Database Design Document

Structure and Rationale by Process Example

Authors:

Tammie Boykin, with help from:

Lincoln Sayger (Week of 2017/03/26)

# Table of Contents

Table of Contents

[Table of Contents 1](#__RefHeading__756_1368297671)

[Tables 1](#__RefHeading__206_1368297671)

[Reserve 1](#__RefHeading__208_1368297671)

[Address 3](#__RefHeading__210_1368297671)

[Volunteer 3](#__RefHeading__642_1368297671)

[Team 4](#__RefHeading__644_1368297671)

[Charges 5](#__RefHeading__1562_539497079)

[Additionals 6](#__RefHeading__1580_539497079)

[Roster 6](#__RefHeading__1592_539497079)

[Payments 7](#__RefHeading__1608_539497079)

[Church 7](#__RefHeading__1628_539497079)

[Blackouts 8](#__RefHeading__1642_539497079)

[Meta 8](#__RefHeading__1654_539497079)

[Forms 8](#__RefHeading__646_1368297671)

[Reservation 9](#__RefHeading__1670_539497079)

[Reports 10](#__RefHeading__648_1368297671)

[Back-end Considerations 10](#__RefHeading__650_1368297671)

[Reservation processor 10](#__RefHeading__1708_539497079)

[30-day reminder 13](#__RefHeading__1178_1571812294)

[60-day reminder 13](#__RefHeading__1724_539497079)

# Tables

## Reserve

The *reserve* table holds reservations. A **reservation** is a request for a specific church to send a team of a specific size on up to three possible dates.

Each team requesting a reservation for a week of work at UMCOR Sager-Brown will be entered into the reservation table, with the following information:

* id (Reservation ID): an automatically generated key. BIGINT
* contact (Timestamp of request): The date and time on which the reservation was received by USB. Requests received before the designated opening of reservations will be processed at the end of the day, in order of their proximity to the designated opening (that is, in reverse chronological order). DATETIME
* contype (Contact type): P(hone), E(mail), M(ail), V(isit), among other possible designations for the way the reservation was made. CHAR(2)
* date1 (Primary date): Date requested (Primary); Also date2 (Secondary date); Also date3 (Tertiary date). DATE;DATE;DATE
* notes (Notes): Any explanatory remarks needed to process the reservation, such as coming for the weekend or coming for two contiguous weeks. TEXT
* volunteerid (Team leader): The id key of the volunteer record (described later) for the team leader making the request. If this leader is not in the database, a new volunteer record will need to be created. BIGINT
* dayphone (Daytime phone #): The daytime phone number of the team leader. VARCHAR(11)
* church (Church): The id key of the sending church’s record (described later). If this record does not exist, it will need to be created. BIGINT
* size (Team size): How many people are on the team. INT
* slips (RV Slips requested): How many RVs does the team already know it wishes to bring? INT
* aparts (Apartments requested): How many apartments does the team already know it desires? INT

Once the information in this table is entered, it will wait for processing. At the end of each day, the reservations should be processed using “ORDER BY contact ASC” (with, for example, “WHERE HOUR(contact) >= ‘10’” for initial processing and “WHERE HOUR(contact) < ‘10’” for end-of-day processing on the opening date of reservations) to assign dates requested.

Each record’s primary date should be tried, with its size, slips, and apartments being compared to the remaining capacity (see the *meta* table for weekly caps and maximums) for that date. If the size would exceed a cap, the team’s secondary date is tried in like manner. If necessary, the team’s third date is tried. If no date is viable, the primary date (date1) is moved to date2, and a 0 value is placed in the primary date field as a marker for reporting unassigned reservations.

If the reservation system is offered online, it should offer only the Sundays for dates that can be chosen (Monday for weeks the Sunday is blocked).

## Address

The *address* table stores address information for a number of other tables. An **address** is a discrete location where a church, team leader, or volunteer resides.

* id (Address ID): A key for referring to the address. BIGINT
* street (Street address): A street number and street name. VARCHAR(128)
* city (City name): The city where the street resides. VARCHAR(32)
* state (State name): The state wherein the city is located. VARCHAR(3)
* post (Postal code): ZIP, or other postal code (e.g., 90210-1234 for the USA, V6B4G3 for Canada, 14-200 for Poland). VARCHAR(10)

This information, once entered, can be linked to a *volunteer* record, church, team record, or *reserve* record.

## Volunteer

The *volunteer* table stores volunteer records. A **volunteer** is a particular person who volunteers at UMCOR Sager-Brown.

Information relating to a particular person includes:

* id (Volunteer ID #): This key, used by other tables, can be given to the volunteer to speed paperwork processing. BIGINT
* given (First name): This is the volunteer’s given name, such as Eloise. VARCHAR(21)
* family (Last name): This is the volunteer’s family name, such as Smith. VARCHAR(21)
* printname (Preferred name): This is what the volunteer prefers to be called, such as Ellie. It will appear on both the nametag and the certificate given to each volunteer. VARCHAR(16)
* male (Male/Female): Is the volunteer male (M) or Female (F)? CHAR
* skills (Special skills): What skills does the volunteer possess? VARCHAR(300)
* birth (Birthdate): When was the volunteer born? DATE
* vaddress (Volunteer’s Address): Where does the volunteer live? Refers to an address record. BIGINT
* email (Email address): Volunteer’s email address. This is used for reservation confirmations and as a login for any future Web interface. VARCHAR(256)
* accom (Special needs): If the volunteer requires any accommodations, they are listed here. VARCHAR(128)

The volunteer record is also used for team leaders.

## Team

The *team* table stores team-wide information about a team. A **team** is a group of people visiting the facility on a particular week.

Team records contain the following data:

* id (Team ID): The team’s key value. BIGINT
* arrive (Arrival): The date the team is scheduled to arrive. DATE
* depart (Departure): The date the team is scheduled to depart. DATE
* leader (Leader ID): The key value of the leader’s record in the *volunteer* table. BIGINT
* laddress (Leader address): The key value of the leader’s address in the *address* table. BIGINT
* dayphone (Daytime phone #): The leader’s daytime phone number. VARCHAR(11)
* evephone (Evening phone #): The phone number where the leader can be reached during evening hours. VARCHAR(11)
* fax (Fax #): The number where information and forms can be faxed to the team leader. VARCHAR(11)
* cell (Leader cell): The cell phone number of the team leader. VARCHAR(11)
* church (Church): The index of the church’s record. BIGINT
* pastor (Pastor’s name): The name of the team’s (or team leader’s) pastor. VARCHAR(75)
* size (Team size): The number of volunteers expected on the team. INT
* contacted (Confirmation date): The date on which the team leader was emailed with a confirmed arrival date.

Once the reservations are processed, certain of these fields will be filled automatically. The arrival date will be recorded as the primary, secondary, or tertiary request date from the *reserve* table. The departure date will be calculated from the arrival as either a Friday, or if that day is blocked in the *blackout* table, the latest date available in the week. The leader index is copied from *reserve.volunteerid*. The leader’s address is gleaned from the volunteer record. The daytime phone # is drawn from the reservation phone number. The church name, phone, and address are copied over. The team size is copied from the reservation. The contact date should be filled by the frontend using CURRENT\_DATE when the team is created, or by a processing program when it has sent an email to the leader.

Remaining fields may be entered by the user when the team creation is initiated, or manually when the team pays some or all of its charges.

## Charges

The *charges* table stores the charges associated with a team. A **charge record** is a list of the fees related to a particular team’s visit.

These are the fees that are most commonly associated with a team:

* team (Team ID): The index of the team for which these charges are due. BIGINT
* beds (Beds): The number of beds needed in dorms. INT
* slips (RV slips): The number of RV slots needed. INT
* aparts (Apartments): The number of apartments requested by the team. INT
* meals (Meals): The number of volunteers taking all meals at the dining hall. INT

Once these fee determinants are entered, the report generator can calculate the amounts of the fees and sum them.

## Additionals

Sometimes, nonstandard charges may be applied to a team, such as when they add a team member late in the trip preparation, or when they receive a special rate. The *additionals* table stores one of these fees and its description, in this manner:

* team (Team ID): The index of the team for which this charge is due. BIGINT
* addsamt (Additional Amount): The amount of the additional fee. FLOAT DEFAULT 0.00
* addsdesc (Additional Amount Description): The type of charge being assessed. VARCHAR(32)

The description is short because it is intended as a shorthand indicator of a type of fee, not a detailed explanation of it.

## Roster

Once the team leader has sent a roster of volunteers, the *roster* table allows the volunteers to be associated with the team in the database. A **roster** is an association between one or more *volunteer* records and a *team* record, as well as data that applies to the volunteers only in connection with a specific visit:

* teamid (Team ID): The index value of the team to which the volunteer belongs. BIGINT
* volunteerid (Volunteer ID): The index value of the volunteer being assigned to a team. BIGINT
* age (Age at visit): How old the volunteer will be during the team’s visit. INT
* room (Room assigned): Which room will the volunteer use? Character, not numeric, in case hall letters are desired. VARCHAR(6)
* forms (Submission of various forms): A bitmask designating the presence or absence of various forms, including (in this order from 1-32 and beyond): L(iability) | M(edical) | Y(outh covenant) | B(ackground check) | M(odel release) | S(afety briefing). Other forms may be added by altering the front-end program to read/alter additional bits in the mask. INT

Initially, the fields for room and forms will likely be blank/zero, so they should not be marked NOT NULL. The age can be calculated using DATEDIFF on the arrival date and the volunteer’s birthdate.

## Payments

When the team has sent a payment to the facility, it may be recorded in the *payments* table. A **payment** is a dollar amount paid and the transaction information that goes with it:

* id (Payment ID): The key value for this individual payment. BIGINT
* team (Team ID): The index value of the team making the payment. BIGINT
* transaction (Check #): The check number or other transaction designation. VARCHAR(7)
* payver (Payment receiver): The initials of the person receiving the payment. This field should be filled automatically by the front-end, based on who is logged into it, or manually by the user, if the payment is being recorded in a MySQL client. CHAR(4)
* amount (Payment amount): The amount being paid. FLOAT
* paydate (Payment date): The date on which the payment was made. DATE
* processing (Is processing?): Whether or not this is a processing fee being paid. CHAR(1)

Payments will be summed and compared to the charges and additionals to determine when the full costs have been paid.

## Church

The *church* table stores information relating to churches. A **church** is a name, address, and phone number for a particular congregation. It consists of:

* id (Church ID): The key value for the church, used in other forms. BIGINT
* name (Church name): The name (preferably the legal name) of the church. VARCHAR(256)
* churchphone (Church phone #): The phone number of the church. VARCHAR(11)
* address (Church address): The index value of the church’s address. BIGINT

The pastor’s name is stored with team records, because it may be desirable at some point to see who was pastor when a particular team was sent.

## Blackouts

The *blackouts* table stores blocked dates. **Blocked dates** are parts of the calendar year when the facility is not accepting volunteers, for one reason or another. Here’s how they’re stored in the database:

* start (Start date): The blockout’s beginning. DATE
* finish (End date): The blockout’s ending. DATE
* reason (Rationale): The rationale for the block. VARCHAR(128)

When reservations are processed, the blackout dates are compared with the requested dates, and when dates are assigned, the blackout dates will be removed from the reservation’s span.

## Meta

The *meta* table stores meta-information that is used by the front-end interfaces to process data along predetermined policies. It contains only one row, and its fields are:

* wkcap (Weekly cap): The soft maximum capacity, based on the prevalence of mixed groups or couples, and which rooms can be filled to capacity, and how many people drop from teams, on average. INT
* wkmax (Weekly max): The hard maximum capacity, the most volunteers who can be properly accommodated on campus. INT
* rvmax (Max RV slips): Total number of RV slips. INT
* apmax (Max apartments): Total number of apartments. INT
* capflex (Capacity flexibility): The amount by which a reservation can exceed the weekly cap and still be approved automatically.

It is very important that the front-ends do not create additional rows and read only from the first row.

# Forms

A number of forms will be used in conjunction with this database. Following is a list of them, with their proposed interfaces and/or programming languages, with considerations attendant on each.

## Reservation

The form used by administration to record reservation requests will present the following visible fields:

* Date, prefilled with the current date. (*reserve.contact*)
* Time, prefilled with the current time. (*reserve.contact*)
* Email checkbox, which if checked, makes the Time field editable. (no DB field)
* Contact type: radio buttons for Phone Call, Email, Mail, Visit. For each checkbox, there is a 1-2 character code the front-end will enter into the database field, when the request is submitted. (*reserve.contype*)
* Team Leader button, which opens a search window where the leader’s family name can be entered for searching. If the leader does not have an existing record, the user can click the Add Leader button in the search window to add their information. (*reserve.volunteerid*)
* Address, prefilled from the *volunteer* record, read-only. A button may be present next to the field allowing the *volunteer* record’s address to be edited. (*volunteer.vaddress*)
* Daytime Phone, data validated field, template (###)###-#### (*reserve.dayphone*)
* Email, prefilled from the *volunteer* record. A checkbox next to this field may update the *volunteer* record if the value changes. (*volunteer.email*)
* Church, button opening a window to search/add/update the church. The form should remind the user to verify the phone number and update it, if necessary. Beside this button, the selected church’s name, address, and phone number should be displayed in text labels. (*reserve.church*)
* Requested Mission Dates, three fields, with calendar buttons. (*reserve.date1*, *reserve.date2*, *reserve.date3*)
* Team Size, numeric. (*reserve.size*)
* RVs, numeric. (*reserve.slips*)
* Apartments, numeric. (*reserve.aparts*)
* Notes, textbox. (*reserve.notes*)
* Cancel, button. This button clears all fields and refills all prefilled fields with updated values.
* Submit, button.

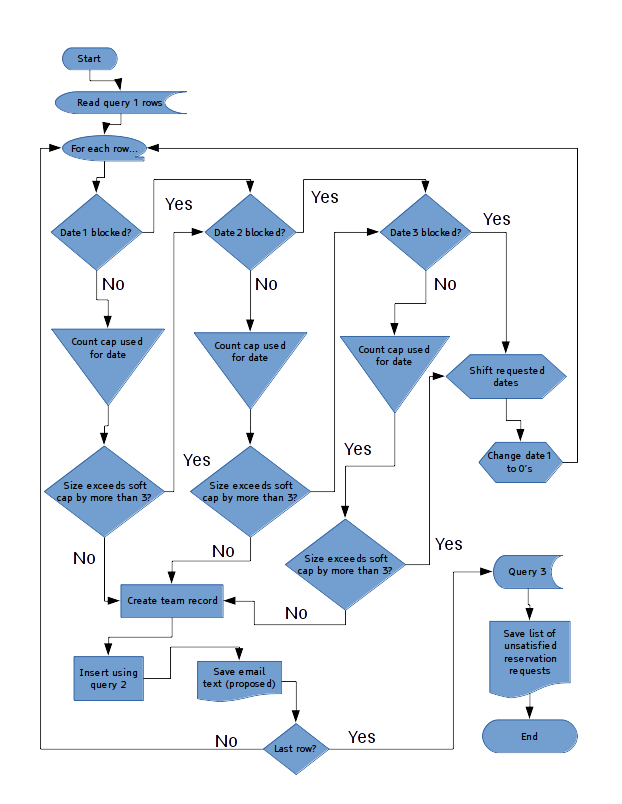
When the Submit button is pressed, the form will validate all fields and prompt for missing information, then perform an INSERT query on the database. The form will then close.

# Reports

# Back-end Considerations

## Reservation processor

At the end of each day, the reservation processor will be run, either manually or automatically. It will perform this procedure:

These are the queries used in this procedure:

1. “SELECT \* FROM reserve WHERE DATE(contact) = CURRENT\_DATE() ORDER BY contact ASC” (on the first day of reservations, the clause “WHERE HOUR(contact) >= ‘10’”, or similar, is inserted before the ORDER BY clause, for a first pass. In a second pass, the ORDER BY clause is replaced by “WHERE HOUR(contact) < ‘10’ ORDER BY contact DESC”, or similar).
2. In this query, “{reserve.fieldname}” is a placeholder. Modify this to use the actual data from the individual row, in your programming. “INSERT INTO team SET arrive={reserve.date[123]}, depart=ADDDATE({reserve.date[123]},5), leader={reserve.volunteerid}, dayphone={reserve.dayphone}, church={reserve.church}, size={reserve.size};”
3. “SELECT \* FROM reserve WHERE date1=’0000-00-00 00:00:00’;”

The processor should have a command-line flag (and/or checkbox in its GUI) that tells the program to send out all proposed emails for the day, if there are no unsatisfied requests (using query 3 to check).

When checking blackout dates, the processor should move the date being checked to a Monday if only the Sunday is blocked.

The check for capacity exceeding by 3 is an example only. The field *meta.capflex* should be used in place of this example value.

## 30-day reminder

## 60-day reminder