THE UNIVERSITY OF SYDNEY

School of Computer Science
[Author]

COMP9120 Relational Database Systems

Tutorial Week 3: The Relational Model

Exercise 1. PostgreSQL Database Login

Throughout this semester, you will be working with PostgreSQL 9.5.1. You can remotely access the PostgreSQL server that is maintained by School of CS via pgAdmin. You can install pgAdmin on your own computer, which can be downloaded from https://www.pgadmin.org/. Note that, if you want to access the PostgreSQL server from outside the university network, you will need to first login VPN. https://sydneyuni.service-now.com/sm?id=kb_article_view&sys_kb_id=c0bf9bd6db41b3485beaf9b7f49619a2&sysparm_tsqueryId=f90a62cbdb937f44c8a5773c349619f2&sysparm_rank=7

Your connection information to the PostgreSQL server maintained by SCS is as follows.

Host: soit-db-pro-2.ucc.usyd.edu.au

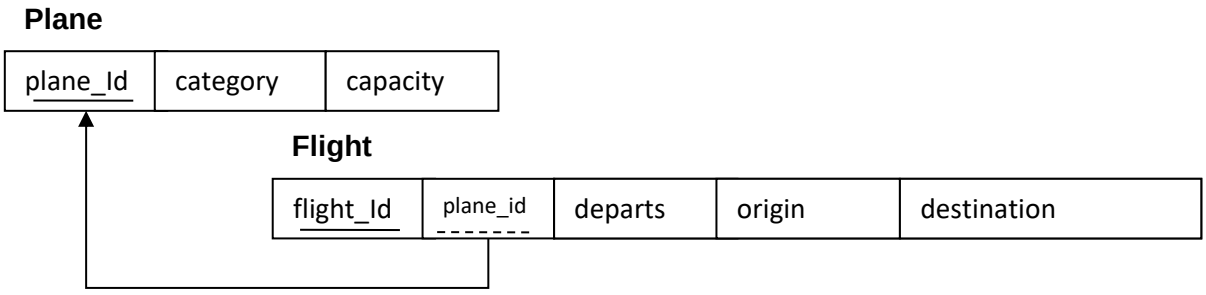
DB: y22s1c9120_UNIKEY (replace UNIKEY with your unikey); this is your database

Username: y22s1c9120_UNIKEY (replace UNIKEY with your unikey); the same as your DB

Password: your password

Exercise 2. Flight Booking Schema

Here is a relational model diagram for part of a flight booking database schema:



Write DDL statements for these two relations. Execute the statements within your PostgreSQL account to test them out. You are using PostgreSQL 9.5 syntax so you may wish to refer to the PostgreSQL 9.5 CREATE TABLE documentation at:

https://www.postgresql.org/docs/9.5/static/ddl.html

You should choose appropriate data types for your attributes based upon the following details:

(i) Each plane has a unique alphanumeric ID of up to 8 characters, a category (either 'jet' or 'turboprop'), and capacity (maximum number of passengers);

(ii) A flight has a unique numerical ID, departure date, the plane making the flight, the origin and destination;

Check the documentation on PostgreSQL data types at:

https://www.postgresql.org/docs/9.5/static/datatype.html

Also make sure you capture all the appropriate key constraints (there is one foreign and two primary keys).

Once the tables are created, add another integrity constraint to capture the extra rule that a plane cannot make more than one flight a day.

ALTER TABLE syntax is documented at:

https://www.postgresql.org/docs/9.5/static/ddl-alter.html

If you'd like to remove one of the tables that you've created, you may do so with DROP TABLE, after which you can create the table afresh.

Solution:

If you have tables of the same name in your database you can remove them with:

```
DROP TABLE Plane CASCADE;
DROP TABLE Flight CASCADE;
```

```
CREATE TABLE Plane (
        plane_id VARCHAR(8) PRIMARY KEY,
        category VARCHAR(10) NOT NULL
                CHECK( category IN ('jet', 'turboprop')),
        capacity INTEGER NOT NULL
);

CREATE TABLE Flight (
        flight_id INTEGER PRIMARY KEY,
        departs DATE NOT NULL,
        plane_id VARCHAR(8) NOT NULL REFERENCES Plane,
        origin VARCHAR(20) NOT NULL,
        destination VARCHAR(20) NOT NULL
);

ALTER TABLE Flight ADD CONSTRAINT one_flight_per_day
                UNIQUE(plane_id, departs);
```

Exercise 3. Populating the DB

Add data to your relations using INSERT statements. Then try inserting, updating and deleting data to violate the integrity constraints of the database. See if you can trigger an error for each of the classes of constraints described in Exercise 2.

You can inspect the contents of each relation with, e.g.:

```
SELECT * FROM Plane;
SELECT * FROM Flight;
```

Answer:

Try inserting some of the values below, and view the data in a table after rows are inserted.

```
-- Add some planes
INSERT INTO Plane VALUES ('aa-1234', 'jet', 250);
INSERT INTO Plane VALUES ('ab-1334', 'turboprop', 17);
INSERT INTO Plane VALUES ('ac-4234', 'jet', 300);
INSERT INTO Plane VALUES ('ad-7234', 'jet', 90);
INSERT INTO Plane VALUES ('ae-6237', 'turboprop', 50);

-- Some examples of breaking constraints:
-- Domain constraint: capacity should be integer
INSERT INTO Plane VALUES ('zz-3456', 'turboprop', ‘ss’);
-- capacity should be integer (but still works):
INSERT INTO Plane VALUES ('ax-9999', 'turboprop', '50');
-- plane_id should be string (but still works):
INSERT INTO Plane VALUES (123456, 'jet', 255);

-- NOT NULL constraints:
-- Explicit NULL when category NOT NULL in schema
INSERT INTO Plane VALUES ('bj-78', NULL, 500);
-- Same thing with implicit NULLs
INSERT INTO Plane (plane_id, capacity) VALUES ('bj-78', 500);

-- Primary key constraints
-- Plane ID must be unique
INSERT INTO Plane VALUES ('ax-6666', 'jet', 90);
INSERT INTO Plane VALUES ('ax-6666', 'jet', 85);
-- Primary keys are also implicitly NOT NULL
INSERT INTO Plane VALUES (NULL, 'jet', 17);

-- Foreign key constraints
INSERT INTO Flight VALUES (1, '25-DEC-12', 'xmas00', 'Lapland', 'Sydney');
-- Need to have the plane in the database first:
INSERT INTO Plane VALUES ('xmas00', 'jet', 13);
INSERT INTO Flight VALUES (1, '25-DEC-12', 'xmas00', 'Lapland', 'Sydney');
```