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

Creative Commons Attribution License http://momjian.us/presentations

Last updated: June, 2019
Outline

1. Forever
2. Software life cycle
3. Open source adoption
4. Postgres innovation
5. Community structure
6. Conclusion
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: ~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
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
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 begins Falcon development
1998: MicroProse releases Falcon 4.0
1999: MicroProse ends development
2000: leak of source code
2003: Benchmark Sims (BMS) releases community modifications
2005: Lead Pursuit releases Allied Force, which includes BMS mods
2015: GOG.com republishes Falcon 4.0
2015: BMS releases version 4.33, plus later minor releases

https://en.wikipedia.org/wiki/Falcon_4.0
Proprietary Development Flow

Developers

- Design Meetings
- Work in Isolation
- Project Meetings
- Testing/Retesting
- Release
- Fix Bugs

Users

- Receive Software
- Acceptance Tests
- Resolve Problems
- Install
- Production
- Resolve Issues

Sales
Open Source Development Flow

Developers

Propose Feature → Internet
Patch Review
Apply / Testing
Beta Testing
Release
Fix Bugs

Users

Discuss Feature
Patch Review
Testing
Beta Testing
Production
Resolve Issues
Rise of Open Source

Time

Open Source

Closed Source

Features
Performance
Reliability

Time
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

New Workloads/Platforms
(Big Data/Cloud)

- Liaisons with other communities
- FDW for common no-SQL DB’s
- Continue to evolve new datatypes: JSON, XML, HStore

Easy to use / deploy

- Diagnosing Problems
- Configuring for success
- Still easier installs
- Tighter integration with frameworks
- Integration with other data stores
- Very simple in the cloud

High-end Enterprise Requirements

- Vertical Scale (parallel query)
- Horizontal Scale
- Performance Diagnostics
- Incremental Backup
- Integration with other data stores
- Zero down time upgrades

Keith Alsheimer, EnterpriseDB
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.
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/
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. Competitive features, innovation
2. Freedom from vendor lock-in
3. Quality of solutions
4. Ability to customize and fix
5. Cost
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 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
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
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

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
CREATE EXTENSION isn;

\d

<table>
<thead>
<tr>
<th>Schema</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>ean13</td>
<td>International European Article Number (EAN13)</td>
</tr>
<tr>
<td>public</td>
<td>ismn</td>
<td>International Standard Music Number (ISMN)</td>
</tr>
<tr>
<td>public</td>
<td>ismn13</td>
<td>International Standard Music Number 13 (ISMN13)</td>
</tr>
<tr>
<td>public</td>
<td>issn</td>
<td>International Standard Serial Number (ISSN)</td>
</tr>
<tr>
<td>public</td>
<td>issn13</td>
<td>International Standard Serial Number 13 (ISSN13)</td>
</tr>
<tr>
<td>public</td>
<td>upc</td>
<td>Universal Product Code (UPC)</td>
</tr>
</tbody>
</table>

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)

Postgres Index Types

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

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

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...
Postgres Innovation: 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

EXPLAIN SELECT data->>'last_name'
FROM friend2
WHERE data::jsonb @> '{"first_name": "Jane"}';

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 ...
Postgres Innovation: Range Types

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

EXPLAIN SELECT *
FROM car_rental
WHERE time_span @> '2007-08-01 00:00:00'::timestamptz;

QUERY PLAN

Index Scan using car_rental_idx on car_rental (cost=0.15..8.17...}

Index Cond: (time_span @> '2007-08-01 00:00:00-04'::timestamp...
Postgres Innovation: 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

EXPLAIN SELECT *
FROM dart
ORDER BY location <-> '(50, 50)':::point
LIMIT 2;

QUERY PLAN

Limit (cost=0.14..0.33 rows=2 width=20)
  -> Index Scan using dart_idx on dart (cost=0.14..92.14...
Order By: (location <-> '(50,50)':::point)
Postgres Innovation: 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';

haectares
------------------
32657.9103824927
Postgres Innovation: Foreign Data Wrappers

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

Postgres Innovation: Foreign Data Wrappers
Postgres Innovation: Data Analytics

- Aggregates
- Optimizer
- Server-side languages, e.g., PL/R
- Window functions
- Bitmap heap scans
- Tablespaces
- Data partitioning
- Materialized views
- Common table expressions (CTE)
- BRIN indexes
- GROUPING SETS, ROLLUP, CUBE
- Just-in-time compilation (JIT)
- Parallelism
- Sharding (in progress)

Postgres Innovation: Data Analytics

Primary

Data Warehouse

Network
Postgres Innovation: Sharding

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

Postgres Innovation: Sharding

SQL Queries

PG FDW

SQL Queries

Foreign Server

Foreign Server

Foreign Server
5. Community Structure
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

- 32 years of development
- 22 years of annual major releases
- ~180 features per major release
- Quarterly minor releases
- Most-loved relational database
PgLife Postgres Community Life

Users
- how to understand checkpoint information in pg_control data

Other
- Small bug in new "Resend email" feature

Announce
- repmgr 4.4 Now Available

Developers
- Re: Commitfest 2018-07, the first of five* for PostgreSQL 13

Hackers
- Fixed misleading comment in nodexindexonlyscan.c.

Versions
- Stable: 11.4+, 10.8+, 9.6.14+, 9.5.16+, 9.4.23+ | Development: 12 beta2+

External
- Bruce Momjian: ibiza: A Different Type of Conference

Blog
- Announcing the release of repmgr 4.4

News
- It's #FeatureFriday! Did you know you can use regular expressions to search over...

Events
- Stream for Azure Database for PostgreSQL Now Available in the Microsoft Azure Marketplace - PRNewswire

IRC (also Slack)
- _ash works:_ do you separate services along databases, schemas or don't care?

_pgsql:_ Hello again! While using postgres docker image (https://hub.docker.com/_/postgres) I noticed a docker-compose rm will not ask to database to stop, requiring a docker exec -it postgres-img-name /bin/bash, a su postgres inside shell and pg_ctl stop after this (With complete path)

_ne2k:_ Is the normal behaviour? I don't see postgres stop messages in log while stopping docker, so something is wrong and to prevent issues, im stopping postgres manually. What people here know or thinks about these containers and about this situation?

Myon, you about? I can do unit(UNIT) / 30 days, but I can't do unit(UNIT) / (timestamptz - now()), it says unit / interval is not valid. what do I need to cast to?

_ne2k:_ failing that, anyone, how can I show which implicit type casts and operators are being used when I do an operation?

_ne2k:_ I can't find how to get pg to show what implicit type conversions are going on in an operation, is this possible?

_xzili:_ ne2k: if you run explain, it usually will show you (assuming I understand your question)

_ne2k:_ my real question is I can do unit(UNIT) / 30 days, but I can't do unit(UNIT) / (timestamptz - now()), it says unit / interval is not valid. what do I need to cast to

http://pglife.momjian.us