

S

Q

TR

T

E
R
D

E

U

G

COVID-19 vaccination management system



By Gal Karasnty
& Elazar Fine

S**Q****TR****T****E
R
D****E****U**

GENERAL INTRODUCTION

- The system oversees a vaccination operation in a city, managing data about its citizens, medical workers, and clinics and their vaccine supplies.
- The purpose of the system is to facilitate vaccination appointments for citizens of varying risk groups, track the vaccination phase for each citizen, and make sure each clinic is adequately staffed and has enough vaccine supplies.

**G**

USERS

The system has three front ends:

- One for citizens - to make appointments and track their status through an app and a website.
- Second for city leaders / project managers - to overlook the operation and get relevant information to make informed logistical and managerial decisions.
- Third for health care workers – to log vaccine administrations and to view their schedule.



ENTITIES

S

Q

TR

T

E
R
D

Vaccine (VaccineID, Name, Company)

Dose (DoseBarcode, *SupplyID)

Citizen (CitizenID, Fname, Lname, PhoneNum, RiskGroup, Age, Weight, Email, District, PhasesComplete)

HealthCareWorker (WorkerID, *ClinicID, FirstName, LastName, PhoneNum, LicenseNum, Seniority)

Clinic (ClinicID, ClinicName, PhoneNum, Email, Street, HouseNum, District)

Supply (SupplyID, *ClinicID, *VaccineID, ExpiryDate)

Appointment (AppointmentID, *CitizenID, Date, *ClinicID, *WorkerID)

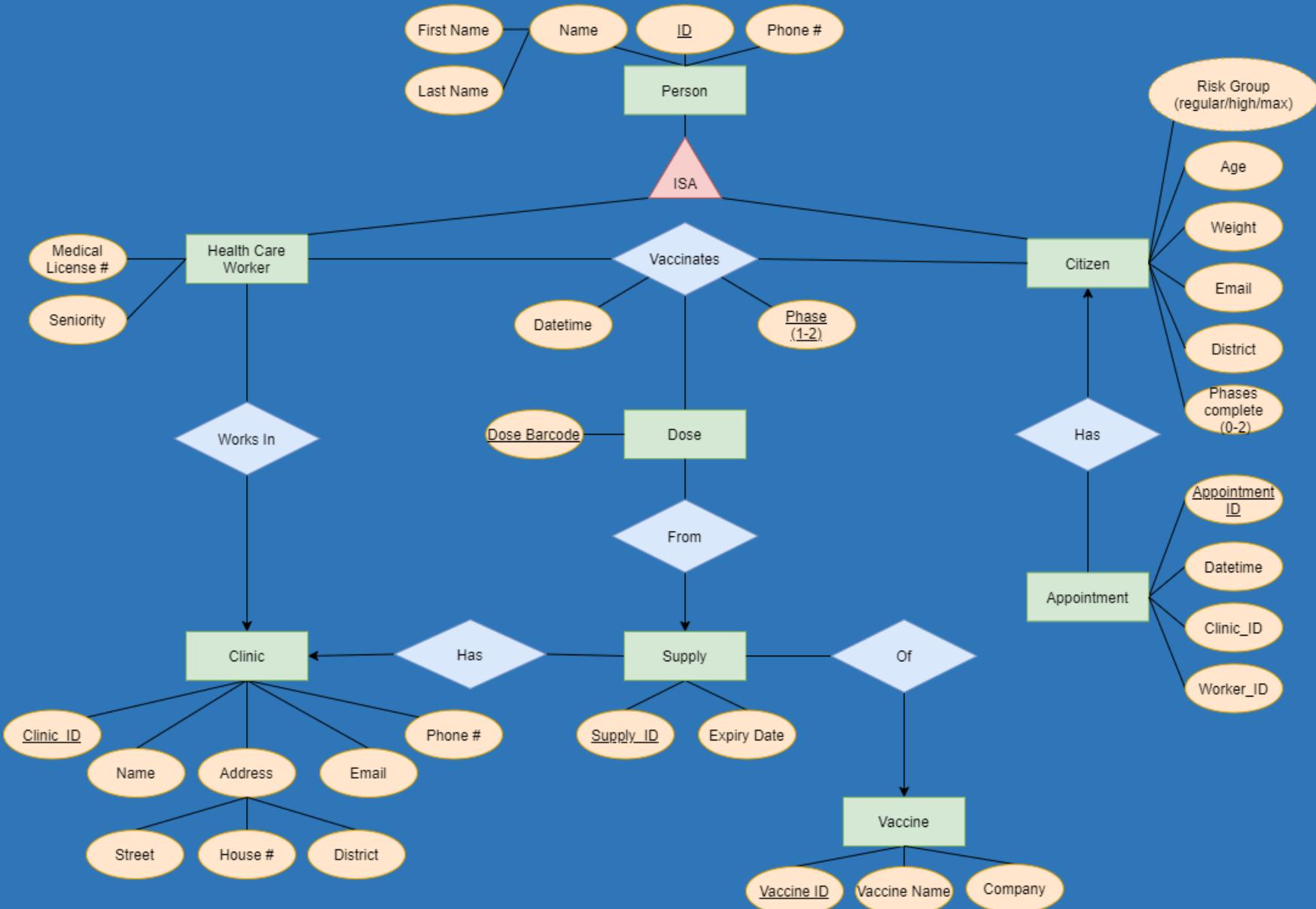
E

U

G



ERD



S

Q

TR

T

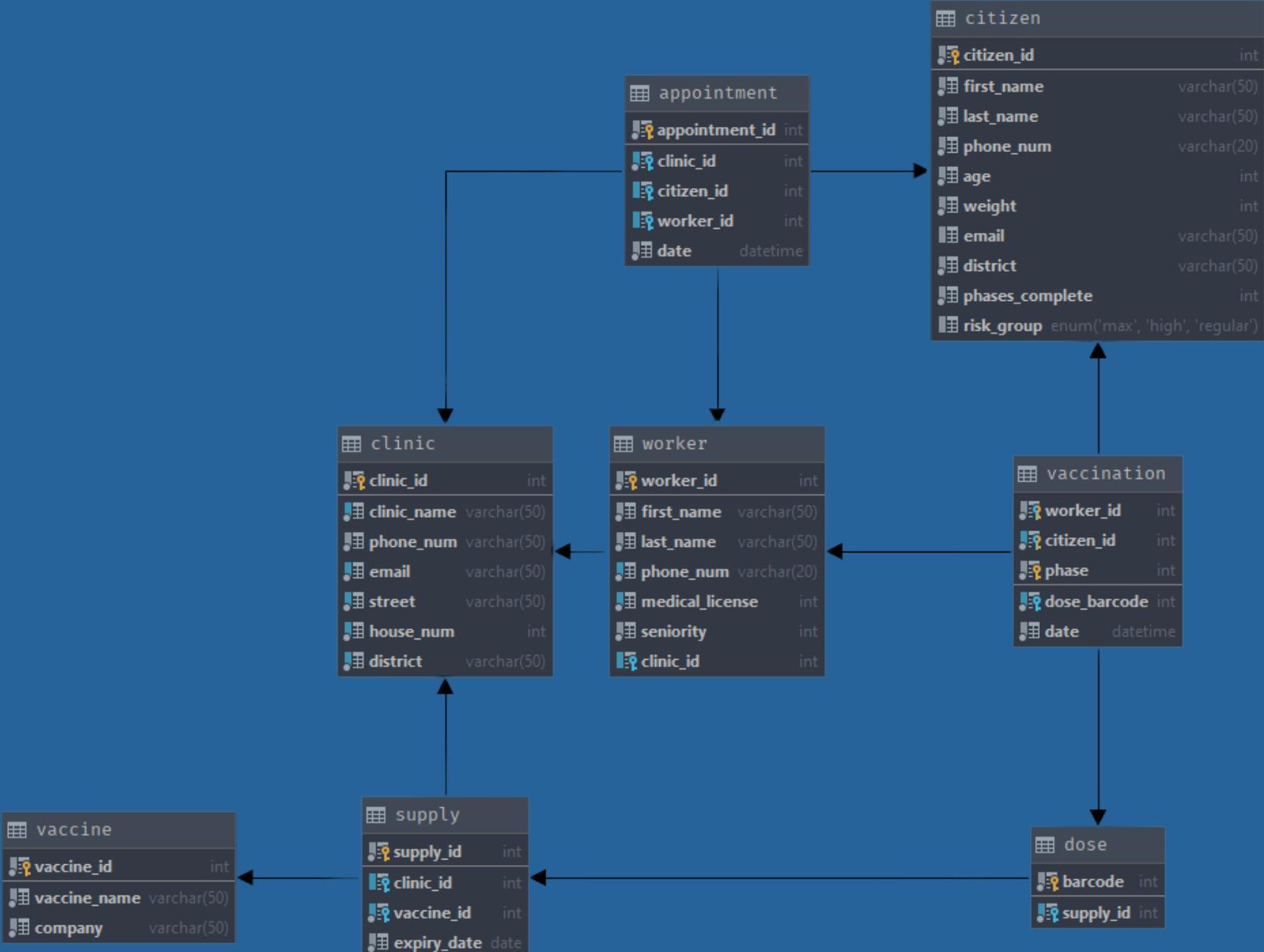
E
R
D

E

U

G

TABLES



S**Q**

TRIGGERS

TR**T****E
R
D****E****U****G**

```
DELIMITER $$  
CREATE TRIGGER delete_fulfilled_appointment  
    AFTER INSERT  
    ON vaccination  
    FOR EACH ROW  
BEGIN  
    /* takes cares of edge cases of vaccination  
     * in a different date or vaccination without appointment */  
    IF (SELECT COUNT(appointment_id) AS aid  
        FROM appointment  
        WHERE appointment.citizen_id = new.citizen_id) > 1  
    THEN  
        DELETE  
        FROM appointment  
        WHERE appointment_id IN (  
            SELECT aid  
            FROM ( /* this weird format helps avoiding errors */  
                SELECT appointment_id AS aid  
                FROM appointment  
                WHERE appointment.citizen_id = new.citizen_id  
                ORDER BY appointment.date  
                LIMIT 1  
            ) AS a);  
    END IF;  
END $$  
DELIMITER ;  
  
CREATE TRIGGER update_phase_for_citizen_after_vac  
    AFTER INSERT  
    ON vaccination  
    FOR EACH ROW  
    UPDATE citizen  
    SET phases_complete = phases_complete + 1  
    WHERE citizen_id = new.citizen_id;
```

S

Q

```
-- make appointment for next phase in 21 days
DELIMITER $$

CREATE TRIGGER make_appointment_for_next_phase
    BEFORE INSERT
    ON vaccination
    FOR EACH ROW
BEGIN
    DECLARE citizen_chosen_clinic INT;

    SELECT clinic_id
    INTO citizen_chosen_clinic
    FROM appointment
    WHERE citizen_id = new.citizen_id
    LIMIT 1;

    /* make sure we have a clinic for the citizen
    in case he vaccinated without an appointment */
    IF citizen_chosen_clinic IS NULL THEN
        SET citizen_chosen_clinic = (
            SELECT clinic_id
            FROM clinic
            WHERE clinic.district IN
                (SELECT citizen.district
                FROM citizen
                WHERE citizen.citizen_id = new.citizen_id)
            LIMIT 1
        );
    END IF;

    INSERT INTO appointment
        VALUE (DEFAULT, citizen_chosen_clinic, new.citizen_id,
               new.worker_id, DATE_ADD(new.date, INTERVAL 21 DAY));
END $$

DELIMITER ;
```

```
CREATE TRIGGER worker_clinic_integrity
    BEFORE INSERT
    ON appointment
    FOR EACH ROW
wci_trigger:
BEGIN
    DECLARE worker_clinic_integrity_error VARCHAR(255);
    DECLARE worker_clinic INT;

    SELECT clinic_id
    INTO worker_clinic
    FROM worker
    WHERE worker_id = new.worker_id;

    CALL get_random_worker_from_clinic_id(
        _clinic: new.clinic_id,
        rand_worker_id: @rand_worker_id
    );

    /* if worker is null we choose a random valid
    worker from the user's chosen clinic and return */
    IF new.worker_id IS NULL THEN
        SET new.worker_id = (SELECT @rand_worker_id);
        LEAVE wci_trigger;
    END IF;

    SET worker_clinic_integrity_error = CONCAT('worker #',
                                                new.worker_id,
                                                ' is not in clinic #',
                                                new.clinic_id);

    /* if worker is not assigned to a clinic
    assign him to the appointment clinic */
    IF worker_clinic IS NULL THEN
        SET worker_clinic = new.clinic_id;

        UPDATE worker
        SET clinic_id = worker_clinic
        WHERE worker_id = new.worker_id;
    END IF;

    IF new.clinic_id != worker_clinic THEN
        SIGNAL SQLSTATE '45000'
            | SET MESSAGE_TEXT = worker_clinic_integrity_error;
    END IF;
```

TR

T

E

R

D

E

G

QUERIES

select queries

```

# amount of people in each phase
|SELECT phases_complete, COUNT(*) AS amount_of_people
|  FROM citizen
| GROUP BY phases_complete
| ORDER BY phases_complete;

# show amount of vaccines in each district
|SELECT district, COUNT(barcode) AS vaccines_total_amount
|  FROM dose
|    JOIN supply s ON dose.supply_id = s.supply_id
|    JOIN clinic c2 ON s.clinic_id = c2.clinic_id
| WHERE barcode NOT IN (SELECT dose_barcode FROM vaccination)
| GROUP BY district;

# show all clinics and their supplies ordered by expiry date
|SELECT (SELECT clinic.clinic_name
|        FROM clinic
|        WHERE clinic_id = supplies.clinic_id) AS clinic,
|        (SELECT CONCAT(company, ' - ', vaccine_name)
|         FROM vaccine
|         WHERE vaccine.vaccine_id = supplies.vaccine_id) AS vaccine,
|        amount,
|        expiry_date
|FROM ((SELECT clinic_id, vaccine_id, COUNT(barcode) AS amount, expiry_date
|        FROM supply
|          JOIN dose d ON supply.supply_id = d.supply_id
|        WHERE barcode NOT IN (SELECT dose_barcode FROM vaccination)
|        GROUP BY clinic_id
|      ) AS supplies
|        JOIN clinic ON supplies.clinic_id = clinic.clinic_id)
|ORDER BY expiry_date;

```

```

# show all citizens who missed their appointments
|SELECT date, first_name, last_name, phone_num
|  FROM appointment,
|       citizen
| WHERE date < DATE(NOW())
|   AND citizen.citizen_id = appointment.citizen_id;

# show amount of workers in each clinic ordered by descending seniority
|SELECT clinic_name,
|        COUNT(worker.worker_id) AS workers_count,
|        AVG(worker.seniority) AS average_seniority_in_years
|  FROM worker
|    JOIN clinic c
|      ON worker.clinic_id = c.clinic_id
| GROUP BY c.clinic_id
| ORDER BY average_seniority_in_years DESC;

# show citizens in the max risk group who did not
# complete the first phase and not don't have an appointment
|SELECT first_name, last_name, phone_num, email
|  FROM citizen
| WHERE phases_complete = 0
|   AND citizen_id NOT IN (SELECT citizen_id
|                           FROM appointment)
|   AND risk_group = 'max';

```

QUERIES

select queries

```
# show workers who have no appointments in a specific
# time range. order by ascending seniority
SELECT *
FROM worker
WHERE worker_id NOT IN (
    SELECT appointment.worker_id
    FROM appointment
    WHERE appointment.worker_id = worker.worker_id
    AND appointment.date BETWEEN NOW() AND DATE_ADD(NOW(), INTERVAL 1 MONTH))
ORDER BY worker.seniority;
```

```
# show schedule for a specific worker in the next month
SELECT *
FROM appointment
WHERE worker_id = 248490510
    AND appointment.date BETWEEN NOW() AND DATE_ADD(NOW(), INTERVAL 1 MONTH)
ORDER BY appointment.date;
```

```
# show a specific citizen how much time is left until his appointment
SELECT DATEDIFF(appointment.date, NOW()) AS days_left
FROM appointment
WHERE appointment.citizen_id = 696200479
ORDER BY appointment.date
LIMIT 1;
```

```
# show how many vaccinated in each age group
SELECT CONCAT(FLOOR(age / 10) * 10, '-',
              FLOOR((age + 10) / 10) * 10) AS age_group,
        COUNT(*) AS amount_vaccinated
FROM citizen
WHERE phases_complete = 2
GROUP BY FLOOR(age / 10) * 10
ORDER BY age;
```

QUERIES

select queries

```

# show citizen and worker details + appointment date for people who are max
# risk group who are scheduled with a worker with less than 5 years seniority
SELECT c.citizen_id,
       c.first_name,
       c.last_name,
       w.worker_id,
       w.first_name,
       w.last_name,
       w.seniority,
       appointment.date
  FROM (appointment JOIN citizen c ON c.citizen_id = appointment.citizen_id)
        JOIN worker w ON w.worker_id = appointment.worker_id
 WHERE c.risk_group = 'Max'
   AND w.seniority < 5;
                                         # show clinics who have more appointments than total vaccine amount
                                         # or don't any supplies but do have appointments
SELECT *
  FROM (SELECT clinic.clinic_id,
              clinic.clinic_name,
              (SELECT COUNT(barcode)
                 FROM dose
                WHERE dose.supply_id = supply.supply_id) AS vaccines_total,
              COUNT(appointment_id) AS appointments
     FROM (clinic JOIN appointment ap ON clinic.clinic_id = ap.clinic_id)
           JOIN supply ON supply.clinic_id = clinic.clinic_id
    GROUP BY clinic.clinic_id) AS all_clinics
 WHERE appointments > vaccines_total
 UNION
 (SELECT clinic_id,
         clinic_name,
         (SELECT COUNT(clinic_id) FROM supply WHERE clinic.clinic_id = supply.clinic_id),
         (SELECT COUNT(clinic_id) FROM appointment WHERE clinic.clinic_id = appointment.clinic_id)
    FROM clinic
   WHERE clinic_id NOT IN (SELECT clinic_id FROM supply)
     AND clinic_id IN (SELECT clinic_id AS cid FROM appointment)
);

```

S

QUERIES

Q**TR****T****E****R**
D**E****U****G**

Update/insert/delete

```

# move supplies from most supplied clinic to the least
# supplied (nested format to work with complex set/where)
UPDATE supply
|SET supply.clinic_id = (SELECT cid
|    FROM (SELECT clinic_id AS cid
|        FROM (SELECT c.clinic_id, COUNT(barcode) AS cnt
|            FROM supply
|                JOIN dose d ON supply.supply_id = d.supply_id
|                JOIN clinic c ON c.clinic_id = supply.clinic_id
|            GROUP BY c.clinic_id
|            ORDER BY cnt
|
|                LIMIT 1) AS min_inner) AS min)
|
|WHERE supply.supply_id = (SELECT sid
|    FROM (SELECT supply_id AS sid
|        FROM (SELECT d2.supply_id, COUNT(barcode) AS cnt
|            FROM supply
|                JOIN dose d2 ON supply.supply_id = d2.supply_id
|                JOIN clinic c2 ON c2.clinic_id = supply.clinic_id
|            GROUP BY c2.clinic_id
|            ORDER BY cnt DESC
|                LIMIT 1) AS max_inner) AS max);

# replace worker in appointments in a specific day
CALL replace_worker_in_specific_date(
|    _old_worker_id_ 460223128,
|    _new_worker_id_ 998843053,
|    _date_ DATE(DATE_ADD(NOW(), INTERVAL 21 DAY))
|);

# remove un-administered expired supplies
DELETE
FROM dose
WHERE supply_id IN (SELECT supply_id
    FROM supply
    WHERE expiry_date < DATE(NOW()))
    AND barcode NOT IN (SELECT dose_barcode
        FROM vaccination);

CREATE PROCEDURE replace_worker_in_specific_date
(IN _old_worker_id_ INT, IN _new_worker_id_ INT, IN _date_ DATE)
UPDATE appointment
SET appointment.worker_id = _new_worker_id_
WHERE appointment.worker_id = _old_worker_id_
    AND DATE(appointment.date) = _date_;

```

QUERIES

Update/insert/delete

```

# add supplies to all clinics with
# less than 1000 total vaccines
CALL add_to_low_supply_clinics();

DELIMITER $$

CREATE PROCEDURE add_doses_to_supply(IN _amount_ INT, IN _supply_id_ INT)
BEGIN
    DECLARE i INT;
    SET i = 0;
    WHILE i < _amount_
        DO
            INSERT INTO dose VALUE (DEFAULT, _supply_id_);
            SET i = i + 1;
        END WHILE;
END $$

DELIMITER ;

```

```

DELIMITER $$
CREATE PROCEDURE add_to_low_supply_clinics()
BEGIN
    DECLARE current_clinic INT;
    DECLARE current_supply INT;

    DROP TABLE IF EXISTS lows;
    CREATE TEMPORARY TABLE lows
    (
        _clinic_id_ INT
    );

    /* collect all low supply clinic IDs */
    INSERT INTO lows (_clinic_id_)
    SELECT cid
    FROM (SELECT c.clinic_id AS cid, COUNT(*) AS amount
          FROM supply
          JOIN dose d ON supply.supply_id = d.supply_id
          JOIN clinic c ON c.clinic_id = supply.clinic_id
          GROUP BY c.clinic_id
         ) AS min_inner
    WHERE amount < 1000;

    /* for each low supply clinic, add a 1000 doses supply */
    WHILE (SELECT COUNT(*) FROM lows) > 0
        DO
            SET current_clinic = (SELECT * FROM lows LIMIT 1);

            INSERT INTO supply VALUE (DEFAULT,
                                      current_clinic,
                                      1,
                                      DATE_ADD(DATE(NOW()), INTERVAL 1 MONTH));

            SELECT LAST_INSERT_ID() INTO current_supply;

            CALL add_doses_to_supply( _amount_: 1000, _supply_id_: current_supply);

            DELETE FROM lows WHERE _clinic_id_ = current_clinic;

        END WHILE;
END $$

DELIMITER ;

```

SECURITY

S

Q

TR

T

E

R

D

E

U

G

The server will direct different users to the database based on permission level:

The operation manager will have access to everything. District managers only to the parts relating to their districts.

Citizens only have access to make and see their appointments, and status.

Health care workers can access everything relating to their work schedule and appointments/patients.

SECURITY

The server will direct different users to the database based on permission level:

The operation manager will have access to everything.
District managers only to the parts relating to their districts.

Citizens only have access to make and see their appointments, and status.

Health care workers can access everything relating to their work schedule and appointments/patients.