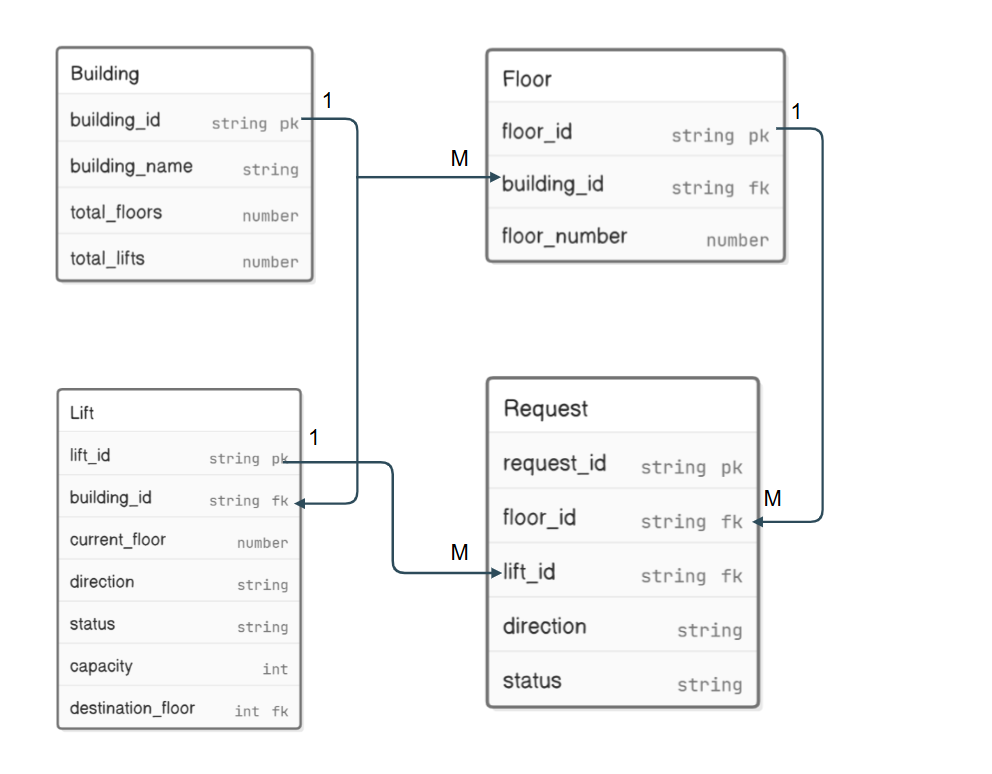
Assignment : Creating a database for the lift Management System

(Updated based on feedback )

Coordinator: Suresh Burde

Submitted by : Bhagyashree Dattatray Watpade.

Entity Relationship Diagram:



Database : PostgreSQL

Tables : Building, Floor, Lift, Request

>> Code :

CREATE TABLE Building (

building\_id SERIAL PRIMARY KEY,

building\_name VARCHAR(10),

total\_floors INT CHECK (total\_floors > 0),

total\_lifts INT CHECK (total\_lifts > 0) DEFAULT 1

);

CREATE TABLE Floor (

floor\_id SERIAL PRIMARY KEY,

floor\_number INT,

building\_id INT REFERENCES Building(building\_id) ON DELETE CASCADE

);

CREATE TABLE Lift (

lift\_id SERIAL PRIMARY KEY,

current\_floor INT NOT NULL DEFAULT 0,

direction VARCHAR(10),

status VARCHAR CHECK (status IN ('ACTIVE', 'INACTIVE', 'MAINTENANCE')),

capacity INT,

destination\_floor INT ,

lift\_building\_id INT REFERENCES Building(building\_id) ON DELETE CASCADE,

lift\_destination\_floor INT REFERENCES Floor(floor\_id) ON DELETE CASCADE

);

CREATE TABLE Request (

request\_id SERIAL PRIMARY KEY,

floor\_id INT,

lift\_id INT ,

direction VARCHAR ,

status VARCHAR CHECK (status IN ('PENDING', 'COMPLETED', 'CANCELLED')),

request\_floor\_id INT REFERENCES Floor(floor\_id) ON DELETE CASCADE,

request\_lift\_id INT REFERENCES Lift(lift\_id) ON DELETE CASCADE

);

-- INSERT DATA INTO TABLES

INSERT INTO Building (building\_name, total\_floors, total\_lifts) VALUES ('Building 1', 10, 3);

INSERT INTO Building (building\_name, total\_floors, total\_lifts) VALUES ('Building 2', 15, 4);

INSERT INTO Building (building\_name, total\_floors, total\_lifts) VALUES('Building 3', 10, 4);

INSERT INTO Floor (building\_id, floor\_number) VALUES (1, 0);

INSERT INTO Floor (building\_id, floor\_number) VALUES (1, 1);

INSERT INTO Floor (building\_id, floor\_number) VALUES(2, 2);

INSERT INTO Floor (building\_id, floor\_number) VALUES (3, 5);

INSERT INTO Lift (current\_floor, direction, status, capacity, destination\_floor) VALUES (1, 'UP', 'ACTIVE', 7, 5);

INSERT INTO Lift (current\_floor, direction, status, capacity, destination\_floor) VALUES (4, 'UP', 'ACTIVE', 15, 3);

INSERT INTO Lift (current\_floor, direction, status, capacity, destination\_floor) VALUES (2, 'DOWN','INACTIVE', 20, 3);

INSERT INTO Request (floor\_id, lift\_id, direction, status) VALUES (2, 2, 'UP', 'COMPLETED');

INSERT INTO Request (floor\_id, lift\_id, direction, status) VALUES (3, 1, 'DOWN', 'PENDING');

INSERT INTO Request (floor\_id, lift\_id, direction, status) VALUES (1, 1, 'UP', 'PENDING');

-- getting data of each table

SELECT \* FROM Building

SELECT \* FROM Floor

SELECT \* FROM Lift

SELECT \* FROM Request

--Q1 retriving the list of lifts whose destination floor is 5

SELECT \* FROM Lift WHERE destination\_floor = 5;

-- Q2 get the details of lift which is currently active and moving up :

SELECT \* FROM Lift WHERE status = 'ACTIVE' AND direction = 'UP';

-- Q3 update lift status :

UPDATE Lift SET status = 'INACTIVE' WHERE lift\_id = 3;

SELECT \* FROM Lift

-- Q4 delete request where request id is 1

DELETE FROM Request WHERE request\_id = 1;

SELECT \* FROM Request

-- Q5 get list of floors where floor number is greater than 4

SELECT \* FROM Floor WHERE floor\_number > 4;

-- Q6 get all lift requests with floor, building details

SELECT Req.request\_id, Build.building\_name, Fl.floor\_number, Li.lift\_id, Req.status

FROM Request Req

JOIN Floor fl ON Req.floor\_id = Fl.floor\_id

JOIN Lift Li ON Req.lift\_id = Li.lift\_id

JOIN Building Build ON Fl.building\_id = Build.building\_id;

OUTPUT : Output of the above queries is available in the below csv file.

