**The first user group: Vet and Vet Technicians.** These users must be able to insert and read data about care being performed on the patients. This information includes vitals measurements, as well as any treatments or procedures. Vitals, such as age, weight, and blood pressure are all numerical in nature. Certain obvious constraints apply to these categories, such as they cannot be negative, age and bp are integers where weight can be a decimal number (only out to the tenths place). There also must be a few text fields with less constraints. There must be a varchar field, perhaps 30 characters, such that non-numerical treatments can be entered: Name of medicine (Prozac, Advil, CPR, etc), and administered dosage (varchar, so we don’t have to mess with unit conversions). Additionally, a large text field (1-2k varchar) must exist for important notes relating to the procedure. There are little constraints on this information, it is left as very freeform text. This field is most important for more complicated/extensive procedures. All of this treatment information must be stored, along with date information, so as to maintain a record of the patient’s health over time. The patient (animal) itself should be stored in a relation of its own and joinable to the vitals/treatments. The patient will have information such as its name, owner, species and breed. These are all varchar fields.  
  
**The second user group: Front desk, clerical, administrative** **staff.** These users are primarily involved in billing and scheduling. These users need access to the care provided to a patient, and they need to be able to join this data into a relation that contains costs of procedures. The treatment relation contains numerical data (out to the hundredths place) for the cost of a standard dosage. This treatment table needs to join against both the patients care table. If a patient receives a non-standard dosage, the cost can be calculated as a fraction either in the SQL itself, or the application that the user is using. Calculating cost on the fly is a difficult task, as dosages are not all measured in the same units (pills, ounces, minutes). Either way, often the amount owed will not typically match the amount in the ‘cost’ table, as the administered dosage likely won’t match the standard dosage we have a cost stored for. This user group must have access to scheduling. Thus, there must be a relation containing date and time information. Not all vet visits are the same length, so there must be a concept of a ‘duration’ as well. This can be implemented as a start and end time (time data type). One might be able set up constraints to prevent double-booking, although this could be tricky. The scheduling application software could also be used to protect against this. Finally, a table of veterinarians is necessary, as there could be multiple vets with appointments simultaneously. Vet will be a first name (varchar), last name (varchar), and id (int, unqiue). It might not be a bad id to set this up with a ‘role’ varchar attribute, in case this DB was expanded later to include things such as payroll or scheduling. This implementation does not plan to service those needs, but is easily extensible in the future.

**The last user group #3: Customers**. Customers need to have access to a lot of the same information as the first two users, but in a much more restricted sense. Customers should have access to scheduling information only pertinent to their pets. In addition, they are given access to itemized receipts for the care of their pets, but do not have access to the full care notes, nor do they have access to the marginal cost of the medicines/procedures. For online billing, customers will have access to their account, which is a relation that contains customer first and last name (varchar), a unique customer id (int), and a FK join in all of the pets that they own. Additionally, the customer should have insert/update access to a relation that contains billing information. The customer should be able to store credit card information, including the CC number (varchar), expiration date (date), and the security code (varchar). Finally, there should be several attributes for billing‑street address (varchar), zipcode (varchar), town and state (varchar for both). The billing information should be one to many, as one user may have several cards on file.

Combining all of my requirements: First and foremost, I did not see the need to make any compromises in terms of attribute data types, so the attributes as laid out above can stay as listed. In the first design, there will be a relation containing information about care (“chart”), which has attributes described in the first section. This relation is accessible by all three user groups, although the customers will not have access to the detailed notes section. This relation will need to be joinable to a “treatment” relation, which contains medicines and procedures, as well as a standard dosage/time, and a marginal cost. The ‘chart’ relation will need to also be joinable against the “vet” table, and the “patient” table. I will not make it joinable against the scheduling table, however, and thus the chart table will have its own date stored.

The vet table will be joinable against the schedule table as well as the appointment table. The patient table (containing information about pets) needs to contain a natural way to join it again chart, schedule, as well as customer. Customer only needs to be joinable against schedule, patient, and billing.  
 Overall, making this system work for the three user groups seems to be more of granting/restricting access, rather than converting between different domains.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Unique, not null | ON DELETE RESTRICT  Don’t leave orphaned patients in the DB |
| fname | varchar | 50 | Not null |  |
| lname | varchar | 50 | Not null |  |
| phone | varchar | 11 | Not null |  |
| email | varchar | 50 |  |  |

**Primary Key(s):**

id

**Foreign Key(s):**

**Indices:**

fname, lname

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PATIENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Unique, not null | None, no tables reference this. |
| name | varchar | 50 | Not null |  |
| species | varchar | 30 | Not null |  |
| breed | varchar | 30 |  |  |
| customer\_id | int | 8 | Not null |  |

**Primary Key(s):**

id

**Foreign Key(s):**

customer\_id REFERENCES customer(id)

**Indices:**

customer\_id, species

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TREATMENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Unique, not null | None, no tables reference this one. |
| treatment | varchar | 30 | Not null |  |
| dosage | varchar | 15 |  |  |
| cost | numeric | 7, 2 | Positive |  |

**Primary Key(s):**

id

**Foreign Key(s):**

**Indices:**

Treatment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VITALS | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Unique, not null | ON DELETE RESTRICT  This should not be deleted unless the whole appointment is. |
| weight | numeric | 5, 1 | Not null |  |
| diastolic\_bp | smallint |  | Positive |  |
| systolic\_bp | smallint |  | Positive |  |

**Primary Key(s):**

id

**Foreign Key(s):**

**Indices:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| APPOINTMENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| date | date |  | Not null | None, no tables reference this one. |
| patient\_id | int | 8 | Not null |  |
| vet\_id | int | 8 | Not null |  |
| vitals\_id | Int | 8 | Not null |  |
| treatment\_id | varchar | 30 | Not null |  |
| administered\_dosage | varchar | 15 |  |  |
| cost | numeric | 7, 2 | Positive |  |

**Primary Key(s):**

date, patient\_id, treatment\_id

**Foreign Key(s):**

patient\_id REFERENCES patient(id)

treatment\_id REFERENCES treatment(id)

vet\_id REFERENCES employee(id)

vitals\_id REFERENCES vitals(id)

**Indices:**

vet\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMPLOYEE | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Not null, unique | Yes, make FK null in other tables, rather than delete tuples. |
| fname | varchar | 50 | Not null |  |
| lname | int | 50 | Not null |  |

**Primary Key(s):**

id

**Foreign Key(s):**

**Indices:**

lname

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BILLING | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| id | int | 8 | Not null, unique | None, no tables references this one. |
| customer\_id | int | 8 | Not null |  |
| card\_no | varchar | 16 | Not null, numerical |  |
| exp\_date | date |  | Not null |  |
| security\_code | varchar | 4 | Not null, numerical |  |
| street\_address | varchar | 30 | Not null |  |
| town | varchar | 30 | Not null |  |
| state | varchar | 30 | Not null |  |
| zip | char | 5 | Not null |  |

**Primary Key(s):**

id

**Foreign Key(s):**

customer\_id REFERENCES customer(id)

**Indices:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SCHEDULE | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| customer\_id | int | 8 | Not null | None, no tables reference this one. Appointment cancellations are expected |
| patient\_id | int | 8 | Not null |  |
| date | date |  | Not null |  |
| time\_start | time |  | Not null |  |
| time\_end | time |  | Not null |  |
| vet\_id | int | 8 | Not null |  |

**Primary Key(s):**

patient\_id, date

**Foreign Key(s):**

customer\_id REFERENCES customer(id)

patient\_id REFERENCES patient(id)

vet\_id REFERENCES vet(id)

**Indices:**

vet\_id, date, time\_start