Design and Development of a Relational Database

Daniel Guthart, Jade Tustin
Professor: Vanessa Aguiar

Submitted in partial fulfillment
of the requirements for the course project
for CSC 423 at the University of Miami

November 24, 2024

Table of Contents

1. Development of Logical Data Model

Now that a clear conceptual data model has been established, a logical data model can be constructed using the conceptual model as a basis and reference point. We work through the steps necessary to create this new model in this section.

1.1. Relations from Conceptual Model

To establish the necessary relations, we can look to the conceptual model and the work completed in section 1.

First, since all of the entities identified were strong entities, these can be expressed as their own relations with the attributes identified in the previous section. The only necessary change would be to convert the complex attributes to simple attributes (for example, name should be split into first name and last name for Clinic, Staff, and Owner). The list below shows all entities and their attributes reflecting this change.

Clinic (clinicNo, name, street, buildingInfo, city, state, ZIPcode, telephone)
Primary Key clinicNo
Alternate Keys name, (street, buildingInfo, city, state, ZIPcode), telephone

Staff (staffNo, firstName, lastName, street, buildingInfo, city, state, ZIPcode, telephone, DOB, position, salary)
Primary Key staffNo
Alternate Keys (firstName, lastName, DOB), telephone

Owner (ownerNo, firstName, lastName, street, buildingInfo, city, state, ZIPcode, telephone)
Primary Key ownerNo
Alternate Key telephone

Pet (petNo, petName, DOB, species, breed, color)
Primary Key petNo
Alternate Key (petName, DOB)

Examination (examNo, chiefComplaint, description, dateSeen, actionsTaken)
Primary Key examNo

We can now look towards the relationships between entities that were described in the previous section and make the appropriate changes to the model. For 1:* relationships in the model, the entity on the 1 side of the relationship posts a foreign key to the entity on the many side of the relationship. This applies to almost every single relationship detailed in the conceptual model (Has, Registers, Belongs To, Undergoes, and Performs).

The only unaffected relationship is Manages, which is a 1:1 relationship. These relationships are handled by checking the requirements of participation of entities in the relationship. In this instance, the Staff side is optional (a staff member may or may not manage their clinic) whereas the Clinic side is mandatory (every clinic must have a member of staff managing it). This type of relationship is handled very similarly to the 1:* relationships; the entity with mandatory participation (Clinic) is the child entity, and the entity with optional participation (Staff) serves as the parent entity. Again, the parent entity posts their primary key into the child entity.

Based on this information, we can now present the finalized sets of attributes for the logical model.

Clinic (clinicNo, name, street, buildingInfo, city, state, ZIPcode, telephone, staffNo)
Primary Key clinicNo
Alternate Keys name, (street, buildingInfo, city, state, ZIPcode), telephone
Foreign Key staffNo **references** Staff(staffNo)

Staff (staffNo, firstName, lastName, street, buildingInfo, city, state, ZIPcode, telephone, DOB, position, salary, clinicNo)
Primary Key staffNo
Alternate Keys (firstName, lastName, DOB), telephone
Foreign Key clinicNo **references** Clinic(clinicNo)

Owner (ownerNo, firstName, lastName, street, buildingInfo, city, state, ZIPcode, telephone)
Primary Key ownerNo
Alternate Key telephone

Pet (petNo, petName, DOB, species, breed, color, ownerNo, clinicNo)
Primary Key petNo
Alternate Key (petName, DOB)
Foreign Key ownerNo **references** Owner(ownerNo)
Foreign Key clinicNo **references** Clinic(clinicNo)

Examination (examNo, chiefComplaint, description, dateSeen, actionsTaken, petNo, staffNo)
Primary Key examNo
Foreign Key petNo **references** Pet(petNo)
Foreign Key staffNo **references** Staff(staffNo)

1.2. Validation of Logical Model

Before proceeding with the construction of the logical model and then implementing it into a real database, it is important to validate the model. Two key methods for doing so are normalization and validation via user transactions. We perform both types of validation in this section.

1.2.1. Normalization to 3rd Normal Form (3NF)

To start, we will assume that the model is already in first normal form (1NF). 1NF demands that there are no row-column intersections (cells) in any table containing multiple values. Since we have not worked with any sample data yet, we can assume that data will be organized in such a way that it complies with this normalization.

To make the logical model adhere to second normal form (2NF), all partial dependencies in the relations must be removed and inserted into a separate relation. A partial dependency occurs where a subset of the primary key uniquely identifies an attribute. However, since no entities use composite primary keys, there are no partial dependencies and the data model adheres to 2NF.

Finally, to convert the logical model into 3rd normal form (3NF), transitive dependencies need to be placed in separate relations. In other words, if the primary key uniquely determines the set of attributes, but any of those attributes in turn determine another, then a transitive dependency exists. This is not the case for any of the entities, as there are no non-primary-key attributes determining other attributes. As such, the model is in 3rd normal form.

1.2.2. User Transactions

We can now use the normalized logical model to verify that we can complete some sample transactions. Below, we have listed some sample transactions, and how they would be carried out in the database.

- **Register a new pet owner and their pet**
    - First, add a record to the Owner table, providing the necessary information.
    - After that, add a record to the Pet table, ensuring the pet references the correct owner and supplying data for the other attributes.
- **Assign a staff member to manage a clinic**

    - Update the Clinic table by setting the staffNo attribute for a staff member. Take care to reference the correct staff member.

- **Log a pet's examination**
    - Add an entry to the Examination table with references to the Pet and Staff involved. Ensure that these references are to the correct pet and staff member.
- **Retrieve all examinations for a specific pet**
    - Query the Examination table filtered by petNo.
- **List all staff working at a specific clinic**
    - Query the Staff table filtered by clinicNo.

1.3. Integrity Constraints

At this stage, we should define a set of constraints in a manner that safeguards the database against inconsistencies, incorrectness, or incongruence with its stated aims and functions.

1.3.1. Primary Key Constraints

Per the definition of a primary key, primary keys must be both unique to each tuple in a relation and non-null. Aside from this, we can determine the following constraints for each primary key. We may define some assumptions to help create an appropriately large range of values for each entity, but these are not used for any other purpose.

- **Clinic**

  The primary key for Clinic is clinicNo. This should be a unique, 5-digit integer. Restricting this key to 5 digits is somewhat of an arbitrary choice, but a small number has been chosen as this entity will likely have the least tuples in the entire database. This allows up to 100,000 clinics to be stored in the database (with IDs ranging from 00000 to 99999), making the probability of running out of unique IDs for new clinics extremely unlikely. We will take this upper bound into consideration when defining constraints for other entities.

- **Staff**

The Staff entity holds a primary key, staffNo. When defining the constraints for clinicNo, we allowed for a maximum of 100,000 clinics. We will assume that a clinic is unlikely to have more than 50 employees. Taking this into account, this gives a maximum staff count of 5,000,000.

To provide a constraint wide enough to account for these staff members, we can use a 7-digit integer. We can also define this field in such a way that there is no overlap between the ranges of Staff and Clinic; this is optional, but may assist future users of the database and avoid confusion. Doing this, we can start the range of staffNo at 0100000 and end it at 9999999. This allows for up to 9,900,000 employees to be recorded in the database, which grants plenty of additional headroom for either hiring additional staff or expanding the enterprise.

- **Owner**

  The Owner entity uses ownerNo as its primary key. To obtain an estimate for the necessary range, we will assume that owners visit the clinic once per year and visit with one pet. We will also assume that the clinic is open for 350 days per year (allowing for 3 business weeks of holiday time but excluding weekends). If the clinic is open for 8 hours and each appointment takes an hour, the clinic will perform 2800 examinations per year. Taking this space and expanding it across the maximum number of clinics, this would imply 280,000,000 examinations, and based on our assumptions, that many owners.

  Thus, a 9-digit integer field would suffice for Owner. This entity will continually grow over time, and it may be helpful to double the estimate we made in order to provide a buffer for tuples added later. We can account for 560,000,000 unique owners (double the original estimate) by using an ID range starting at 440000000 and ending at 999999999. This also makes ownerNo completely distinct from clinicNo and staffNo when evaluating range.

- **Pet**

  The primary key for Pet is petNo. As we just established, the Owner entity can account for up to 560,000,000 unique owners. For the field estimate, we will assume that owners may have up to 10 pets. Given this, we would need to account for up to 5,600,000,000 pets across the entire system.

  Starting with a 10-digit integer value would be appropriate here. However, it may be helpful to define the range in such a way that it is distinct from

ownerNo. We will not make a range sufficient for double the field in this case, as it is highly unlikely that there would be 560,000,000 unique owners in the first place, and even more unlikely that they would each have 10 pets. As such, we can define the range to start at 1000000000 and end at 6599999999. This accounts for the estimate we created for the number of pets in addition to making petNo entirely distinct in range from ownerNo, clinicNo, or staffNo.

- **Examination**

  Examination is the table that we are perhaps most concerned with, as new tuples will be regularly added to it. We previously estimated that 280,000,000 examinations per year would occur if they each took an hour, the maximum number of clinics exist, each one is open 8 hours per day, and they operate for 350 days per year. Using a field one digit larger than the number of digits indicated by the estimate (10 digits instead of 9) would be useful, but since we already used this range for petNo, it may be useful to use an 11-digit integer. Since we want to provide plenty of space for growth here, we will use the entire range (00000000000 to 99999999999), accounting for up to 100 billion examinations. This should be more than enough.

1.3.2. Foreign Key Constraints

We have already defined some important constraints on the primary keys for each relation, but it is also important to define conditions for the foreign keys. None of the foreign keys in any of the relations may be null, as they are all required by the particular relationship (e.g. an examination must be carried out by a member of staff and on a particular pet).

What is less clear is how to handle cases where primary keys in a table are deleted or otherwise modified. Since we have already stated that these foreign keys are required in all cases and cannot be null, we must evaluate the nature of the relationship between entities and make appropriate decisions.

For Owner, any changes or deletions of the ownerNo should cascade throughout the table, deleting any pets they own from the Pet entity and in turn removing any examinations conducted from the Examination table. The same should be done for Pet; any changes or deletions to petNo should cascade through the table.

The Staff entity presents a more complicated situation. If staffNo was changed or deleted, we could not make these values null where they are used as foreign keys, and we would not want to remove entries in either Clinic or Examination. This

really leaves us with only two options; set the key to some default value or disallow the action. Since it may be necessary to remove staff members from the database, it makes more sense to set this key to some default value. In this case, the ideal default value would be the manager of the clinic.

The only remaining case to handle is changes or deletions of primary keys in Clinic. One of the affected entities here would be Pet, and entries there should not be deleted if entries in Clinic are, so the key should be updated to an appropriate default value. An example of an acceptable default value would be the clinicNo of the nearest clinic to the manager of the deleted clinic.

1.3.3. Alternate Key Constraints

No constraints will be placed on alternate keys; any constraints on attributes not including the primary key will be mentioned in section 1.3.5.

1.3.4. Required Data

For each entity, we can specify what attributes are required and which may be left null. By definition, primary keys may not be null. Foreign keys may not be null either for reasons outlined in section 1.3.2.

For Clinic, the attributes name (firstName, lastName), telephone, and address (street, buildingInfo, city, state, ZIPcode) are required. This information is important for clients and should be stored. Additionally, based on the foreign key constraints we defined, the address could be used to set the default/fallback foreign keys in the event of a tuple deletion.

For Staff, the attributes name (firstName, lastName), telephone, position, and salary are required. These are essential attributes, as they establish a point of contact, role within the firm, and payout rate, respectively. DOB and address (street, buildingInfo, city, state, ZIPcode) may be null.

For Owner, the attributes name (firstName, lastName), address (street, buildingInfo, city, state, ZIPcode) and telephone are required. Making this information mandatory has practical uses; for example, the owner’s address may be used to send billing documents.

For Pet, the attributes petName and species are required. Maintaining these attributes makes it easy to identify an owner’s pets and administer the proper care. The other non-foreign-key fields (DOB, breed, color) may be null.

For Examination, the attributes chiefComplaint, description, dateSeen, and actionsTaken are all required. Since a pet's owner or the staff member operating on a pet could need that information at any time, it is necessary for it to be stored.

1.3.5. Attribute Domain Constraints

In our data model, many of the tables are responsible for storing similar types of attributes. As such, we will refer to these attributes generally as opposed to going entity-by-entity. For clarification, we define a short string as being less than 30 characters, an intermediate-length string as being less than 75 characters, and a long string as being over 75 characters.

Attributes storing a name value (firstName, lastName, petName) should be a short string.

All fields holding a telephone number (telephone) should be constrained to a 10-digit number in accordance with United States telecommunications standards. (We assume this case study is of a clinic located within the country.)

Fields containing a date (DOB, dateSeen) should be restricted to a standard date format. Month-first formatting (MM-DD-YYYY) would be appropriate.

Any attribute containing a street name, building information, or city (street, buildingInfo, city) should be constrained to a short string.

Attributes containing a state (all instances of state) should be restricted to two characters only. This should be further restricted to matching the two-character abbreviations of any of the 50 US states, including the District of Columbia, but this is ultimately optional.

The position and salary attributes in Staff should be set to a short string and a decimal number, respectively.

All attributes in Examination that are not a primary key, a foreign key, or a date should be strings. Specifically, chiefComplaint should be an intermediate-length string, whereas actionsTaken and description should be long strings.

The fields species, breed, and color in Pet should all be short strings. Furthermore, entries in species should be restricted to valid names for animal species if possible.

1.3.6. General Constraints

We opt not to place any general constraints on the attributes within the database.

1.4. Entity-Relationship Diagram for Logical Level