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](https://momjian.us/presentations)  
Creative Commons Attribution License

Last updated: March, 2020
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/
- 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%
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 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
- Patch Review
- Apply / Testing
- Beta Testing
- Release
- Fix Bugs

Users

- Discuss Feature
- Patch Review
- Testing
- Beta Testing
- Production
- Resolve Issues

Internet
Rise of Open Source

Features
Performance
Reliability

Time

Open Source
Closed Source
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 WorkloadsPlatforms (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
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.
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
Advantages of Open Source for Database Decision Makers

Managers
- No vendor lock-in
- Cost
- Cheaper development

Choose PG

Administrators
- Quality
- Customize
- Interoperability
- Breadth of solutions

Developers
- Innovation
- Speed of development
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

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
CREATE EXTENSION isn;

\dT

| Schema | Name   | Description                                                      |
|--------+--------+-----------------------------------------------------------------|
| public | ean13  | International European Article Number (EAN13)                    |
| 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)                                    |

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/pgresql/central.pdf](https://momjian.us/main/writings/pgresql/central.pdf)
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');

---

Bitmap Heap Scan on fortune (cost=12.41..94.25 rows=21 width=36)
  Recheck Cond: (to_tsvector('english'::regconfig, line) @@ to_tsquery('pandas'))
  -> Bitmap Index Scan on fortune_idx_ts (cost=0.00..12.40 rows=21)
    Index Cond: (to_tsvector('english'::regconfig, line) @@ to_tsquery('pandas'))
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"}'
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"}'::jsonb)
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

Postgres Innovation: Geometric Types

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';

hectares
------------------
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
- ora_tab
- tw_tab
- mon_tab
- MongoDB
- Oracle
- Twitter
Postgres Innovation: Data Analytics

- Aggregates
- Optimizer
- Server-side languages, e.g., PL/R
- Window functions
- Bitmap heap scans
- Tablesspaces
- 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: Sharding

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

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

- 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

<table>
<thead>
<tr>
<th>Users</th>
<th>General</th>
<th>Re: Restrict user to create only one db with a specific name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>Re: JSON array is not updated if updating with empty array</td>
<td></td>
</tr>
<tr>
<td>Announce</td>
<td>pgAdmin4 v4.19 released</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developers</th>
<th>Hackers</th>
<th>Questions about the CI process and proposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Re: Revert &quot;initdb: Change authentication defaults&quot;</td>
<td></td>
</tr>
<tr>
<td>Versions</td>
<td>Stable: 12.2+, 11.7+, 10.12+, 9.6.17+, 9.5.21+</td>
<td>Development: 13 devel, in commitfest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External</th>
<th>Blogs</th>
<th>Mark Wong: Creating a PostgreSQL procedural language - Part 4 - Handling Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>News</td>
<td>dbMigration.NET v12 released</td>
<td></td>
</tr>
<tr>
<td>Tweets</td>
<td>Your regularly scheduled PostgreSQL update release is coming out this Thursday, ...</td>
<td></td>
</tr>
<tr>
<td>Media</td>
<td>EnterpriseDB looks to grow market for PostgreSQL - TechTarget</td>
<td></td>
</tr>
<tr>
<td>Events</td>
<td>PostgreSQL@SCaLE18x</td>
<td></td>
</tr>
</tbody>
</table>

**IRC (also Slack)**

Sargun: FunkyBob: shouldn't the serializable one fail

FunkyBob: Sargun: yes, that would be my understanding, but I'm no expert on the topic

Sargun: FunkyBob: madcide: awake?

FunkyBob: if both txms are marked serializable, it errors

FunkyBob: ah, interesting

FunkyBob: can you see which order the DB actually commits them?

FunkyBob: I could imagine since the former is serializable, the latter is delayed

gp3st: I'd like to use the same trigger for insert, update, and delete, but for updates and deletes I need the old table and for inserts and updates I need the new table. Is there a way to, in the trigger function, sort of create an empty table with the appropriate names based on the trigger op?

http://pglife.momjian.us