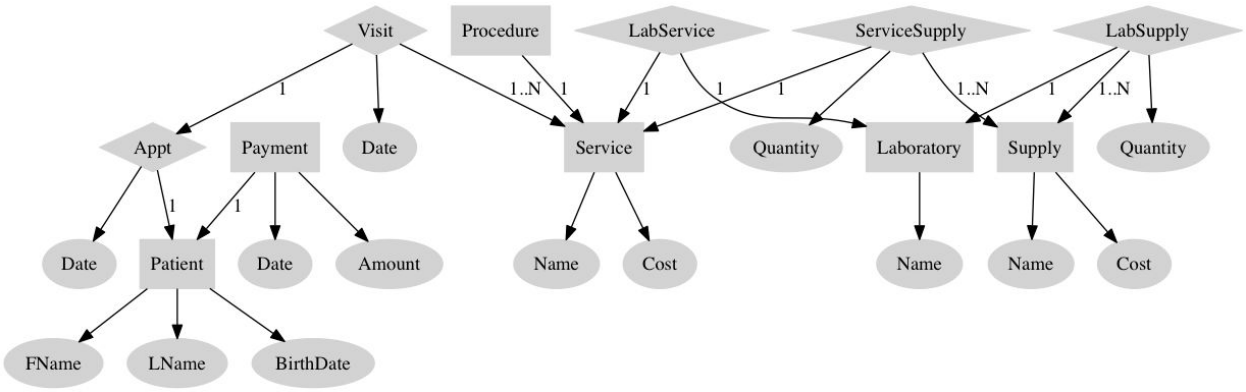
Conceptual Database Design



There are several considerations we made in designing our schema.
We wanted a separation between visits and appointments. Patients can create appointments and never visit. We knew that each visit should have an appointment but the reverse isn't always true.
We wanted to reduce the number of relationships. We realized that procedures were essentially a service offered by the dentist office much in the same way that a service is offered by a lab. By modeling the procedure as a weak-entity (i.e. it only contains a reference to the pk of service), we were able to get procedure supplies by virtue of service supplies. This reduced the number of relations necessary. At the same time it is trivial to get services offered by the dentist office v.s. those offered by a lab.
We decided to connect payments to a patient which is quite a bit looser than attaching them to a visit or service. We thought this flexibility would allow for unique scenarios like a charge to a customer who decided to cancel an appointment.
We imposed several constraints on our tables to ensure integrity. Each foreign key uses a "FOREIGN KEY … REFERENCES" clause to enforce referential integrity. We also make use of the unique constraint. In designing our application it was convenient to be able to refer to a tuple in a relation via a single identifier. Many of our relations have an id field which serves as a primary key. This extra identifier is extraneous to the record, so we use the unique constraint on the remaining fields.

Schema/Normalization analysis

Below we list the schemas with a justification for how each is in 3NF.

Patient	patient#	lname	fname	dob

Each non-prime key (lname, fname, dob) depends solely on the candidate key (patient#).

Appt	appt#	patient#	time

Each non-prime key (patient#, date) depends solely on the candidate key (appt#).

Visit	visit#	appt#	service#

The non-prime attributes are FFD on the only prime attribute.

Procedure	service#

A procedure is a classification of a service. It's a service offered by the dentist office (as opposed to one offered by a lab). There are no dependencies.

Service	service#	name	cost

Each non-prime key (name, date) depends solely on the candidate key (service#).

Supply	supply#	name	cost

Each non-prime key (name, date) depends solely on the candidate key (supply#).

ServiceSupply	servicesupply#	service#	supply#	quantity

The non-prime attributes are FFD on the only prime attribute.

LabService	labservice#	lab#	service#

There are only 2 FD for this relation: labservice# -> lab#, labservice# -> service#. So each non-prime is FFD on the only candidate key labservice#.

Lab	lab#	name

The only non-prime key (name) depends solely on the candidate key (lab#).

LabSupply	labsupply#	lab#	supply#	quantity

The non-prime attributes are FFD on the only prime attribute.

Payment	payment#	patient#	time	amount

Each non-prime key depends solely on the candidate key (payment#).

Query description

1. Transactions by patient
   The user is prompted for a patient#. For the given patient all charges and payments are returned in a table view. The columns include a timestamp the name of the transaction and the amount.
2. Appointments where the patient did not show up
   This user can view all appointments where a patient never showed up. In the schema we match all appointments that are missing visits, we ignore appointments that have not passed yet.
3. Today's appointments
   View all of today appointments.
4. Most used supplies
   We take each visit and enumerate all services performed. For each service performed we count all supplies used. We then join with all supplies. The join ensures that supplies are represented (with a use count of 0) even if they are not part of a service.
5. Patients with birthdays this month
   Filter all patients by date of birth matching the current month.