cs4111-Project

Group - 45 Nihar Maheshwari - nm3223 Aayush Kumar Verma - av2955

Postgres Connection Settings

- "user": "nm3223",
- "passphrase": "369638"
- "db-server": "34.73.36.248/project1"

Features Expanded

As part of Part-4 of the project, we have chosen to implement the following

- **Text Attributes** :
  - **New attribute introduced** : **`consult_notes_txt`**
  - **Rationale and Application** : As a hospital management system, we would wish to maintain the doctors notes taken as part of a patient visit or surgery. This is often useful information passed on to other doctors or used in insurance healthcare claims. This field is introduced as part of the **`consult`** table.
- **List Attributes**:
  - **New Attribute Introduced** : **`med_history`**
  - **Rationale and Application**: It would be good to have a documented list of diseases and pre-conditions every patient has for a hospital management system. Having this information would ideally reduce the effort on doctors and nurses while treating patients. Example : A pre-condtion of allergy to peanuts would be very helpful for doctors while recommending a diet to patients. With this field, we aim to capture all preconditions a patient has as a list. This field will be available in the **`patient`** table.
- **Triggers**
  - **Trigger Added** : **`check_inventory`**
  - **Rationale and Application**:
    - Our Hospital Management System maintains an inventory of medicines present in the hospital pharmacy. Upon prescribing and dispatching medicines to patients, the current stock of the medicine prescribed should reduce by 1 for each patient. We model this scenario via a trigger.
    - Note that we need to handle the corner case when the medicine stock drops below 0. We incorporate the exception withing the trigger. Any attempt to dispatch a medicine with 0 units in the inventory will throw an exception and the transaction will roll back.

Trigger Details

What it is meant to achieve and Why

We aim to maintain correct real-time count of medicines present in the Pharmacy with this trigger. This will be helpful in assignment of medicines. For instance, The abscence of a particular medicine may indicate the doctor to prescribe an alternative medicine or order a new batch of the empty medicine.

Event

- The trigger listens for changes (New Inserts / Order Changes) to the `prescription` table. This table maintains a mapping from an appointment (doctor visit) to the medicines that were prescribed during that appointment. For this scope, we assume that at most only 1 unit of medicine is prescribed to a patient.
- Upon a new entry in this table the `current_stock` of the medicine (identified by `fda_id`) in the `medicine` table should reduce by 1.
- Further, if there is no stock left to dispatch, the transaction should be aborted.
- Sample Queries to Recreate the EVENT
  - Select any 1 `consult_id` from the `consult` table
  - Select any 1 `fda_id` from the `medicine` table. Note the `current_stock` for this id.
  - With the `consult_id` and the `fda_id` create an entry in `prescription`. Ensure that this pair does not already exist.

Explaination and Data Modification

Following is the query that can be run to see how the data is modified. Upon running the trigger, the inventory of the medicine prescribes should be reduced by 1.

```
-- Using Consult_ID = 700010 and fda_id = 13240002 as an example
select * from medicine where fda_id = 13240002; --Note current_stock
INSERT INTO prescription values(700010, 13240002); -- Prescribe medicines after an appointment
select * from medicine where fda_id = 13240002; -- check the current_stock (should be reduced by 1)
```

Queries

Query 1 (Text Attributes)

- Patients who were recommended additional tests by their doctors or Patients who had some surgery done during their hospital visit

```
SELECT ps.first_name as "Patient First Name", ps.last_name "Patient Last Name",
em.first_name "Employee First Name", em.last_name "Employee Last Name",
cs.consult_id "Consult ID", cs.consult_notes_txt "Doctor Notes"
FROM consult cs, advise ad, employee em, patient ps
WHERE
  to_tsvector(cs.consult_notes_txt) @@ to_tsquery('test | surgery')
AND ad.consult_id = cs.consult_id
AND ad.employee_id = em.employee_id
AND cs.patient_id = ps.patient_id;
```

Query 2 (List Attributes)

- All pairs of patients who have **SOME** common medical history amongst them

```
SELECT p1.patient_id "First Patient - ID",
p1.first_name "First Patient - First Name",
p1.last_name "First Patient - Last Name",
p1.med_history "First Patient - Medical History",
p2.patient_id "Second Patient - ID",
p2.first_name "Second Patient - First Name",
```

```
p2.last_name "Second Patient - Last Name",
p2.med_history "Second Patient - Medical History"
FROM patient p1, patient p2
WHERE
  p1.med_history && p2.med_history
AND p1.patient_id < p2.patient_id;
```

Query 3 (Both in Combination)

- This query uses both Text and List attribute : Get details of patients who came in for a peanut allergy and also had a demonstrated medical history of peanut allergy

```
SELECT *
FROM
  patient pt, consult cs
WHERE
  cs.patient_id = pt.patient_id
AND pt.med_history && ARRAY['peanut allergy']::varchar(255)[]
AND to_tsvector(cs.consult_notes_txt) @@ to_tsquery('peanut*');
```