University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: April 2nd, 2025

Group Number: 124

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
QueAnh Ngo	83827527	g4j1w	emeraldnqa@gmail.com
Senlin Sun	50218502	u0a8c	sun.senlin0@gmail.com
Raymond Li	58216474	x3c6y	li.raymond04@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

1. Project Description:

a. A brief description of the projection:

Our astronomy database project is an application that can host multiple information regarding the knowledge of astronomy, and that includes different type of celestial bodies information, such as names, their coordinates, distance from Earth, color, temperature, size, etc. In addition, other information such as telescopes, observatories, astronomers, astronomical theories and phenomena can be hosted on this application. With this application, the users can collect simple information such as the furthest away celestial bodies in the database, which observatories have taken all the picture in the database, find the name of the observatory that house a specific telescope, etc.

b. Changes in the schema

There’s only one change in the schema since our Milestone 2 submission. That changes is data type of all the columns in the relation that has Boolean datatype to a binary datatype of 0 and 1, where 0 represent FALSE in Boolean and 1 represent True in Boolean. The changes need to be made since the version of SQLPLUS offered by UBC CS remote server does not support Boolean data type. The relation with their column data type has to be change are:

- Celestial_body with visible columns
- Planet1 with habitable and water columns
- Planet with habitable column
- Astronomer with active column
- Ph_location_found with explained column
- Th_explained_by with solved column

c. Schemas

Celestial_Body: (cb_name: VARCHAR(40), coordinate: VARCHAR(100), visible: NUMBER(1), distance: DOUBLE PRECISION, diameter: DOUBLE PRECISION)

Star(**cb_name** VARCHAR(40)**, coordinate** VARCHAR(100)**,** age INTEGER, **temperature** INTEGER)

Star1(temperature: INTEGER, spectral_class: VARCHAR(40), luminosity_class VARCHAR(40), color VARCHAR(40))

Planet(**cb_name** VARCHAR(40)**, coordinate** VARCHAR(100)**, habitable** NUMBER(1) , rotational_tilt DOUBLE PRECISION, axial_tilt DOUBLE PRECISION, shape VARCHAR(10))

Planet1(habitable NUMBER(1), water NUMBER(1))

Blackhole: (**cb_name:** VARCHAR(40), **coordinate:** VARCHAR(100)**,** charge: DOUBLE PRECISION, angular_momentum: DOUBLE PRECISION, mass: DOUBLE PRECISION)

Galaxy:(**cb_name:** VARCHAR(40), **coordinate:** VARCHAR(100), shape: VARCHAR(10), color: VARCHAR(20))

Observatory(obs_id INTEGER, **obs_name** VARCHAR(40))

Observatory1(obs_name VARCHAR(40), address VARCHAR(100))

Astronomer: (ast_id: INTEGER, ast_name: VARCHAR(40), active: NUMBER(1)): ast_name are unique and not null

ph_location_found (ph_name:VARCHAR(50), **obs_id:** INTEGER, explained: NUMBER(1))

picture_taken_by(picture_id: INTEGER, date_taken DATE, link VARCHAR(2048)**, tel_name** VARCHAR(40)**)**

tel_housed_at(**tel_name** VARCHAR(40), manufactured_date DATE, model VARCHAR(40))

tel_housed_at1(tel_name VARCHAR(40), obs_id INTEGER)

found (**ast_id:** INTEGER, **cb_name:** VARCHAR(40), **coordinate:** VARCHAR(100))

described(**ast_id:** INTEGER, **ph_name:** VARCHAR(40))

th_explained_by: (th_name: VARCHAR(40), **ph_name:** VARCHAR(40)**,** date_found: DATE, content: VARCHAR(4000), solved: NUMBER(1))

authored: (**ast_id:** INTEGER**, th_name:** VARCHAR(40)**, ph_name:** VARCHAR(40))

has(**ph_name:** VARCHAR(40), **cb_name:** VARCHAR(40)**, coordinate:** VARCHAR(100))

taken_of(**picture_id:** INTEGER,**, cb_name:** VARCHAR(40)**, coordinate:** VARCHAR(100))

d. SQL Queries:

- Insert (Line 157-159 in appService.js)

  INSERT INTO celestial_body (cb_name, coordinate, visible, distance, diameter) VALUES (:1, :2, :3, :4, :5);

  INSERT INTO star1 (temperature, spectral_class, luminosity_class, color) VALUES (:1, :2, :3, :4)";

```
INSERT INTO star (cb_name, coordinate, age, temperature) VALUES (:1, :2, :3, :4)";
```

- Delete (Line 313 in appService.js)
  ```
  DELETE FROM celestial_body
  WHERE cb_name = :cb_name
  AND coordinate = :coordinate
  ```
- Update (Line 343, 434-435 in appService.js)
  ```
  UPDATE celestial_body
  SET ${celestialUpdates.join(', ')}
  WHERE cb_name = :cb_name
  AND coordinate = :coordinate

  UPDATE Star SET ${starSet.join(', ')}
  WHERE cb_name = :cb_name
  AND coordinate = :coordinate
  ```
- Selection (Line 453 in appService.js)
  ```
  SELECT * FROM celestial_body
  WHERE distance < :distance
  ```
- Projection (Line 476 in appService.js)
  - Show all theory names associated with phenomena name and astronomer who authored it
    ```
    SELECT ${selectedColumns.join(', ')}
    FROM Authored au, Astronomer ast
    WHERE au.ast_id = ast.ast_id
    ```
- Join (Line 492 in appService.js)
  - Find the name of the observatory that house a specific user input telescopes
    ```
    SELECT obs.obs_name FROM tel_housed_at1 th1, observatory obs
    WHERE th1.tel_name = :tel_name
    AND th1.obs_id = obs.obs_id
    ```

Aggregation queries:

- **Aggregation with group by** (Line 510 in appService.js)
  - Find the max distance celestial body for every observatory
    ```
    SELECT obs.obs_name, cb.cb_name, cb.distance
    ROM celestial_body cb, taken_of to1, tel_housed_at1 th1, observatory obs,
    picture_taken_by ptb
    WHERE to1.cb_name = cb.cb_name
    AND to1.picture_id = ptb.picture_id
    AND ptb.tel_name = th1.tel_name
    ```

```
AND th1.obs_id = obs.obs_id
AND cb.distance =
        (SELECT MAX(cb.distance) FROM celestial_body cb, taken_of to1,
        tel_housed_at1 th1, Observatory obs2, picture_taken_by ptb
        WHERE to1.cb_name = cb.cb_name
        AND to1.picture_id = ptb.picture_id
        AND ptb.tel_name = th1.tel_name
        AND th1.obs_id = obs.obs_id
        AND obs2.obs_name = obs.obs_name
        GROUP BY obs2.obs_name)
```

- **Aggregation with Having** (Line 511 in appService.js)
  - Find oldest manufacturing date of telescope among observatories that have 2 or more telescopes

    ```
    SELECT th1.obs_id, MIN(manufactured_date)
    FROM tel_housed_at th, tel_housed_at1 th1
    WHERE th.tel_name = th1.tel_name
    GROUP BY th1.obs_id
    HAVING COUNT(*) >= 2
    ```

- **Nested aggregation with group by (**Line 512, 517 in appService.js)
  - Find the average count of pictures taken overall from the observatories

    ```
    DROP VIEW Temp;
    CREATE VIEW Temp(obs_id, amount) as
    SELECT th1.obs_id, COUNT(*) AS amount
    FROM tel_housed_at1 th1, tel_housed_at tha, picture_taken_by ptb
    WHERE th1.tel_name = ptb.tel_name
    AND th1.tel_name = tha.tel_name
    GROUP BY th1.obs_id;

    SELECT AVG(amount) as average
    FROM Temp
    ```

- **Division** (Line 513 in appService.js)
  - Find the observatories that have taken a picture of all the celestial bodies in the database

    ```
    SELECT obs.obs_id, obs.obs_name
    FROM Observatory obs, Observatory1 obs1
    WHERE obs.obs_name = obs1.obs_name AND NOT EXISTS (
            SELECT cb_name
    ```

```
FROM celestial_body
MINUS
SELECT to1.cb_name
FROM tel_housed_at1 th1, picture_taken_by ptb, taken_of to1
WHERE th1.obs_id = obs.obs_id
AND th1.tel_name = ptb.tel_name
AND ptb.picture_id = to1.picture_id
)
```