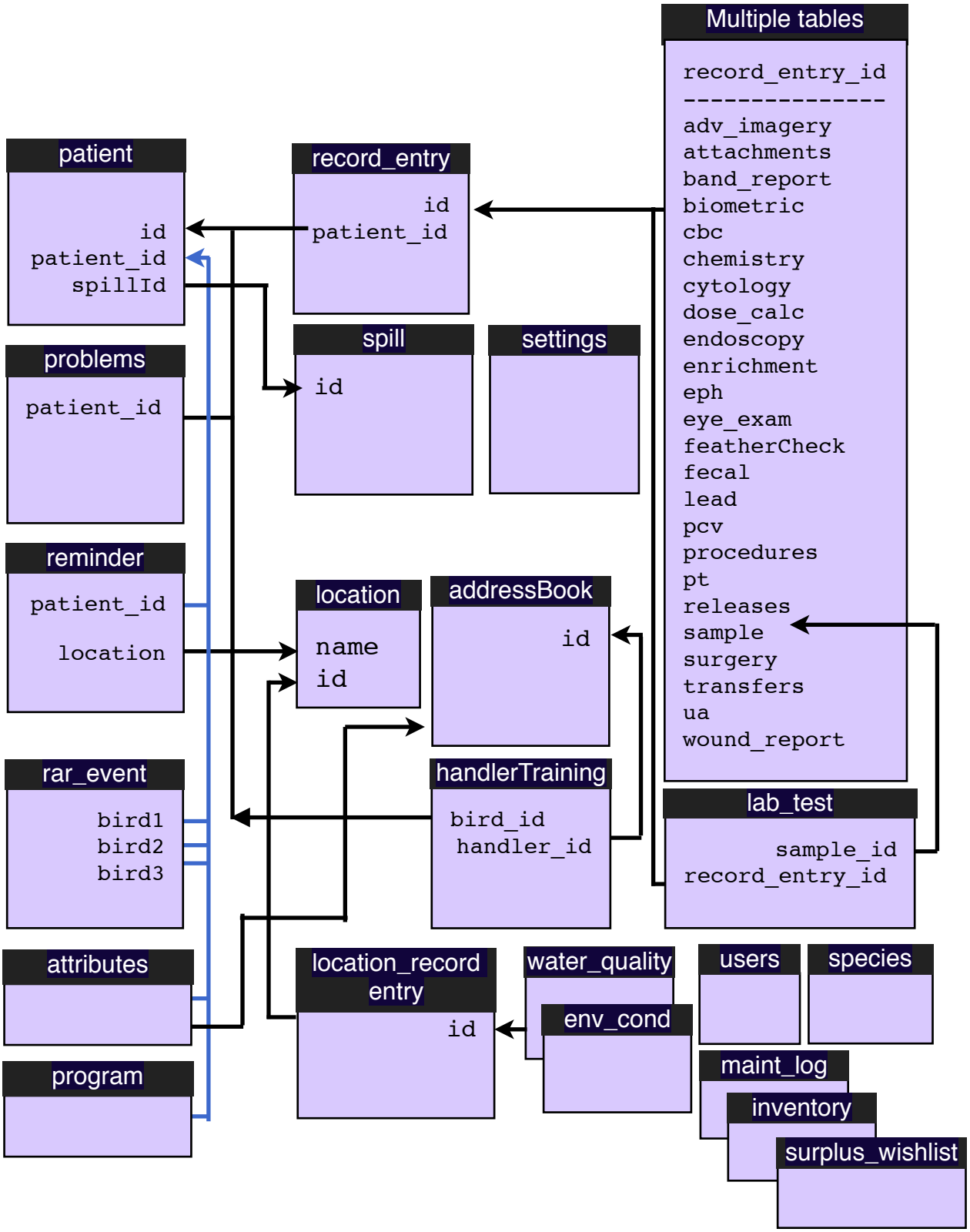
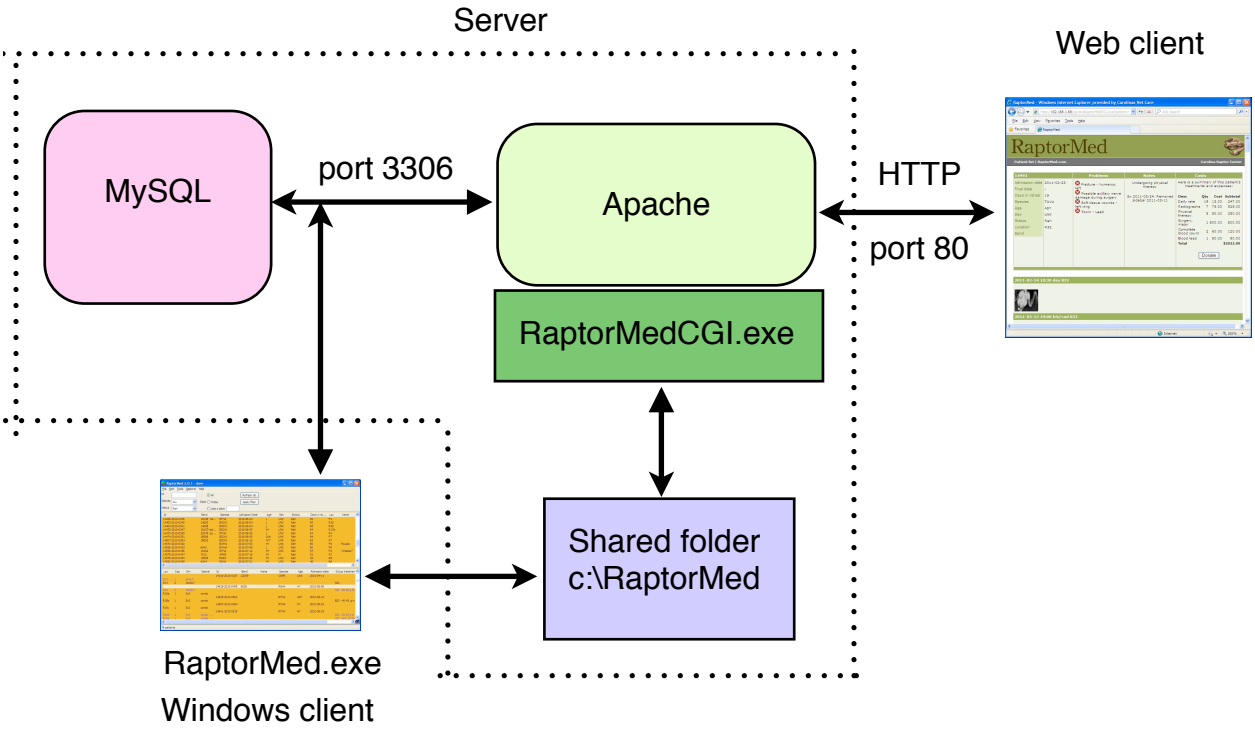
July 16, 2016

Technical Reference Manual

Table of Contents


Multiple tables
record_entry_id
adv_imagery
attachments
band_report
biometric
cbc
chemistry
cytology
dose_calc
endoscopy
enrichment
eph
eye_exam
featherCheck
fecal
lead
pcv
procedures
pt
releases
sample
surgery
transfers
ua
wound_report
patient
id
patient_id
spillId
record_entry
id
patient_id
problems
patient_id
spill
id
settings
reminder
patient_id
location
location
name
id
addressBook
id
rar_event
bird1
bird2
bird3
handlerTraining
bird_id
handler_id
lab_test
sample_id
record_entry_id
attributes
program
location_record entry
id
water_quality
env_cond
users
species
maint_log
inventory
surplus_wishlist

Architecture



Database version: 3.12.3
MySQL version: 5.0.67 or 5.7.12 community edition
Apache 2.2.17 with openssl

Database Schema

Table: addressBook
This table is a generic address book that stores contact information for various types of people and/or entities and organizations. For example, it can store data for volunteers and rehabbers as well as for suppliers, government agencies as well as for commonly used release sites. If a latitude and longitude are provided, these entries can be plotted on a Google Map by the system.

name	type	description
id	int	The PRIMARY KEY.
org	char(63)	The name of the organization.
contactName	char(31)	The name of the person/main contact.
addr	char(32)	
addr2	char(32)	
city	char(32)	
dob	date	Date of birth
state	char(2)	
zip	char(10)	
county	char(15)	
country	char(15)	
phone1	char(32)	
phone2	char(32)	
phone3	char(32)	
email	char(63)	
webAddress	char(64)	
permit	char(127)	Permit information for organizations that keep resident animals or do rehab.
lookingFor	char(63)	Free form text describing animals that this organization is looking to add to their permanent collection.

name	type	description
lat	float	The latitude of the organization. Necessary in order to place this entry on the Google Map.
lon	float	The longitude of the organization. Necessary in order to place this entry on the Google Map.
type	int	This field stores several yes/no values for the various types. More than one can be “yes”. ADDRESS_BOOK_TYPE_TRANSPORTER 0x00000001 ADDRESS_BOOK_TYPE_ANIMAL_CONTROL 0x00000002 ADDRESS_BOOK_TYPE_REHABBER 0x00000004 ADDRESS_BOOK_TYPE_FALCONER 0x00000008 ADDRESS_BOOK_TYPE_VOLUNTEER 0x00000010 ADDRESS_BOOK_TYPE_GOV_AGENCY 0x00000020 ADDRESS_BOOK_TYPE_SUPPLIER 0x00000040 ADDRESS_BOOK_TYPE_HANDLER 0x00000080 ADDRESS_BOOK_TYPE_EDUCATOR 0x00000100 ADDRESS_BOOK_TYPE_STAFF 0x00000200 ADDRESS_BOOK_TYPE_INACTIVE 0x00000400 ADDRESS_BOOK_TYPE_ZOO 0x00000800 ADDRESS_BOOK_TYPE_VET 0x00001000 ADDRESS_BOOK_TYPE_ACCEPT_TEXT_MSG 0x00002000 ADDRESS_BOOK_TYPE_SITE 0x00004000 ADDRESS_BOOK_TYPE_COLLECTION_TEAM 0x00008000 ADDRESS_BOOK_TYPE_FISHERMAN 0x000010000
availability	char(64)	A description of when this person is available to help with transport. For example “Weekday nights and weekends”.
notes	char(128)	Any special notes about this contact/address book entry.
cellProvider	char(31)	The cell phone provider. This is used for primarily for transporters and allows the RaptorMed software to send an automated text message when an animal needs transport.

name	type	description
daily_availablility	unsigned int	This field stores several yes/no values for availability time slots. More than one can be “yes”. This field is used primarily by the system to indicate when it is permissible for a text message to be sent to transporters. ADDRESS_BOOK_AVAIL_SUN_AM 0x0001 ADDRESS_BOOK_AVAIL_MON_AM 0x0002 ADDRESS_BOOK_AVAIL_TUE_AM 0x0004 ADDRESS_BOOK_AVAIL_WED_AM 0x0008 ADDRESS_BOOK_AVAIL_THU_AM 0x0010 ADDRESS_BOOK_AVAIL_FRI_AM 0x0020 ADDRESS_BOOK_AVAIL_SAT_AM 0x0040 ADDRESS_BOOK_AVAIL_SUN_PM 0x0080 ADDRESS_BOOK_AVAIL_MON_PM 0x0100 ADDRESS_BOOK_AVAIL_TUE_PM 0x0200 ADDRESS_BOOK_AVAIL_WED_PM 0x0400 ADDRESS_BOOK_AVAIL_THU_PM 0x0800 ADDRESS_BOOK_AVAIL_FRI_PM 0x1000 ADDRESS_BOOK_AVAIL_SAT_PM 0x2000

Table: adv_imagery
This table records procedures involving advanced imagery techniques such as MRI, CT, or U/S. These records are attached to individual record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the band report.
type	char(31)	The type of imagery (i.e. CT, MRI, Ultrasound)
who	char(15)	The person performing the procedure and/or interpreting the results
equipment	char(32)	The machine used
location	char(127)	The location of the body imaged
anesthesia	char(127)	The anesthesia protocol used
comments	TEXT	General comments regarding results.

Table: attachments
This table allows various files to be attached to individual medical record entries or to entries for an enclosure. The attached files can be of any type. Some (such as jpeg files) are viewable directly in the RaptorMed system. Others are viewable when launched with the appropriate application. These records are attached to individual record entries.

name	type	description
id	int	The PRIMARY KEY.
One of the next two columns must have a valid value.		
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
location_record_entry_id	int	FOREIGN KEY to the LOCATION_RECORD_ENTRY table.

name	type	description
name	char(64)	The name/short description for this attachment. Only used for attached documents to provide a short description.
type	char(16)	The file extension of the attachment. This is used like a MIME type in order to launch the correct viewer application.
image_type	char(32)	For all image attachments, this field specifies what type of image it is. Some example include, radiograph, photograph, microscopic image. This list is configurable.

Table: attributes
This table stores random name/value pair attributes that can be stored for individual animals or for an entry in the address book. These records are attached to individual PATIENT records or ADDRESS_BOOK records.

name	type	description
id	int	PRIMARY KEY.
patient_id	int	FOREIGN KEY to PATIENT table.
address_book_id	int	FOREIGN KEY to ADDRESSBOOK table.
date	date	
name	char(32)	The name of the attribute. This is mandatory.
value	char(64)	The value of the attribute. This optional.

Table: band_report
This table stores band report data and it is compatible with the data required and used by the USGS Bird Banding Lab. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the band report.

name	type	description
lat	float	The latitude (in decimal degrees) of the report.
lon	float	The longitude (in decimal degrees) of the report.
finder_info	char(255)	Information about the person who made the report.
location	char(255)	A description of the where the report was made.
notes	char(128)	Any additional notes.
present_condition	unsigned int	The 'condition' code as outlined on the USGS BBL web site.
how	unsigned int	The 'how' code as outlined on the USGS BBL web site.
band	char(65)	The band number.
aux_marker	char(15)	The auxiliary marker.

Table: biometric

This table stores biometric data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the measurement.
weight	float	The weight (in grams)
wc	float	The wing chord (in mm)
hallux	float	The hallux measurement (in mm)
culmen	float	The culmen measurement (in mm)
length	float	In mm
fork_length	float	In mm
standard_length	float	In mm
girth	float	In mm
disc_width	float	In mm

name	type	description
disc_length	float	In mm
straight_NPN_length	float	In mm
curved_NPN_length	float	In mm
straight_carapace_width	float	In mm
curved_carapace_width	float	In mm
sv_length	float	The snout to vent length (in mm)

Table: cbc
This table stores complete blood count data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the blood was collected.
count	int	The total white count.
corrected_count	int	The total white count after correction for the PCV.
pcv	int	The packed cell volume
het_perc	int	The heterophil percentage.
lymph_perc	int	The lymphocyte percentage.
eos_perc	int	The eosinophil percentage.
mono_perc	int	The monocyte percentage.
baso_perc	int	The basophil percentage.
azur_perc	int	The azurophil percentage
het_abs	int	The absolute heterophil count.
lymph_abs	int	The absolute lymphocyte count.
eos_abs	int	The absolute eosinophil count.

name	type	description
mono_abs	int	The absolute monocyte count.
baso_abs	int	The absolute basophil count.
azur_abs	int	The absolute azurophil count
bands	int	An assessment of the number of bands (0 or 1+ to 4+)
toxics	int	An assessment of the toxic heterophils (0 or 1+ to 4+)
pi	int	An assessment of the polychromatophilic index (1 to 5)
hemoproteus	int	An assessment of the number of hemoproteus organisms (0 or 1+ to 4+)
leukocytozoon	int	An assessment of the number of leukocytozoon organisms (0 or 1+ to 4+)
plasmodium	int	An assessment of the number of plasmodium organisms (0 or 1+ to 4+)
interpretation	char(63)	The interpretation.
readBy	char(32)	Initials or name of the person/lab reading the slide.
thrombo	char(32)	An assessment of the thrombocytes/platelets.
mcv	float	Mean corpuscular volume
mch	float	Mean corpuscular hemoglobin
mchc	float	Mean corpuscular hemoglobin concentration
hb	float	hemoglobin

Table: chemistry
This table stores blood chemistry data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the blood was collected.
type	char(4)	AE for avian/exotic, M for mammalian
UA	float	Uric acid

name	type	description
AST	float	AST
CK	float	Creatinine kinase
BA	float	Bile acids
GLU	float	Glucose
CA	float	Calcium
PHOS	float	Phosphorus
TP	float	Total protein
ALB	float	Albumin
GLOB	float	Globulin
K	float	Potassium
NA	float	Sodium
AMY	float	Amylase
CRE	float	Creatinine
BUN	float	BUN
CL	float	Chloride
ALP	float	ALP
ALT	float	ALT
GGT	float	GGT
TBIL	float	Total bilirubin
CHOL	float	Cholesterol
TRIG	float	Triglycerides
CO2	float	CO_2
AG	float	Albumin/Globulin ratio
BICARB	float	Bicarbonate
GAP	float	Anion gap
source	char(20)	The source of the chemistry values (i.e. the machine or lab that produced the results).

name	type	description
hem	char(8)	An assessment of the hemolysis in the sample.
lip	char(8)	An assessment of the lipemia in the sample.
ict	char(8)	An assessment of the icterus in the sample.
mg	float	Magnesium
ldh	float	Lactate dehydrogenase
nh3	float	Ammonia
fe	float	Iron
pH	float	pH
total_ca	float	Total calcium

Table: cytology
This table stores cytology data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the sample was collected.
location	char(64)	Where the sample was collected from.
stain	char(32)	Which type of stain was used. Choices are made from a pre-configured list.
findings	text	A description of the findings.
dx	char(128)	The diagnosis.

Table: dose_calc
This table stores drug dosage calculation. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.

name	type	description
startDate	datetime	The start date for this drug.
endDate	datetime	The end date for this drug.
drug	char(64)	The drug or medication.
who	char(32)	Who prescribed the drug and made the dosage calculation.
calc	char(255)	The actual calculation. An example entry is: `2 mg/kg x 1.5 kg / 10  mg/ml = 0.3 ml`
problem	int	The problem that this drug is supposed to treat.

Table: endoscopy
This table stores endoscopy reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
location	char(64)	The anatomic location.
performedBy	char(16)	Who performed the procedure.
anesthesia	char(128)	A brief description of the anesthetic protocol.
description	text	A description of the procedure and findings.
dx	char(128)	The diagnosis.

Table: enrichment
This table records procedures involving enrichment activities. These records are attached to individual record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the band report.

name	type	description
type	unsigned tinyint	ENRICHMENT_TYPE_TRAINING 0x0001 ENRICHMENT_TYPE_VISUAL 0x0002 ENRICHMENT_TYPE_AUDITORY 0x0004 ENRICHMENT_TYPE_OLFACTORY 0x0008 ENRICHMENT_TYPE_PHYSICAL 0x0010 ENRICHMENT_TYPE_SOCIAL 0x0020 ENRICHMENT_TYPE_TACTILE 0x0040 ENRICHMENT_TYPE_DIET 0x0080
who	char(16)	The person doing the enrichment
what	char(28)	A description of what was done
goal	char(128)	The goal for the enrichment
location	char(64)	Where the activity took place
interactionLevel	int	INTERACTION_LEVEL_NONE 0x0001 INTERACTION_LEVEL_AVOIDED 0x0002 INTERACTION_LEVEL_BRIEF 0x0004 INTERACTION_LEVEL_EXTENDED 0x0008
interactionTime	char(64)	The time of the interaction
comments	TEXT	Additional comments
targetType	char(15)	Target type such as: Pole, Light, Sound
bridgeType	char(15)	Bridge type such as: Clicker, Verbal
reinforcement	char(15)	Reinforcement type such as: Food, Verbal
flightType	char(15)	Flight type such as: Creance, Free
sessionRating	int	

Table: env_conditions
This table stores the test results for various environment condition parameters for terrestrial enclosures..

name	type	description
id	int	PRIMARY KEY.
location_id	int	FOREIGN KEY to the LOCATION table.
location_record_entry_id		FOREIGN KEY to the LOCATION_ENTRY table.

name	type	description
date	datetime	The date/time that the water was collected for testing.
who	char(32)	The name/initials of the person running the test
air_temp	float	The air temperature
humidity	float	The humidity
basking_temp	float	The temperature in an particular location (i.e. under a heat lamp)
uv	float	The uv level.
low_temp	float	The temperature in an area that is coolest.

Table: eph

This table stores serum protein electrophoresis records reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the sample was collected.
tp	float	Total protein
albGlobRatio	float	
preAlb	float	
alb	float	
alpha1	float	
alpha2	float	
beta	float	
gamma	float	
sample	char(32)	Details about the sample
source	char(32)	Details about the sample source

name	type	description
interpretation	char(255)	

Table: eye_exam
This table stores eye exam reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the examination.
who	char(32)	Who performed the examination.
left_score	int	A subjective score of the overall vision/function in the left eye. Ranges from 1 (blind) to 5 (perfect).
left_plr	int	The PLR. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
left_menace	int	The menace response. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
left_stain	int	Fluorscein stain. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
left_notes	char(255)	A description of the abnormalities.
right_score	int	A subjective score of the overall vision/function in the right eye. Ranges from 1 (blind) to 5 (perfect).
right_plr	int	The PLR. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
right_menace	int	The menace response. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
right_stain	int	Fluorscein stain. Values are 0 = did not examine, 1 = NEG, 2 = +, 3 = ++.
right_notes	char(255)	A description of the abnormalities.

Table: featherCheck
This table stores feather check reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the examination.
Each primary remiges and retrice is assigned a status or score. The values are as follows: FEATHER_STATUS_OK 90 FEATHER_STATUS_IB_MATURE 80 FEATHER_STATUS_IB 70 FEATHER_STATUS_IMPED 60 FEATHER_STATUS_FRAYED 50 FEATHER_STATUS_TIPPED 40 FEATHER_STATUS_BENT 35 FEATHER_STATUS_BROKEN_NEEDS_IMP 30 FEATHER_STATUS_SINGED 27 FEATHER_STATUS_BROKEN_NOT_IMPABLE 20 FEATHER_STATUS_MISSING 10 FEATHER_STATUS_UNKNOWN 0		
L1_10	int	The status of **left** primary #10 thru #1
L1_9	int	
L1_8	int	
L1_7	int	
L1_6	int	
L1_5	int	
L1_4	int	
L1_3	int	
L1_2	int	
L1_1	int	
R1_10	int	The status of **right** primary #10 thru #1
R1_9	int	

name	type	description
R1_8	int	
R1_7	int	
R1_6	int	
R1_5	int	
R1_4	int	
R1_3	int	
R1_2	int	
R1_1	int	
LT_1	int	The status of **left** tail #1 thru #6
LT_2	int	
LT_3	int	
LT_4	int	
LT_5	int	
LT_6	int	
RT_1	int	The status of **right** tail #1 thru #6
RT_2	int	
RT_3	int	
RT_4	int	
RT_5	int	
RT_6	int	

Table: fecal
This table stores fecal exam data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.

name	type	description
date	datetime	The date the sample was collected.
type	char(32)	The actual test such Zinc sulfate floatation, Sheather's floatation, Direct smear, etc.
comments	char(63)	Details about the sample
results	char(127)	The results are simply a comma-delimited list of parasites. The choices for this list are configurable.

Table: flight_eval
This table stores flight evaluations. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	datetime	The date of the evaluation.
who	char(32)	The person who did the flight evaluation.
flight_score	int	The subjective score of the bird's flight ability. Ranges from 1 (unable to fly) to 5 (perfect).
sound_score	int	The subjective score of the bird's sound level in flight. Ranges from 1 (very loud) to 5 (silent).
notes	char(255)	Any descriptive notes.

Table: handlerTraining

This table stores training records for a specific handler/person with a specific animal. Each training record pertains to a specific training category. The categories are configurable and some examples may be natural history, working on a glove, transport in vehicle, etc. The number of required training records in each category before being approved with an animal is also configurable and this may be dependent on the animal and its species. For instance, some species or individuals may be considered beginner level while others may be more advanced. This level is configured for each patient in the PATIENT table.

name	type	description
id	int	PRIMARY KEY.

name	type	description
date	date	The date this training record entry was made.
bird_id	int	FOREIGN KEY to PATIENT table;
handler_id	int	FOREIGN KEY to ADDRESS_BOOK
initials	char(8)	The initials of the person approving this person for this animal.
category	char(16)	The category that this training record pertains to.

Table: inventory
This table stores inventory for items in stock. It is used to generate a reorder list.

name	type	description
id	int	PRIMARY KEY
description	char(31)	The description/name of inventoried item
part_number	char(31)	The re-order number
desired_number	float	The number that is desired to be on hand
current_number	float	The current number on hand
reorder_number	float	The number at which an order should be made
supplier	unsigned int	FOREIGN KEY to ADDRESSBOOK table
unit	char(16)	The units for the number columns above. Examples of values include ‘boxes’, ‘bottles’, ‘tablets’, ‘bags’, etc.
notes	char(128)	
expirationDates	char(127)	One or more dates separated by spaces

Table: lab_test
This table stores lab tests and the results. These records are attached to individual sample entries.

name	type	description
id	int	The PRIMARY KEY.

name	type	description
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
sample_id	unsigned int	FOREIGN KEY to the SAMPLE table;
submission_date	date	The date the sample was submitted for this test.
type	char(32)	The type of test run. The list of choices is configurable.
source	char(32)	The lab/source of the results. The list of choices is configurable.
results	char(255)	The results.

Table: lead
This table stores blood lead data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the blood was collected.
level	float	The actual measurement.
source	char(20)	The source (instrument or lab) that produced the results. This value is configurable.
units	char(8)	The units of the result (i.e. ug/dL or ppm). This value is configurable.

Table: location
This table stores information about each available patient enclosure.

name	type	description
id	unsigned int	The PRIMARY KEY.
name	char(32)	The name of the location/enclosure.
dimensions	char(32)	A description of the dimensions. For informational use only.

name	type	description
capacity	unsigned int	An estimate of the maximum capacity. For informational use only.
curator	char(32)	The person responsible for this enclosure
department	char(32)	The organization's department that this enclosure is part of or managed by.
description	char(64)	A short description of this enclosure.
special	char(64)	A description of any special characteristics of this enclosure (i.e. “OK for live prey test”).
skip	int/bool	If 1, then this cage will be skipped on the treatment sheets on the configured skip day. Note that there may not be a configured skip day specified.
treatments	text	The group treatment for all patients in this location. Ignored if only one patient in this location.
treatmentPlan	text	This field is used when the Enclosure Treatment interface is used. This allows specific treatments to be scheduled with different medications, intervals and dates. This is generally not used when the normal group treatment is used but there is no reason why both cannot be used simultaneously.
feed_order	float	The order that this location will display on the treatment/ feeding sheets. This is a floating point number so that new locations can easily be inserted between existing ones.
display_order	float	The order that this location will display in the location list on the RaptorMed main screen. This is a floating point number so that new locations can easily be inserted between existing ones.
freq	char(3)	The frequency of the group treatment (i.e. SID, BID, TID).
cleaned_date	date	The date of the last cleaning.
cleaned_by	char(8)	The initials/name of the person that did the last cleaning.
disinfect_date	date	The date of the last disinfection.
disinfect_by	char(8)	The initials/name of the person that did the last disinfection.
watered_date	date	The date that water was changed.

name	type	description
watered_by	char(8)	The initials/name of the person that changed the water last.
carbon_change_date	date	The date that the carbon filter was last changed.
carbon_change_by	char(8)	The initials/name of the person that changed the carbon filter last.
algal_tx_date	date	The date that water was last treated for algae.
algal_tx_by	char(8)	The initials/name of the person that performed an algae treatment last.
sterilization_date	date	The date that water was last sterilized.
sterilization_by	char(8)	The initials/name of the person that performed a sterilization last.
waterChangePercent	smallint	The percentage of the water that was changed the last time the water was changed.
alert	char(128)	Any special instructions that pertain to the enclosure. They are typically printed on the treatment sheets.
flags	unsigned int	This field stores several boolean bit values for various characteristics. More than one can be set to true. Some, such as INSIDE and OUTSIDE are mutually exclusive. LOCATION_FLAG_INSIDE 0x00000001 LOCATION_FLAG_OUTSIDE 0x00000002 LOCATION_FLAG_KENNEL_ROOM 0x00000004 LOCATION_FLAG_UNDER_CONSTRUCTION 0x00000008 LOCATION_FLAG_INACTIVE 0x00000010 LOCATION_FLAG_SKIP 0x00000020 LOCATION_FLAG_OFFSITE 0x00000040 LOCATION_FLAG_FRESH_WATER 0x00000080 LOCATION_FLAG_SALT_WATER 0x00000100
x_coord	int	The x location in pixels from the upper left corner on the site map image.
y_coord	int	The y location in pixels from the upper left corner on the site map image.
combo_lock	char(31)	The combination lock code for this location. This can be printed on the treatment sheets for convenience.
skillLevel	char(32)	The skill level required in order to do this group treatment.

name	type	description
volume	float	The volume of this enclosure. Used for aquatic systems. The units are specified in the SETTINGS table with the “aquarium_volume_units” key.
level	smallint	The level that this location is on. Used for multi-level sites only. Default is 0, otherwise.

Table: location_record_entry
This table stores individual entries for a location/enclosure. These can include maintenance reports or water quality tests/treatments for aquatic enclosures.

name	type	description
id	int	The PRIMARY KEY.
location_id	unsigned int	FOREIGN KEY to the LOCATION table.
date	date	The date the entry was made.
who	char(32)	The name/initials of the person making the entry
notes	text	The actual body of the record entry.
ts	timestamp	The date and time when this entry was last updated.

special	tinyint	The entry type
		`0x01 Normal entry`
		`0x02 Veterinary entry`
		`0x04 Important entry`
		`0x08 New visit`
		`0x10 Entry requires review`
		`0x20 User defined entry type`
		`0x40 User defined entry type`

Table: maint_log
This table stores entries that describe system maintenance events such as database updates or backups.

name	type	description
id	int	PRIMARY KEY.
ts	timestamp	The timestamp when this entry was made. This is auto-filled by the system.

name	type	description
description	char(128)	A description of the maintenance event. This stores entries for database updates, backups and changes to user accounts.

Table: patient

This is the main table in the database. It stores information about each patient. Each patient should have a unique value in the ‘patient_id’ field. All RECORD_ENTRY records are attached to records in this table.

name	type	description
admission_date	date	The date of the admission
admission_time	TIME	The time of the admission. This was added after the admission_date column. Ideally they would be combined into a DATETIME column.
admitted_by	char(31)	The initials/name of the person who admitted the patient.
age	char(16)	The current age of the patient. The valid values for this field are configurable. This field must be updated as necessary.
ageIn	char(16)	The age at admission.
ageRelatedCharacteristics	char(255)	Text describing age and/or sex related characteristics such as plumage, iris color, etc.
away_from_date	date	Starting date when patient will be away/offsite. Used to exclude the patient from the daily treatment sheets. Only used for resident birds.
away_to_date	date	Ending date when patient will be away/offsite. See 'away_from_date'.
band	char(63)	Any uniquely identifying characteristic or id such as a band number or color. Also used to store permanent band/id numbers when animals are released.
aux_marker	char(16)	The optional Bird Banding Lab auxiliary marker code.
collection_depth	int	The depth (in meters) where this aquatic animal was found/ collected.
collection_info	char(255)	Random information about the collection.
collection_method	char(64)	The method used during the collection. Examples include 'Dip net', 'Hook and line', etc.
collection_temp	float	The temperature (in degrees C) of the water where the collection was made.
donation	float	Used to store any donation that may have been received by the finder/presenter when the animal was admitted.
donation_type	char(8)	Cash, check, credit card.
donation_details	char(16)	Check number, credit card digits, etc.
expected_release_ date	date	The expected release date. Used by the Event Planner to schedule events.
final_date	date	The date when the patient is no longer an active patient. This is set when the patient dies, is euthanized or release, or when it is transferred.

name	type	description
flags	64 bit integer	This field stores several boolean bits values for various characteri PATIENT_FLAG_CRITICAL 0x00000001 PATIENT_FLAG_EASILY_STRESSED 0x00000002 PATIENT_FLAG_NO_BAND 0x00000004 PATIENT_FLAG_RELEASE_WHERE_FOUND 0x00000008 PATIENT_FLAG_ANESTHETIC_RISK 0x00000010 PATIENT_FLAG_FOOD_AGGRESSIVE 0x00000020 PATIENT_FLAG_NO_WATER 0x00000040 PATIENT_FLAG_NEEDS_IMPING 0x00000080 PATIENT_FLAG_NEEDS_MOUSE_SCHOOL 0x00000100 PATIENT_FLAG_HAS_PLACEMENT 0x00000200 PATIENT_FLAG_BODY_GRAB 0x00000400 PATIENT_FLAG_DONT_GRAB_LEFT_LEG 0x00000800 PATIENT_FLAG_DONT_GRAB_RIGHT_LEG 0x00001000 PATIENT_FLAG_NR 0x00002000 PATIENT_FLAG_WAITING_FOR_MOLT 0x00004000 PATIENT_FLAG_ALL_MEDS_PO 0x00008000 PATIENT_FLAG_CARRY_WEIGH_IN_BOX 0x00010000 PATIENT_FLAG_FEED_SEPARATELY 0x00020000 PATIENT_FLAG_RELEASE_APPROVED 0x00040000 PATIENT_FLAG_NEEDS_COPE 0x00080000 PATIENT_FLAG_NEEDS_MOLT_CHAMBER 0x00100000 PATIENT_FLAG_PR_AWAY 0x00200000 PATIENT_FLAG_PR_STAFF_WILL_FEED 0x00400000 PATIENT_FLAG_PR_IN_REHAB 0x00800000 PATIENT_FLAG_NEEDS_OPHTHO_CONSULT 0x01000000 PATIENT_FLAG_NEEDS_FALCONER_EVAL 0x02000000 PATIENT_FLAG_BAND_RETURN 0x04000000 PATIENT_FLAG_GERIATRIC 0x08000000 PATIENT_FLAG_ESCAPE_ARTIST 0x10000000 PATIENT_FLAG_UNDERGOING_PT 0x20000000 PATIENT_FLAG_NROA 0x40000000 PATIENT_FLAG_EASILY_IMPRINTABLE 0x80000000 PATIENT_FLAG_CONTAGIOUS 0x100000000 PATIENT_FLAG_NON_NATIVE 0x200000000 PATIENT_FLAG_COLD_INTOLERANT 0x400000000 PATIENT_FLAG_LONG_TERM_FEEDER 0x800000000 PATIENT_FLAG_EXCLUDE_FROM_WEB 0x1000000000 PATIENT_FLAG_DOES_NOT_NEED_MOUSE_SCHOOL 0x2000000000 PATIENT_FLAG_BLEEDER 0x4000000000 PATIENT_FLAG_NEONATE 0x8000000000 PATIENT_FLAG_NOT_RWF 0x10000000000 PATIENT_FLAG_REPORTED_GT_180 0x20000000000 PATIENT_FLAG_ACTIVE_PATIENT 0x40000000000 PATIENT_FLAG_FINDER_PRESENT_AT_RELEASE 0x80000000000 PATIENT_FLAG_CHEMOTHERAPEUTIC 0x100000000000 PATIENT_FLAG_DANGEROUS 0x200000000000 PATIENT_FLAG_RESIDENT 0x400000000000 PATIENT_FLAG_ON_DISPLAY 0x800000000000 PATIENT_FLAG_FOSTER 0x1000000000000 PATIENT_FLAG_DISP_LETTER_SENT 0x2000000000000 PATIENT_FLAG_LOCKED 0x4000000000000 PATIENT_FLAG_TEMPORARY 0x8000000000000

name	type	description
found_addr	char(31)	The street address where the patient was found.
found_city	char(31)	The city where the patient was found.
found_country	char(15)	The country where the patient was found.
found_county	char(15)	The county where the patient was found.
found_date	date	The date on which the patient was found.
found_district	char(20)	The district where the patient was found. Useful for foreign addresses.
found_email	char(63)	The email address of the person who found the patient.
found_hx	char(255)	A short history describing how/where the patient was found.
found_lat	float	The latitude where the patient was found.
found_lon	float	The longitude where the patient was found.
found_name	char(48)	The name of the person who found the patient.
found_phone1	char(19)	A phone number for the person who found the patient.
found_phone2	char(19)	Another phone number for the person who found the patient.
found_state	char(20)	The state where the patient was found.
found_zip	char(10)	The zip code where the patient was found.
HandlerTrainingLevel	unsigned int	The handler training level for this patient. This is for resident animals that are handled and trained and allows each patient to be identified as requiring only basic or more advanced training before handling. The values are: 0 - animal not to be handled 1 - beginner 2 - intermediate 3 - advanced 4 - master 5 - eagle
husbandryNotes	char(255)	Any special husbandry notes are stored here.
id	unsigned int	The PRIMARY KEY.
ideal_weight	unsigned int	The ideal summer weight (in grams). See ‘winter_weight’.
injury_cause	char(31)	The cause of injury. The choices are configurable and include things like HBC, gunshot, etc.
keywords	char(63)	Any special keywords that are associated with this patient. The choices are configurable and can be used to track and easily recall patients with special issues or treatments.
location	char(32)	Where the patient is currently housed.
morph	char(15)	Color morph or subspecies such as ‘red’ or ‘grey’ for screech owls.
name	char(32)	The name. Usually for resident birds only but can be used for any patient.
notes	char(255)	Any notes that need to be kept in mind. These are displayed at the top of the patient record.

name	type	description
patient_id	char(15)	The patient id. This is a alphanumeric string that typically increments with each new patient. The numbering scheme is configurable.
The presenter fields allow another set of contact info to be stored for each patient. The 'found' fields above are for where the patient was actually found. The 'presenter' fields are used if the patient came from another organization and was transferred in for continued care, for example.		
presenter_addr	char(31)	The presenter address.
presenter_city	char(31)	The presenter city.
presenter_country	char(15)	The presenter country.
presenter_county	char(15)	The presenter county.
presenter_district	char(20)	The presenter district (useful for foreign addresses).
presenter_email	char(63)	The presenter email address.
presenter_name	char(48)	The presenter name (i.e. for a person).
presenter_org	char(63)	The presenter organization name.
presenter_phone1	char(19)	The presenter phone number.
presenter_phone2	char(19)	Another presenter phone number.
presenter_state	char(20)	The presenter state.
presenter_transporter	char(31)	The name of the person who actually transported the birds.
presenter_zip	char(10)	The presenter zip code.
rar	char(64)	Release-A-Raptor. This is used to plan events to release this bird/animal at. Can contain something like 'Release at July 4th event'.
sex	char(3)	M,F, or UNK.
species	char(16)	The species. The choices are configurable. Typically the 4 letter abbreviation used by the USGS Bird Banding Lab is used for birds. For non-avian patients, the first 2 letters of the genus and species can be used. For example, the common garter snake (Thamnophis sirtalis) would be THSI. Note that any set of abbreviations can be used.
species_group	char(64)	The species group include choices such as avian, avian:raptor, avian:songbird, mammal, mammal:rodent, reptile, etc. Each species is categorized with a species group and this group is used to calculate proper drug dosages for treatments.

name	type	description
status	char(3)	The basic status values are: EOA - euthanized on arrival DOA - dead on arrival D24 - died within 24 hours E24 - euthanized within 24 hours D - died (after 24 hours) E - euthanized (after 24 hours) Reh - in rehab R - released PR - permanent resident T - transferred ESC - escaped
subStatus	char(3)	When the patient status is 'COA' for client-owned animal, the animal can also have a sub-status. Valid values are D, R, E, T, or ESC.
team	char(15)	The treatment team that this patient belongs to. Ignored if treatment teams are not used.
treatment	text	The current treatment plan. This is an xml structure that is described elsewhere.
whenCaptured	char(15)	Information about when the animal was captured.
winter_weight	unsigned int	The ideal winter weigh (in grams). See also 'ideal_weight'.
admission_question_1	char(32)	The answer to the optional admission question #1
admission_question_2	char(32)	The answer to the optional admission question #2
admission_question_3	char(32)	The answer to the optional admission question #3
admission_question_4	char(32)	The answer to the optional admission question #4
admission_question_5	char(32)	The answer to the optional admission question #5
reason_for_euth	char(32)	The reason for euthanasia.
summary	text	This field is used to store a patient clinical summary.
microchip	char(32)	Stores the microchip number
found_crosstreets	char(32)	The major cross streets where animal found for cases where a street address is not available.
dob	date	The date of birth/hatch
clientId	unsigned int	FOREIGN KEY to ADDRESSBOOK table. Used to set the owner information by referring to an entry in the Address Book.
spillId	unsigned int	FOREIGN KEY to the SPILL table. Used to link this animal to a spill event.

Table: pcv

This table stores packed cell volume/total solids data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
date	datetime	The date the blood was collected.
pcv	int	The packed cell volume (%).
ts	float	The total solids.
serum	char(31)	A description of the serum.
buffy	float	The thickness of the buffy layer (%).

Table: problems
This table stores a list of problems and diagnoses for each patient. These records are attached to individual PATIENT records.

name	type	description
id	int	PRIMARY KEY.
patient_id	int	FOREIGN KEY to PATIENT table.
description	char(63)	The description of the problem. This is free-form but is typically populated from a configurable list.
resolved	int/bool	1 = resolved, 0 = unresolved.
initial_date	date	The date the problem was first reported.
resolved_date	date	The date the problem was resolved.

Table: procedures
This table stores a line item for each procedure performed during this record entry. These records are attached to individual medical record entries.

name	type	description
id	int	PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
description	char(64)	A description of the procedure

name	type	description
cost	float	The cost in the local currency.
type	smallint	`ANE_USED 0x01`

Table: program
This table stores program usage for resident animals. These records are attached to individual PATIENT records.

name	type	description
id	int	PRIMARY KEY.
patient_id	int	FOREIGN KEY to PATIENT table.
date	date	
org	char(64)	The contact info for the program
event	char(32)	
addr	char(32)	
city	char(32)	
state	char(21)	
zip	char(11)	
contact_name	char(32)	
phone	char(32)	
email	char(64)	
on_site	char(1)	Y' for onsite, 'N' for not onsite
numberPeople	smallint	The number of people at the program
type	char(16)	Examples: Display, Presentation, Mixed
ages	char(16)	The ages of the audience such as: 3-5, children, adults, etc.

Table: pt
This table stores physical therapy reports. These records are attached to individual medical record entries.

name	type	description
id	int	PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
location	char(15)	The appendage such as right wing, left leg, etc. The choices are configurable.
beforeProx	int	The starting angle of the proximal joint.
beforeDistal	int	The starting angle of the distal joint.
afterProx	int	The ending angle of the proximal joint.
afterDistal	int	The ending angle of the distal joint.

Table: rar_event
This table stores information on a release event. It allows an animal to be released to matched with an upcoming event. This is used for fund raising.

name	type	description
id	int	PRIMARY KEY.
name	char(32)	The name of the event.
date	date	The date for the reminder.
contact	char(32)	Contact info for the person organizing the event.
addr	char(64)	The address of the event.
priority	char(4)	Low, Med, High
timeOfDay	char(8)	Free form. Used to indicate when the event is to take place in order to match an appropriate species (i.e. nocturnal or diurnal).
notes	char(255)	Any descriptive notes.
bird1	char(16)	The first choice for the patient to release at the event.
bird2	char(16)	The second choice for the patient to release at the event.
bird3	char(16)	The third choice for the patient to release at the event.

name	type	description
siteApproved	int/bool	Is the site approved?
species	char(16)	The requested or required species.

Table: record_entry

This table contains the text of each entry in the medical record. Each entry is attached to a patient in the PATIENT table. Many other tables have foreign keys to this table in order to attach lab results and reports to each entry in this table.

name	type	description
date	datetime	The date and time when the entry was made.
id	unsigned int	The PRIMARY KEY.
keel	float	The body condition score. Currently uses a range of 1 to 5 with 0.5 increments.
leftovers	int	The amount of leftovers. Assumes units of grams. -1 means that it was not recorded.
location	char(32)	The patient's location when this entry was made.
notes	text	Free-form text describing this entry.
patient_id	unsigned int	FOREIGN KEY to PATIENT table
ts	timestamp	Auto-populated field indicating when the record was created or last updated.
weight	float	The patient's weight in grams.
who	char(32)	The initials/name of the person making this entry.

special	tinyint	The entry type
		
		`0x01 Normal entry`
		`0x02 Veterinary entry`
		`0x04 Important entry`
		`0x08 New visit`
		`0x10 Entry requires review`
		`0x20 User defined entry type`
		`0x40 User defined entry type`

amountFed	smallint	The amount of food fed. -1 is the default value

name	type	description
aneTime	smallint	The amount of time in minutes that the animal was anesthetized to perform the procedures listed in this record entry.
procedureTime	smallint	The amount of time in minutes to perform all the procedures listed in this record entry.

Table: releases

This table stores release/banding reports. These records are attached to individual medical record entries.

name	type	description
id	int	PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date of the release.
patient_id	char(15)	FOREIGN KEY to the PATIENT table
how_sex	char(48)	The code indicating how the patient was sexed. These codes are used by the Bandit software and are outlined on the USGS BBL web site.
how_age	char(48)	The code indicating how the patient was aged. These codes are used by the Bandit software and are outlined on the USGS BBL web site
band_number	char(64)	The federal band number. This will match the value in the PATIENT table.
band_size	char(7)	The band size (e.g. 8, 9R, etc).
aux_marker	char(16)	The optional Bird Banding Lab auxiliary marker code. This will match the value in the PATIENT table.
weight	int	The weight (in grams)
released_by	char(31)	Who release the patient.
location	char(63)	The location/address where the release took place.
city	char(31)	The city where the release took place.
district	char(20)	The district where the release took place. Useful for foreign addresses.
state	char(2)	The state where the release took place.

name	type	description
county	char(15)	The county where the release took place.
country	char(15)	The country where the release took place.
comments	char(127)	Any comments regarding the release.
lat	float	The latitude where the release took place.
lon	float	The longitude where the release took place.
id_type	char(15)	The type of identification device placed on the patient when released. Typical choices are None, Leg band, or Ear tag.
tracker_type	char(31)	The type of tracking device placed on the patient when released. Typical choices are None, Backpack, Implant, etc.
details	char(127)	Use this field to store any additional details about the identification or tracking devices such as frequencies, etc.
pub_priv	char(7)	Indicates whether the animal was release on public or private land. Choice include PUB, PRIV, UNK.

Table: reminder

This table stores reminders which can be attached to an individual patient, an enclosure or to the clinic as a whole. A reminder can have a specific date or can be auto-repeating if a date is not specified.

name	type	description
id	int	PRIMARY KEY.
category	char(32)	An optional category for the reminder. This can be used to sort the reminders.
date	date	The date for the reminder.
task	char(128)	The description of the task. Note: the reminder can be for a patient, for a location or a generic reminder for the clinic/ organization as a whole.
patient_id	char(16)	FOREIGN KEY to the PATIENT table if this reminder is for an individual patient.
location	char(32)	FOREIGN KEY TO THE LOCATION table if this reminder is for a location.

name	type	description
type	char	REMINDER_TYPE_REGULAR 'R' regular REMINDER_TYPE_DAILY 'D' daily auto-repeat REMINDER_TYPE_WEEKLY 'W' weekly auto-repeat REMINDER_TYPE_MONTHLY 'M' monthly auto-repeat
day	tinyint	The day of the week (1-7) or the day of the month (1-31) for type = W and M, respectively

Table: sample
This table stores sample data. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY.
record_entry_id	unsigned int	FOREIGN KEY to the RECORD_ENTRY table.
patient_id	unsigned int	FOREIGN KEY to the PATIENT table.
collection_date	date	The date the sample was collected.
type	char(32)	The sample type such as whole blood, etc. The choices are configurable.
notes	char(255)	Any descriptive notes about this sample.

Table: settings
This table stores many configuration settings and choices lists for various parts of the graphical user interface. It is basically just a name-value pair hash table.

name	type	description
id	int	PRIMARY KEY.
category	char(32)	The category or group that this setting applies to. This table allows one or more name-value pairs to be stored based on the category. For example, some category values are species, problem, and age. These categories are used to populate the choices in various list controls in the RaptorMed program. There are also categories that only have one value. For example, org_name or org_phone which store the name and phone number of the organization.

name	type	description
value	char(255)	The value of the category-value pair.

Table: species
This table stores information about each possible species.

name	type	description
id	int	PRIMARY KEY.
abbrev	char(15)	The abbreviation for this species. Typically the 4 letter abbreviation used by the USGS Bird Banding Lab is used for birds. For non-avian patients, the first 2 letters of the genus and species can be used. For example, the common garter snake (Thamnophis sirtalis) would be THSI. Note that any set of abbreviations can be used.
scientific_name	char(64)	The genus and species. Somewhat redundant with the genus and species columns below but used in different circumstances.
common_name	char(64)	The common name. For example: Red-tailed hawk.
kingdom	char(24)	The full taxonomy from kingdom to species
phylum	char(24)	
class	char(24)	
_order	char(24)	
family	char(24)	
genus	char(24)	
species	char(24)	
nord	char	N for nocturnal, D for diurnal.
native	char	Y for native, N for non-native.
hms	char	Highly migratory species. Y for yes, N for no.

habitat	char	`U Unknown`
		`T Terrestrial`
		`A Amphibian`
		`F Fresh water`
		`S Salt water`
		`B Brackish`

name	type	description
rvs	char	Rabies vector species Y = yes N = no
venomous	char(15)	Is this a venomous species Y = yes N = no
info_page	char(128)	The URL of a web page that provides information on the species. http:// should not be appended at the front. For example: en.wikipedia.org/wiki/Broad-winged_Hawk
species_group	char(64)	The species group that this species belongs to. There is a configurable list of choices. For example: avian:raptor
cost_category	int	The cost category is a estimate of how expensive it is to treat this species. The values are: 1 Low 2 Med 3 High 4 Very high
state_soc	tinyint	SPECIES_SOC_NO 0 SPECIES_SOC_THREATENED 8 SPECIES_SOC_ENDANGERED 10
federal_soc	tinyint	SPECIES_SOC_NO 0 SPECIES_SOC_THREATENED 8 SPECIES_SOC_ENDANGERED 10
value	float	The dollar amount associated with this species (primarily used for zoo/aquarium collections)

Table: spill
This table stores information about oil spills.

name	type	description
id	int	The PRIMARY KEY
date	date	
location	char(31)	The address of the spill if on land
body_of_water	char(31)	The body of water of the spill if on water
city	char(31)	The city of the spill if on land
state	char(20)	The state of the spill if on land

name	type	description
county	char(15)	The county of the spill if on land
country	char(15)	
lat	float	
lon	float	
source	char(31)	The source of the spill
crude_type	char(15)	The type of crude
crude_amount	char(15)	The amount of crude spilled
depth	float	The depth of the spill
comments	char(255)	

Table: surgery
This table stores patient surgery reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
procedureName	char(63)	This is configurable list of choices. For example: Fracture repair - humerus or Mass removal.
complexity	char(15)	major or minor
performedBy	char(63)	The name of the surgeon.
totalTime	int	The total anesthesia time (in minutes).
sxTime	int	The actual procedure time (in minutes).
anesthesia	char(127)	A summary of the anesthetic protocol.
monitor	char(127)	A summary of the anesthetic monitoring employed.
premeds	char(127)	A summary of the pre-medications.
fluids	char(127)	A summary of the fluids administered.
antibiotics	char(127)	A summary of the antibiotics the patient received.

name	type	description
postop	char(127)	A summary of the post-op medications.
description	text	A detailed description of the procedure.

Table: surplus_wishlist
This table stores information about animals that are wanted or available for trade.

name	type	description
id	int	The PRIMARY KEY
type	int	0 = available for trade 1 = wanted for trade
species	char(64)	The species of interest - can be free form and as detailed or a vague as needed
number	int	The number that is available or wanted
age	char(16)	The age of interest
sex	char(8)	The sex of interest
who	char(64)	Contact information for the person who made this entry

Table: transfers
This table stores patient transfer reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
patient_id	char(15)	The patient id (alphanumeric) of the transferred patient
date	date	The date of the transfer
name	char(63)	Who the animal is transferred to. This can be the name of the person and/or the organization.
addr	char(31)	The address of the place where the animal was transferred to.
city	char(31)	
state	char(20)	

name	type	description
zip	char(10)	
county	char(15)	
country	char(15)	
permit	char(127)	The permit number(s) for the receiving organization
reason	char(15)	The reason for the transfer. This must be one of the valuse allowed on the US FWS year-end report and include: • Release • Continued care • E/S - Educational/scientific • F/P - Falconry/propagation • Other

Table: ua

This table stores patient urinalysis reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
date	date	The date the sample was corrected
source	char(15)	Notes about the sample
color	char(15)	
clarity	char(15)	
ph	float	
spgr	float	Specific gravity
protein	char(15)	
glucose	char(15)	
ketones	char(15)	
bilirubin	char(15)	
urobilinogen	char(15)	
wbc	char(15)	

name	type	description
rbc	char(15)	
nitrite	char(15)	
sediment	char(15)	
notes	char(255)	Notes about the interpretation and results.

Table: users
This table stores the various privileges for each user.

name	type	description
id	int	PRIMARY KEY.
name	char(32)	The usename. This must match a name in the MySQL user table.

privileges	int	This field stores several boolean bit values for various characteristics. More than one can be set to true.
		`USER_PRIVILEGE_VIEW 0x00000001`
		`USER_PRIVILEGE_ADD 0x00000002`
		`USER_PRIVILEGE_UPDATE 0x00000004`
		`USER_PRIVILEGE_DELETE 0x00000008`
		`USER_PRIVILEGE_USER_MANAGEMENT 0x00000010`
		`USER_PRIVILEGE_ADD_PATIENT 0x00000020`
		`USER_PRIVILEGE_MAINT 0x00000040`

Table: wound_report
This table stores patient wound reports. These records are attached to individual medical record entries.

name	type	description
id	int	The PRIMARY KEY
record_entry_id	int	FOREIGN KEY to the RECORD_ENTRY table.
name	char(32)	The name of the wound report template. This refers to an entry in the `WoundReports.xml` resource file
max_score	int	The maximum score that can be selected for each lesion
max_size	int	The maximum size that can be specified for each lesion.
who	char	Who recorded this wound.

name	type	description

scores	TEXT	The wound scores are stored in this CLOB field. Each entry is terminated with a carriage return. The format is as follows:
		
		`name=right:sole,wound=4,swelling=2,size=6`
		`name=right:digit1,wound=4,swelling=3,size=2`

Table: water_quality

This table stores the test results for various water quality parameters for aquatic enclosures..

name	type	description
id	int	PRIMARY KEY.
location_id	int	FOREIGN KEY to the LOCATION table.
location_record_entry_id		FOREIGN KEY to the LOCATION_ENTRY table.
date	datetime	The date/time that the water was collected for testing.
who	char(32)	The name/initials of the person running the test
pH	float	The pH
temp	float	The temperature. The units are determined by the system configuration (typically degrees C or F).
fecal_coliform	float	The number of CFU per unit volume. The units are to be determined by the system configuration (typically CFUs/ 100 ml H_2O).
chlorine	float	The chlorine level. The units are to be determined by the system configuration (typically ppm).
salinity	float	The salinity level. The units are to be determined by the system configuration.
nh3	float	The NH_3 level. The units are to be determined by the system configuration.
no2	float	The NO_2 level. The units are to be determined by the system configuration.
no3	float	The NO_3 level. The units are to be determined by the system configuration.
do	float	The DO level. The units are to be determined by the system configuration.

name	type	description
phosphate	float	The Phosphate level. The units are to be determined by the system configuration.
cu2	float	The Cu^{2++}level. The units are to be determined by the system configuration.
percent_abs	float	The Percent Abs level. The units are to be determined by the system configuration.
hardness	float	The Hardness level. The units are to be determined by the system configuration.
residual_ozone	float	The Residual Ozone level. The units are to be determined by the system configuration.
orp	float	The ORP level. The units are to be determined by the system configuration.
alk	float	The Alkalinity level. The units are to be determined by the system configuration.
ca2	float	The Ca^{2++} level. The units are to be determined by the system configuration.
copper_sulfate	float	The copper sulfate level. The units are to be determined by the system configuration.
chloroquine	float	The Chloroquine level. The units are to be determined by the system configuration.
kh	float	The kh level. The units are to be determined by the system configuration.
nh4	float	The Ammonia level. The units are to be determined by the system configuration.
fe	float	The Iron level. The units are to be determined by the system configuration.
sio2	float	The Silicon dioxide level. The units are to be determined by the system configuration.
co2	float	The CO_2 level. The units are to be determined by the system configuration.
TAN	float	The TAN level. The units are to be determined by the system configuration.

Example Queries

Joins to record entries

Each medical RECORD_ENTRY has a foreign key (FK) to the PATIENT table and each lab test or examination is linked via FK to the RECORD_ENTRY table. So, in order to find all blood lead tests, for example, for a specific patient, the appropriate SQL query would be:

```
select l.date,l.level from lead l, patient p,
record_entry re where p.patient_id='20120049-01' and
l.record_entry_id=re.id and re.patient_id=p.id;
```

In this query, three tables are joined together and the actual join is highlighted. This type of join query can be used to access the following tables: ATTACHMENTS, BAND_REPORT, BIOMETRIC, CBC, CHEMISTRY, CYTOLOGY, DOSE_CALC, ENDOSCOPY, EYE_EXAM, FEATHERCHECK, FECAL, FLIGHT_EVAL, LEAD, PCV, PT, RELEASES, SAMPLE, SURGERY, and TRANSFERS.

Installation instructions

MySQL server installation

- Log in as a Windows administrator user
- Download the MySQL installation zip file
- Unzip it by double-clicking on it
- Run the **setup.exe** program. This will start the installer
- Choose **Typical** when prompted for the type of installation
- Click **Install**
- A screen will appear that has a **Configure the MySQL server now** check box. Make sure it is checked and press **Finish**
- The server configuration wizard will start
- Choose **Detailed configuration** when prompted
- Choose **Server machine** when prompted
- Choose **Multifunctional database** when prompted
- Accept the default settings for **InnoDB tablespace settings** and the **number of concurrent connections**
- On the next page check **Enable TCP/IP Networking** and accept the default port number. **UNCHECK** the **Enable strict mode** check box.
- Accept the **Standard character set**
- On the next page, check **Install as Windows service** and **Include Bin directory in Window PATH**
- On the next page, enter a password for the root account. Note that it must be entered three times. Check the **Enable root access from remote machine** check box. Do **not** check the **Create anonymous account**
- Press **Execute**. When this completes, the database server should be up and running.
- Confirm that the Windows firewall is not blocking port 3306
- Copy the following folder structure from the install CD to the server:

```
c:\RaptorMed
c:\RaptorMed\Attachments
c:\RaptorMed\Documents
c:\RaptorMed\Help
c:\RaptorMed\http
c:\RaptorMed\http\cgi-bin
c:\RaptorMed\http\images
c:\RaptorMed\http\log
c:\RaptorMed\Query
c:\RaptorMed\Resources
c:\RaptorMed\Updates
c:\RaptorMed\Updates\Resources
c:\RaptorMed\Updates\Resources\Templates
```

- **Share** all folders\files for **read and write access.** On Windows 7, you may need to remove the Guest account password to make this work.

- At a command prompt, change to the folder that contains the **RaptorMed.sql** and **RaptorMedConfig.sql** scripts. Type:

```
mysql --host=ip --user=root --password=password

Type: drop database test;
Type: source RaptorMed.sql
Type: source RaptorMedConfig.sql
```

Apache installation

Install the Apache Web Server and accept the default location.

Replace the installed **httpd.conf** file in

```
C:\Program Files\Apache Software Foundation\Apache2.2\conf
```

with the file from the installation disk.

Software updates

The application (**RaptorMed.exe**) will look for updates whenever it is started. In addition, an update can be initiated anytime by choosing **File ▶ Check for update** from the main menu. The determination of whether an update is needed is based on the file timestamp and the process will update the main application as well as any support files. If the **RaptorMed.exe** file needs to be updated, then it will start a small application (**RaptorMedUpd8r.exe**) that prompts the user to quit the main application, copies over the new file, and prompts the user to restart the main app. Any support files are also copied over, if necessary. The process generally works well but the security features and UAC in Win 7 can make this very frustrating so the auto-update feature can be disabled by the following entries in the SETTINGS table:

```
EnableAutoUpdate_Software - set to false to disable
EnableAutoUpdate_Other - set to false to disable (not
              recommended)
```

Turn the UAC off in Windows 7 and later resolves most of the problems with the auto-update feature.

When the software was installed and configured, a shared folder on the network was specified as the update folder. In order to deliver an update, the new files simply need to be copied to the update folder and each workstation will auto-update itself the next time

it is restarted. In some instances, **RaptorMed** may need to be restarted after an update. You will be prompted when this is necessary.

The update folder is actually a hierarchy of folders and it is organized as follows:

```
\RaptorMed\Updates
\RaptorMed\Updates\Resources
\RaptorMed\Updates\Resources\Templates
```

Maintenance and configuration

Backing up the database
In order to backup the database:

- The MySQL service must be stopped.
- The following folders are copied to a secure location or burned onto a CD/DVD:
  `C:\Program Files\MySQL\MySQL Server 5.0\data\raptormed`
  `C:\Program Files\MySQL\MySQL Server 5.0\data\mysql`
  `C:\RaptorMed`
- The MySQL service must be restarted.

This process is partially automated by using the Backup Wizard function in the RaptorMed application (**Tools ▶ Maintenance ▶ Backup database**). It can also be done manually by using the Windows Services control panel applet to start and stop the database.

User management
Any number of users can be created to access the **RaptorMed** database. This function is accessed from the **RaptorMed** main menu (**File ▶ User manager**). Each user can have one or more the following privileges:

- View data
- Add or update data
- Delete data
- Add new patients
- Manage users - currently only the MySQL '**root**' login can manage users.
- Perform maintenance and configuration functions
- Requires review - users with this option checked can only make record entries with `Entry Type = Requires Review` indicating that these entries need to be reviewed before they can be considered finalized.

Alerts
The system can be configured to alert the user when any sort of event occurs. The alerts are run periodically in the background and are configured in the `Alerts.xml` file. Each alert has the following fields:

- **Enable** = yes or no
- **Name**
- **Who** is a list of RaptorMed logins that are interested in this alert.
- **Importance** = low, medium or high. Alerts are displayed using different colors depending on the Importance level.
- **SQL query**

```
<Alert>
<Alert enable="yes" name="Admissions today" who="dave root" importance="low">
      select p.patient_id Id,p.name Name,sp.scientific_name Species from
patient p,species sp where date_format(admission_date,'%Y-%m-%d') = curdate()
and p.species=sp.abbrev;
</Alert>

<Alert enable="yes" name="Overtemp" who="ALL" importance="med">
      select loc.name Enclosure,wq.date Date,wq.temp Temp from location loc,
water_quality wq where wq.location_id=loc.id and wq.temp > 80 and
      date_format(wq.date,'%Y-%m-%d') = curdate();
</Alert>
</Alerts>
```

Bandit integration
The **BanditLocations.xml** is required in order to create Bandit bird banding reports. This file contains a duplicate of the location data that is already stored in the Bandit program. The file has the following format:

```
<locations>
<location id="Y008" precision="10"
            lat="32 10 0" lon="-80 40 0" description=”Aiken, SC" />
</locations>
```

The format is self-explanatory and the **precision** field can have the following values:

0 - the lat/lon must match exactly.
1 - defines a square that is 1 minute wide and 1 minutes tall. The coordinates define the southeast corner of the square.
10 - defines a square that is 10 minutes wide and 10 minutes tall. The coordinates define the southeast corner of the square.

To export location information from Bandit version 3.01:
1. On the Location tab, press the Export button

2. Choose Excel format and choose a file name/location
3. For Excel options, check “use field names…” and leave the other fields blank. Press continue.
4. This displays a form that allows you to choose what fields to export. I think you can just accept it as is.
5. Press Export
6. Using notepad, edit this file into the XML file format described above.

Cage Card Template

The CageCageTemplate.xml file contains a design for a cage card that can be printed for each animal. It contains simple html code but can also contain various macros that are surrounded by **double-triangle** brackets as in: **<<SPECICES>>**.

The following macros are supported. Most are self-explanatory.

- TODAY
- ADDRESSED_TO - the name and contact info of a person or other entry from the Address Book.
- ADDRESSEE - the name only of a person or other entry from the Address Book.
- PATIENT_ID
- NAME
- STATUS - this disposition
- SPECIES - the specie abbreviation
- SPECIES_COMMON_NAME
- SPECIES_SCIENTIFIC_NAME
- AGE
- SEX
- ADMISSION_DATE
- FINAL_DATE
- FINDER_NAME
- FOUND_LOCATION
- PROBLEMS - displays all problems, both resolved and unresolved.
- UNRESOLVED_PROBLEMS
- INJURY_CAUSE
- BAND/TAG
- INTAKE_WEIGHT
- TREATMENT_PLAN - the current treatment plan
- WEIGHT - the last/current weight
- VENOMOUS_BANNER - displays a banner if this is a venomous animal

City/state/zip code/county lookups

The zip code, county and state can be auto-filled when a city is chosen in many dialogs including the New Patient or Finder Info dialogs. These choices are stored in an xml file called **CityLookup.xml** file.

This file must be manually edited. It has the following overall structure:

```
<entries>
<entry city="Addison" zip="60101" county="DuPage" state="IL" country="USA"/>
<entry city="Aurora" zip="60502" county="DuPage" state="IL" country="USA"/>
<entry city="Bartlett" zip="60103" county="DuPage" state="IL" country="USA"/>
</entries>
```

Data Entry Forms

The **DataEntryFormsTemplates.xml** file contains a description of any number of forms that can be used for data entry. Each form is defined by a **<template>** tag. The **name** attribute is used to display this as a choice in the **Medical Record Entry** form. The **caption** is displayed as the caption in the form's title bar. The **instructions** are additional text that is displayed at the top of the form to provide any advice or guidance when filling out the form.

Each form is composed of multiple data entry fields that follow the same format and structure as the **Physical Exam** templates described below.

```
<DataEntryFormTemplates>
<template name="EEDRF" instructions="" caption="-- Escape, Euthanasia, Death
Report --" >

      <entry name="Type" type="checkboxes" columns="2">
            <checkbox name="Escape"/>
            <checkbox name="Euthanasia"/>
            <checkbox name="Death"/>
      </entry>
      <entry name="Staff in charge"/>
      <entry name="Location prior to event" />
      <entry name="Notes/history" lines="3"/>

      <entry name="Euthanasia" edit="false">
            <entry name="Method" type="checkboxes" columns="1">
                  <checkbox name="MS-222 via bath/immersion"/>
                  <checkbox name="Induction via isoflurane + Pentobarbital"/>
                  <checkbox name="Cervical dislocation"/>
            </entry>
            <entry name="Dose/volume/route"/>
            <entry name="Administered by"/>
      </entry>

      <entry name="Corrective action taken" lines="3"/>
      <entry name="Vet services notes" lines="3"/>
      <entry name="Disposition/location of carcass"/>
```

```
</template>

</DataEntryFormTemplates>
```

Diets

The **Diets.xml** file stores predefined diets that can be selected from a list in each animal's treatment/feeding plan. The diets are arranged hierarchically by the species group as shown in the example below. The system will find the section that most closely matches the animal in order to find the diet options. For example, an animal in the **reptile:turtle** group would use the **reptile:turtle** section. However, an animal in the **reptile:snake** group would use the more generic **reptile** section since there isn't a section defined specifically for snakes.

Each diet entry has a **name** that is simply to make the file more self-explanatory. It is not used by the program. The actual diet description is what is presented to the user in a drop-down list.

```
<Diets>

<avian>
      <avian:raptor>
            <diet name="Nestling diet">20 g furless mouse cutup small</diet>
            <diet name="Osprey diet">200 g whole fish + vitahawk</diet>
      </avian:raptor>
</avian>

<reptile>
      <diet name="Diet 1">...</diet>
      <diet name="Diet 2">...</diet>

      <reptile:turtle>
            <diet name="Diet 1">...</diet>
            <diet name="Diet 2">...</diet>
      </reptile:turtle>
</reptile>

</Diets>
```

Dosages

The dosages that **RaptorMed** uses to automatically calculate the dose for each patient are stored in an xml file called **c:\RaptorMed\Resources\Dosages.xml**.

The file is based on a hierarchical structure that mimics the structure of the species groups that are in use at each clinic. This allows for a separate set of drug dosages for each species group. Each section in the formulary can contain a formulary and the structure of the formulary entries is described below.

Here is a typical hierarchy that allows for separate formularies for the following species groups: avian:raptor, avian:songbird, mammal, reptile:turtle.

```
<Dosages>
      <avian>
            <raptor>
            </raptor>

            <songbird>
            </songbird>
      </avian>

      <mammal>
      </mammal>

      <reptile>
            <turtle>
            </turtle>
      </reptile>
</Dosages>
```

Note that the system will use the set of dosages that most closely matches the patient's species group. So a patient with species group avian:raptor would use the avian:raptor dosages but a patient with species group avian:waterfowl would use the generic avian dosages since a formulary for avian:waterfowl hasn't been specified.

Also note that the system will use the species group, route and frequency when attempting to find a dosage for specific drug. They all must match in order for a dosage to be calculated.

The actual format of each formulary entry is straightforward. There is an entry for each drug that has the following fields:

- **name** - The name of the drug
- **form** - The form that the drug comes in. This is typically a concentration of tablet size
- **planned_dosage** - The dosage that you intend to give
- **freq** - SID, BID, TID. Note that you can specify more than one frequency as in "SID, BID"
- **route** - The route that the drug is to be given. Choices include PO, SQ, IV, IO. Note that you can specify more than one route as in "IO,IV"
- **type** - Either **calc** or **closestMatch**

  Calc is used for drugs that are in liquid form and the dose can be calulated exactly by muliplying the **dosage** by the weight. The specified dosage is in ml\kg. The units field is currently ignored. Here is an example:

```
<Drug name="Meloxicam" form="1.5 mg\ml sol" planned_dosage="0.2 mg\kg"
freq="SID,BID" route="PO,SQ" type="calc" dosage="0.13" units="ml" />
```

dosage - The actual dose in ml/kg that is to be given to the patient.
units - This is currently ignored

ClosestMatch is for drugs in pill or capsule form that must be split and administered in as close to the proper dose as is possible. For instance, it is very hard to administer 1\3 of a tablet so you may decide the best dose is 1\4 or 1\2 of a tablet. For these types of drugs, there are several entries that contain a weight and a dose. The system will choose the closest match when determining a dose. You can add as many entries as are needed for each of these drugs. Here is an example:

```
<Drug name="baytril" form="22.7 mg tabs" planned_dosage="15 mg\kg" freq="BID"
route="PO" type="closestMatch">
            <choice weight="200" dose="NA"/>
            <choice weight="300" dose="1\4 tab"/>>
            <choice weight="400" dose="1\4 tab"/>
            <choice weight="500" dose="1\4 tab"/>
            <choice weight="600" dose="1\2 tab"/>
            <choice weight="700" dose="1\2 tab"/>
            <choice weight="800" dose="1\2 tab"/>
            <choice weight="900" dose="1\2 tab" />
            <choice weight="1000" dose="3\4 tab"/>
            </Drug>
```

This file can be edited with any text-based editor such as Notepad. Be careful, however, since the XML file format is very picky about missing triangle brackets and it is easy to make the file unreadable. Always backup the file before editing it and view it in Internet Explorer(IE) (by double-clicking on it) when you are done. If it can be loaded by IE, then it should be ok.

Once the file edits are complete, each workstation must be restarted in order to get the changes.

Email templates
The **EmailTemplates.xml** file is used to generate pre-formatted emails. The following built-in email templates are available but any number of other templates can also be added to the file:

- Illegal cause of injury - used to send a notification about a patient injured from gunshot or another reportable cause of injury.
- Special species admitted - used to send a notification about a special species admission such as bald or golden eagles.
- Patient status update - used to send a status update message to a member of the public.
- 180 day request - used to send a request to keep a patient in rehab longer than the typical 180 day limit.

The format is as follows:

```
<EmailTemplates>

<template name="Illegal cause of injury">
      <email>bob@fws.gov;another email address..</email>
      <Subject>Illegal cause of injury in patient admitted to CRC</Subject>
      <MessageForRecord>US FWS notified via email: illegal cause of injury
      </MessageForRecord>

</template>

<template name="">
</template>

</EmailTemplates>
```

Each template has the following fields:
The **<name>** tag is used to choose the template from a list in the RaptorMed program. There are specific values listed above that cannot be changed for the built-in templates.
The **<email>** tag contains the list of emails to send the message to. Multiple email addresses are separated by semicolons.
The **<Subject>** tag contains the text that will be in the email subject line.
The **<MessageForRecord>** tag contains text that will be automatically inserted into the animal's medical record.

The body of the **<template>** tag contains the actual message text. The text can contain various macros that will auto-expand to actual data from the animal's record. Here is an example:

```
Hi Resee,

Here is another case for you.

Patient id: [PATIENT_ID]
Admission date: [ADMISSION_DATE]
Species:  [SPECIES]
Age: [AGE]
Found:  [FOUND_LOCATION]
Injuries/Problems: [PROBLEMS]
Cause of injury: [INJURY_CAUSE]
Prognosis:

Thanks,

Dave Scott, DVM
Staff Veterinarian
```

```
Carolina Raptor Center
....
```

The following macros are supported. Each macro must be surrounded in square brackets. Most are self-explanatory.

- TODAY
- ADDRESSED_TO - the name and contact info of a person or other entry from the Address Book.
- ADDRESSEE - the name only of a person or other entry from the Address Book.
- PATIENT_ID
- SPECIES
- AGE
- ADMISSION_DATE
- FINAL_DATE
- FINDER_NAME
- FOUND_LOCATION
- PROBLEMS
- INJURY_CAUSE
- BAND/TAG
- AUX_MARKER
- INTAKE_WEIGHT
- SIGHTINGS - a list of all previous sightings for this animal. This includes the date and location.
- SIGHTING_NAME - the name of the last person to make a sighting of this animal

Enclosure treatments

The treatment options and dosages for enclosure treatments are configured in **EnclosureTreatmentChoices.xml** and **EnclosureDosages.xml**.

The **EnclosureTreatmentChoices.xml** file specifies what treatments are available for treating an enclosure. It is very similar in format to the **TreatementChoices.xml** file that describes medications that are used to treat individual animals. There are a few differences:

- There are no sub-sections for the drug type (i.e. antibiotics, analgesics, etc).
- There is a new attribute **nonReleasable** that can have a value of either yes or no. If yes, animals in an enclosure that were treated with this chemical will be considered non-releasable in the future.

```
<Treatments>
      <choice select="yes" name="Acetozolamide" nonReleasable="no">
            <forms>
                  <form value="250 mg/tab"/>
            </forms>
            <freqs>
```

```
                <freq value="q4h"/>
            </freqs>
            <until show="yes"/>
      </choice>
      <choice select="yes" name="Copper_sulfate" nonReleasable="no">
            <forms>
                <form value="1 mg/ml sol"/>
            </forms>
            <freqs>
                <freq value="SID"/>
            </freqs>
            <until show="yes"/>
      </choice>
</Treatments>
```

The **EnclosureDosages.xml** file specifies the amounts to be used of each chemical defined in the **EnclosureTreatmentChoices.xml** file. It is very similar to the Dosages.xml file used to treat individual animals but there are a few differences:

- There are no sub-sections species group.
- There is no **route** attribute since these chemical are always simply added to the enclosure's water.
- There is no **type** attribute since the dosage calculation is always calculated in one way (i.e. x units per volume).
- There is a **per** attribute that is used to perform the dosage calculation. It can either **liter** or **gallon.**

```
<Dosages>
<Drug name="Fenbendazole" form="12 mg/ml" planned_dosage="100 mg/L"
      freq="SID" dosage="8" units="ml" per="liter"/>
<Drug name="Copper_sulfate" form="1 mg/ml sol" planned_dosage="0.15 mg/L"
      freq="SID" type="calc" dosage="0.15" units="ml" per="liter"/>
</Dosages>
```

Each entry defines a chemical form (usually a concentration), the desired concentration in the system after the medication is applied (planned_dosage) and the actual dose/ units to add for every unit of measure in the system. In the example above, Copper sulftate comes in a 1 mg/ml solution. Our desired concentration is 0.15 mg/L in the system so we would add 0.15 mls per liter.

Handler training levels

A **Training level** may be selected for each animal that is handled or trained. Each training level has a corresponding to a number of signoffs that the handler must get in order to be certified on the animal. The number of signoffs for each training level is configured with the following entries in the **Settings** table.

Training level	Setting in database	Example
Beginner	HandlerTrainingReqsForLevel1	3

Training level	Setting in database	Example
Intermediate	HandlerTrainingReqsForLevel2	3
Advanced	HandlerTrainingReqsForLevel3	4
Master	HandlerTrainingReqsForLevel4	

Note that if the value is not supplied, as in the **Master** level above, then that training level is considered open-ended and there is no set number of signoffs needed for certification.

In addition to the number of signoffs required, the actual categories that the handler must be signed off on are also configurable and there can be any number of them. The categories are configured as follows:

#	Setting in database	Example
1	HandlerTrainingCategory1	Food prep
2	HandlerTrainingCategory2	Getting on glove
3	HandlerTrainingCategory3	Kennel
4	HandlerTrainingCategory4	Health checks

Medical record text templates

The **MedicalRecordInsertTemplates.xml** file is used to populate the list of text shortcuts in the medical record entry dialog box. The format is as follows:

```
<MedicalRecordInsertions>

      <Template name="Imp feathers" useForEnclosureEntries="true"
      useForMedicalEntries="false">

      ----------- IMP REPORT -------------
      Left wing:  1 2 3 4 5 6 7 8 9 10
      Right wing: 1 2 3 4 5 6 7 8 9 10
      Tail: L 1 2 3 4 5 6, R 1 2 3 4 5 6

      </Template>

      <Template name="Change in orders" >

      ----------- ORDER CHANGE -------------

      </Template>

</MedicalRecordInsertions>
```

The **name** attribute is displayed in the list in the medical records entry dialog box. The text between the **<Template>** tags is inserted when the template is selected by the user. The **useForEnclosureEntries** and **useForMedicalEntries** determine whether these text templates are available when making Enclosure entries and Patient Medical entries, respectively.

Non-releasable letter template
The **NonReleasableLetterTemplate.rtf** is a template file for generating a letter stating that an animal is non-releasable. This letter is required by the US FWS and it uses their pre-defined format. This letter contains macros that are auto-expanded when the letter is generated. Each macro is surrounded by double triangle brackets as in **<<PATIENT_ID>>**. The following macros are currently used:

- PATIENT_ID
- SPECIES
- AGE
  SEX
- PROBLEMS
- TODAY

All macros described in the **Email Templates** section are supported but, in general, this file should not be edited. Also note that some editors, like Microsoft Word will often save the file incorrectly by inserting rtf codes inside one or more of the macros. Because of this, it is best to edit the file with Notepad.

Physical exam templates
The **PhysicalExamTemplates.xml** file is used to define a physical exam entry form for an animal. A different form can be defined for each major species group.
The correct template is found by matching the "root" of the patient's species group with the template's species group. That is, a patient with species group = **avian:raptor** will match with the **avian** template. If a template's `speciesGroup` is left blank, then it will match with any animal. The template file is searched from top to bottom so it is best to put the blank `speciesGroup` template last in the file so a more speciesGroup-specific template will be found first.

The format is as follows:

```
<PhysicalExamTemplates>

      <template speciesGroup="avian">

      <entry name="Eyes" edit="false">
            <entry name="Left" edit="false">
                  <entry name="PLR" label="Left PLR"/>
                  <entry name="Menace" label="Left menace response"/>
                  <entry name="Cornea" label="Left cornea"/>
```

```
                <entry name="Lens" label="Left lens"/>
            </entry>

        <entry name="Mucous membranes" type="checkboxes" columns="3">
            <checkbox name="WNL"/>
            <checkbox name="Dry"/>
            <checkbox name="Tacky"/>
            <checkbox name="Muddy"/>
            <checkbox name="Petechia"/>
        </entry>

        </template>

        <template speciesGroup="mammal">
            ...
        </template>

</PhysicalExamTemplates>
```

As the example shows, each template is really just a nested structure of **<entry>** tags. The nesting level of each **<entry>** tag determines the indent of the entry on the physical exam form and this helps make the form more readable. Each template can contain any number of entry fields and each entry field is created with an **<entry>** tag. The **<entry>** tags have the following format:

```
<entry name="" label="" edit="true or false" type="" lines=""/>
```

Each tag has the following attributes:

- **name** - This is the text that is displayed on the actual physical exam form. If the value for the name is "**Blank**", then a blank line will be shown on the form and this helps to make the form more readable. This value will also be used and inserted into the medical record unless it is overridden with the **label** tag.
- **type** - Default value is "**text**" for a normal text edit field. It can also have a value of "**checkboxes**" as shown in the example.
  - For "**text**" fields:
    - **edit** - Either "**true**" or "**false**". Default value is "**true**". This determines if an edit field will appear on the form. If not, then this entry serves as a visual "container" for subsequent entries that will be indented below it. This is only used for "**text**" type entry fields. It is ignored for all others.
  - For "**checkboxes**" fields:
    - **columns** - This indicates the number of columns for layout purposes. The default value is 5 if not specified and the ideal number really depends on the length of the text values used for each check box.
    - Each checkbox field can contain any number of **<checkbox>** tags. Each tag simple contains a name value that will be displayed next to the checkbox and will be printed into the patient record.

- The **lines** parameter defaults to 1 but can be set to a higher number for multiline edit fields.
- The **edit** parameter indicates whether this field is editable or not. If not specified, it assumes **true**. If **false** is specified, then this field simply serves as a placeholder heading and is helpful to organize the visual appearance of the form.
- **label** - This text will override what is actually inserted into the record. This is used in cases like the following:

```
<entry name="Ears" edit="false">
     <entry name="Left" label="Left ear" lines="2" >
     <entry name="Right" label="Right ear"/>
</entry>
```

In this case, there is already a heading for “Ears” and the “Left” and “Right” ears are subheadings underneath it. So it would be redundant to display “Ears”, the “Left ear” and “Right ear” underneath it. It is simpler to display “Ears”, then simply “Left” and “Right”. However, we still want the actual entry in the medical record to be specific so we can define **name**=”Left” but **label**=”Left ear”.

Procedures

The **Procedures.xml** file is used to define various procedures that can be recorded and itemized in each medical record entry. The format is as follows:

```
<procedures>

<category name="Patient evaluation" />

<category name="Diagnostics">
     <choice name="Blood collection" cost="75.00"/>
     <choice name="Necropsy"/>
     <choice name="Culture"/>
     <choice name="Biopsy"/>
     <choice name="Skin scraping"/>
     <choice name="Ultrasound"/>
     <choice name="Corneal stain"/>
     <choice name="Other"/>
</category>

</procedures>
```

The **procedures** are arranged in **categories** and each **category** can have any number (including none) of **choices**. Each **choice** can have an optional **cost** specified and this can be used to generate invoices.

Reports Cascading Style Sheet
The **ReportsCSS.html** file defines a stylesheet that is inserted into many of the repots.

Site Viewer
If the site map requires just one image, then **site.png** is the only file that is needed and it exists in **\Resources** in the folder that the application is installed in on the **client** machine.

If the site map is composed of multiple site images, the **Site1-n.png** files are needed and they exist in **c:\RaptorMed\Resources** on the server machine.

Species of concern
The **SpeciesOfConcernTemplates.xml** file is used to define the level of concern for each species. The level of concern can range from No concern to Endangered and the level can be different for the state and federal level. The format is as follows:

```
<speciesOfConcern>
      <region name="TX">
            <level description="No" abbrev="No" value="0"/>
            <level description="Species of concern" abbrev="Sco" value="2"/>
            <level description="Threatened" abbrev="ST" value="8"/>
            <level description="Endangered" abbrev="SE" value="10"/>
      </region>

      <region name="FEDERAL">
            <level description="No" abbrev="No" value="0"/>
            <level description="Species of concern" abbrev="Fco" value="2"/>
            <level description="Candidate" abbrev="FC" value="6"/>
            <level description="Threatened" abbrev="FT" value="8"/>
            <level description="Endangered" abbrev="FE" value="10"/>
      </region>

</speciesOfConcern>
```

There should be two **<region>** tags, one for the state that the organization is in and one for **"FEDERAL"**. The **description** is displayed in the **RaptorMed** interface. The **abbreviation** may be used on various reports and the **value** is actually stored in the database. The following **values** are defined

- 0 - no concern
- 8 - threatened
- 10 - endangered

Other **values** can be added as needed but these should not be changed.

Treatment options on the treatment sheet
All the non-medication treatment options on a patient's individual treatment plan are configured by choosing **Tools ▶ Maintenance ▶ Treatment options editor** from the main menu. These choices are stored in **TreatmentSheetOptions.xml**.

All medication options are configured using in **TreatmentChoices.xml**.

file. This file can be manually edited or it can be edited using the Medication Editor interface. It has the following overall structure:

```
<Treatments>
      <Antibiotics>
            <choice/>
      </Antibiotics>

      <Analgesics>
            <choice/>
      </Analgesics>

      <Other>
            <choice/>
      </Other>
</Treatments>
```

Each `<choice>` entry has the following structure:

```
<choice select="yes" name="Meloxicam" onClick="method" >
      <forms>
            <form value="1.5 mg/ml susp"  />
            <form value="5 mg/ml inj"  />
      </forms>
      <freqs>
            <freq value="BID" default=”yes” />
            <freq value=”dates” number=”2”/>
      </freqs>
      <routes>
            <route value="IM" default=”yes” />
            <route value="PO" />
      </routes>
      <until show="yes"/>
</choice>
```

- **select** - Is this option to be selected and therefore visible as an option on the treatment sheet?
- **name** - The text that will be displayed on the treatment printouts and on the patient's treatment sheet. This name that will also be used when looking in the **Dosages.xml** in order to auto-calculate the dosage.
- **onClick** - This optional attribute allows different operations to be performed when this treatment choice is selected. The currently supported operations are:

  - **approvalRequired** - This will force the user to enter their initials and the initials of the person approving the use of this treatment. Once the initials are entered, an entry documenting this treatment change will be appended to the patient's record.
  - **fluids** - This will allow the user to specify and save a fluid protocol and can be used when potentially nephrotoxic drugs are ordered.
- The **<forms>** list allows any number of different drug forms to be specified.
- The **<freqs>** list allows any number of frequencies to be specified. Typical choices include **SID**, **BID**, **TID**. In addition, you can also specify **Dates** and a **number**. In this case, the specified number of date entry fields will also be displayed. When the **Dates** option is chosen, **SID** is assumed for the frequency. You can specify **default="yes"** to make an option the default choice.
- The **<routes>** list allows any number of different administration routes to be specified. Typical choices include PO, IM, SQ, IV, IO, or Topical. You can specify **default="yes"** to make an option the default choice.
- The **<until show="yes">** field allows a "treat until" date box to be displayed. Note that if you choose the **Dates** option for the **frequency**, then an **Until** date box will never be displayed.

In order for the dosage auto-calculation feature to work, an matching entry must be found in the **Dosages.xml** file. A match is only possible when:

- The **name** in **TreatmentChoices.xml** matches a **<Drug> name** attribute in the same species group as the patient in question. See the next section for more information on the how the species group hierarchy is structured.
- The form matches.
- The specified route and frequency match.

Drug names

The drug name listed in the **TreatmentChoices.xml** file must exactly match an entry in the **Dosages.xml** file.

The name cannot contain a forward slash /

Any spaces should be replaced with an underscore. The program will display all underscores as a space.

WoundReports

The **WoundReports** function is configured in the **WoundsReports.xml** file. A wound report uses a image with pre-defined areas on which the user can assign a size, a

severity and a swelling score. Each report allows for a **maxScore**, a **maxSize** and a specific **image file** to be defined. Each report also allows any number of locations to be specified. Each location has a **name** and an **x/y coordinate**. The coordinates can be determined using any image editor program and the (0,0) is the upper-left corner.

Note: The **wound**, **swelling** and **size** attributes in each **<area>** tag are for internal use only and should be left blank.

```
<WoundReports version="1.0">

<type name="BumbleFoot" imageFile="WoundReports-Bumblefoot.jpg" maxScore="5"
maxSize="15">
       <area name="left:sole" x="177" y="106" wound="" swelling="" size=""/>
       <area name="left:digit1:1" x="177" y="143" wound="" swelling="" size=""/>
       <area name="left:digit2:1" x="160" y="94" wound="" swelling="" size=""/>
       </type>

<type name=...” >
</type>

</WoundReports>
```