Project Description

Our final project is an application that allows users to log and manage their character builds for the video game *Honkai: Star Rail*. Each build includes information about the selected character, light cone, and relics. Users can view detailed information regarding the different relations, such as character stats and rarities. The application also provides aggregated summary statistics to give users insights into their overall build collection. Ultimately, the app helps users organize and better understand their builds, as well as the game's characters, light cones, and relics in an interactive way.

Our final schema differed from the original version we submitted in Milestone 2 in a few ways:

- We changed the builds and characters and builds and light cones relationship to many-to-one (from many-to-many), since each build is now associated with a single character and a single light cone. This meant we removed the intermediate table and instead added the character ID (cid) and light cone ID (cone_id) foreign keys directly to the Builds table.
- We also updated the relics and builds relationship to reflect that builds can include multiple relics, and the same relic can appear in multiple builds. This required implementing a many-to-many relationship between builds and relics, with an associative table that includes the relationship attributes.

These changes were necessary because the purpose and functionality of the project shifted slightly as we built it. The way users would interact with builds, as well as the relationships between different relations, became clearer to us as we developed our application, so we adjusted our schema accordingly.

Final Schema

- **Characters(name: VARCHAR2(100), element: VARCHAR2(100), rarity: NUMBER, path: VARCHAR2(100))**
    - name is the Primary Key.

- **LightConeDetails(name: VARCHAR2(100), rarity: NUMBER, path: VARCHAR2(100), effect: VARCHAR2(500))**
  - name is the Primary Key
- **LightCones(cone_id: NUMBER, name: VARCHAR2(100))**
  - Primary Key: cone_id.
  - Foreign Key: name references **LightConeDetails**
- **CharacterRelations(cid: NUMBER, name: VARCHAR2(100), cone_id: NUMBER)**
  - Primary Key: cid.
  - Foreign Key: name references **Characters**
  - Foreign Key: cone_id references **LightCones**
- **Builds(bid: NUMBER, name: VARCHAR2(100), playstyle: VARCHAR2(100), cid: NUMBER, cone_id NUMBER)**
  - Primary Key: bid.
  - Candidate key: name.
  - Foreign Key: cid references **CharacterRelations**
  - Foreign Key: cone_id references **LightCones**
- **Stats(sid: NUMBER, stat_type: VARCHAR2(30), stat_value: NUMBER, cid: NUMBER)**
  - Composite Primary Key: (sid, cid).
  - Candidate key: (cid, stat_type).
  - Foreign Key: cid references **CharacterRelations**
- **Abilities(name: VARCHAR2(100), ability_type: VARCHAR2(100), ability_level: NUMBER, cid: NUMBER, description: VARCHAR2(500))**
  - Composite Primary Key: (name, cid).
  - Foreign Key: cid references **CharacterRelations**
- **Basic(name: VARCHAR2(100), cid: NUMBER, damage: VARCHAR2(100))**
  - Composite Primary Key: (name, cid).
  - Foreign Key: (name, cid) references **Abilities**
- **Skills(name: VARCHAR2(100), cid: NUMBER, duration: NUMBER)**
  - Composite Primary Key: (name, cid).
  - Foreign Key: (name, cid) references **Abilities**
- **Ultimates(name: VARCHAR2(100), cid: NUMBER, energy_cost: NUMBER)**
  - Composite Primary Key: (name, cid).
  - Foreign Key: (name, cid) references **Abilities**
- **Talents(name: VARCHAR2(100), cid: NUMBER, trigger_condition: VARCHAR2(100), effect: VARCHAR2(200))**
  - Composite Primary Key: (name, cid).

    - Foreign Key: (name, cid) references **Abilities**
- **Consumables(name: VARCHAR2(100), consumable_type: VARCHAR2(100), effect: VARCHAR2(500))**
    - name is the Primary Key.
- **MaterialDetails(name: VARCHAR2(100), material_type: VARCHAR2(100), location: VARCHAR2(100), rarity: NUMBER)**
    - name is the Primary Key.
- **Materials(mid: NUMBER, name: VARCHAR2(100))**
    - Primary Key: mid
    - Foreign Key: name references **MaterialDetails**
- **RelicSet(set_name: VARCHAR2(100), two_pb: VARCHAR2(500), four_pb: VARCHAR2(500))**
    - set_name is the Primary Key.
- **RelicDetails(name: VARCHAR2(100), relic_type: VARCHAR2(100), set_name: VARCHAR2(100))**
    - name is the Primary Key.
    - Foreign Key: set_name references **RelicSet**
- **Relics(rid: NUMBER, relic_level: NUMBER, name: VARCHAR2(100), main_stat: VARCHAR2(30), rarity: NUMBER)**
    - Primary Key: rid.
    - Foreign Key: name references **RelicDetails** on delete cascade.
- **Builds_Relics(bid: NUMBER, rid: NUMBER, rec_main: VARCHAR2(100), rec_substat: VARCHAR2(100))**
    - Composite Primary Key: (bid, rid).
    - Foreign Key: bid references **Builds**
    - Foreign Key: rid references **Relics**
- **Characters_Consumables(name: VARCHAR2(100), cid: NUMBER)**
    - Composite Primary Key: (name, cid).
    - Foreign Key: name references **Consumables**
    - Foreign Key: cid references **CharacterRelations**
- **Characters_Materials(cid: NUMBER, mid: NUMBER)**
    - Composite Primary Key: (cid, mid).
    - Foreign Key: cid references **CharacterRelations**
    - Foreign Key: mid references **Materials**
- **Abilities_Materials(cid: NUMBER, name: VARCHAR2(100), mid: NUMBER)**
    - Composite Primary Key: (cid, name, mid).
    - Foreign Key: (name, cid) references **Abilities**

  - Foreign Key: `mid` references **Materials**
- **LightCones_Materials(cone_id: NUMBER, mid: NUMBER)**
  - Composite Primary Key: (`cone_id`, `mid`).
  - Foreign Key: `cone_id` references **LightCones**
  - Foreign Key: `mid` references **Materials**
- **Relics_Materials(rid: NUMBER, mid: NUMBER)**
  - Composite Primary Key: (`rid`, `mid`).
  - Foreign Key: `rid` references **Relics**
  - Foreign Key: `mid` references **Materials**

SQL Queries

1. Insert

```
INSERT INTO Builds (bid, name, playstyle, cid, cone_id)
VALUES (build_seq.nextval, :name, :playstyle, :cid, :cone_id)
RETURNING bid INTO :bid_out

INSERT INTO Builds_Relics (bid, rid) VALUES (:bid, :rid)
```

- src/appService.js lines 280-282, 300
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L280C14-L282C41
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L300

2. Update

```
UPDATE Builds
SET name = :newName,
      playstyle = :newPlaystyle,
      cid = :newCid,
      cone_id = :newCone_id
WHERE bid = :bid
```

- src/appService.js lines 323-325:
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L323C14-L325C30

3. Delete

```
DELETE FROM Builds WHERE bid = :bid
```

- src/appService.js line 340
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L340

4. Selection

```
SELECT * FROM Characters WHERE ${whereClauses.join(' ')}
```

- src/appService.js lines 355-389
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L355C5-L389C1

5. Projection

```
SELECT ${columns} FROM LightConeDetails
```

- src/appService.js line 235
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L235

6. Join

```
SELECT md.name
FROM MaterialDetails md, Characters_Materials cm, Materials m
```

```
WHERE m.mid = cm.mid AND m.name = md.name AND cm.cid = (SELECT cid
                                                                                                          FROM CharacterRelations
                                                    WHERE name=:name)
```

- src/appService.js lines 183-187
  - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L183C13-L187C86

7. Aggregation with Group By:

```
SELECT c.name AS NAME, c.element AS ELEMENT, c.rarity AS RARITY, c.path AS PATH,
       COUNT(b.bid) AS BUILD_COUNT
FROM Characters c, CharacterRelations cr, Builds b
WHERE c.name = cr.name AND cr.cid = b.cid
GROUP BY c.name, c.element, c.rarity, c.path
```

- src/appService.js lines 418-423
  - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L418C10-L423C54
- For characters with at least one build, we return the total number of builds they have by using a GROUP BY clause on character attributes and counting build IDs.

8. Aggregation with Having:

```
SELECT c.NAME, c.ELEMENT, c.RARITY, c.PATH,
            COUNT(DISTINCT a.name) AS SINGLE_TARGET_COUNT
FROM Characters c, CharacterRelations cr, Builds b, Abilities a
WHERE c.NAME = cr.NAME
           AND cr.cid = b.cid
      AND cr.cid = a.cid
      AND a.ability_type = 'Single Target'
GROUP BY c.NAME, c.ELEMENT, c.RARITY, c.PATH
HAVING COUNT(DISTINCT a.name) > 1
```

- src/appService.js lines 434-442
  - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L434C10-L442C43
- From the list of characters with at least one build, we return the number of single target abilities they have by aggregating abilities with a GROUP BY clause and filtering with a HAVING clause.

9. Nested Aggregation with Group By:

```
SELECT c.name AS NAME, c.element AS ELEMENT, c.rarity AS RARITY, c.path AS PATH,
       s.stat_value AS HP
FROM Characters c, CharacterRelations cr, Builds b, Stats s
WHERE c.name = cr.name
      AND cr.cid = b.cid
      AND cr.cid = s.cid
      AND s.stat_type = 'HP'
GROUP BY c.name, c.element, c.rarity, c.path, s.stat_value
HAVING s.stat_value > (
       SELECT AVG(s2.stat_value)
       FROM Stats s2
       WHERE s2.stat_type = 'HP'
)
```

- src/appService.js lines 453-464

    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L453C10-L464C36
- From the list of characters with at least one build, we return the list of characters whose HP exceeds the overall average HP by using a nested query in the HAVING clause for nested aggregation.

10. Division:

```
SELECT c.name AS NAME, c.element AS ELEMENT, c.rarity AS RARITY, c.path AS PATH
FROM Characters c, CharacterRelations cr
WHERE c.name = cr.name
   AND NOT EXISTS (
     SELECT lc.cone_id
    FROM LightCones lc
    WHERE NOT EXISTS (
      SELECT 1
     FROM Builds b
     WHERE b.cone_id = lc.cone_id
        AND b.cid = cr.cid
    )
   )
```

- src/appService.js lines 476-488
    - https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_f7d3s_w0t6f_x4o7u/blob/719eb4335fbdd92e02362c468c46db01c957de31/src/appService.js#L476C10-L488C13
- Returns characters whose builds, taken together, cover every light cone in the database by using a double NOT EXISTS approach to implement division