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

Vaccine (VaccineID, Name, Company)

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

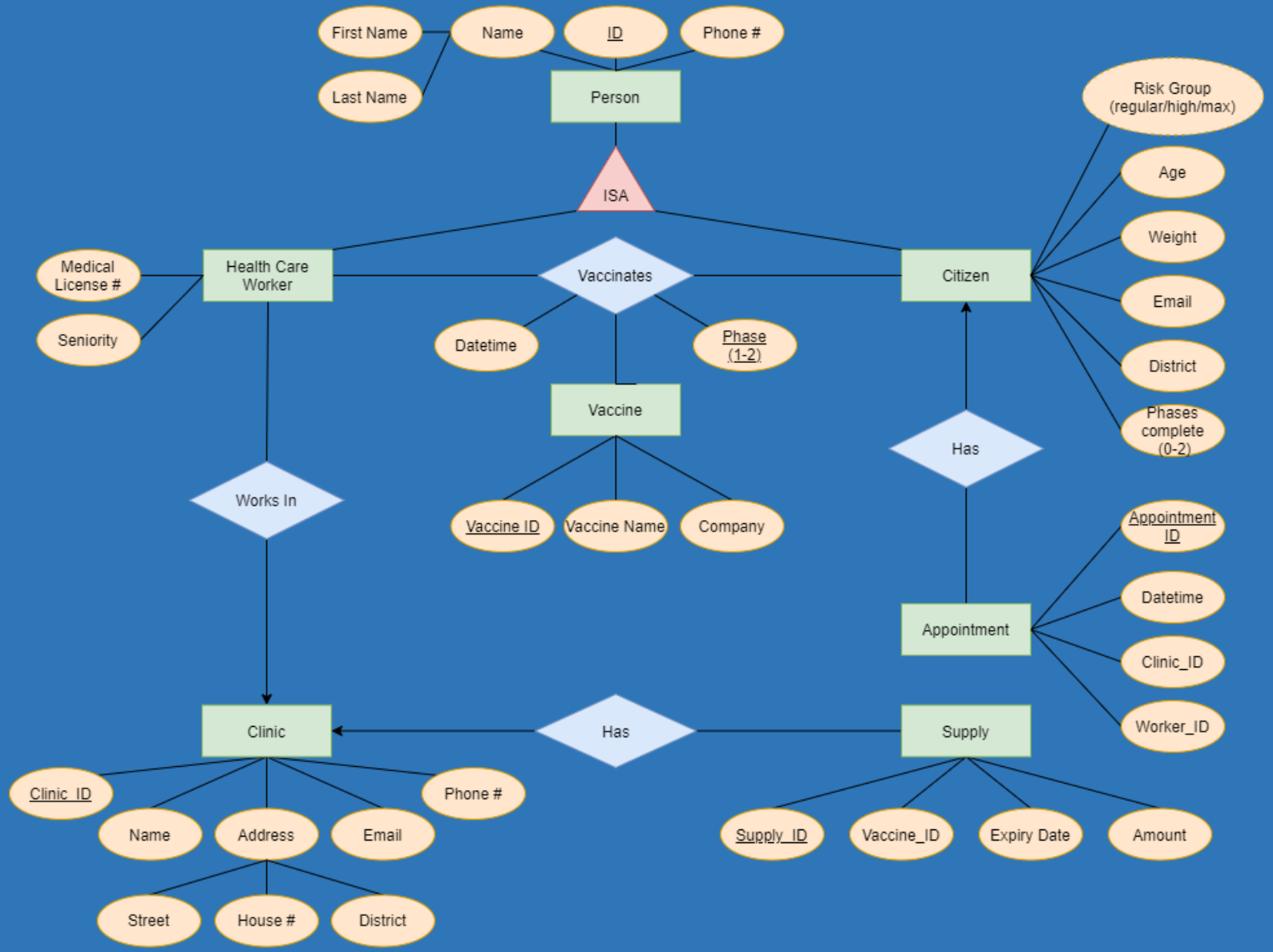
HealthCareWorker (WorkerID, FirstName, LastName, PhoneNum, LicenseNum, Seniority)

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

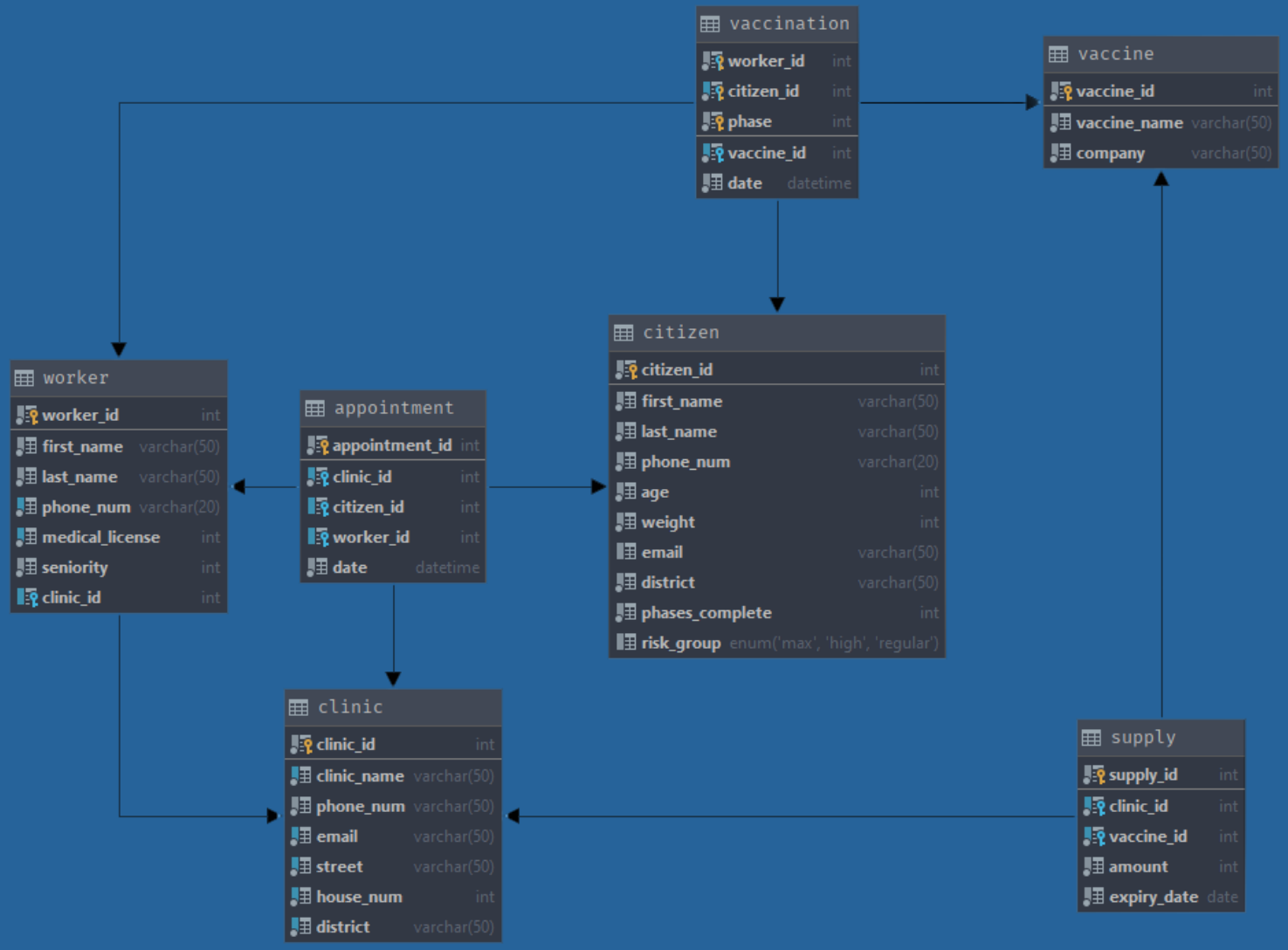
Supply (SupplyID, VaccineID*, ExpiryDate, Amount)

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

ERD


First Name
Name
ID
Phone #
Last Name
Person
ISA
Risk Group (regular/high/max)
Age
Weight
Email
District
Phases complete (0-2)
Medical License #
Seniority
Health Care Worker
Vaccinates
Citizen
Datetime
Phase (1-2)
Vaccine
Has
Works In
Vaccine ID
Vaccine Name
Company
Appointment ID
Datetime
Appointment
Clinic_ID
Worker_ID
Clinic
Has
Supply
Clinic_ID
Phone #
Name
Address
Email
Supply_ID
Vaccine_ID
Expiry Date
Amount
Street
House #
District

TABELS


vaccination
worker_id int
citizen_id int
phase int
vaccine_id int
date datetime
vaccine
vaccine_id int
vaccine_name varchar(50)
company varchar(50)
worker
worker_id int
first_name varchar(50)
last_name varchar(50)
phone_num varchar(20)
medical_license int
seniority int
clinic_id int
appointment
appointment_id int
clinic_id int
citizen_id int
worker_id int
date datetime
citizen
citizen_id int
first_name varchar(50)
last_name varchar(50)
phone_num varchar(20)
age int
weight int
email varchar(50)
district varchar(50)
phases_complete int
risk_group enum('max', 'high', 'regular')
clinic
clinic_id int
clinic_name varchar(50)
phone_num varchar(50)
email varchar(50)
street varchar(50)
house_num int
district varchar(50)
supply
supply_id int
clinic_id int
vaccine_id int
amount int
expiry_date date

TRIGGERS

```sql
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;
```

```sql
CREATE TRIGGER update_vaccine_supply_amount
    AFTER INSERT
    ON vaccination
    FOR EACH ROW
    UPDATE supply
    SET amount = amount - 1
    WHERE new.vaccine_id = supply.vaccine_id
      AND supply.clinic_id = (
        SELECT worker.clinic_id
        FROM worker
        WHERE worker.worker_id = new.worker_id
    );
```

```sql
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 ;
```

TRIGGERS

```sql
-- 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 ;
```

```sql
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;
```

QUERIES

select queries

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

```sql
# show amount of vaccines in each district
SELECT district, SUM(amount) AS vaccines_total_amount
FROM supply
         JOIN clinic c ON supply.clinic_id = c.clinic_id
GROUP BY c.district;
```

```sql
# 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, amount, expiry_date
       FROM supply) AS supplies
         JOIN clinic ON supplies.clinic_id = clinic.clinic_id)
ORDER BY expiry_date;
```

```sql
# 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;
```

```sql
# 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;
```

```sql
# 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';
```

S

QUERIES

select queries

```sql
# show how many vaccinated in each age group
SELECT age, COUNT(*) AS amount_vaccinated
FROM citizen
WHERE phases_complete = 2
GROUP BY age
ORDER BY age;
```

```sql
# 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;
```

```sql
# 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;
```

```sql
# 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;
```

QUERIES

QUERIES

select queries

```sql
# 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
SELECT *
FROM (SELECT clinic.clinic_id,
             clinic.clinic_name,
             SUM(supply.amount)    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 vaccines_total < appointments;
```

QUERIES

Update/insert/delete

```sql
# remove expired supplies
DELETE
FROM supply
WHERE expiry_date < DATE(NOW());
```

```sql
# 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 supply
                              ORDER BY amount
                              LIMIT 1
                             ) AS min)
WHERE supply.supply_id = (SELECT sid
                          FROM (SELECT supply_id AS sid
                                FROM supply
                                ORDER BY amount DESC
                                LIMIT 1) AS max);
```

```sql
# 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))
    );
```

```sql
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_to_low_supply_clinics()
BEGIN
    DECLARE current_clinic INT;
    CREATE TEMPORARY TABLE lows
    (
        _clinic_id_ INT
    );

    INSERT INTO lows (_clinic_id_)
    SELECT clinic_id
    FROM supply
    WHERE clinic_id NOT IN (SELECT clinic_id
                            FROM (SELECT clinic_id, SUM(amount) AS sum
                                  FROM supply
                                  GROUP BY clinic_id) AS high
                            WHERE sum >= 1000);

    WHILE (SELECT COUNT(*) FROM lows) > 0
        DO
            SET current_clinic = (SELECT * FROM lows LIMIT 1);

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

            DELETE FROM lows WHERE _clinic_id_ = current_clinic;

        END WHILE;
END $$
DELIMITER ;
```

SECURITY

Possible Implementation

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. And clinic managers to their clinics.

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

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

SECURITY

Possible Implementation

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. And clinic managers to their clinics.

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

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

S Q TR T E R D E U G