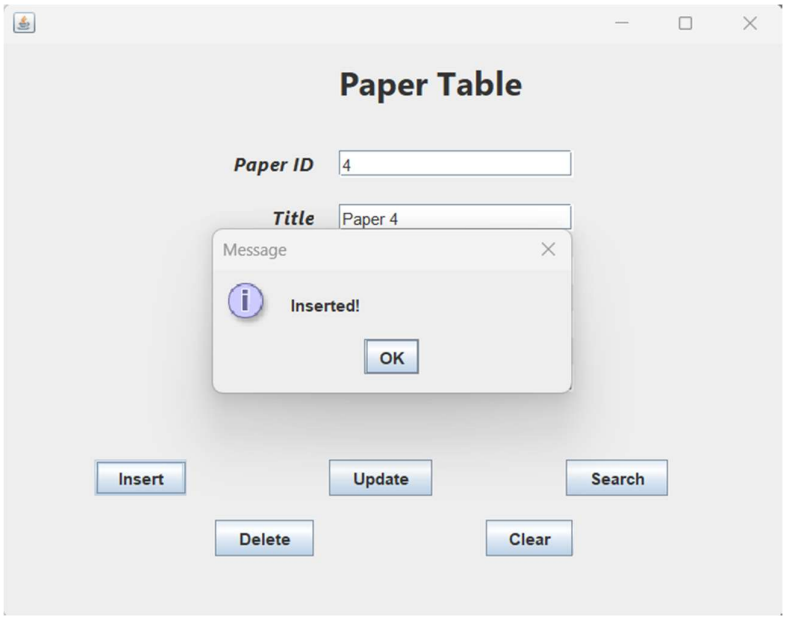


Inserting:

Before Insertion:

```
        ID TITLE                ABSTRACT                          TASKID
---------- -------------------- ------------------------------ ----------
TOTALSCORE
----------
         1 Paper 1              Abstract of Paper 1                    1
        17

         2 Paper 2              Abstract of Paper 2                    2
        14

         3 Paper 3              Abstract of Paper 3                    3
         7
```

After Successful Insertion:




```
        ID TITLE               ABSTRACT                         TASKID
---------- ------------------- ------------------------------ ----------
TOTALSCORE
----------
         1 Paper 1             Abstract of Paper 1                   1
        17

         2 Paper 2             Abstract of Paper 2                   2
        14

         3 Paper 3             Abstract of Paper 3                   3
         7


        ID TITLE               ABSTRACT                         TASKID
---------- ------------------- ------------------------------ ----------
TOTALSCORE
----------
         4 Paper 4             Abstract of Paper 4                   3
         0
```

Deleting:

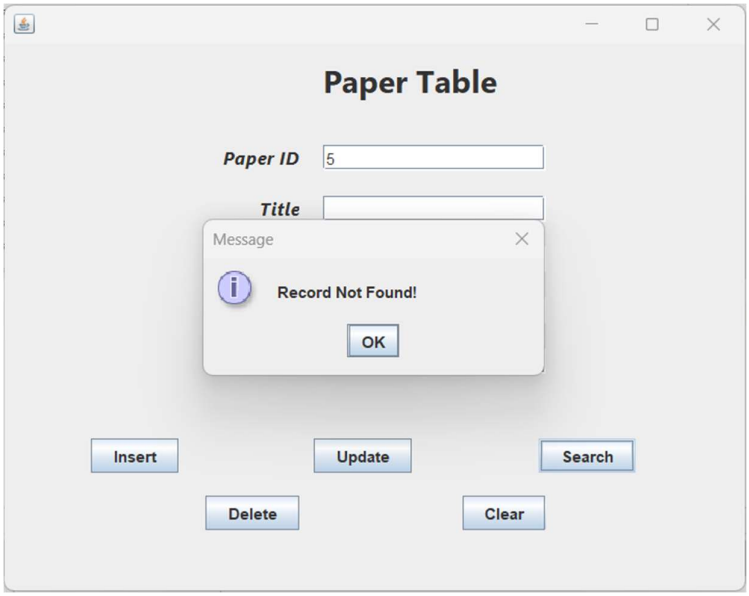
Before Deleting a Record:

```
        ID TITLE               ABSTRACT                          TASKID
---------- ------------------- ------------------------------ ----------
TOTALSCORE
----------
         1 Paper 1             Abstract of Paper 1                    1
        17

         2 Paper 2             Abstract of Paper 2                    2
        14

         3 Paper 3             Abstract of Paper 3                    3
         7


        ID TITLE               ABSTRACT                          TASKID
---------- ------------------- ------------------------------ ----------
TOTALSCORE
----------
         4 Paper 4             Abstract of Paper 4                    3
         0
```

After Deleting a Record Successfully:

```
        ID TITLE               ABSTRACT                          TASKID
---------- ------------------- ------------------------------ ----------
TOTALSCORE
----------
         1 Paper 1             Abstract of Paper 1                    1
        17

         2 Paper 2             Abstract of Paper 2                    2
        14

         3 Paper 3             Abstract of Paper 3                    3
         7
```

Incorrect Deletion:

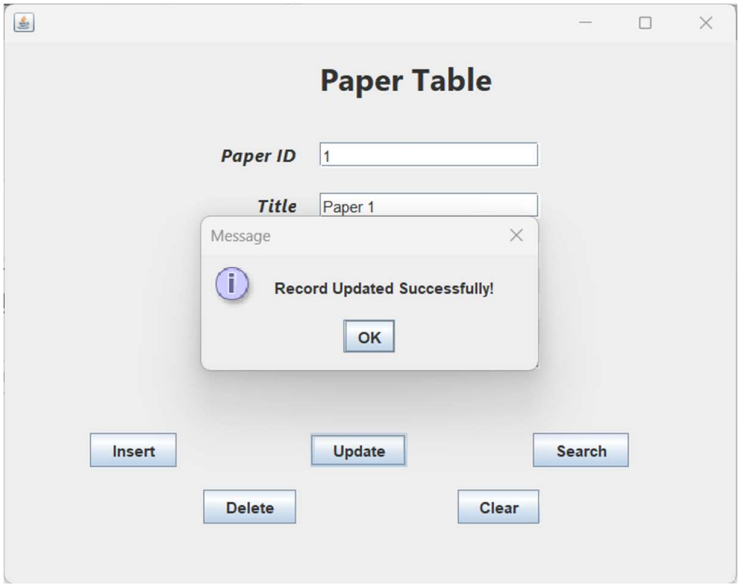


Updating:

Before Updating:

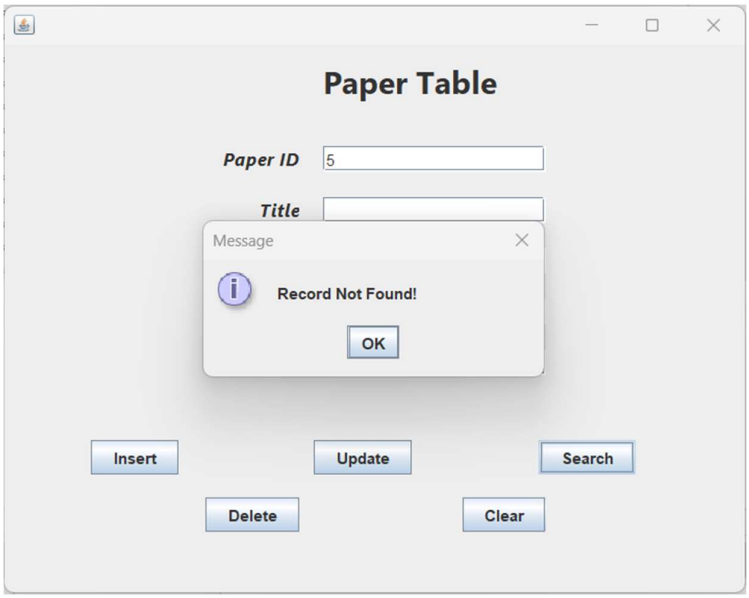
ID	TITLE	ABSTRACT	TASKID	TOTALSCORE
1	Paper 1	Abstract of Paper 1	1	17
2	Paper 2	Abstract of Paper 2	2	14
3	Paper 3	Abstract of Paper 3	3	7

After Successful Update:



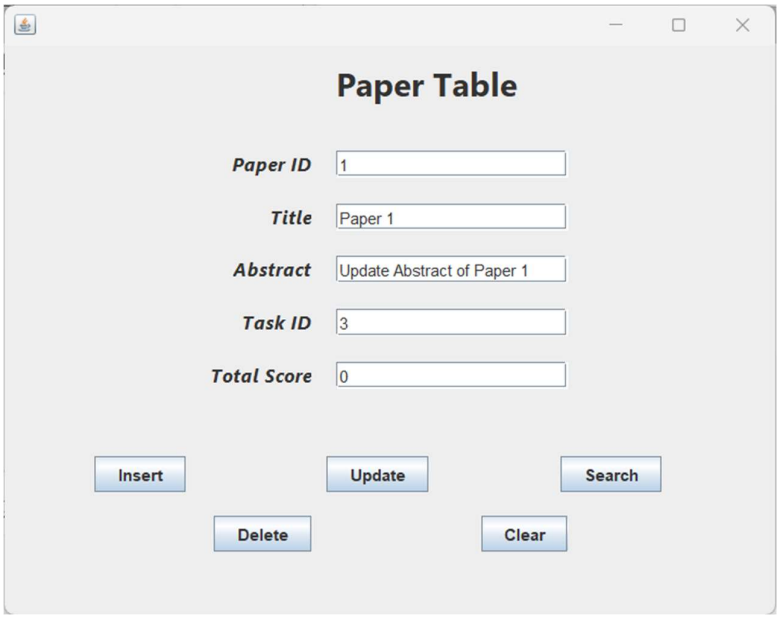
ID	TITLE	ABSTRACT	TASKID	TOTALSCORE
1	Paper 1	Update Abstract of Paper 1	3	0
2	Paper 2	Abstract of Paper 2	2	8
3	Paper 3	Abstract of Paper 3	3	7

Updating incorrectly:

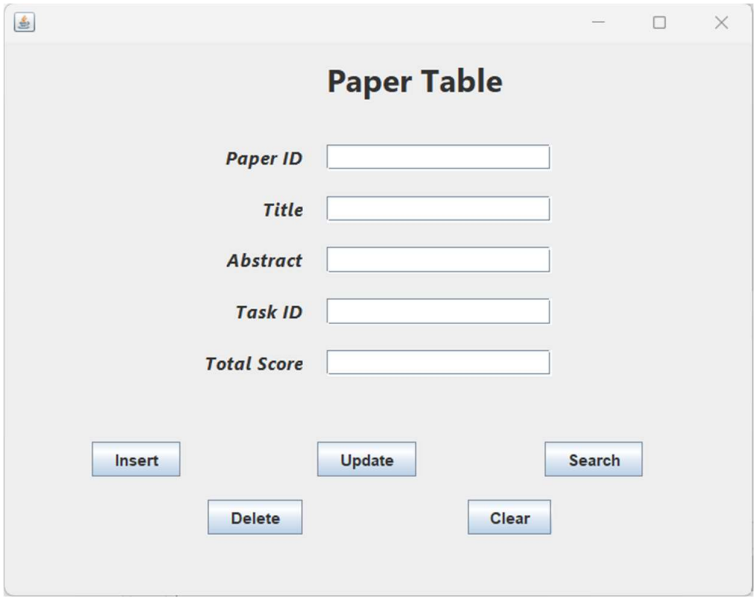


Clear:

Before:

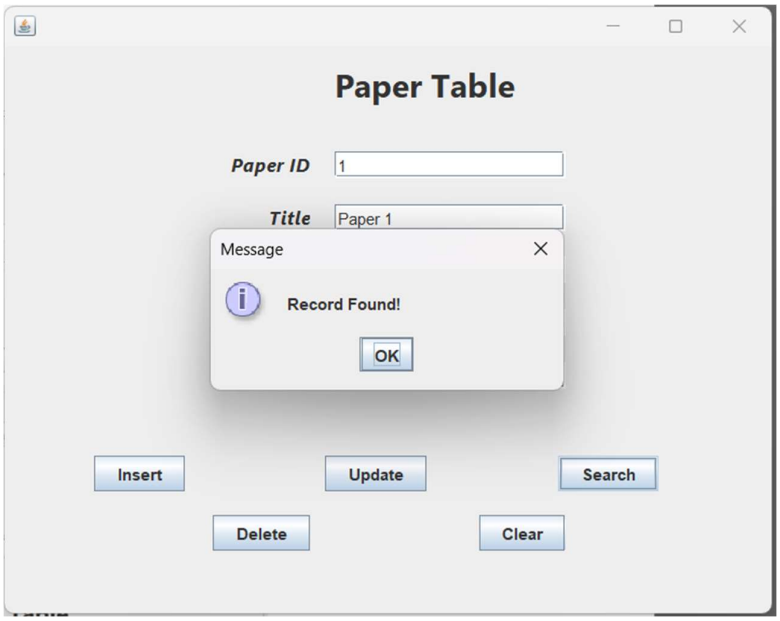


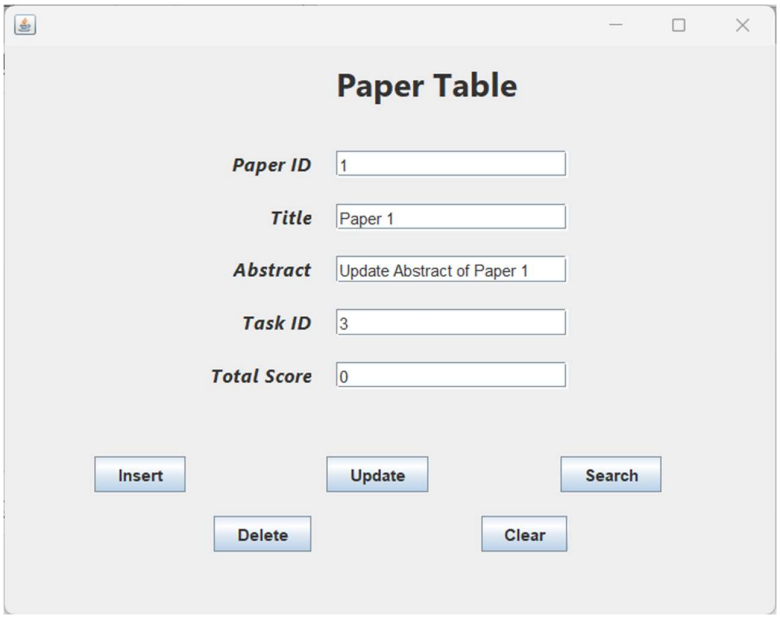
After:



Search:

If Record is Found:





If Record is Not Found: