Project 2
ITIS 6120

Kelsey Kornegay, Matt Barksdale, Stephanie Karp, Will Lapinel

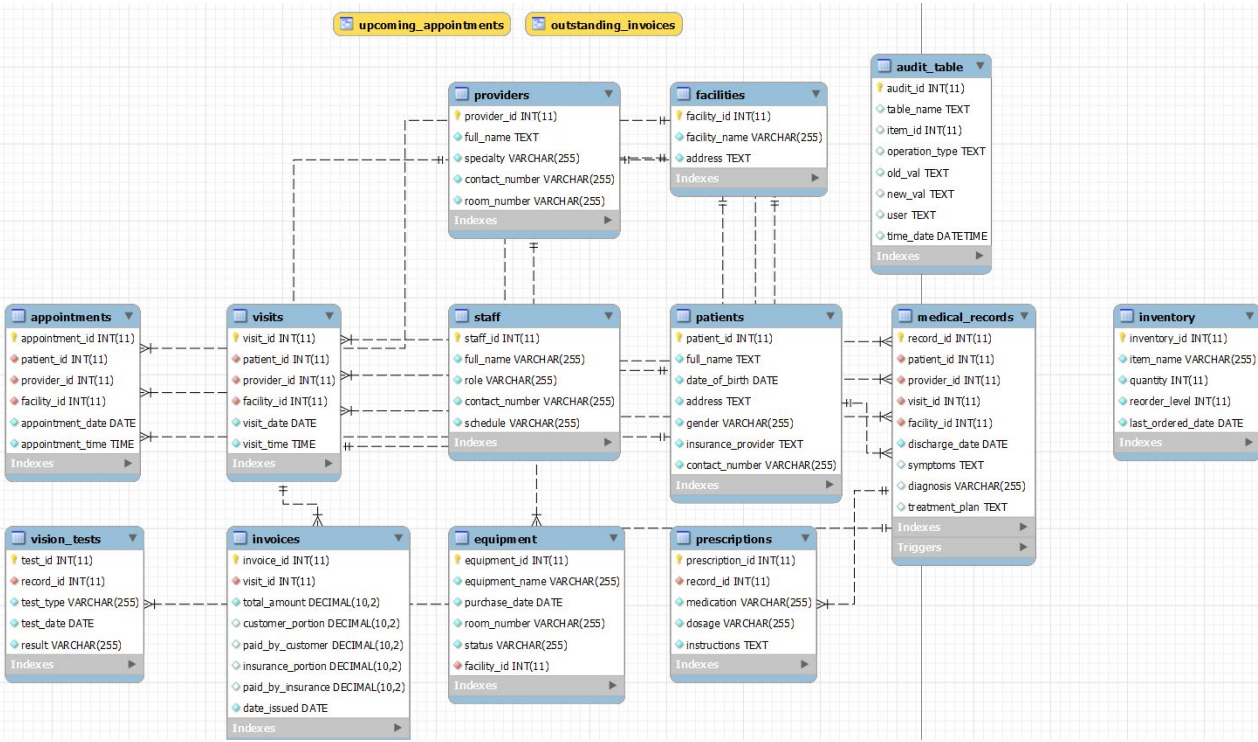
(Improved) Ophthalmology Database

Our database is designed to support the operations and administration of a hypothetical ophthalmology practice, with several different facility locations.

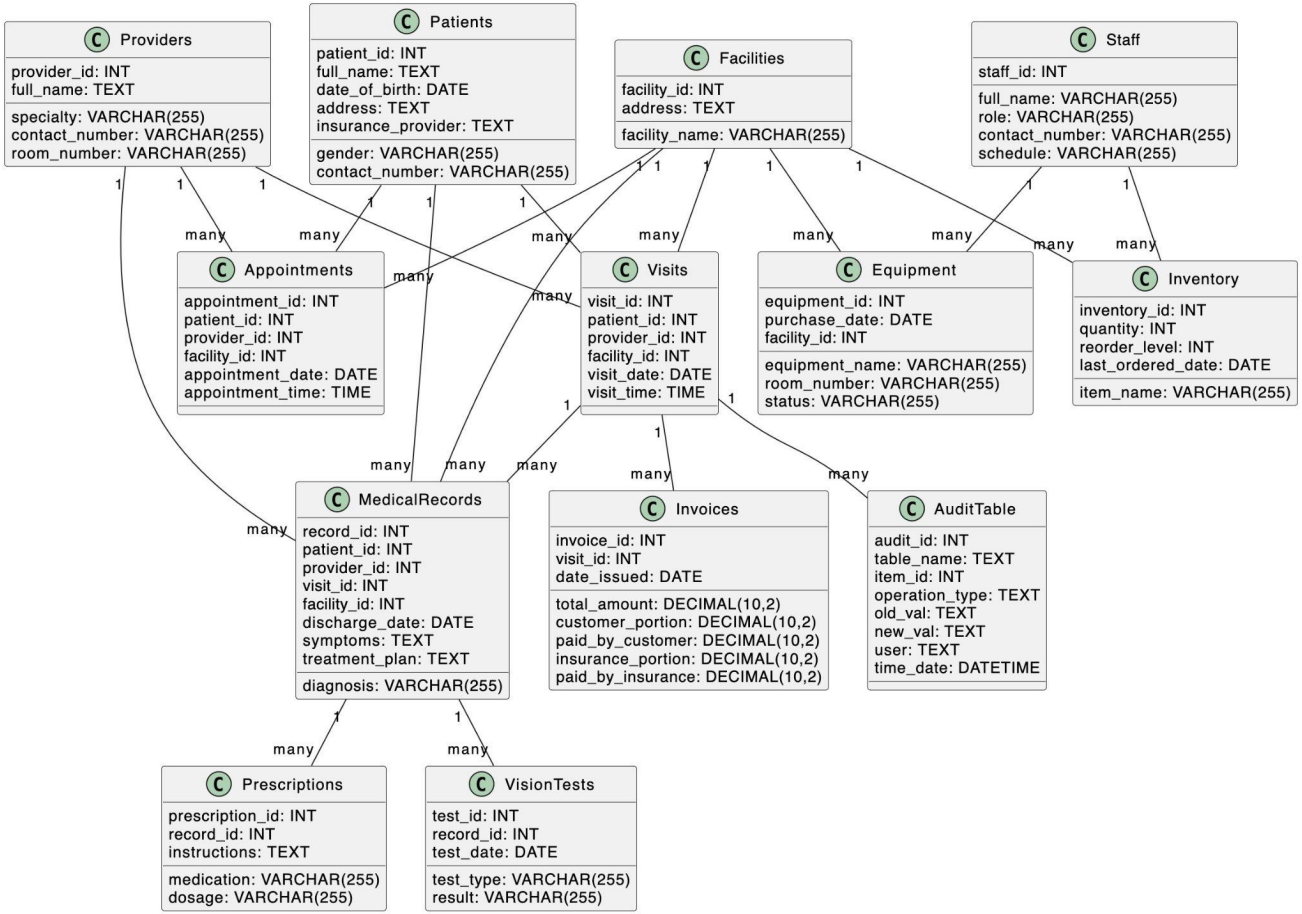
Entity Relationship Diagrams

Changes include:
views:
outstanding_balances and upcoming_appointments

table: audit_table



Unified Modeling Language Diagrams



Stored Procedures: Create Patient

```
DELIMITER $$
CREATE PROCEDURE `create_patient`(
    IN p_full_name text,
    IN p_dob date,
    IN p_address text,
    IN p_gender varchar(255),
    IN p_insurance text,
    IN p_contact_no varchar(255)
)
BEGIN
    DECLARE v_patient_id int;
    INSERT into patients (full_name, date_of_birth, address, gender, insurance_provider,
contact_number)
    VALUES (p_full_name, p_dob, p_address, p_gender, p_insurance, p_contact_no);

    SELECT LAST_INSERT_ID() INTO v_patient_id;

    SELECT v_patient_id as new_patient_id;
end$$
DELIMITER ;
```

Stored Procedures: Create Prescription

```
DELIMITER $$
CREATE PROCEDURE `create_prescription`(
    IN p_record_id int,
    IN p_medication varchar(255),
    IN p_dosage varchar(255),
    IN p_instructions text
)
BEGIN
    DECLARE v_prescription_id int;
    INSERT into prescriptions (record_id, medication, dosage, instructions)
        VALUES (p_record_id, p_medication, p_dosage, p_instructions);
    SELECT LAST_INSERT_ID() INTO v_prescription_id;
    SELECT v_prescription_id as new_prescription_id;
end$$
DELIMITER ;
```

Stored Procedures: Create Appointment

```
DELIMITER $$
CREATE PROCEDURE `make_appt`(
    IN p_date DATE,
    IN p_patient_id int,
    IN p_provider_id int,
    IN p_facility_id int,
    IN p_time time
)
BEGIN
    DECLARE appointment_id int;
    INSERT INTO appointments (patient_id, provider_id, facility_id, appointment_date,
appointment_time)
    VALUES (p_patient_id, p_provider_id, p_facility_id, p_date, p_time);
    SELECT LAST_INSERT_ID() INTO appointment_id;
    SELECT appointment_id AS new_appointment_id;
end$$
DELIMITER ;
```

Stored Procedures: Record Test Results

```
DELIMITER $$
CREATE PROCEDURE `record_test_results`(
    IN p_record_id int,
    IN p_test_type varchar(255),
    IN p_test_date date,
    IN p_result varchar(255)
)
BEGIN
    DECLARE v_test_id int;
    INSERT into vision_tests (record_id, test_type, test_date, result)
        VALUES (p_record_id, p_test_type, p_test_date, p_result);

    SELECT LAST_INSERT_ID() INTO v_test_id;

    SELECT v_test_id as new_test_id;
end$$
DELIMITER ;
```

Stored Procedures: Record Visit

```
DELIMITER $$
CREATE PROCEDURE `record_visit`(
    IN p_patient_id int,
    IN p_provider_id int,
    IN p_facility_id int,
    IN p_date date,
    IN p_time time
)
BEGIN
    DECLARE visit_id int;
    INSERT into visits (patient_id, provider_id, facility_id, visit_date, visit_time)
    VALUES (p_patient_id, p_provider_id, p_facility_id, p_date, p_time);

    SELECT LAST_INSERT_ID() INTO visit_id;

    SELECT visit_id as new_visit_id;
end$$
DELIMITER ;
```

Python CLI Application

```
Welcome to Patient Services. Please login.
Username: root
Password:
Welcome, root!
[1] Create Appointment
[2] Check in Patient
[3] Record Test Result
[4] Create Patient
[5] Create Prescription
Please choose an option:
```

```
Welcome to Patient Services. Please login.
Username: root
Password:
Appointment: 16
Visit: 12
Test Results: 11
Patient: 11
Prescription: 11
```

Python API functions: Create Patient

```
def create_patient(name, dob, address, gender, insurance, phone):
  cursor = db.cursor()
  args = (name, dob, address, gender, insurance, phone)
  cursor.callproc("create_patient", args)
  result = cursor.fetchone()
  cursor.close()
  return result[0]
```

Python API functions: Create Prescription

```
def create_prescription(record, medication, dosage, instructions):
  cursor = db.cursor()
  args = (record, medication, dosage, instructions)
  cursor.callproc("create_prescription", args)
  result = cursor.fetchone()
  cursor.close()
  return result[0]
```

Python API functions: Create Appointment

```
def make_appointment(date, patient, provider, facility, time):
  cursor = db.cursor()
  args = (date, patient, provider, facility, time)
  cursor.callproc("make_appt", args)
  result = cursor.fetchone()
  cursor.close()
  return result[0]
```

Python API functions: Record Test Results

```
def record_test_results(record, type, date, result):
  cursor = db.cursor()
  args = (record, type, date, result)
  cursor.callproc("record_test_results", args)
  result = cursor.fetchone()
  cursor.close()
  return result[0]
```

Python API functions: Record Visit

```
def record_visit(patient, provider, facility, date, time):
  cursor = db.cursor()
  args = (patient, provider, facility, date, time)
  cursor.callproc("record_visit", args)
  result = cursor.fetchone()
  cursor.close()
  return result[0]
```

Triggers: Create Audit Entries (Example)

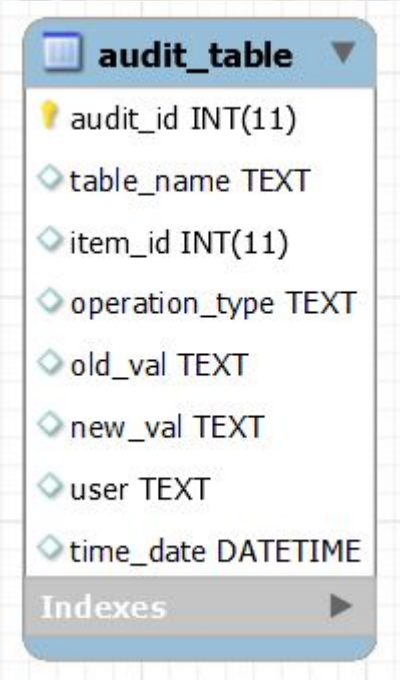
```
CREATE TRIGGER audit_table_update_medical_records
   AFTER UPDATE ON project_2.medical_records
   FOR EACH ROW
   BEGIN
      INSERT INTO audit_table (table_name, item_id, operation_type, old_val, new_val, user,
time_date)
         VALUES (
               'Medical_records', OLD.record_id, 'UPDATE',
               CONCAT_WS(',', CONCAT('provider_id:', OLD.provider_id), CONCAT('facility_id:',
OLD.facility_id), OLD.symptoms, OLD.diagnosis, OLD.discharge_date, OLD.treatment_plan),
               CONCAT_WS(',', CONCAT('provider_id:', NEW.provider_id), CONCAT('facility_id:',
NEW.facility_id), NEW.symptoms, NEW.diagnosis, NEW.discharge_date, NEW.treatment_plan),
USER(), NOW());
 end;
```

Users: Provider and Receptionist

```
create user 'provider'@'%' identified by 'password';

grant select, insert, update, delete on project_2.* to 'provider'@'%';


create user 'receptionist'@'%' identified by 'password';

grant select, insert, update, delete on project_2.appointments to 'receptionist'@'%';

grant select on project_2.prescriptions to 'receptionist'@'%';
```

New Table: Audit

Every change to the database will create a new row in the audit table due to aforementioned trigger



Indexes

```
CREATE INDEX idx_appointment_date_facility_id ON appointments(appointment_date, facility_id);

CREATE INDEX idx_patient_facility ON medical_records(patient_id, facility_id);
```

Views: Outstanding Invoices

```
create view outstanding_invoices AS
select v.visit_date, i.customer_portion, i.paid_by_customer, i.insurance_portion,
i.paid_by_insurance, p.full_name, p.contact_number, p.insurance_provider,
(total_amount - (paid_by_insurance + paid_by_customer)) as outstanding_balance
from invoices i
left join visits v
on i.visit_id = v.visit_id
left join patients p
on v.patient_id = p.patient_id
where (total_amount - (paid_by_insurance + paid_by_customer)) > 0
order by visit_date;
```

Views: Upcoming appointments

```
create view upcoming_appointments AS

select a.appointment_date, a.appointment_time, f.facility_name,
p.full_name from appointments a

Join patients  p ON a.patient_id = p.patient_id

JOIN facilities f ON a.facility_id = f.facility_id

WHERE appointment_date >= CURDATE()

ORDER BY appointment_date;
```