The Federation Website

Sidney Marshall

June 21, 2012 (updated February 27, 2024)

Contents

Abstract

The Federation website is described with details on how the website is put together. Also described are the tools used to automatically build the website. All of the software that is used for maintaining the website is free.

Add-on domains are also on the Federation website and the administration of these is also described.

1 Introduction

The Federation website contains information about square and round dance clubs in the Rochester area. One of the most important parts of the site is the dance schedule for the area clubs. Most of the current website pages are generated automatically from a database by special scripts. Because of this, changes to the content and format are easy to accomplish.

All of the information to create the website are contained in a database along with the database statements needed to actually create the website. Schedules, club officers etc. are updated by changing the appropriate database table. Structural changes to the website are made by editing the tables containing the SQL statements used in building the website.

The database can be updated using the database editor `dbe.jar`. The website is updated by running `makeweb.jar`. Modified files are copied to the website image area using `copyfiles.jar`. Then either `copyfiles.jar` or a program such as `WinSCP` is used to actually update the "live" website. Either of the local copies of the website can be browsed with a browser to check if the updates were correctly made.

Add-on domains can be updated by editing their raw `.html` pages but these pages are usually updated by someone else.

2 Top-Level View of the Website

The `squaredancingrochester.org` website contains web pages and additional files such as `.jpg` files and `.pdf` files. Most of the `.html` pages that are not contained in subdomains or specific to clubs are generated by scripts, especially pages containing data like dance schedules and club information.

The various types of generated web pages are:

Announcements	Federation news
Articles	Various news articles, roving reporter, obituaries of interest to the square dance community
Awards	A description of some awards in the square dance community
Callers and Cuers	A list of callers and cuers in the area

Class/Club dances	A list of class/club dances in the area
Club Dance List	A list of dances sorted by club
Club News	News for each club in the federation
Clubs	A list of clubs in the area
Editor's Message	A message from the editor
FAQ Page	A list of general questions and answers
Federation Home Page	The home page for the Rochester Federation
Federation Staff	A list of federation staff members
History	Currently contains scanned past issues of the Promenader
Home Page	The initial page of the website
Caller Dance List	A list of dances sorted by caller and cuer
Dance List by Day	A list of dances sorted by month day
Dance List keys	A list of letter codes used on dance lists
Dance List by Month	A list of dances in a month calendar format
Dance List by Weekday	A list of dances sorted by day of week
Federation Meetings	A list of federation meeting days
Federation Minutes	Minutes of federation meetings
New Dancer Information	Information for the new dancer
President's Message	A message from the federation president
Promenader	Top page listing promenader sections and a list of special dances
Promenader Staff	Listing of staff members of the Promenader
Single Page	Many odd pages have this type and are just copied from the database
Cascading Style Sheet	A cascading style sheet setting the styles for the website pages
Flyers	A page containing images of flyers sorted by date
Site Map	A site map for the federation website

There is also a proofreading section containing some experimental pages and:

Articles for Proofing	Articles for proofreading before going on the website proper
Club News for Proofing	Club news for proofing before going on the website proper
Sandbox Home	The home page for the proofreaders and experimental pages
Full Site Map	An experimental site map for the entire website including all subdomains

The three main top pages are (currently) the home page, the promenader home page, and the federation home page. Most pages have direct links to these pages.

The dance schedules link to a view of the current month's dance schedule. There are three different monthly views available — one with the days of the month in order, one with the days grouped by weekday, and one with a layout like a conventional calendar. Also available is a list of dances sorted by caller/cuer and another page sorted by club. All data comes from the same database so all of the information is consistent.

The clubs page lists the square dance clubs and other information from the database.

There are five directories containing content not in the database:

compressed	Compressed files for articles, club news, and flyers
flyers	All flyers - both `pdf`s and `.jpg`s
gifs	Logos for the club news
jpg	Various `.jpg`s for page headers etc.
png	Various `.jpg`s for articles

3 Workflow for Building the Website

All of the data necessary for creating the html pages of the website is in the database `squaredance.db` except for `.pdf` files, `.jpg` files, and other image-type documents. The tools need a `Java` environment to run. You do not need the `JDK` if you are not rebuilding the tools.

3.1 Programs Used

Several programs are used to build and maintain the website. They can be run in either a linux or Microsoft (or probably Apple) environment. All of the special software is stored on the website so it can be easily downloaded.

For making thumbnail images for the flyers page I use `imagemagick` under linux but any appropriate program can be used. Photos for the articles section generally are submitted with very high resolution so I reduce the resolution to 500 pixels wide using `gimp` under linux, but again, any suitable image processing program can be used. This saves a lot of space on the website and the images download quicker.

3.2 Useful Java Programs

There are several Java programs used in maintaining the website. All of these programs are on the website.

Clean.class	Checks a file for characters that annoy a web browser
dbe.jar	A display editor for editing the database
makeweb.jar	This program recreates the entire website from the database
copyfiles.jar	Copies files that have been changed from one directory to another
filebrowser.jar	A very general file transfer program

3.2.1 Clean

Usage: `java Clean files....`

All of the files on the command line are scanned for "funny" bytes. A "funny" byte is any byte less than `0x20` (control character other than `0x0a` (new-line) or any character greater than `0x7e`. If one is found the line in which it appears is printed with a `^` pointing to the offending byte. This program is used to find characters that need to be replaced in a web page.

Here is a useful table of character replacements:

Quotation mark	`&quot;`
Ampersand	`&amp;`
Less-than sign	`&lt;`
Greater-than sign	`&gt;`
Open single quote	`&lsquo;`
Close single quote	`&rsquo;`
Open double quotes	`&ldquo;`

Close double quotes	`&rdquo;`
En dash	`&ndash;`
Em dash	`&mdash;`
Bullet	`&bull;`
Non-breaking space	`&nbsp;`
Horizontal ellipsis	`&hellip;`

3.2.2 dbe.jar

Usage: `java -jar dbe.jar [database-name]`.

If the optional database name is not included a file browser is displayed to choose the database to edit. Complete documentation of this editor can be found in `DBEditor.ps`, and `DBEditor.pdf`.

All of the data required for generating the website are stored in an SQLite database called `squaredance.db`. You can modify this database with the command

```
java -jar dbe.jar squaredance.db &
```

You can modify tables or execute SQL commands with this editor. The documentation for using this editor is in `DBEditor.ps` and `DBEditor.pdf`.

3.2.3 makeweb.jar

Usage: `java -jar makeweb [-map [root]]`.

If the `-map` option is used alone the `makeweb` program first creates in the sitemap table an image of the current directory and subdirectories. If a directory name appears after the `-map` option then a map is made of that directory. I generally use

```
java -jar makeweb.jar -map ../upload
```

so that the dates are closer to what is actually up on the web. (You would have to run the makeweb, copyfiles sequence twice to be more accurate.)

The database used is hard coded to be `squaredance.db`. The `makeweb` program reads SQL statements from the database and creates several web pages in the current directory and subdirectories. Any errors in the process such as SQL statement errors will terminate the process with an obscure error message. You will also get an error message if the scripts try to create files in a nonexistant directory so you will have to make any new directories manually.

3.2.4 copyfiles.jar

Usage: `java -jar copyfiles.jar [-m] source-dir target-dir`.

This copies all files that do not have the same contents from the source directory to the target directory. If the `-m` flag is used then extra files in the target directory will be deleted and the file dates in the target directory will be set equal to the file dates in the source directory. You probably don't want to use the `-m` flag for website operations.

The Java program `copyfiles.jar` copies all changed files from one directory to another. I keep the working website files in a directory called `squaredance` and the files that are an image of the actual website in a directory called `upload`. Then, when I am finished updating the database and adding `.pdf` and `.jpg` files to the working copy in `squaredance` I execute the command

```
java -jar copyfiles.jar . ../upload
```

to copy the changed files to the `upload` directory.

3.2.5 filebrowser.jar

The program, `filebrowser.jar`, is a general purpose file transfer program that mimics WinSCP or FileZilla. It has the capability of comparing and transfering between two local file systems, two remote file systems, or a local and remote file system. It also has editors for editing files and a diff program for comparing two files. It also has a remote shell capability. The documentation for this program is in a separate file.

3.2.6 Java / sqlite-jdbc

The sqlitejdbc project has a free sqlite3 interface totally implemented in Java. I used this interface to write Java programs to use `SQLite3` databases. The two programs used for maintaining the website are `dbe.jar` and `makeweb.jar`. The program `dbe.jar` is used to enter and update data for the website. The program can also execute raw sql commands for changing the structure of the database or you can use other tools (like the SQLite shell) for this. The program `makeweb.jar` uses the information in the `squaredance.db` database and creates most of the `.html` files on the website.

3.3 WinSCP and ftp

`WinSCP` is a free program that provides a windows interface to the unix scp program. It can be downloaded from `winscp.net`. It allows a quick way to copy files between machines and indicate which files need to be updated. I generally use the `filebrowser.jar` for transferring files. I use this program to upload files to the web server. This program works on most platforms.

Do not upload or change any files in the subdomains `callers/MikeCallahan`, `clubs/CloverleafSquares`, or `SSDUSA` as these pages are managed by someone else. If you refresh in both directions you won't overwrite changes to subdomains. You will want to propagate changes to these subdomain websites to the squaredance directory so that the next copyfiles won't change them. This will allow backing up the subdomains easily.

3.4 emacs

I use emacs for all of my text editing. Any plain-text editor can be used including a text editor window in the Database Editor. Note that `test.db` is a scratch db so editing anything in this database is OK and can be used for trying out new ideas. The main editing tasks are to edit the content in the database. This could be done using the database editor but some extra editing tools may be useful. I also use emacs to edit the documentation (in the `.tex` files).

4 Building The Website

I keep an image of the website in two parallel directories: `squaredance` and `upload`.

The directory `squaredance` is where I do my modifications. The date is set to the current date so you don't have to set anything. (The database engine apparently runs on GMT so it will "see" the next day after 7 or 8 p.m. of the current day. To edit the database type

```
java -jar dbe.jar squaredance.db &
```

or click on the `dbe.jar` icon and select `squaredance.db` to run the database editor.

After editing the database (and any other files) I run

```
java -jar makeweb.jar -map ../upload
```

This will update the sitemap table and then rebuild the website using the contents of the database. You should then be able to browse the updated site in the squaredance directory.

After making the `squaredance` directory the way you want it to be, execute

```
java .jar copyfiles.jar . ../upload
```

to copy the changed files to the `upload` directory. I generally run

```
java -jar makeweb.jar -map ../upload
java -jar copyfiles.jar . ../upload
```

again to get the full site map updated.

After copying the changed files these two directories will be identical (except for the dates on the files that have been remade but whose contents haven't changed). I do this copying so that files that have been remade with identical content won't have to be uploaded to the website.

Now the directory `upload` contains all of the files that the website needs. I then run

```
java -jar filebrowser.jar
 /home/sidney/home/upload/
 //squaree8@squaredancingrochester.org/home1/squaree8/public_html
```

(all on one line).

The copying process defaults to downloading files on the website that are not in `upload`. You should make sure you do not upload files to any of the add-on domains. Just click on the green arrow to delete the local copy.

4.1 Add-on Domains

Add-on domain web sites could be edited in the `squaredance` directory and then proceed as above. But these files are generally maintained by someone else. If someone else is maintaining these pages you must be careful not to upload pages to the add-on website pages as this will revert any changes that they may have made. Generally you want to only download or delete local files in a subdomain.

4.2 Transferring "Ownership" of the Website

Only one person should be updating the website at a time. If several people attempt to update the website at the same time chaos will result. The tools require an exact local image of the website (e.g., in directory `upload`). This copy will be stale if someone else updates the website causing the other person's updates to be lost when the original person does another update. Therefore the following ritual should be followed for transferring "ownership".

Only one person can have the "baton" at a time. This is the person currently updating the website. To pass the baton this person finishes all updates and unambiguously gives up the baton to another person. At this point the website contains the "truth" about the contents of the website.

The person receiving the baton then downloads changes to mirror the website on his/her machine in the `upload` directory. This directory is then (locally) mirrored to the `squaredance` directory preserving dates and removing files that do not appear in the `upload` directory. The new baton owner is now free to manage the website.

Note that this process of handing over the baton only requires downloading the changed files since the last time the baton was owned. (A new baton owner will have to download the entire site.)

4.3 Typical Website Operations

Most operations can be done by clicking on the `dbe.jar` icon or running

```
java -jar dbe.jar squaredance.db &
```

and editing the various database tables. Note that views cannot be edited but triggers have been added to some views to give the appearance of editing a table. The `schedule` view has such triggers associated with it so some editing functions can be done by editing this view although not in an obvious way. You must click the `Update Table` button to preserve your edits or they will be lost. If you make a mistake you can click `Revert Table` or `Revert Selected` to abandon your edit(s). For views the `Update Table` button also refreshes the view with any updated data from the tables after executing the appropriate triggers.

After completing a sequence of edits the command

```
java -jar makeweb.jar -map ../upload
```

should be executed to update the website. The `-map ../upload` just updates the `fullsitemap` table in the `squaredance.db` database from the specified directory.

4.3.1 Adding A Dance

I add a dance by first adding a row to the `event` table and filling in the date (the date format must be exact - four-digit year, hyphen, two-digit month, hyphen, two-digit day). Multiple lines can be added at the same time to enter multiple dances. You must click `Update Table` on the `event` table before proceeding. You must also click `Update Table` on the `schedule` view to see the new dates.

I then edit the `schedule` view to enter the various facts about the dance. Then click `Update Table` to verify your changes. You must enter the `clubid` from the `club` table in the `clubname` column. Each `clubid` is added to the list of clubs sponsoring the dance. You must click `Update Table` after each `clubid` is added. To remove a club you must delete all clubs associated with the event by dragging a `null` to the `clubname` column and starting over.

The `venuename` should be filled in with the `venueid` (found on the `venue` table). Only one venue is allowed with an event. The `venueadd1`, `venueadd2`, and `latlong` fields will be filled in automatically (from the venue table) when `Update Table` is clicked.

The `caller` and `cuer` fields work like the `clubname` field and allow multiple callers and cuers to be associated with an event. You must enter the `personid` from the `personview` view. Just like the `clubname` column you must drag a `null` to the `caller` or `cuer` field and start over if you wish to remove a name.

The fields `time`, `flags`, `eventname`, and `comment` are just text fields and can be edited normally.

4.3.2 Adding A Flyer to a Dance

You should make the flyer a `.pdf` file and put it in the flyers directory (with a name describing the dance and date). It is important that the extension `.pdf` be in lower case. Then edit the `eventpdf` table adding a row and putting the `eventid` in the `eventid` field and the filename of the flyer in the `flyer` field. It is OK to have the same flyer associated with several events or several flyers associated with the same event. Flyers should be named consistently. I generally do all work in the `tmp` directory and move the results to the `flyers` directory.

4.3.3 Adding A Flyer or Ad to the Classifieds

After putting the `.pdf` file in the flyers directory as above you should make a `.jpg` of the first page of the flyer with a height of 500 pixels. It is important that the extensions `.pdf` and `.jpg` be in lower case. I use `convert` from the `ImageMagick` suite of tools although any program capable of converting a `.pdf` to a `.jpg` with a height of 500 pixels can be used. The `.jpg` should have the same base name as the `.pdf` and be stored in the flyers directory. I use the command `convert -thumbnail x500 x.pdf[0] -flatten x.jpg`

Then add a row to the `classifieds` table. The `startdate` is the starting date the flier should be displayed. The `sortdate` is the date used for sorting the classifieds. The `enddate` is the date after which the flyer should not appear. I generally set this to be one day after the event. The `name` is the root name of the `.pdf` and `.jpg` — the thumbnail will be linked to the `.pdf` file.

4.3.4 Updating a Flyer

Since all of the database tables and flyer information are already on the website you just need to replace the flyer. To update a flyer, make a new `.jpg` of the new flyer and replace the flyer `.pdf` and `.jpg` files in the flyers directory and run `java -jar copyfiles.jar . ../upload/`. Then upload the `upload` directory to the website. The full site map won't have the correct date for the new flyers but I don't think anyone will notice.

4.3.5 Adding A Venue

To add a new venue just add a row to the `venue` table and enter the appropriate data in the proper fields. As a hint, if you go to `maps.google.com` and right click on the location, and select `Whats here?`. This will display the latitude and longitude of the clicked location. Left-click on it then copy and paste this value into the `latlong` field.

4.3.6 Adding A Person or Changing a Person's Information

To add a new person to the database just add a row to the `person` table or `personview`. The `personview` view has triggers to allow editing all fields except the `personid` column. This column will be filled in by the database when the table/view is updated.

If a person changes their name/email etc., just edit the appropriate cell in the table or view. When `java -jar makeweb.jar -map ../upload/` is run all instances of that person's information will be updated.

4.3.7 Releasing a new edition of the promenader

For proofreading, the new articles, along with the editor's message and the presitent's message, go in the `articleproof` table and `articlecontent` tables. All of the new club news goes in the `clubnewsproof` table.

This will put the new articles, editor's message, president's message, and club news in the sandbox for proofreading.

To release a new issue of the promenader

> Copy all new articles from articleproof to article. (Note that articlecontent does not change.)
>
> ```
> insert into article select * from articleproof
> order by date, rank, articleid
> ```
>
> Copy all club news from clubnewsproof to clubnews.
>
> ```
> insert into clubnews select * from clubnewsproof
> ```
>
> Update table content — row promenader/date — to reflect the current issue number. There are 4 issues a year. Increment the roman numeral for the WINTER edition.

Then remake the website as usual. The entries in the `articleproof` and `clubnewsproof` can be deleted sometime later in preparation for the next issue. The editor's message and presidents message will go in the proper place and the other articles will be collected in the photos and features section.

5 The Database

Most of the changeable data on the website is stored in a database. This database includes information about people, clubs, events, dance locations, and other kinds of information that can be reduced to a table form. The database engine that is used is `sqlite3` which is a free relational database engine that is used in many applications.

To generate most of the pages on the website a script is run that queries the database and generates the pages. Common modifications to the website are made by updating the database and running the script. This insures that all of the pages are consistent with each other. Modifications to website styles can be made in one place.

5.1 Automatically Generated Views

Sometimes it is convenient to have a view on a table that is sorted in a particular order. `SQLite` does not permit operations on views but it does allow triggers on views that can update a backing table instead of updating the view.

For simple cases there are scripts in the database machinery that makes this easy. There are two tables that implement this functionality: `vieworder` and `views`. Only one view per table is allowed using this mechanism because of name collisions. This restriction could be lifted if there were a need.

The table `vieworder` contains a column `tablename` giving the name of the table which to view and a column `tableorder` giving the sort order as a comma-separated list of columns in the base table to sort on.

The table `views` has a column `tablename` with the name of the table, a column `position` giving the relative position of a column in the view, and a column `column` giving the name of the column in the table and view.

The scripts create the view with the specified column order (which can be different from the base table order) and triggers that handle inserting rows into the view, deleting rows from the view, and updating cells in the view.

5.2 The Main Tables in the Square Dance Database

5.3 Tables, Views, and Triggers

Here are the tables and views used in the database with a short description. All tables with an

```
INTEGER PRIMARY KEY AUTOINCREMENT
```

designation use this value as a key for other tables to reference an entry in this table. These keys are guaranteed never to be changed or reused. You should not attempt to set this kind of field.

5.3.1 Table article

This table contains the published articles for the photos and features section of the promenader.

The `date` field is the date of the promenader issue.

The `rank` field is the rank order for articles in a particular date.

The `type` field is one of news, president, reporter, obit, or editor. The news, reporter, obit types go in the photos and features section. The president type goes in the President's Corner under the federation page. The editor type goes to the Editor's page linked from the Promenater page.

The `heading` field is the title of the piece.

The `subheading` field (if needed) is the subheading of the piece.

The content of the article is in the `articlecontent` or `articleproof` table.

5.3.2 Table articlecontent

This table contains the actual contents of an article. An article can contain text, photos, or headings.

The `articleid` field is the articleid of the article in the `articlecontent` or `articleproof` table.

The `position` field is the position in the article where this part of the article should appear.

The `type` field is one of jpg for pictures, h6 for a heading, or text for text, or author for an author line.

The `content` field is the .jpg file name for jpgs (these jpgs belong in the `png` directory), the heading for type h6, or the text for type text.

5.3.3 View articlecontentview

Automatically generated editable view on the `articlecontent` table.

5.3.4 Table articleproof

This table is identical in format to the `article` table but the articles etc. will appear in the sandbox for proofing.

5.3.5 View automation

The `automation` view is there to help understand how the `mkkeweb` program builds the website and is a view on the `substitution`, `pagesets`, and `scripts` tables ordered by how the automation process will process these tables. The `automation` view is not editable (maybe in the future). This view is not used by any of the machinery but this view might be helpful if you are trying to figure out how a particular web page is made.

5.3.6 Table breakingnews

This is where "breaking news" on the home page is stored. The `sortdate` column stores the date of the "breaking news". The `enddate` stores the date when the news should disappear. The `heading` column contains the headline for the news. The `news` column contains the news. Double newlines indicate paragraph breaks.

If there is no news to display then the breaking news section does not appear.

5.3.7 View callers

This view is used by the schedule view.

5.3.8 Table callerscuers

Which people are callers or cuerers and if they are members of the CCR.

5.3.9 Table classifieds

This table puts flyers in the ads and flyers section of the promenader. The automation process uses this table to generate the ads and flyers page in sorted order and remove outdated flyers.

The `startdate` field is the date at which the flyer should start being displayed. It can be set to sometime in the past to immediately display a flyer.

The `sortdate` field is the date used to order the flyers. For dances I set this to the date of the dance. For flyers with multiple dances I set this to the first dance on the flyer.

The `enddate` field is the date when the flyer should no longer be displayed. For dances I set this to the day after the dance. Note that the date is GMT so it will be the next day 4 or 5 hours before midnight.

The `name` field is the name of the pdf and jpg of the flyer with the `.pdf` or `.jpg` suffix removed. The `.pdf` and `.jpg` files should be in the `flyers` directory.

5.3.10 Table club

This table lists all of the clubs and organizations that put on a dance.

The `name` field is the name of the club or organization.

The `url` field is a link to the club's web page if it has one.

The `gif` field is the name of a `.gif` file logo in the `gifs` directory for the club. This is the logo used in the club news.

5.3.11 Table clubnews

This table contains all of the club news for the promenader.

The `date` field is for the date of the issue.

The `clubid` field is for the clubid (from the club table) of the club.

The `news` field contains the actual text of the club news.

The `information` field is for where to contact for more information about the club.

The `reporters` field is for the reporters of this club news.

5.3.12 Table clubnewsproof

The format for this table is identical to that of the `clubnews` table except the result will show up in the proofreaders part of the website.

5.3.13 Table clubofficer

The clubid, position (like `president`), and `personid` of a club officer.

5.3.14 View cuers

This view is used by the schedule view.

5.3.15 View clubs

This view presents the club names associated with an event. Multiple clubs are separated by commas. Used by the schedule view.

5.3.16 Table content

This table holds fragments of web pages that are assembled by various scripts.

The `page` field indicates what web page is using this content.

The `part` field contains a string to allow several fragments on a single page.

The `content` field is intended to contain the actual web page content. You need to examine the actual SQL using this fragment to see how it is used.

5.3.17 Table dancechanges

This table tracks updates to the dance schedules. It is populated via triggers on the `event`, `eventcaller`, `eventclub`, and `eventpdf` tables. This table should be cleaned out as the updates contained in it are applied to other websites.

5.3.18 Table event

This table lists all of the square dance events.

The `date` field is the date of the event and must be in the format `YYYY-MM-DD` with leading zeros if necessary for weeks and days.

The `venue` field is the venueid of the venue of the dance.

The `time` field is text indicating the time of the dance (with other information about times).

The `flags` field is a text field containing the flags for the dance. A "O" (not in the word "DEMO") in the flags indicates an open house. A "*" indicates a class/club dance. These special flags are used to add dances to special lists on the website.

The `eventname` field is the name of the event if any.

The `comment` field is used for contact information for open houses.

5.3.19 Table eventcaller

This table associates a caller with an event. More than one caller can be associated with an event (even no callers).

5.3.20 Table eventclub

This table associates a club with an event. More than one club can be associated with an event (even no clubs).

5.3.21 Table eventcuer

This table associates a cuer with an event. More than one cuer can be associated with an event (even no cuers).

5.3.22 Table eventpdf

If an event has a flyer (or even more than one) that information is entered into this table. The event is the eventid and the flyer is the short name for the pdf file (x.pdf). Multiple flyers can be associated with an event.

5.3.23 Table eventurl

If an event has a web page (or even more than one) that information is entered into this table. The event is the eventid and the url is the complete url. Multiple urls can be associated with an event.

5.3.24 Table faq

This table contains all of the Frequently Asked Questions. The contents of this table are used to generate the FAQ page on the website.

The `rank` field indicates the order of the questions and answers.

The `question` field is the question.

The `answer` field is the associated answer to the question.

5.3.25 Table federationemail

This table is for email addresses of federation officer positions (not a person's email address). These addresses will (hopefully) remain static when officers change.

The `section` field refers to the section of the officer table (see the table `federationstaff`).

The `position` field refers to the position of the officer (see the table `federationstaff`).

The `email` field is the email address.

5.3.26 Table federationmeetings

This table contains all of the meeting places and times and meeting minutes of the federation. Meetings without minutes are listed as future meetings followed by the minutes of the meetings that have been published.

The `date` field is the date (past or future) of the meeting.

The `venue` field is the venueid (same as a dance) where the meeting will/did take place.

The `room` field is a text field for the room information.

The `time` field is the time of the meeting `-` a text field).

The `refreshments` field is a text field for who should bring refreshments.

The `minutes` field contains the minutes of meetings that have happened.

5.3.27 Table federationstaff

This table contains all of the federation officers. Currently there are two sections - officer and committee.

The `rank` field indicates the order for listing officers and committee members.

The `section` field (currently) is either “officer” or committee”.

The `position` field is the name of the position.

The `person` field contains the personid of the person with this position.

5.3.28 Table monthname

This is a utility table mapping month number to month name.

5.3.29 Table number

This table contains consecutive numbers used for generating the calendar month view.

5.3.30 View orphancallers

This table gives the eventid of events that have been deleted but still have a event-caller association in the `eventcaller` table. These entries in the `eventcaller` table can be deleted. There should probably be a trigger to do this automatically.

5.3.31 View orphanclubs

This table gives the eventid of events that have been deleted but still have a event-club association in the `eventclub` table. These entries in the `eventclub` table can be deleted. There should probably be a trigger to do this automatically.

5.3.32 View orphancuers

This table gives the eventid of events that have been deleted but still have a event-cuer association in the `eventcuer` table. These entries in the `eventcuer` table can be deleted. There should probably be a trigger to do this automatically.

5.3.33 View orphanspecial

This table gives the eventid of events that have been deleted but still are listed in the `specialdances` table. These entries in the `specialdances` table can be deleted. There should probably be a trigger to do this automatically.

5.3.34 View orphanpdfs

This table gives the eventid of events that have been deleted but still have a event-pdf association in the `eventpdf` table. These entries in the `eventpdf` table can be deleted. There should probably be a trigger to do this automatically.

5.3.35 Table pagesets

This table contains information for the website building automation process. A full explanation of this table is in the Automation Process section below.

The `pageset` field gives the pageset this row belongs to and matches the corresponding `pageset` field in the `substitutions` table.

The `position` field gives the order within a pageset.

The `pagepart` field matches the `pagepart` field in the `scripts` table.

5.3.36 View pagesetsview

Automatically generated editable view on the `pageset` table.

5.3.37 Table person

All references to people in the database are referenced by their personid and the actual information about the person is stored in this table. Updating information for a person in this table will spread updated information everywhere on the website.

The `personid` field is automatically generated and is unique.

The `firstname` field is the person's first name.

The `lastname` field is the person's last name.

The `address1` field is the street address (if available) of the person.

The `address2` field is the city, state part of the address (if available) of the person.

The `zip` field is the zipcode part of the address (if available) of the person.

The `phone` field is the phone number with area code (if available) of the person.

The `email` field is the email address (if available) of the person.

The `spouseid` field is the personid of the spouse (if available) of the person. This field is currently unused by any of the scripts.

The `gender` field is the gender (M or F) of the person. This field is currently unused by any of the scripts.

5.3.38 personview

Automatically generated editable view on the `person` table.

5.3.39 Table promenaderstaff

This table lists the people on the staff of the Promenader.
The `rank` field tells the order to display the staff members.
The `position` field is the name of the position.
The `person` field is a personid from the `personview` view¿

5.3.40 Table rafclubs

This table lists the clubids of clubs that are members of the Rochester Area Federation.

5.3.41 View schedule

This view assembles useful information about events from multiple tables and views. Some but not all columns have triggers to allow editing of the underlying table. Some of these triggers do nonobvious things so you need to read the details below.

The `date` field is the date of the event. This field is editable and will edit the date field of the event in the `event` table.

The `clubname` field is a list of clubs sponsoring the event. To add a club to this list replace the contents of this field with the clubid from the `club` table. To delete all clubs drag a "NULL" entry to this field.

The `venuename` field is the name of the venue of the event. To change a venue replace this field with the venueid from the `venue` table. This will update the venuename, venueadd1, venue2, and latlong fields from the `venue` table.

The `venueadd1` field is the address1 field from the `venue` table.

The `venueadd2` field is the address2 field from the `venue` table.

The `latlong` field is the latlong field from the `venue` table.

The `caller` field is a list of callers for this event. To add a caller to this list replace the contents of this field with the personid from the `personview` table. To delete all callers drag a "NULL" entry to this field.

The `cuer` field is a list of cuers for this event. To add a cuer to this list replace the contents of this field with the personid from the `personview` table. To delete all cuers drag a "NULL" entry to this field.

The `time` field is an editable text field indicating the time for this event. Editing this field will edit the underlying time field in the `event` table.

The `flags` field is an editable text field indicating the flags for this event. Editing this field will edit the underlying flags field in the `event` table.

The `eventname` field is an editable text field containing the name of this event. Editing this field will edit the underlying eventname field in the `event` table.

The `comment` field is an editable text field containing a comment about this event. It is used for contact information for open houses. Editing this field will edit the underlying comment field in the `event` table.

The `pdfs` field is not editable and is a list of html-formatted pdf references associated with this event. The information for this field comes from the `eventpdf` table which must be edited directly.

The `subpdfs` field is not editable and is a list of html-formatted pdf references associated with this event. The url assumes that this will be used one level down in the directory structure of the website. The information for this field comes from the `eventpdf` table which must be edited directly.

5.3.42 Table scripts

This table contains information for the website building automation process. A full explanation of this table is in the Automation Process section below.

The `pagepart` field is the pagepart selected by the pagepart field in the `pagesets` table.

The `position` field gives the relative ordering within a pagepart.

The `script` field contains the actual SQL command that is run during the automation process. All occurances of $\$n$ are replaced by the corresponding result of the select statement from the `substitutions` table.

5.3.43 View scriptsview

Automatically generated editable view on the `scripts` table.

5.3.44 Table singlepage

This table makes single pages on the website. The automation process surrounds the content with appropriate html to make a valid page.

The `location` field contains the name of the file of the web page. It should be at the top level of the directory hierarchy.

The `type` field is not currently used.

The `heading` field contains the headline for the page.

The `content` field contains the html page content. This field is copied to the web page.

5.3.45 Table sitemap

This table is automatically generated by the `makeweb.jar` program when the `-map` option is used. The previous table contents are deleted and replaced with the result of a tree-walk of the website.

This table is used to automatically generate the full site map set of pages and the list of scanned promenaders.

The `entryid` field is a unique identifier for this file/directory for this run of the program. It will change with successive runs of the program.

The `name` field is the base name of the file or directory

The `type` field is 0 for a file, 1 for a directory and -1 for something else.

The `path` field is a relative path (including the file/directory name) from the top of the hierarchy.

The `parentid` field is the entryid of the directory containing this entry. An entryid of 0 is used for the top of the hierarchy and there is no table entry with an entryid of 0.

The `length` field is the length of the file in bytes.

The `date` field is the last-modified time of the file/directory as milliseconds past the unixepoch in GMT. The automation process does not understand daylight savings time (yet).

5.3.46 Table specialdances

This table lists the eventids of special dances.

5.3.47 Table styles

This table is used to generate the cascading style sheet used on the website. The automation process collects information from this table and makes a correctly formatted file `style.css`.

The `selector` field sets the selector.

The `property` field sets the property.

The `value` field sets the value of the property.

5.3.48 View stylesview

Automatically generated editable view on the `styles` table.

5.3.49 Table substitutions

This table contains information for the website building automation process. A full explanation of this table is in the Automation Process section below. Each row specifies a group of similar pages to be made or specifies initialization code.

The `pageset` field specifies the pageset to be built.

The `substitutions` field contains an SQL select statement. The first column of the result set is the name of the file to be built and the following columns are strings used for replacing the $\$n$ in the script column of the `scripts` table. If the first column of the result set is NULL then no file is generated but the pageset code is still run, e.g., for initialization of temp tables.

5.3.50 View substitutionsview

Automatically generated editable view on the `substitutions` table.

5.3.51 Table venue

This table holds information about event locations locations.

The `venueid` field is a unique identifier for this venue.

The `latlong` field is the latitude and longitude of the venue in a format acceptable to google maps.

The `name` field is the name of the venue.
The `address1` field is the street address of the venue.
The `address2` field is the city and state of the venue.
The `zip` field is the zipcode of the venue.
The `shortname` field is not currently used.
The `url` field is the url of the venue.

5.3.52 Table vieworder

This table is used to create editable views on tables.

The `tablename` field is the name of the table on which the view is to be constructed.

The `tableorder` field is a list of fields specifying the desired sort order of the view.

5.3.53 Table views

This table is used to create editable views on tables. It specifies the columns in the view and the column order. The name of the view will be called [table-name]view and the triggers will be called [table-name[field-name] and [table-name]insert and [table-name]delete.

`tablename` field is the name of the table.
`position` field is the relative order of the columns for this table.
`column` field is the name of the column.

5.3.54 Table weekday

This is a utility table mapping day number to day name.

5.3.55 Trigger sch...

These triggers are used for updating the appropriate tables when editing the `schedule` view.

5.4 Overall File Structure

The Makefile makes the Website.pdf and associated files and runs the makeall bash script. The makeall script sets the dates to make calendars and the current month and then runs scripts for each type of web page. The calendar pages are divided into day, week, and month views and a script is run for each month and view combination.

The last thing that makeall does is copy files whose contents differ from the update directory or don't exist in the upload directory.

The current directory structure of the website is:

./articles	News articles and federation president messages
./awards	Awards-related pages

./cgi-bin	I don't use this yet except for two experimental scripts
./clubs	Club web pages for clubs without their own website
./compressed	Compressed `.pdf`s of club news and articles
./flyers	`pdf` flyers and their `jpg` thumbnails
./fullsitemap	pages for the experimental full site map
./gifs	logos for the clubs used in club news
./history	scanned images of the paper promenader are in the subdirectory `history/scanned`
./jpg	various `jpg` files used by various web pages
./photo	remnant of old website - could be deleted
./plesk-stat	no idea - it is empty
./png	`jpg` files used in articles
./sandbox	proofreading and experimental area
./schedule	all of the dance schedule pages
./thumbnails	left over from the conversion process - should be deleted
./tmp	random stuff in process for making a new edition of the website
./tools	some useful information for setting up a cygwin environment

6 The Automation Process

The new automation process keeps more data in the database. In particular, all scripts, club news, article texts, and more, are now in the database. Only flyers and club websites and pictures still reside in files. This should allow for a more flexible operation. By editing the contents of the database, the number and kind of generated pages can be easily changed. All of the code that accesses the database and writes the files is written in Java and uses libraries that work on windows, Apple, and linux platforms. Maintenance of the website can be easily done on any of these platforms. Only Apple lacks the ability to run `WinSCP` for file transfer although ftp can be used. (A future version of this software may include an ftp-like client written in Java.)

There are three special database tables used by the automation process: `substitutions`, `pagesets`, and `scripts`.

6.1 The substitutions Table

Table `substitutions` columns
`pageset` - the set of pages controlled by this row
`substitutions` - text substitutions performed on pageset scripts
`comment` - a human readable comment

The `pageset` column is a tag denoting the type of page(s) generated in this page set. The name is arbitrary but will be matched in the `pagesets` table. The page sets will be executed in collating order so some scripts can be made to run before other scripts. Multiple rows with the same `pageset` will work but are probably not useful.

The `substitutions` column contains an sql statement that returns a table whose first column is the file name of the file to be created and the remaining columns are strings to be substituted into the page generation scripts. If the first field is “null” then the scripts will still be run but no file is written. This is useful for scripts that set up temporary tables for use by the following scripts. The substitutions are performed by looking for `$0` through `$9` in the page generating script and replacing these two characters by the contents of the corresponding column 0 through 9 from the returned table. (Note that `$0` will be replaced by the file name field.) The sequence `$$` will be replaced by a single `$`.

The `comment` column contains a human readable comment explaining the purpose of this row and how the substitutions are intended to be used.

Recommended substitutions are:

Table `pagesets` columns
`$0` - the filename
`$1` - the current date (for top-level menus referring to schedules)
`$2` - the distance to the home directory, e.g., `../` or empty.
`$3` - the date before this calendar page
`$4` - the date of this calendar page
`$5` - the date after this calendar page

6.2 The pagesets Table

Table `pagesets` columns
`pageset` - the pageset we are creating
`position` - the collating order of this field determines the execution order
`pagepart` - the name of the pagepart scripts to be run to calculate insertions
`comment` - a human readable comment

The `pageset` field labels the page group of this line and is used to link with the corresponding row(s) of the `substitutions` table.

The `position` field contains a string indicating the ordering of pageset rows belonging to the same page set. Rows will be ordered by the collating order of the database.

The `pagepart` field references the rows in the `scripts` table that contain scripts that generate content for the piece. A generated page results from concatenating all if the pieces in order.

The `comment` column contains a human readable comment explaining the purpose of this row and how the substitutions are intended to be used.

6.3 The scripts Table

Table `scripts` columns

- `pagepart` - the page part this script belongs to
- `position` - the collating order of this field determines the order
- `script` - an sql statement generating the content of part of a page
- `comment` - a human readable comment

The `pagepart` field labels the page group of this line and is used to link with the corresponding row(s) of the `pagesets` table.

The `position` field is used to order multiple lines in the same page part. Rows will be ordered by the collating order of the database.

The `script` field contains a database sql statement that generates a part of a page. The substitutions from the `substitutions` table are applied before executing the statement. For generating similar pages the same script is used with different substitutions. All of the table output generated by running this script (generally a select statement) is written to the current file. All of the fields of each table output are concatenated and each resulting string is written to the current file followed by a new-line.

The `comment` column contains a human readable comment explaining the purpose of the script in this row.

6.4 The Algorithm

```
for each substitution row in table substitutions
    run substitutions sql and get table of substitutions
    for each substitution row
        get and open file (if not null)
        get matching pagegroup rows in order from table scripts
            for each script (after substitutions) get table
                for each row of table
                    concatenate fields and write as line to file
        close file (if not null)
```

If the script does not result in a table, i.e., a `select` statement, or the file name is `null`, the script is still run for side effects but no data is written.

The recommended way of using these tables is to group similar pages to be generated in a pageset. Each page of the pageset will be generated by an identical set of scripts but writing a different file and with different substitutions. These alternatives are generated by the select statement contained in the substitutions column of the `substitutions` table. For example, all of the month schedules can be generated by substituting the month to be generated and the filename for the month.

6.5 The Code

```
// Get and process pagegroups from master table substitutions
Table subs = new Table("select * from substitutions order by pagegroup",
                       "pagegroup", "substitutions");
ArrayList<String> subsLine;
while((subsLine = subs.next()) != null) {
  //System.out.println("subsLine: " + subsLine);
  String pagegroup = subsLine.get(0);
  // run substitution sql to get list of substitutions for pagegroup
  Table subsList = new Table(subsLine.get(1));
  ArrayList<String> subsListLine; // actual substitutions vector
  while((subsListLine = subsList.next()) != null) {
    //System.out.println("subsListLine: " + subsListLine);
    String filename = subsListLine.get(0);
    FileWriter out = null;
    if(filename != null) {
      out = new FileWriter(filename);
    }
    // get scripts for pagegroup
    Table scripts
      = new Table("select script from scripts where pagegroup = '"
                  + pagegroup + "' order by piece",
                  "script");
    ArrayList<String> scriptsLine;
    while((scriptsLine = scripts.next()) != null) {
      // run substituted sql to get tables to write out
      if(filename != null) {
        Table page =
          new Table(substitute(scriptsLine.get(0), subsListLine));
        ArrayList<String> pageLine;
        while((pageLine = page.next()) != null) {
          // write scriptsLine concatenated
          StringBuilder sb = new StringBuilder();
          for(int i = 0; i < pageLine.size(); i++) {
            sb.append(pageLine.get(i));
          }
          sb.append('\n');
```

```
                out.write(sb.toString());
            }
        } else {
            String s = substitute(scriptsLine.get(0), subsListLine);
            Statement stat = conn.createStatement();
            stat.execute(s);
            stat.close();
        }
    }
    if(out != null) {
        out.close();
    }
  }
}
```

7 New Promenader Website

7.1 New Automation Process

7.2 Current Promenader

Front cover - breaking news and special dances
list of club presidents etc.
classified ads
promenader info - deadlines - promenader officers
Ads and ad info
federation officers
President's corner article
News with pictures - Daphne-Norma news - Letter from Halsted
Club news - logos - pictures
Flyers in order with ads - both local and nonlocal dances
Club schedules with flags - symbol key
Inside cover - callers and cuers - both CCR and non-CCR
Back cover - index - other notes

7.3 Top-Level Pages

7.3.1 Home Page

Listing of breaking news and special dances
Listing of class/club dances
Miscellaneous links

7.3.2 Clubs

schedule with month layout
schedule with week layout
schedule with day layout
schedule page for each club
Club news
roving reporter
federation news

7.3.3 Wanted Pages

history of caller / cuer schedules?
history of club schedules?

8 Tools For Maintaining the Federation Website

This section explains how the tools used to maintain the website can be modified and rebuilt. If you are just maintaining the website you do not need anything in this section.

The tools need a `Java` and `latex` environment to run. I compile all Java programs with Java8. You can run the tools in unix, linux or windows (or the mac — I haven't tried this but it should work). For maintaining the documentation on the website I use latex in a linux texlive environment. Using latex in a linux environment also works.

If you want to directly access the database from the command line you can use the sqlite3 shell program or you can use a command window in the database editor. For most uses the command window is quite adequate. If you need to use the sqlite3 shell program it must be recompiled for each environment. The sources for this program are on the website. You need to recompile the shell.c and sqlite3.c programs for each different environment. The `Makefile` has some comments that might be useful for compiling from the sources.

8.1 make

The `make` program is used to automatically remake the documentation and the Java programs. A version of `make` needs to be in the path if you want to remake the documentation or any of the `.jar` files. The makefile is `Makefile`. It makes the documentation and all of the `.jar` files.

You only need `make` if you want to remake the documentation files or recompile the `.jar` files.

The Java Database Editor can be compiled by executing `make dbe.jar`. The build process deletes all old `.class` files, compiles `DatabaseEditor.java`, uses `manifest.txt` as the manifest, and creates the `dbe.jar` file with the newly created `.class` files and `DatabaseEditor.java` file, and then deletes the `.class` files. All of this is specified in the `Makefile`. To experiment with another sqlite3 version just edit the `manifest.txt` file and do a `make`.

8.2 latex and Friends

The documentation is generated using latex. The source for the website documentation is in `Website.tex`. The source for the database editor documentation is in `DBEditor.tex`. The source for the file browser documentation is in `Filebrowser.tex`. If you are not remaking the documentation you do not need latex.

8.3 Rebuilding the Documentation

The documentationfor the website is contained in `.dvi`, `.ps`, and `.pdf` files. To rebuild either the `Website` or the `DBEditor` documentation files you edit the corresponding `.tex` files and then run `make`. This will generate the `.dvi`, `.ps`, and `.pdf` files (as well as some other files used by `latex`). You must have `latex` installed for this to work.

8.4 Rebuilding the Tools

There are five `java` programs used for maintaining the web. They are:

Clean.java	A program that detects “funny” characters in a text file
DatabaseEditor.java	An editor for a `SQLite` database
MakeWeb.java	Creates the entire website from the database
Copyfiles.java	Copies changed files from one directory to another
FileBrowser.java	A program for inspecting and moving files

To change or augment the behavior of any of these programs, edit the `.java` and run `make`. The “executables” made from these source code files are:

Clean.java	`Clean.class` (run with `java Clean files ...`)
DatabaseEditor.java	`dbe.jar`
MakeWeb.java	`makeweb.jar`
Copyfiles.java	`copyfiles.jar`
FileBrowser.java	`filebrowser.jar`

The database editor and the makeweb programs require a `sqlite-jdbc .jar` file to access the database. The way this works is that in the manifest of the `.jar` files `dbe.jar` and `makeweb.jar` have a class-path entry pointing to the proper `sqlite-jdbc .jar` file. This is set from the file `manifest.txt` and is currently set to `Class-Path: sqlite-jdbc-3.27.2.jar`. To change the `sqlite-jdbc` version used edit the `manifest.txt` file.

The file browser program requires a `myjsch.jar` file to access remote files. The way this works is that in the manifest of `filebrowser.jar` there is a class-path entry pointing to the `myjsch.jar` file. This is set from the file `jschmanifest.txt`. This is an edited version of `jsch` located in the `jsch-0.1.54` directory. It's `Makefile` is at `jsch-0.1.54/src/main/java`. To change the `jsch` version used, download the new jsch version and edit the appropriate files. Also edit the `Makefile` to copy the appropriate `.jar` file. WARNING: I have added several methods and made other corrections to the released version of `jsch` so you will have to track down these changes.

A Setting Up The Tools

A.1 Java

The tools will work with any of the Java versions Java8 or higher. I compile all Java programs with Java8. The `sqlite-jdbc-3.27.2.jar` file is needed to supply the interface between Java and sqlite3. Included in the .jar file are native libraries that speed up the interface. The supplied `Makefile` has a target for `dbe.jar` that will compile `DatabaseEditor.java` and add these files into a copy of the sqlite jar file with the correct manifest. The file `manifest.txt` contains the manifest that contains the class path of the sqlite `.jar` file.

A.2 SQLite3

`SQLite3` is a free relational database system used in many applications. All of the tables of an sqlite database are stored in a single file so backing up the database is as simple as copying this file. The code for accessing the database is contained in a `sqlite-jdbc-xxx.jar` file which must be in the same directory as the Java code accessing the database. The `dbe.jar` and `makeweb.jar` files contain a class path pointer to this file. I am using version `sqlite-jdbc-3.27.2.jar` although most versions are compatible with each other. Versions of the `Java jdbc` can be downloaded from the maven repository at `mvnrepository.com/artifact/org.xerial/sqlite-jdbc`.

B Overview of the Internet and Websites

(This section is for a fuller understanding of how websites work. It is not necessary for day-to-day maintaining of the website.)

The world-wide-web is a collection of web pages hosted on web servers spread around the world. A web browser given the name of a web page, e.g., `squaredancingrochester.org`, first looks up the name using a domain name server to get an IP address, e.g., 69.195.124.252, and sends a GET request to this address. This request contains the name of the page desired and the host name. The web server then sends the requested page to the web browser (with additional information).

In order for this to work the domain name server needs to have a correct map from names to IP addresses and the web server needs to be able to process requests and deliver requested web pages, images etc.

For a yearly fee a domain registrar will add a mapping from a specified domain name to an IP address. Currently, the federation is paying for four domain registrations: `squaredancingrochester.org`, `ssdusa.org`, `callermike.com`, and `cloverleafsquares.org`. They all point to the same IP address at bluehost. Mike Callahan, Cloverleaf Squares, and SSDUSA then reimburse the federation for the cost of his domain name registration. SSDUSA also has dar2018.ssdusa, dar2019.ssdusa, dar2020.ssdusa, and dar2021.ssdusa as subdomains for which the cost is free.

The web pages for the federation and all add-on domains are currently hosted by bluehost. We have "unlimited" space for web pages. Bluehost says that "unlimited" means less than 50,000 inodes (or files) and less than 50 GBytes of storage but they only get excited at around 200,000 inodes. We are currently using about 8,000 inodes and are using about 4.5 GBytes.

If, for some reason we want to change our website host, all we have to do is transfer the website files to the new host and make the domain name servers point to the IP address of the new host. This is conceptually easy but requires several bureaucratic steps (proving we own the domain, unlocking the domain, etc.). The subdomains, unless they want to be transferred too, will have to make other arrangements.

B.1 Domains on the Federation Website

The following table lists the domains hosted on the federation website and where the content for these domains is located. Each domain appears, from the outside, as a totally independent website.

squaredancingrochester.org	`public_html/`
callermike.com	`public_html/callers/MikeCallahan/`
cloverleafsquares.org	`public_html/clubs/CloverleafSquares/`
ssdusa.org	`public_html/SSDUSA/`
dar2018.ssdusa.org	`public_html/SSDUSA/dar2018/`
dar2019.ssdusa.org	`public_html/SSDUSA/dar2019/`

dar2020.ssdusa.org	`public_html/SSDUSA/dar2020/`
dar2020.ssdusa.org	`public_html/SSDUSA/dar2021/`

The first domain `squaredancingrochester.org` is the master domain for the website and contains the content for all domains. The locations of the content for the other domains is set in bluehost's cpanel accessed by a user with the master password.

Only the maintenance of the `squaredancingrochester.org` website is described here as the other websites are administered by someone else.

B.2 Domain and Website Fees

Currently `sidneym@frontiernet.net` is listed as the domain owner for all domains on the website. All domains are renewed using the credit card of Sidney Marshall. This means that no one else will be notified if the renewal process fails for some reason and the domain names will then go into redemption and someone else could acquire these domain names. This is a Federation issue that I currently don't know how to resolve. The website itself has the same problem.

It would be nice if the Federation had an email address, not associated with the website, for these notifications and a Federation credit card.