PostgreSQL Goes to Eleven!

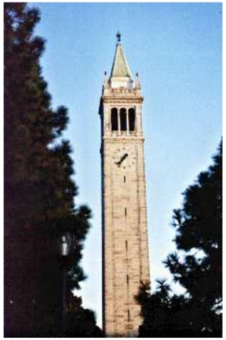
Joe Conway
joe@crunchydata.com
mail@joeconway.com

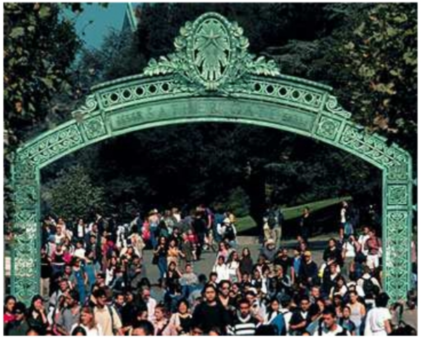
Crunchy Data
February 02, 2019

PostgreSQL Community and Future Development

- Community
  - History
  - Development Process
- PostgreSQL 11
  - Committed patches
- 12 and Beyond
  - Committed patches
  - Being worked/discussed

University of California Berkeley





Courtesy: Bruce Momjian

History: INGRES

- 1974 – 1985: University INGRES
  - INteractive Graphics REtrieval System
  - Prototype of a relational DBMS
  - University of California at Berkeley
  - Prof. Stonebraker
  - Spawned commercial databases: Sybase, MSSQL, NonStop SQL, others
- 1980 – Present: Commercialization
  - Commercial success
  - Relational Technologies ⇒ Ingres Corp. ⇒ Computer Associates ⇒ Ingres Corp. ⇒ Actian
- 2006 – Present: Open Source

crunchy data

History: POSTGRES

- 1986 – 1994: University POSTGRES
  - ”Post Ingres”
  - University of California at Berkeley
  - Prof. Stonebraker
  - Prototype of an object-relational DBMS
  - POSTQUEL query language
  - Spawned commercial databases: Illustra ⇒ Informix, others

History: PostgreSQL

- 1994 – 1995: Postgres95
  - University of California at Berkeley
  - Andrew Yu and Jolly Chen
  - Conversion to SQL
  - More liberal license
- 1996 – present: PostgreSQL
  - Open-source project
  - PostgreSQL Global Development Group
  - Spawned many derived products

Originators

Postgres

Michael Stonebraker

Postgres95

Jolly Chen and Andrew Yu

Courtesy: Bruce Momjian

First Core Team + Jolly and Andrew

Top row: Thomas Lockhart, Jolly Chen, Vadim Mikheev
Jan Wieck, Andrew Yu, Tom Lane
Bottom row: Bruce Momjian, Marc Fournier

PostgreSQL 10 Year Anniversary Summit - 2006

Courtesy: Alvaro Herrera

PostgreSQL Community

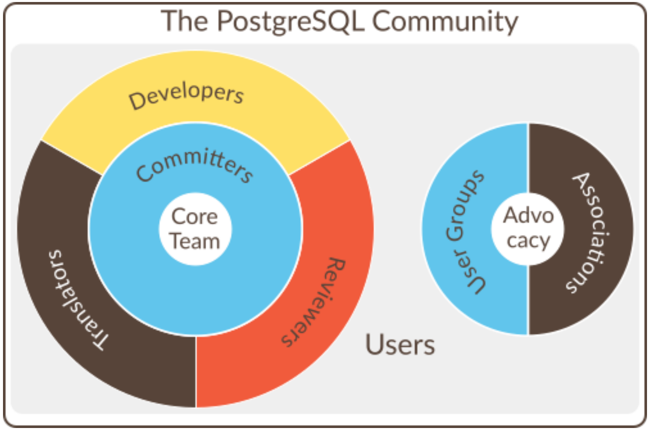
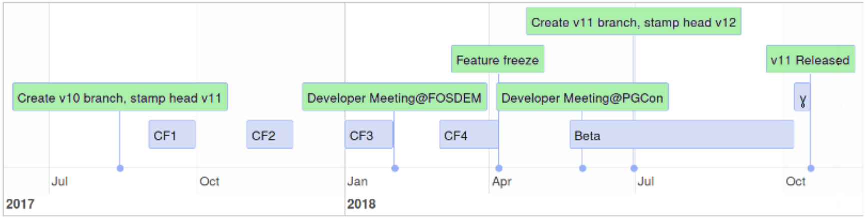


Diagram Courtesy of Lætitia Avrot

PostgreSQL Development Cycle



PostgreSQL Versioning

- Old Versioning scheme
  - Up to version 9.6
  - Three segments
  - Major X.Y (e.g. 9.6)
  - Minor X.Y.Z (e.g. 9.6.8)
- New Versioning scheme
  - Starting with version 10
  - Only two segments
  - Major X (e.g. 10)
  - Minor X.Z (e.g. 10.3)

Commitfests

- Commitfest App: https://commitfest.postgresql.org/
- CF-entries:
  - Patch or set of patches
  - Implements some goal
  - Proposed for inclusion
  - One or more authors/reviewers
  - Examples: new feature, bugfix, refactoring, improved docs

Commitfests

- Goals:
  - Patches do not get lost
  - Reviewers/committers can quickly find patches deserving attention
  - Single place tracking discussion and documenting state of CF-entries
  - Encourage:
    - people review other people’s patches
    - review patches proportionally to patches submitted

Commitfests

- CF-entry States:
  - Needs Review
  - Waiting for Author
  - Ready for Committer
  - Committed
  - Rejected
  - Returned with Feedback
  - Moved to next CF

Commitfests

- Process:
  - Start on certain date
    - submission deadline date is day before start date, AoE (UTC-12)
    - deadline not passed if, anywhere on earth, deadline date has not yet passed
    - must be in Needs Review or Ready for Committer state
  - CF manager: ensures CF-entries in appropriate state, moves process along
  - Reviewers analyze/test patches, provide feedback
  - Authors adapt entries per reviews/discussion
  - When satisfied, reviewer marks CF-entry Ready for Committer
  - Committer either commits or resets state

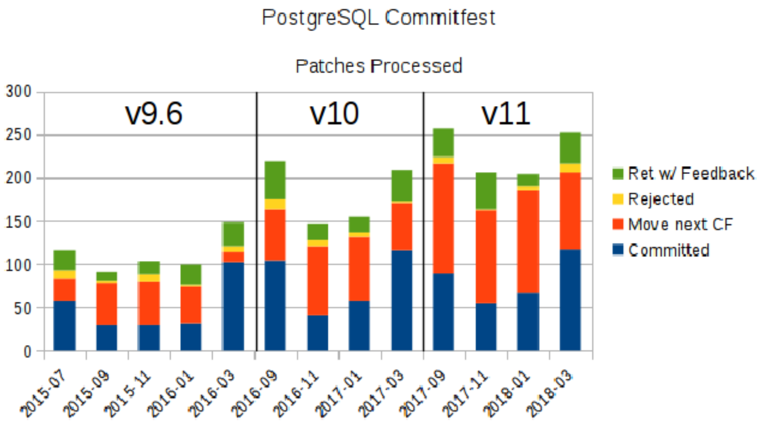
Commitfests

- Process:
  - Waiting for Author too long $\Rightarrow$ Returned with Feedback
  - At end of commitfest:
    - Needs Review $\Rightarrow$ Moved to next CF
    - Waiting on Author $\Rightarrow$ Returned with Feedback

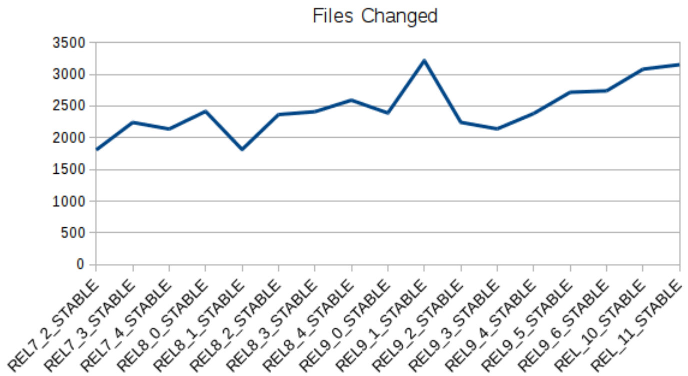
Commitfests

- Last Commitfest:
  - CF before the expected feature freeze date
  - Extra restrictions:
    - No nontrivial CF-entries unless previously submitted to earlier CF
    - CF-entries deemed unlikely to finish in last CF aggressively be moved early

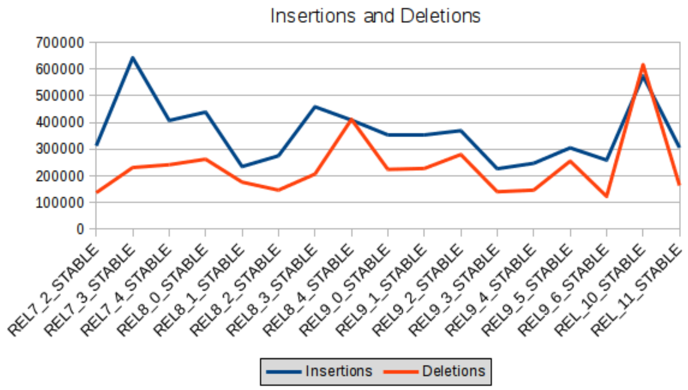
Commitfests



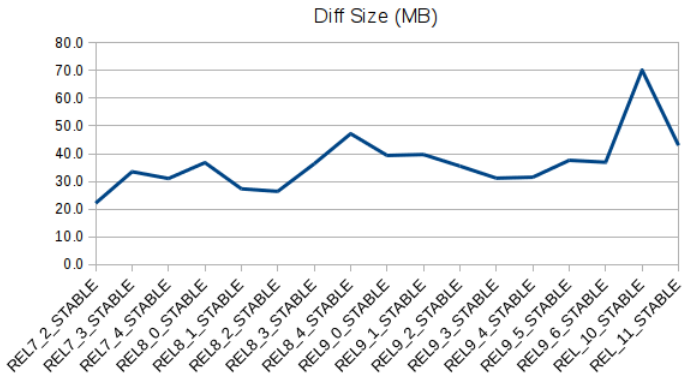
Release Stats



Release Stats



Release Stats



Committed by the Numbers (CF1-CF4)

- Bug Fixes - 84
- Clients - 27
  - ECPG - 2
  - pg_dump - 3
  - pg_receivewal - 2
  - pgbench - 11
  - psql - 8
  - pg_rewind - 1
- Code Comments - 5
- Documentation - 26
- Miscellaneous - 34
- Monitoring and Control - 6

Committed by the Numbers (CF1-CF4)

- Performance - 38
    - caching - 3
    - index - 5
    - memory - 2
    - miscellaneous - 6
    - parallel query - 7
    - partitioning - 6
    - plan/opt - 8
    - JIT - 1
- Procedural Languages - 13
- Refactoring - 38
- Replication and Recovery - 10

Committed by the Numbers (CF1-CF4)

- Security - 2
- Server Features - 29
  - authentication - 1
  - miscellaneous - 10
  - parallel query - 1
  - partitioning - 11
  - security - 1
  - predicate locking - 3
  - indexes - 2
- SQL Commands - 6
- System Administration - 9

crunchy data

Committed - Notable Features

- Partitioning
  - Many bug fixes and miscellaneous improvements
  - Hash Partitioning
  - Default partition
  - Partition-wise `JOIN` for partitioned tables
  - `pg_dump`/`pg_restore` reload through parent
  - Automatic creation of similar indexes on each partition
  - `UPDATE` moves rows between partitions
  - `PRIMARY KEY` and `UNIQUE` indexes on partitioned tables
  - Faster and Runtime (for prepared statements) partition pruning
  - Partition-wise aggregation
  - Allow updating partition key
  - Support `INSERT .. ON CONFLICT`
  - Tuple routing for foreign partitions
  - Foreign keys on partitioned tables

Committed - Notable Features

- Parallelization
  - Many bug fixes and miscellaneous improvements
  - Queries with InitPlans
  - `CREATE TABLE ... AS`, `SELECT INTO`, and `CREATE MATERIALIZED VIEW` queries
  - Prepared statements with generic plans.
  - Queries with Append plan nodes
  - Hash joins
  - Btree index builds
  - `LIMIT` clause passed to workers

Committed - Notable Features

- Performance
  - Pushdown `LIMIT` through subqueries to underlying sort, where possible
  - `SET STATISTICS` on expression indexes
  - Automatic ”prewarm” for `pg_prewarm`
  - Configurable WAL segment size at initdb time
  - Index-only Bitmap scans
  - Generational Memory Allocator
  - Improve performance of MemoryContext creation
  - Push down UPDATE/DELETE joins to remote postgres_fdw servers
  - Clone extended stats in CREATE TABLE (LIKE INCLUDING ALL)
  - Just-In-Time (JIT) Compilation expressions and tuple deform
  - `ALTER TABLE ADD COLUMN` fast `DEFAULT`
  - Predicate locking in Gist/Hash indexes
  - Covering Indexes

crunchydata

Committed - Notable Features

- Logical replication
  - Many bug fixes and miscellaneous improvements
  - `TRUNCATE` support
- Testing
  - Coverage Analysis improvements
  - Additional tests
  - Improved code coverage
- Authentication
  - Custom search filters for LDAP auth
  - LDAPS support
  - libpq connection parameter `scram_channel_binding`
  - SCRAM channel bindings `tls-unique` and `tls-server-end-point`

Committed - Notable Features

- Miscellaneous
  - Build with Visual Studio 2017
  - Arrays over domains
  - Domains over composite types
  - Fewer superuser checks, more GRANT-based
  - Convert documentation to DocBook XML
  - SQL procedures with transaction control
  - Transaction control in PL procedures
  - User-callable SHA-2 functions
  - `pg_stat_statements` 64 bit queryid
  - Window frame clauses now full SQL:2011 support
  - Allow external command for obtaining passphrases for SSL key files
  - Verify Checksums during Basebackups
  - Support huge pages on Windows

crunchydata

Committed - Notable Features

- pgbench
  - Allow non-ASCII characters in variable names
  - Add approximated Zipfian-distributed random generator
  - Add `pow()`, aka `power()`, function
  - Improve scripting language in pgbench
- psql
  - Use `PSQL_PAGER` in preference to `PAGER` if set
  - `\gdesc` command
  - Variables to track success/failure of SQL
  - Test for variable existence
  - `quit` and `exit` commands

Committed (as of November 25)

- `git log` from PG11 branch to date is almost 15000 lines
- Performance
  - Use optimized bitmap set function for membership test in postgres_fdw
  - Improve the performance of relation deletes during recovery.
  - Mark built-in btree comparison functions as leakproof where it's safe.
  - Hand code string to integer conversion for performance
  - Allow multi-inserts during COPY into a partitioned table
  - Make assorted performance improvements in snprintf.c
  - Provide fast path in snprintf.c for conversion specs that are just "%s"
  - Avoid O(N2̂) cost in ExecFindRowMark() (many partitions perf optimized)
  - Reorder FmgrBuiltin members, saving 25% in size
  - Improve parallel scheduling logic in pg_dump/pg_restore

Committed (as of November 25)

- Logging
  - Add application_name to connection authorized msg
  - Improve autovacuum logging for aggressive and anti-wraparound runs
  - Implement ”pg_ctl logrotate” command
- psql
  - psql: Show IP address in conninfo
  - psql: Describe partitioned tables/indexes as such

Committed (as of November 25)

- Miscellaneous
  - Allow replication slots to be dropped in single-user mode
  - Allow CALL with polymorphic type arguments
  - Add pg_dump –on-conflict-do-nothing option
  - Add toast tables to most system catalogs
  - Require C99 (and thus MSVC 2013 upwards)
  - Allow extensions to install built as well as unbuilt headers
  - Add support for nearest-neighbor (KNN) searches to SP-GiST
  - Integrate recovery.conf into postgresql.conf
  - Add option SKIP_LOCKED to VACUUM and ANALYZE
  - Improve the accuracy of floating point statistical aggregates
  - Allow to rename index in concurrent mode

Committed (as of November 25)

- New functions/settings
  - Add pg_ls_archive_statusdir function
  - Add pg_ls_tmpdir function
  - Add pg_promote function
  - Add pg_partition_tree to display information about partitions
  - Add a 64-bit hash function for type citext
  - Add a 64-bit hash function for type hstore
  - Add settings to control SSL/TLS protocol version
- Removed
  - Remove timetravel extension
  - Remove deprecated abstime, reltime, tinterval datatypes
  - Remove WITH OIDS support, change oid catalog column visibility

Ongoing Discussions

- Pluggable Storage
  - zheap - In-place updates, reduce VACUUM
  - Columnar Storage
- MERGE
- Allow `VACUUM` to use more than 1G
- Online enabling of checksums
- SQL/JSON Standard (functions, `JSON_TABLE`, jsonpath)
- SQL ASSERTION
- 64bit transaction IDs
- Parallelize more operations (VACUUM?)
- More improvements in partitioning (automatic creation?)
- Database Encryption

Ongoing

- Logical Replication conflict handling
- Synchronous logical replication
- Chained transactions
- Generated columns
- Alternate to OpenSSL (e.g. GnuTLS) Support
- Global Indexes
- Index-Organized Tables
- Autonomous Transactions
- Cron-like Background Worker

Ongoing

- Building more on logical replication
- Using FDWs from parallel workers
- Asynchronous Append with FDWs
- `CREATE VARIABLE`
- Shared-memory based stats collector
- Concurrent REINDEX
- Precalculate STABLE functions
- Inline CTEs

Questions?

Thank You!
joe@crunchydata.com
mail@joeconway.com