

# Running libraries on PostgreSQL

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

---

## Contents

<b>1</b>	<b>License</b>	<b>1</b>
<b>2</b>	<b>Evergreen library system</b>	<b>1</b>
<b>3</b>	<b>Who is Dan Scott?</b>	<b>1</b>
<b>4</b>	<b>Evergreen library adoption (2011)</b>	<b>2</b>
<b>5</b>	<b>GPLS Pines</b>	<b>4</b>
<b>6</b>	<b>BC Sitka</b>	<b>5</b>
<b>7</b>	<b>King County Library System</b>	<b>6</b>
<b>8</b>	<b>Project Conifer</b>	<b>7</b>
<b>9</b>	<b>Library CONSTRAINTs</b>	<b>7</b>
<b>10</b>	<b>It's not all bad</b>	<b>8</b>
<b>11</b>	<b>Horrible, horrible library data</b>	<b>8</b>
<b>12</b>	<b>Mike Rylander, Evergreen's eeevil database genius</b>	<b>9</b>
<b>13</b>	<b>Indexing library data the Evergreen way</b>	<b>10</b>
<b>14</b>	<b>Random access by field-subfield</b>	<b>10</b>
<b>15</b>	<b>Indexing title / author / subject / keyword</b>	<b>11</b>
<b>16</b>	<b>Adventures in text search: Evergreen 1.0</b>	<b>11</b>
<b>17</b>	<b>Adventures in text search: Evergreen 1.6</b>	<b>11</b>
<b>18</b>	<b>Adventures in text search: Evergreen 2.0</b>	<b>12</b>
<b>19</b>	<b>Adventures in text search: Evergreen 2.2</b>	<b>12</b>
<b>20</b>	<b>Bad news for text search</b>	<b>12</b>
<b>21</b>	<b>Outsource to Solr?</b>	<b>13</b>
<b>22</b>	<b>Functions / stored procedures</b>	<b>13</b>
<b>23</b>	<b>Active tables</b>	<b>13</b>

---

---

<b>24 Debian/Ubuntu packaging</b>	<b>14</b>
<b>25 Materialized views</b>	<b>14</b>
<b>26 Hstore</b>	<b>14</b>
<b>27 Connection pooling</b>	<b>14</b>
<b>28 Replication</b>	<b>15</b>
<b>29 Inheritance</b>	<b>15</b>
<b>30 Schema evolution</b>	<b>15</b>
<b>31 Upgrading PostgreSQL</b>	<b>15</b>
<b>32 Kudos to PostgreSQL</b>	<b>16</b>
<b>33 Help us with our mission</b>	<b>16</b>

---

## 1 License

This talk is licensed under a [Creative Commons, Attribution, Share Alike license](#).



Available from [http://bzc.coffeecode.net/2012/PGCon/PostgreSQL\\_in\\_Libraries.html](http://bzc.coffeecode.net/2012/PGCon/PostgreSQL_in_Libraries.html) and [horrible PDF](#)

Many of the generalizations contained in this presentation are based on a methodologically flawed, self-selecting survey of Evergreen library system administrators. Others simply reflect the author's own biases.

## 2 Evergreen library system

### MISSION STATEMENT

Evergreen: highly-scalable software for libraries that helps library patrons find library materials, and helps libraries manage, catalog, and circulate those materials, no matter how large or complex the libraries.

Open-source (GPL2+): <http://evergreen-ils.org>

If "Libraries are the beating heart of a (community|university)", PostgreSQL is in turn at the heart of libraries that run Evergreen.

- We go a bit beyond the canonical relational example of a library database
  - Current install creates 355 tables, 96 views, > 50 functions in 23 different schemas
  - Handles hold requests, reservations, purchases and fund management, reporting, library information, staff permissions, and more...

## 3 Who is Dan Scott?

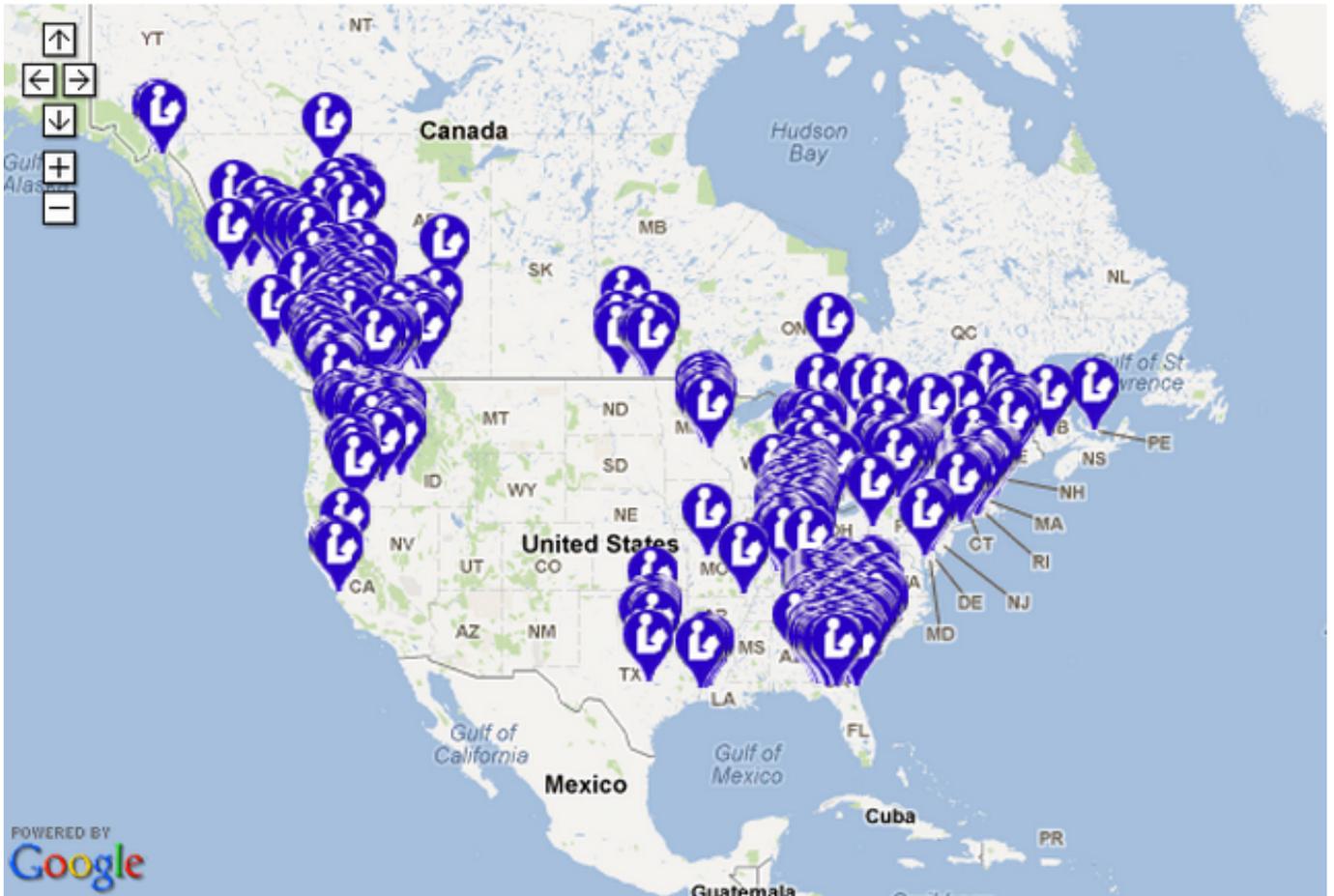
Systems Librarian at the J.N. Desmarais Library, [Laurentian University](#) in Sudbury, Ontario (a founding member of Project Conifer)

- Employed by IBM Canada from 1998-2006 in various positions including technical writer, support, development, and product planner
- All for **DB2 for Linux, UNIX, and Windows** - with a focus on Linux and open source



- Co-author of *Apache Derby: Off to the Races*
- Core Evergreen developer since 2007
- Still feel like a PostgreSQL n00b

#### 4 Evergreen library adoption (2011)





## 5 GPLS Pines

The screenshot shows the Georgia Public Library Service (GPLS) website for Pines. At the top left is the GPLS logo. To its right is a search bar with a dropdown menu for 'Staff by Department' and a 'Search GPLS' input field. Below the logo is a navigation bar with links for 'Find a Public Library', 'About Us', and 'DKB password access'. The main content area is titled 'PINES' and includes a 'PINES Online Library Catalog' section with a search prompt. A sidebar on the left lists resources for Libraries, Public, Kids, and Trustees. A right sidebar titled 'GEORGIANS & PINES' provides statistics on library card usage and borrowing. A footer section mentions a 'PINES 2012 ANNUAL PATRON SATISFACTION SURVEY'.

<http://www.georgialibraries.org/public/pines.php>

- The birthplace of Evergreen (started 2004, 1.0 in 2006)
- 275 libraries on a single system in the state of Georgia
- 2.6 million patrons
- 9.6 million items
- 18.6 million transactions / year

## 6 BC Sitka

The screenshot shows the Sitka website homepage. At the top left is the Sitka logo, which consists of a stylized tree made of green and blue blocks. To the right of the logo is a search bar with a magnifying glass icon and the word 'Search'. Below the logo and search bar is a horizontal navigation menu with the following items: HOME, IMPLEMENTATIONS, DOCUMENTATION, WORKING GROUPS & POLICY, NEWS, CONTACTS, SUPPORT, and STATUS. Below the navigation menu is a breadcrumb trail that reads 'You are here: Home'.

The main content area is titled 'Welcome to Sitka' and contains the following text:

Sitka is a group of libraries in the Provinces of BC and Manitoba, Canada, using the award-winning open-source [Evergreen software](#) as its shared Integrated Library System (ILS). Sitka is governed by the non-profit, member-driven [BC Libraries Cooperative](#), supported by a dedicated team and funded by the member libraries in partnership with the [Province of British Columbia](#).

This initiative is truly a collaborative effort, creating tremendous economies of scale, opportunities for shared growth and increasing access to library resources.

Here are some [links](#) and a [listserv](#) that help facilitate discussion around Evergreen within BC. As well, a set of [Evergreen Talking Points](#) strives to inform library boards and staff about the initiative.

Below the main text is a small graphic of the Sitka logo.

On the left side of the page is a 'NAVIGATION' sidebar with the following items: Home, Implementations, Documentation, Working Groups & Policy, News, Contacts, Support, and Status. Below this is a 'QUICK LINKS' section.

On the right side of the page is a 'NEWS' sidebar with the following items:

- Nicola Valley Institute of Technology joins Sitka! (2012-05-01)
- Evergreen 2013 Conference website & registration launched (2012-04-28)
- Manitoba PLSB joins Sitka! (2012-03-07)
- Kimberley Public Library Joins Sitka! (2012-02-07)
- 100% of North Coast on Sitka!

<http://sitka.bclibraries.ca/>

- 60 libraries on a single system in British Columbia

## 7 King County Library System

The screenshot shows the King County Library System (KCLS) website. At the top left is the KCLS logo. To its right is the tagline "Turn to us. The choices will surprise you." Below this are navigation links for "BOOKS", "MUSIC", "MOVIES", "MAGAZINES", "DOWNLOADS", "DATABASES", "WEB SITES", and "CONNECT". On the right side, there is a "LOG IN TO YOUR ACCOUNT" button with a link for "Get a library card" if the user doesn't have one. Below that is a search bar with the text "FIND" and "Items in the Catalog" in a dropdown menu, followed by "Search the catalog for books and more" and a "GO" button. At the bottom of the search bar are links for "Catalog", "Ask a Librarian", and "Mobile Catalog".

The main content area is divided into several sections:

- Attention Teens!**: A blue box with the text "Read, Film, Win 2012 has begun, enter your videos today!" and a "video book review contest for Teens" graphic.
- USING THE LIBRARY**: A section with a list of services including "Log In to Your Account, Locations & Hours, Meeting Rooms, Saving to the 'Cloud', Library Cards, Borrowing Items, Library Computers, Español, Library Elf, Emergency Closures".
- BOOKS & READING**: A section with links to "Book Talk - Reader's Blog, New This Month, Books We Like, Books of the Week, Ready to Read (Early Literacy)".
- RESEARCH & HOMEWORK**: A section with links to "Ask a Librarian, Job & Career Info, AnswerFile, Homework Help, Study Zone, Accelerated Reader, MSP/HSPE/SAT Help, InfoBiz".
- PROGRAMS & CLASSES**: A section with links to "Search for Story Times, ESL Classes, Computer Classes, STARS Workshops, Writing Workshops, Museum Passes, Tax Assistance, Español".
- EVENTS**: A section with links to "Meet the Authors, Romance Extravaganza, Book Sales, Making Sense of the American Civil War, Home Improvement, Opera Preview, Playing with Words, DIA (Children's Day/Book Day)".
- ABOUT KCLS**: A section with links to "Board of Trustees, Library Foundation, Volunteer, Bond Projects, Future Services Strategy".

At the bottom of the main content area, there is a "FIND A LIBRARY:" section with a dropdown menu labeled "Choose a library..." and a "GO" button. Below this is a "THIS WEEK AT KCLS" section and a "PAGES FOR..." section. In the bottom right corner, there are "Tools" and "AA" icons.

<http://kcls.org>

- Library system surrounding Seattle, Washington
- 1.2 million patrons
- 3.3 million items
- 19 million transactions / year

## 8 Project Conifer



# A Shared Instance of the Evergreen ILS in Ontario

### Conifer News

Conifer personnel win OCUL Outstanding Contribution Award, cited for work in "revisiting, infrastructure development, data migration, and production support for a collaborative open source ILS" at the OLA Super Conference 2011.

Dan speaks! Check out Dan Scott's presentation on Authorities in Evergreen 2.0, given at the EG2011 conference.

Project Conifer turned two years old on May 4, 2011!

The Niagara Evergreen Consortium is highlighted in Niagara Falls Review.

Northern Life and the Sudbury Star recognize Laurentian's Conifer achievements

Project Conifer wins **two** Awards

### Welcome to the Project Conifer Site

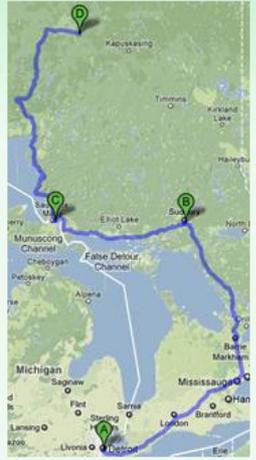
In September 2006, the library world changed dramatically when the most ambitious and sustained open source ILS initiative ever undertaken was unveiled with the deployment of Evergreen, a multi-year project of the PINES (Public Information Network for Electronic Services) Consortium, representing over 250 libraries in Georgia with the backing of full-time developers. Evergreen was constructed from start to finish as an open source application, and one that needed to scale to a very high level of processing load. One in five residents of Georgia are serviced by a PINES library, and it represents one of the busiest library systems in North America.

Two months after Evergreen went into production, the University of Windsor hosted a one day symposium on the state of the ILS and invited representatives from PINES to present their experiences from adopting an OSS solution. There has been long standing interest in OSS solutions at the University of Windsor, and the Leddy Library at Windsor had been a participant in an international gathering of software developers, information access advocates and library representatives at the launching of the eFL FOSS program in the Italian province of Ancona just one month after Evergreen made its debut.

Discussions among several Ontario post-secondary institutions led to a partnership of interested organizations, and on May 4, 2009, Conifer went live.

Why did we chose Evergreen? We were in good company (shout out to our Canadian brethren at SITKA, Innisfil Public Library, Mohawk College, Natural Resources Canada and the University of Prince Edward Island). Evergreen has the most agile protocol imaginable (OpenSRF) at its core, within our Consortium alone we have been able to wire

### Conifer Locations



The Conifer Consortium stretches almost **1600** kilometers between participating organizations.

### Inside Conifer

<http://projectconifer.ca>

- 38 libraries spanning Ontario - a mix of academic and special libraries
- 2.5 million items

## 9 Library CONSTRAINTs

Libraries are generally resource-challenged and their systems people are asked to be responsible for many software and hardware systems, not just the library system. Thus:

- Many Evergreen system administrators have *just enough* skill to get the system up and keep it running
- Despite the critical role it plays in system performance, PostgreSQL is often learned on a need-to-know basis in production
  - "All-in-one" underprovisioned server
  - Logs and data on same partition
  - Limited tuning; `pg_tune` or `bust`
  - Default statistics target at 50
  - Backups via `pg_dump` or incremental file system backups

## 10 It's not all bad

- Many sites rely on a third party company for setup and support, although too much dependency is always a concern
- Several Evergreen system administrators at PGcon this year; collectively, we will be stronger (and perhaps develop a set of Evergreen-specific best practices)
- Our development practices are maturing:
  - Code reviews are mandatory before committing to master
  - We have (some) standard sample data, unit tests, and a CI server
  - We have more documentation and broader communication
- Opportunities for consulting and training for PostgreSQL experts; help us make Evergreen a success throughout the world, and earn a living do it :)

## 11 Horrible, horrible library data

Central element of most library data is the MARC record, a combination of fixed-length fields and variable-length fields that encodes the bibliographic description of an object.

```
LDR 00969cam a22002774a 4500
001 14338589
005 20070508144242.0
008 060412s2005 cc 001 0 eng c
010 #a 2006273753
020 #a9780596007591 (pbk.)
020 #a0596007590 (pbk.)
050 0 0 #aQA76.76.D47 #bF634 2005
082 0 0 #a005.1 #222
100 1 #aFogel, Karl.
245 1 0 #aProducing open source software :
#bhow to run a successful free software project / #cKarl Fogel.
250 #a1st ed.
260 #aBeijing ; #aSebastopol, CA : #bO'Reilly, #c2005.
300 #axx, 279 p. ; #c24 cm.
500 #aIncludes index.
650 0 #aComputer software #xDevelopment.
650 0 #aOpen source software.
```

## 12 Mike Rylander, Evergreen's eevil database genius



Figure 1: Mike Rylander was sent from the future to defend the open source library system world from the tyranny of MARC

## 13 Indexing library data the Evergreen way

Generally, start with MARC (serialized as MARCXML) in `biblio.record_entry.marc`:

```
<record xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<leader>00969cam a22002774a 4500</leader>
<controlfield tag="001">14338589</controlfield>
<controlfield tag="005">20070508144242.0</controlfield>
<controlfield tag="008">060412s2005    cc            001 0 eng c</controlfield>
<datafield tag="010" ind1=" " ind2=" ">
  <subfield code="a"> 2006273753</subfield>
</datafield>
<datafield tag="020" ind1=" " ind2=" ">
  <subfield code="a">9780596007591 (pbk.)</subfield>
</datafield>
<datafield tag="082" ind1="0" ind2="0">
  <subfield code="a">005.1</subfield>
  <subfield code="2">22</subfield>
</datafield>
<datafield tag="100" ind1="1" ind2=" ">
  <subfield code="a">Fogel, Karl.</subfield>
</datafield>
<datafield tag="245" ind1="1" ind2="0">
  <subfield code="a">Producing open source software :</subfield>
  <subfield code="b">how to run a successful free software project /</subfield>
  <subfield code="c">Karl Fogel.</subfield>
</datafield>
</record>
```

- Yes, XML *does* make it better!

## 14 Random access by field-subfield

To support a MARC expert search, we populate `metabib.full_rec`:

1. source FK pointing to `biblio.record_entry.id`
2. value containing normalized text
3. `index_vector` index column with associated trigger

```
SELECT * FROM metabib.full_rec WHERE record = 884755 AND tag = '245';
-[ RECORD 1 ]+-----
id          | 22640054
record      | 884755
tag         | 245
ind1        | 1
ind2        | 0
subfield    | a
value       | producing open source software
index_vector | 'open':2 'produc':1 'softwar':4 'sourc':3
```

83M `metabib.full_rec` rows in Conifer's production database

Challenge: some fields such as general notes are lengthy, blowing past the btree maximum.

Eventual solution: Create a `SUBSTR(value, 1, 1024)` expression index on `metabib.full_rec`, rename the table to `metabib.real_full_rec`, and create a view called `metabib.full_rec` on top of it.

## 15 Indexing title / author / subject / keyword

1. Transform MARCXML into more human-friendly, semantic XML (generally MODS)
2. Define index classes with weighted fields (class, field, XML transform, XPath, weight)
3. Extract corresponding chunks into `metabib.*_field_entry.value`
4. `index_vector` index column with associated trigger

```

-[ RECORD 1 ]+-----
id           | 4234610
source      | 884755
field       | 6
value       | Producing open source software
            | how to run a successful free
            | software project
index_vector | 'a':8 'free':10 'how':5 'open':2
            | 'produc':1 'project':12 'run':7
            | 'softwar':4,11 'sourc':3
            | 'success':9 'to':6

```

29M `metabib.*_field_entry` rows in Conifer's production database

## 16 Adventures in text search: Evergreen 1.0

Circa 2006, PostgreSQL 8.0/8.1

- Text search built on TSearch2 contrib module ca. PostgreSQL 8.0
  - *Thank you Oleg and Teodor!*
- All indexed values created externally via Perl scripts, then initially loaded via COPY
  - Good for parallelized bulk loading
  - Brittle due to potential for ID conflict
  - Terrible for consistency, as updates to indexed values were managed by the application (and thus often did not happen)

## 17 Adventures in text search: Evergreen 1.6

Circa 2009, PostgreSQL 8.3/8.4

- Integrated full text search in PostgreSQL!
  - *Thank you Oleg and Teodor!*
- Still using TSearch2 contrib for compatibility
- Revelations about LCCOLLATE and LCCTYPE:
  - Debian / Ubuntu created UTF8 clusters by default
  - Negative performance impact on search was obfuscated until a real set of data is loaded

---

## 18 Adventures in text search: Evergreen 2.0

Circa 2011, PostgreSQL 9.0

- Evolved to database functions (plperl, plpgsql, SQL) & triggers for indexing and updates, avoiding ID conflicts and improving consistency
  - Trigger applies a series of customizable normalizations, implemented as database functions, for each value for a given field before insertion into the `tsvector` column
  - Search against a given field applies the same normalizations to the incoming search term(s)
- New features for users:
  - Wildcard searches
  - Exposed the Boolean OR operator (joining NOT and AND)
    - \* Librarians rejoiced! Nobody else noticed :)
- Some sites adopting GIN indexes

## 19 Adventures in text search: Evergreen 2.2

Circa 2012, PostgreSQL 9.1

- Still installing TSearch2 contrib module (force of habit; not really required)

## 20 Bad news for text search

- Serialized `serial` operations seem to be a bottleneck for bulk loading and reingesting
  - `ORDER BY rank` with `ARRAY_AGG(DISTINCT source)` kills performance for large results: 600MB merge sort for 500K hits
    - Granular index design compounds problems for general searches, requiring `DISTINCT` & therefore disk-based sort due to outlandish memory demands
    - **Good news:** many nights of `EXPLAIN ANALYZE` later, committed a change yesterday that improves performance significantly (in at least one environment): `CTE` and avoidance of `ARRAY_AGG(DISTINCT source)`
  - Stemming - desired, used, but problematic for academics and their multilingual collections in our implementation
  - Stop words are not an option:
    - *or* is gold to a university that focuses on mining
    - *It* is a popular novel
    - *The The* is a band
-

## 21 Outsource to Solr?

Solr comes up as an option for sub-second results:

- Broader adoption throughout library development community
- Perceived as having more mature and diverse analyzers / tokenizers / token filters
- Several branches exist for synchronizing Evergreen contents with a Solr instance

However, convenience and consistency of having full-text search managed by PostgreSQL generally outweighs perceived advantages of Solr.

Still not fun explaining this advantage to users and staff when their overly general query simply times out.

## 22 Functions / stored procedures

- Integral to indexing and search
  - Custom functions sometimes required to overcome PostgreSQL limitations
  - LOWER() on Unicode strings insufficient; thus we use plperl to invoke lc()
- Similarly, increasingly embedding heavy lifting into the database
- Borrowing periods, fines, and other policies based on the complex matrix of borrower, item, and library attributes that libraries demand
- All custom routines written in SQL, plpgsql, or plperl
  - Recently started tweaking default attributes like COST, ROWS, and IMMUTABLE/STABLE/VOLATILE for performance purposes
  - GSoC student will be hunting bottlenecks that can be addressed via rewrites in SQL or C
  - Adoption of new native functions like STRING\_AGG() vs. ARRAY\_TO\_STRING(ARRAY\_AGG()) and rewriting connectby() as WITH RECURSIVE CTEs

## 23 Active tables

The bibliographic record table is one of the more active tables in our schema:

### biblio.record\_entry triggers

```
a_marcxml_is_well_formed BEFORE INSERT OR UPDATE
a_opac_vis_mat_view_tgr AFTER INSERT OR UPDATE
aaa_indexing_ingest_or_delete AFTER INSERT OR UPDATE
audit_biblio_record_entry_update_trigger AFTER DELETE OR UPDATE
b_maintain_901 BEFORE INSERT OR UPDATE
bbb_simple_rec_trigger AFTER INSERT OR DELETE OR UPDATE
c_maintain_control_numbers BEFORE INSERT OR UPDATE
fingerprint_tgr BEFORE INSERT OR UPDATE
```

---

## 24 Debian/Ubuntu packaging

- Most Evergreen sites rely on packages and don't have expertise
  - Therefore Martin Pitt's backports are a godsend
- But packaging decisions introduce well-known compatibility pain points as well
  - Conflicting approaches to starting / stopping clusters
  - Location of configuration files
  - Upgrade challenges (`pg_upgrade` vs `pg_upgradecluster`)

## 25 Materialized views

For reporting simplicity and increased performance, materialized views (AKA *materialized query tables*) rock

- We fake materialized views using triggers and rules—but occasionally get things subtly wrong
  - A mistake with `money.materialized_billable_xact_summary` was painful, because it led to patrons expecting refunds they weren't owed
- Would love to have `CREATE TABLE` or `CREATE VIEW` options for `REFRESH IMMEDIATE` and `REFRESH DEFERRED` that would do the work for us
- Also, would love a pony

## 26 Hstore

Currently using `hstore` effectively in two places:

- *Single-valued fields*
  - Bibliographic record attributes that can have only one instance per record (such as year of publication)
  - Even though there are already many of them, librarians seem to continually spawn new record attributes
- Function arguments: avoids torturous variations of the same function definition with different signatures
  - For example, specifying different levels of limits: `unapi.bre(..., 'acn=>5,acp=>10')`
- It works!

## 27 Connection pooling

Would like to implement connection pooling to reserve server resources for core database processes

- (Local anecdote): `pgpool-II` failed in production after a few hours with a hard lockup
    - Could be a packaging issue; didn't have time to dig further
    - Only one site is still running `pgpool` successfully
  - Plan to investigate `pgBouncer`
-

---

## 28 Replication

- Slony has been the go-to option for reporting replicas
  - Limitations on commands such as TRUNCATE have bitten us, as developers typically don't test in a Slony environment
- WAL archiving / log shipping has been the go-to option for backup and disaster recovery, but many moving parts and options were daunting
- Streaming replication is simple to set up and great for disaster recovery
  - However, in a naive implementation (ours at Conifer), many queries time out
  - Will be looking into `vacuum_defer_cleanup_age` and `hot_standby_feedback` thanks to Phillip Sorber's replication tutorial

## 29 Inheritance

- Used sparingly but effectively for modelling objects with similar behaviour
  - Things like copies of books (`asset.copy` is a parent of `serial.unit`)
  - Transactions that might have costs attached (`action.circulation` is a child of `money.billing`)
- Occasionally stab ourselves by forgetting triggers, unique / FK / PK constraints (or having to customize them to be more flexible)

## 30 Schema evolution

- Evergreen has no automated solution for creating point-to-point upgrades
  - Currently, we write serially incrementing upgrade scripts that get concatenated & munged at release time
- (9.2) Avoiding table rewrites when we add a column with a default value will be appreciated
- DISABLE TRIGGER ALL helps performance, when we remember and when appropriate

## 31 Upgrading PostgreSQL

Libraries are generally averse to frequent system change, for the usual business reasons (avoiding downtime, risk and retraining).

- One Evergreen upgrade per year is about right
  - Generally prefer to avoid upgrading distributions or major components (such as PostgreSQL) at the same time
    - **Thank you** for your generous support policies; many libraries will be jumping from 8.4 to 9.1 in the next six months
  - Definitely want to avoid downtime; with rise in electronic resources, libraries are 24x7 businesses
    - `pg_dump / pg_restore` cycle was a bit painful, even with parallel restore
    - `pg_upgrade` definitely helps; 148 minutes for a 90 GB database
      - \* Not yet integrated into Debian/Ubuntu packagers' `pg_upgradecluster`, which does a full dump/restore
-

## 32 Kudos to PostgreSQL

- PostgreSQL has never been responsible for Evergreen data loss
- PostgreSQL has never been a bottleneck for Evergreen operations, for the largest and busiest of Evergreen sites (outlier queries excepted)
- Thorough documentation: release notes, core docs, active community of bloggers
- Supportive, welcoming community (#postgresql, mailing lists)
- Continual improvement and evolution

## 33 Help us with our mission

### MISSION STATEMENT

Evergreen[:] highly-scalable software for libraries that helps library patrons find library materials, and helps libraries manage, catalog, and circulate those materials, no matter how large or complex the libraries.

- PostgreSQL is close to your heart, and it's at the heart of Evergreen
  - Help bring Evergreen (and PostgreSQL) to a library near you
  - Make our heart beat faster!

Evergreen: <http://evergreen-ils.org>

---