PostgreSQL Past, Present, and Future
Outline

- Past: A Brief History of PostgreSQL
- Present: New Features In PostgreSQL 9.5
- Future: Features in PostgreSQL 9.6 and Beyond
POSTGRES began as a research project at UC Berkeley in 1986.

In 1994, support for SQL was added and it was released the following year as Postgres95.

In 1996, it was renamed to PostgreSQL and the “modern era” of PostgreSQL as an open source project began.
When I Started Using PostgreSQL...

- No Built-In Replication.
- No Slony, either. No pgpool.
- No Windows Support.
- No Schemas.
- No Autovacuum.
- Couldn't drop table columns.
- Really slow.

- This was PostgreSQL 7.2, circa 2001.
So What Happened?

- 2002: PostgreSQL 7.3 adds schemas and support for DROP COLUMN.
- 2004: First release of Slony.
- 2005: PostgreSQL 8.0 adds Windows support.
- 2006: First release of pgpool-II.
- 2008: PostgreSQL 8.3 adds Heap Only Tuples (HOT) and 1-byte varlena headers. It also improves autovacuum and turns it on by default.
- 2010: PostgreSQL 9.0 adds Streaming Replication and Hot Standby. Also adds pg_upgrade.
Five Years Ago

- By the time PostgreSQL 9.0 was reduced, most of the really serious performance problems had been fixed.
- Improvements to VACUUM and the autovacuum daemon had fixed many of the troubling maintenance problems of earlier releases.
- Slony was in version 2.x, and pgpool-II was in version 3.x; both were mature projects.
- You could now deploy PostgreSQL in environments that required high availability, and even do some limited clustering.
Further Progress

• Vertical Scalability. PostgreSQL 9.2 and 9.4 improved PostgreSQL's ability to scale on large systems.

• Foreign Data Wrappers. PostgreSQL 9.1 introduced foreign tables, which became writeable in 9.3.

• JSON. PostgreSQL 9.2 introduce a json datatype. PostgreSQL 9.4 adds jsonb and advanced indexing.

• Checksums. PostgreSQL 9.3 adds the optional capability to checksum all data blocks.

....and many others.
PostgreSQL 9.5 Features

- INSERT .. ON CONFLICT UPDATE
- GROUPING SETS, CUBE, ROLLUP
- Row-Level Security
- Block Range Indexing (BRIN)
- Faster Sorting
- Better Scalability
ON CONFLICT DO NOTHING: Try to insert a row, but skip the insert if it would fail with a duplicate key violation.

ON CONFLICT DO UPDATE SET: Try to insert a row, but if there is already an existing row with the same key, update the existing row instead of inserting.

Very desirable feature for application developers – allows pushing rows into the database without regard to whether those rows are already present.

Most other databases have a similar feature (often SQL-standard MERGE, but not always).
• INSERT .. ON CONFLICT UPDATE uses different syntax because it is less general than MERGE. It requires a UNIQUE index matching the key columns.

• This restriction allows correct behavior under concurrency.

• Getting the behavior under heavy concurrency right was very difficult.
PostgreSQL 9.5: GROUPING SETS (Andrew Gierth, Atri Sharma)

- SELECT a, sum(c) FROM foo GROUP BY a;
- SELECT b, sum(c) FROM foo GROUP BY b;

→

- SELECT a, b, sum(c) FROM foo GROUP BY GROUPING SETS ((a), (b));

- Unused grouping columns are filled in with NULLs.
- More efficient than running multiple queries.
PostgreSQL 9.5: RLS (Craig Ringer, Stephen Frost, and many others)

- Allow access to a table while restricting access to some rows.
- The rows to which the user does not have access are simply invisible – no errors as with column-level or table-level security.

- CREATE POLICY classify ON foo USING (classification_level < 10);
- ALTER TABLE classify ENABLE ROW LEVEL SECURITY;
Block Range Index

Intended to create small indexes on large data sets.

The indexes won't always be as accurate as a btree index, but they're much smaller, which is a big advantage.

Simple way to think about it: Stores the minimum and maximum range for the indexed column for each page in the table.

Actually, could be ranges of pages, and might be something other than minimum and maximum.
Abbreviated keys make sorting varchar, text, and numeric fields much faster in most cases.

Basically, we compare the first eight bytes of the string (after transforming it with strxfrm if appropriate) and only compare the rest of the string if that's equal.
PostgreSQL 9.5+: Better Scalability (Amit Kapila, Andres Freund, Robert Haas)

• Improved concurrency of buffer eviction can provide a massive benefit on workloads where the working set exceeds the size of shared_buffers.

• Improved LWLock implementation dramatically improves performance on systems with 4+ sockets.

• Work continues for 9.6 and beyond.
Parallel query refers to the ability to use multiple CPUs to answer a single query.

Deeply embedded assumption: a transaction happens within a single process.

Basically every layer of the system needed updating.

Parallel query plans are different from serial query plans – you can't just run the same query plan with more CPUs.

9.6 will have many limitations … but there's still some pretty cool stuff there.
pglogical_output is a proposed new loadable module that integrates with the logical decoding facility introduced in PostgreSQL 9.4. It streams changes in JSON format.

pglogical is a proposed new contrib module that allows logical replication via a publish/subscribe model, powered by background workers and pglogical_output.

Together, these would give us high-performance log-based logical replication in the core distribution.
PostgreSQL 9.6: FDW Pushdown

• Sort Pushdown

– Given “SELECT stuff FROM ft WHERE conditions ORDER BY something”, we can now transmit the “ORDER BY something” clause to the remote side.

• Join Pushdown

– Given “SELECT stuff FROM ft1, ft2, ft3 WHERE conditions”, we can now transmit the entire join (all foreign tables on the same server) to the remote side.

• DML Pushdown + Sort/Join Pushdown Still Being Worked On...
PostgreSQL 9.6?

- Causal Reads
- Declarative Partitioning
- Column Store
- Freeze Map
- Better Monitoring Facilities
Where Is PostgreSQL Today?

- Excellent scalability on 2-socket systems; beginning to tackle scalability issues specific to 4+-socket systems
- Tackling new workloads, like high-security environments, unstructured data, and analytics.
- Rich ecosystem of tools
- Ready for mission-critical enterprise applications
What Is The Future of PostgreSQL?

- Logical Replication is an incredibly powerful tool. Streaming replication is great for high availability, but logical replication will allow partial replication, cross-version replication, heterogeneous replication, even multi-master replication.

- Parallelism will improve PostgreSQL's ability to use all of the available hardware, continuing a trend toward analytic workloads, and easing the pain of bulk operations.
Foreign Data Wrappers – with enough work on “pushdown” optimizations – will allow queries to access both local and remote data transparently.

Performance and scalability work will continue. Some of this work will involve investigating new storage formats for PostgreSQL, highly optimized for specific use cases.
Technical Challenges

- Continuous Availability. Streaming Replication helps, and so does pg_upgrade. Logical Replication will help more, but there are still challenges around keeping the system 100% available.

- Connection Pooling. Connecting to the database server is a heavyweight operation, and connections use a lot of resources.

- No In-Place Update. HOT is a huge advance, but there are some workloads where controlling table bloat is difficult.
Summary

- PostgreSQL has gone from a low-performance database suitable for personal projects to a highly scalable enterprise-class database.
- PostgreSQL continues to innovate and add major new technologies that will allow it to continue to reach more users and more use cases.
- More users and developers needed!
Thank You

- Any questions?