Will Postgres Live Forever?

BRUCE MOMJIAN

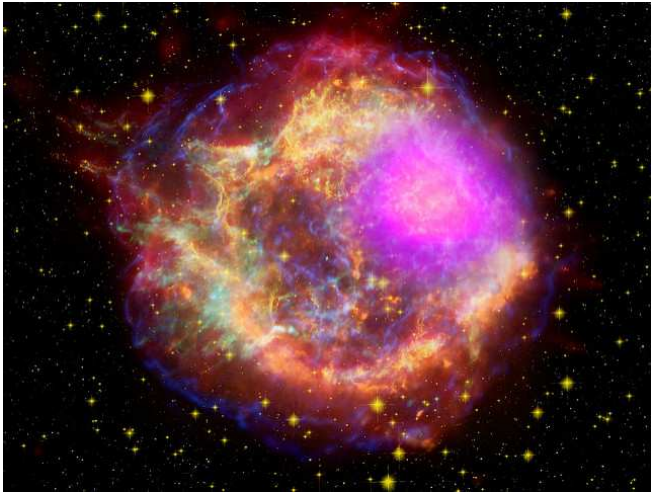
This presentation explains the long life of open source software, and the life cycle differences between proprietary and open source software. *Title concept from Renee Deger*

https://momjian.us/presentations

Last updated: March 2024

Outline

1. Forever



https://www.flickr.com/photos/gsfc/

Forever Is a Long Time

- Age of the Universe: 13.7 billion years
- Age of the Earth: 4.5 billion years
- Age of civilization: 6,000 years
- Civilized era vs. Earth years: 0.00001%
- Digital era vs. Earth years: $\sim$0%

Brief Digital History

1804: Jacquard loom
1945: ENIAC
1970: E. F. Codd Relational Theory
1974: System R
1977: Ingres
1986: University-based Postgres
1994: Postgres95
1996: Internet-based Postgres

2. Software Life Cycle

https://www.flickr.com/photos/tarynmarie/

Proprietary Software Life Cycle

1. Innovation
2. Market growth
3. Market saturation
4. *Maximize profit, minimize costs (development, support)*
5. Maintenance mode (no new features, no innovation)
6. End-of-life

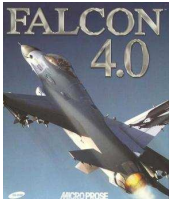
Open Source Software Life Cycle

1. Parity with proprietary software, low cost
2. Market growth
3. *Continue innovation or decline*
4. Source code is always available to continue

Illustrative Example of Open Source Growth

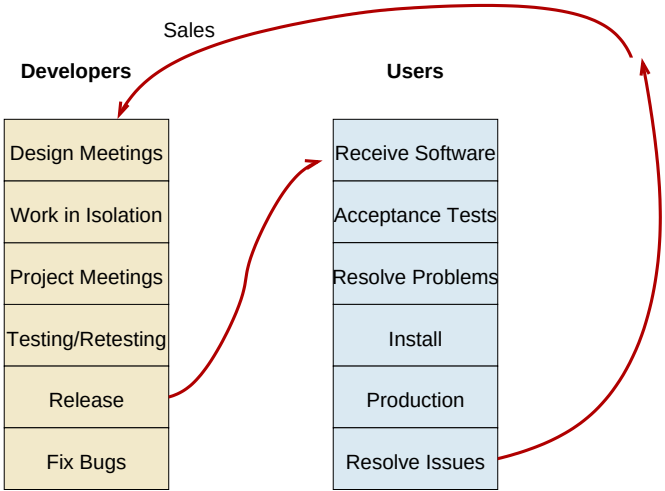
One of the longest-developed computer games:

- 1984: Spectrum HoloByte began Falcon development
- 1998: MicroProse released Falcon 4.0
- 1999: MicroProse ended development
- 2000: Falcon source code leaked
- 2003: Benchmark Sims (BMS) released community modifications
- 2005: Lead Pursuit released Allied Force, which included BMS mods
- 2015: GOG.com republished Falcon 4.0 (on Steam since 2016)
- 2021: BMS released version 4.35 U3, continued development

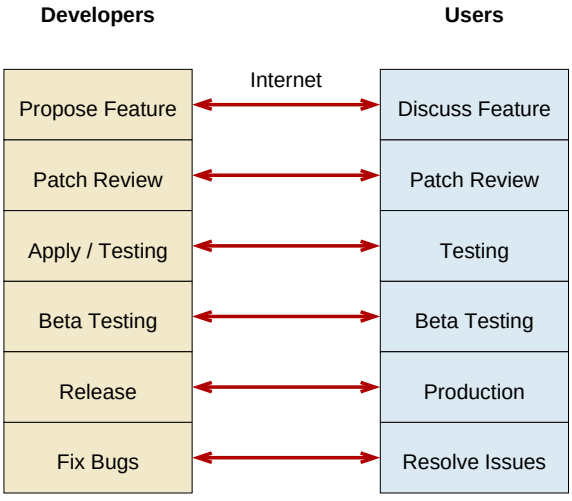


https://en.wikipedia.org/wiki/Falcon_4.0

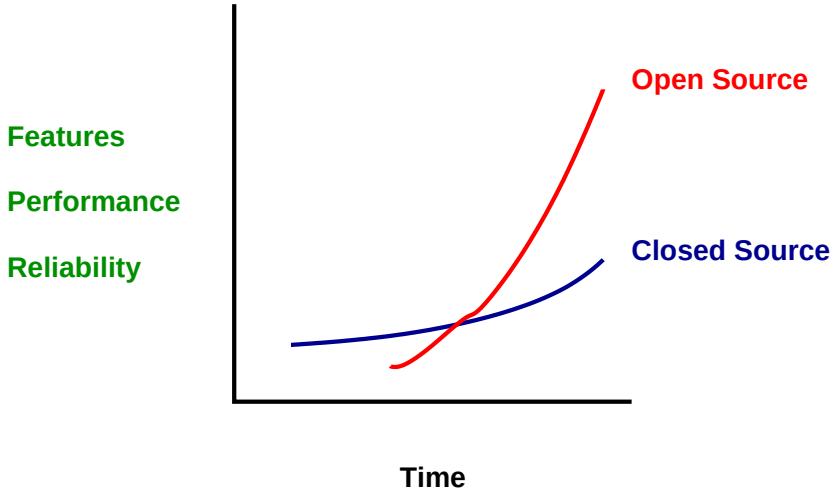
Proprietary Development Flow



Open Source Development Flow



Rise of Open Source



Linux

Linux attained feature parity with:

- HP-UX
- AIX
- Solaris

and then went on to innovate beyond them.

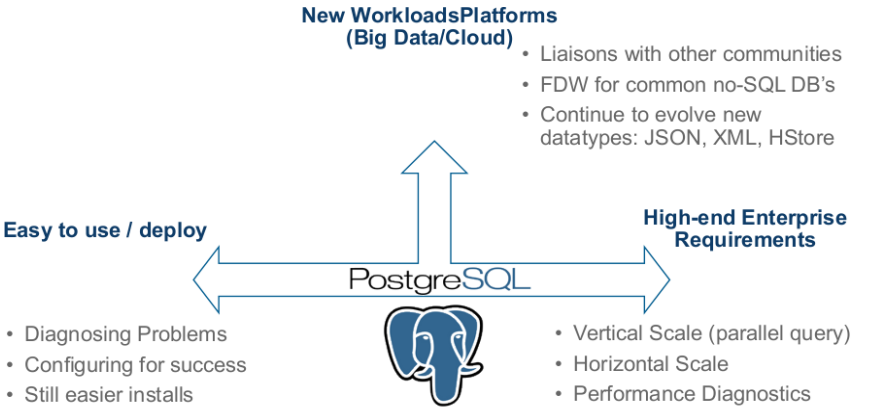
Postgres

Postgres nearing feature parity with:

1. Oracle
2. DB2
3. MS-SQL
4. Sybase
5. Informix
6. Ingres Corp.

and then going on to innovate beyond them.

Many Focuses



Keith Alsheimer, EDB, 2013

When Does Software Die?

- Proprietary software dies when the owner of the source code can no longer profit from it.
- It declines long before death due to profit maximization.
- Open source cannot die in the same way.
- Open source remains active while it serves a purpose.
- It can always be resurrected if useful.
- Postgres was given new life in 1996.

Ideas Don’t Die

1. Ideas don’t die, as long as they are shared.
2. Ideas are shared, as long as they are useful.
3. Postgres will live, as long as it is useful.

3. Open Source Adoption

https://www.flickr.com/photos/99438314@N02/

Open Source Survey, 2016

When the first survey launched 10 years ago, hardly anyone would have predicted that open source use would be ubiquitous worldwide just a decade later, but for many good reasons that's what happened. Its value in reducing development costs, in freeing internal developers to work on higher-order tasks, and in accelerating time to market is undeniable. Simply put, open source is the way applications are developed today.

Lou Shipley
President And CEO
Black Duck Software

https://www.slideshare.net/blackducksoftware/2016-future-of-open-source-survey-results

Advantages of Open Source

1. Innovation, competitive features
2. Freedom from vendor lock-in
3. Quality of solutions
4. Ability to customize and fix
5. **Cost (initially #1)**
6. Speed application development
7. Reduce development costs
8. Interoperability
9. Breadth of solutions

https://www.slideshare.net/blackducksoftware/2016-future-of-open-source-survey-results

Open Source Today

Open source today is unequivocally the engine of innovation; whether that's powering technology like operating systems, cloud, big data or IoT, or powering a new generation of open source companies delivering compelling solutions to the market.

Paul Santinelli
General Partner
North Bridge

https://www.slideshare.net/blackducksoftware/2016-future-of-open-source-survey-results

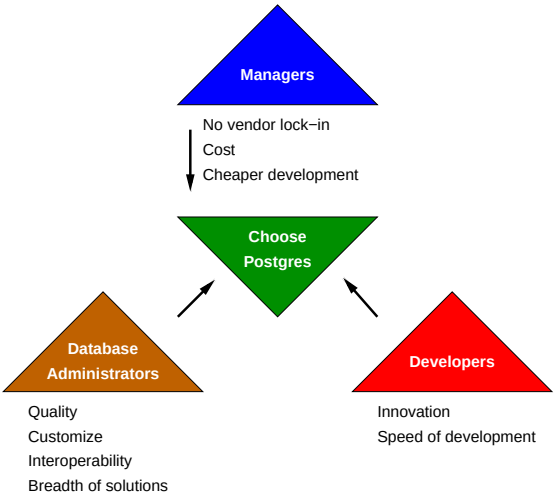
Open Source Usage, 2016

1. Operating Systems
2. Database
3. Development tools

Database didn't appear in the top three the previous year's survey (2015).

https://www.slideshare.net/blackducksoftware/2016-future-of-open-source-survey-results

Advantages of Open Source for Database Decision Makers

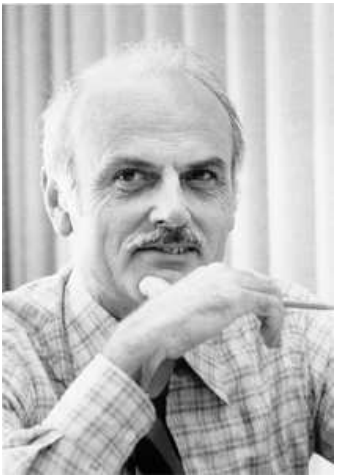


4. Postgres Innovation

https://www.flickr.com/photos/tomas_vondra/

Relational Innovation

- E. F. Codd introduces relational theory
- Row, column, table
- Constraints
- Normalization, joins
- Replaces key/value data storage systems
- Pre-Postgres



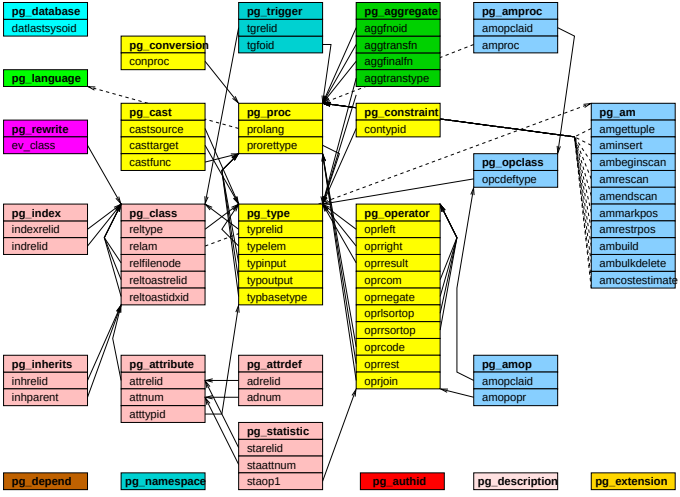
https://en.wikipedia.org/wiki/Edgar_F._Codd

University Postgres Innovation

- Michael Stonebraker creates university Postgres
- Allows extendability via system table contents:
  - Data types
  - Indexing methods
  - Server-side languages

https://en.wikipedia.org/wiki/Michael_Stonebraker

Postgres Extendability



Postgres Extension Data Type

```
CREATE EXTENSION isn;

\dT
                          List of data types
 Schema |  Name  |                    Description
--------+--------+-------------------------------------------------
 public | ean13  | International European Article Number (EAN13)
 public | isbn   | International Standard Book Number (ISBN)
 public | isbn13 | International Standard Book Number 13 (ISBN13)
 public | ismn   | International Standard Music Number (ISMN)
 public | ismn13 | International Standard Music Number 13 (ISMN13)
 public | issn   | International Standard Serial Number (ISSN)
 public | issn13 | International Standard Serial Number 13 (ISSN13)
 public | upc    | Universal Product Code (UPC)
```

https://momjian.us/main/writings/pgsql/central.pdf

Postgres Server-Side Languages

- PL/Java
- PL/Perl
- PL/pgSQL (like PL/SQL)
- PL/PHP
- PL/Python
- PL/R (like SPSS)
- PL/Ruby
- PL/Scheme
- PL/sh
- PL/Tcl
- PL/v8 (JavaScript)
- SPI (C)

https://momjian.us/main/writings/pgsql/central.pdf

Postgres Index Types

- BRIN
- BTree
- Hash
- GIN (generalized inverted index)
- GiST (generalized search tree)
- SP-GiST (space-partitioned GiST)

https://momjian.us/main/writings/pgsql/indexing.pdf

Postgres Innovation: Full Text Search

- Supports full text search capabilities in a relational database
- Whole-word, word prefix, *and, or,* and *not* searches
- Stemming for 21 languages
- *Pg_trgm* extension allows search of letter combinations and similarity
- Specialized indexing, operators, and functions
- Full transaction semantics

https://momjian.us/main/writings/pgsql/non-relational.pdf

Full Text Search

```
EXPLAIN SELECT line
FROM fortune
WHERE to_tsvector('english', line) @@ to_tsquery('pandas');
                                        QUERY PLAN
-----------------------------------------------------------------------…
 Bitmap Heap Scan on fortune  (cost=12.41..94.25 rows=21 width=36)
   Recheck Cond: (to_tsvector('english'::regconfig, line) @@ to_ts…
   ->  Bitmap Index Scan on fortune_idx_ts  (cost=0.00..12.40 rows…
         Index Cond: (to_tsvector('english'::regconfig, line) @@ t…
```

NoSQL

- Supports NoSQL capabilities in a relational database
- Mix structured and unstructured data in the same row and query; the best of both worlds
- Specialized indexing, operators, and functions
- Full transaction semantics

https://momjian.us/main/writings/pgsql/yesql.pdf

```
EXPLAIN SELECT data->>'last_name'
FROM friend2
WHERE data::jsonb @> '{"first_name" : "Jane"}'
ORDER BY 1;                    QUERY PLAN
-------------------------------------------------------------------…
 Sort  (cost=24.03..24.04 rows=1 width=139)
   Sort Key: ((data ->> 'last_name'::text))
   ->  Bitmap Heap Scan on friend2  (cost=20.01..24.02 rows=1 …
         Recheck Cond: (data @> '{"first_name": "Jane"}'::jsonb)
         ->  Bitmap Index Scan on friend2_idx  (cost=0.00..20.01 …
               Index Cond: (data @> '{"first_name": "Jane"}'::js…
```

Range Types

- Combines start and stop times into a single field
- Allows sophisticated indexing and comparisons
- Allows automatic range overlap prevention

https://momjian.us/main/writings/pgsql/non-relational.pdf

Range Types

```
EXPLAIN SELECT *
FROM car_rental
WHERE time_span @> '2007-08-01 00:00:00'::timestamptz;
                                    QUERY PLAN
-------------------------------------------------------------------------------------------
 Bitmap Heap Scan on car_rental  (cost=4.27..28.35 rows=16 width=36)
   Recheck Cond: (time_span @> '2007-08-01 00:00:00-04'::timestamp with time zone)
   ->  Bitmap Index Scan on car_rental_idx  (cost=0.00..4.27 rows=16 width=0)
         Index Cond: (time_span @> '2007-08-01 00:00:00-04'::timestamp with time zone)
```

Geometric Types

- Handle multi-dimensional data
  - Points
  - Lines
  - Circles
  - Polygons
- Multi-dimensional indexing and operators
- Allows efficient nearest neighbor searches
- Avoids using a separate geometric data store

https://momjian.us/main/writings/pgsql/non-relational.pdf

Geometric Types

```
EXPLAIN SELECT *
FROM dart
ORDER BY location <-> '(50, 50)'::point
LIMIT 2;
                                    QUERY PLAN
-------------------------------------------------------------------------------------
 Limit  (cost=0.14..0.30 rows=2 width=28)
   ->  Index Scan using darts_idx on darts  (cost=0.14..80.14 rows=1000 width=28)
         Order By: (location <-> '(50,50)'::point)
```

GIS

- PostGIS is a full-featured Geographical Information System (GIS)
- Implemented as a extension
- Independent development team and community

https://postgis.net/

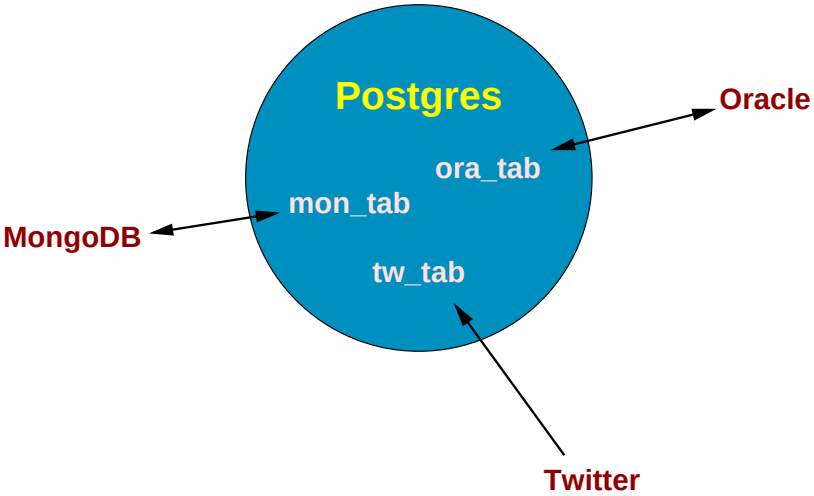
```
SELECT ST_Area(the_geom)/10000 AS hectares
FROM bc_municipality
WHERE name = 'PRINCE GEORGE';
    hectares
------------------
32657.9103824927
```

Foreign Data Wrappers

- 100+ interfaces to foreign data
- Read/write
- Sophisticated push down of joins, sorts, and aggregates

https://momjian.us/main/writings/pgsql/central.pdf

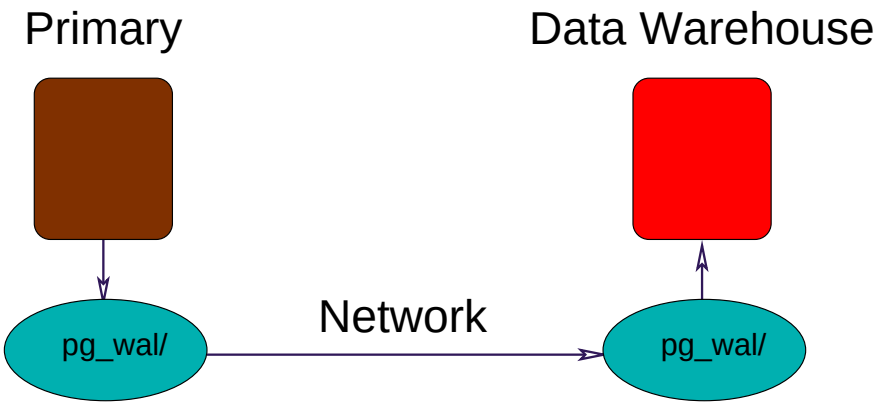
Foreign Data Wrappers



Data Analytics

- SQL
  - aggregates, GROUPING SETS, ROLLUP, CUBE
  - window functions
  - common table expressions (CTE)
  - server-side languages, e.g., PL/R
- Performance
  - optimizer
  - bitmap heap scans
  - BRIN and bloom indexes
  - materialized views
  - just-in-time compilation (JIT)
- Large data sets
  - data partitioning
  - tablespaces
  - parallelism
  - sharding (in progress)

https://momjian.us/main/writings/pgsql/central.pdf

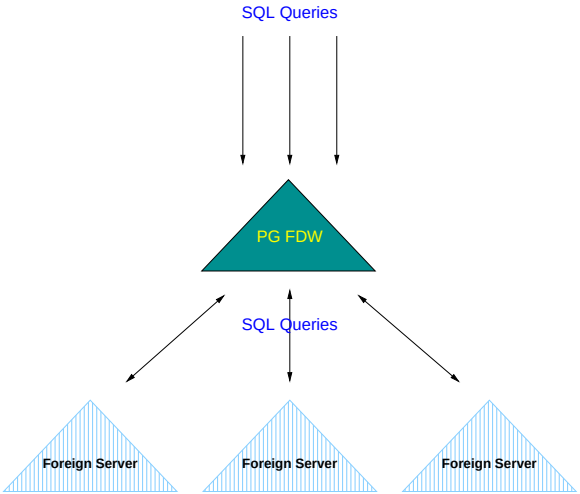

Primary
Data Warehouse
pg_wal/
Network
pg_wal/

Sharding

- Allows multi-host databases
- Uses existing functionality
  - Partitioning
  - Parallelism
  - Foreign data wrappers
  - Logical replication
- Needs new functionality
  - Global transaction manager
  - Global snapshot manager

https://momjian.us/main/writings/pgsql/sharding.pdf

Sharding


SQL Queries
PG FDW
SQL Queries
Foreign Server
Foreign Server
Foreign Server

5. Community Structure

https://www.flickr.com/photos/tomas_vondra/

Community Structure

- BSD license guarantees software will be available forever, including for proprietary use.
- Development and leadership is diversified geographically, culturally, and is multi-company.

Still Going Strong

- 35+ years of development
- 25+ years of annual major releases
- ~180 features per major release
- Quarterly minor releases
- Most-loved relational database
  - https://insights.stackoverflow.com/survey/2018/#technology-most-loved-dreaded-and-wanted-databases

PgLife

PgLife Postgres Community Life

Users

General	Re: Getting wrong datetime in database using insert into table query.
Other	Re: Deadlock
Announce	pgmoneta 0.10

Developers

Hackers	Re: Cutting support for OpenSSL 1.0.1 and 1.0.2 in 17~?
Commit	Avoid function call overhead of pg_popcount() in syslogger.c.
Versions	Stable: 16.2+, 15.6+, 14.11+, 13.14+, 12.18+ | Development: 17 devel, in commitfest

External

Blogs	Álvaro Hernández: Why Postgres Extensions should be packaged and distributed as OCI images
News	Ora2Pg 24.3 have been released
Media	Index Creation in PostgreSQL Large Tables: Essential Checklist for Developers - Medium
Events	PGDay at FOSSASIA Summit

IRC (also Slack)

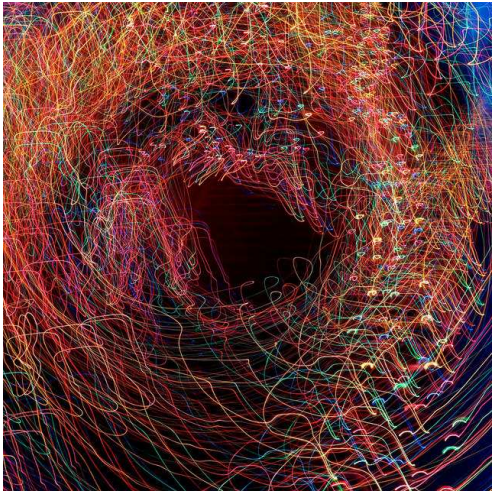
buu: but with treeparser it should be possible to pull out the sql bits
buu: *treesitter
pnorman: I try to keep my coding making sense if you looked at any jinja2 part in isolation or took them out. so if I open up a (in a conditional I also close it as well
buu: yeah those old school types are obnoxious
buu: I'm so happy I never have to read <* if 1 { *> ... <* } *>
pnorman: so WHERE (foo {% if bar %}AND baz){% else %}{% endif %} would always generate valid SQL, I avoid it
buu: augh
pnorman: actually, the worst things are commas. if you have a condition like foo IN (1,2,3,4) where depending on some other variable you keep on adding on to the list of 1,2,3,4 and 1 is always present you need to really tie the comma with the item *after* it, not before

London 19:59 Berlin 20:59 Moscow 21:59 Mumbai 00:29 Beijing 02:59 Tokyo 03:59 Los Angeles 11:59 New York 14:59 São Paulo 15:59

Content updates automatically | *About* | *Submit Feedback*

http://pglife.momjian.us

Conclusion



https://momjian.us/presentations

https://www.flickr.com/photos/pagedooley/