PostgreSQL Universal Database

Oleg Bartunov
Moscow University,
Postgres Professional

Dec 5, 2017, PGCONF.ASIA, Tokyo
When I started using Postgres

- No UTF-8, even no 8-bit
- No WAL
- No MVCC
- No replication
- No usable non-scalar data types
- No subselects, no window functions, no CTE
- It was Postgres95
Oleg Bartunov, major PostgreSQL contributor since Postgres95
How to choose a right database?

- People usually choose a database looking on:
  - Functionality, Performance
  - Availability - License, price
  - Local expertise, Personal experience
  - Compatibility to existing environment
  - Support
  - Other

- People start to think about one important feature of database after the project needs:
  - Need new functionality, Better performance

- Project is in production, no way to change database:
  - Starting to use various ugly «solutions»
  - System works, but looks pretty strange
Your system with wrong database
Any project could start with PostgreSQL
PostgreSQL is a reliable and stable database with rich functionality and long history
PostgreSQL has liberal BSD license, cross platform (~30)
Developed by international community, no vendor lock

PostgreSQL is **EXTENSIBLE**, this is the very important feature, which people miss! It allow database to support
• New workloads
• New functionality
• New environment
• Often without restarting a server, no need core programmer.
“The main design goals of the new system are to:
2) provide user extendibility for data types, operators and access methods,”

Stonebraker M., Rowe L. A. The design of Postgres.

“It is imperative that a user be able to construct new access methods to provide efficient access to instances of nontraditional base types”

Michael Stonebraker, Jeff Anton, Michael Hirohama.
Extendibility of PostgreSQL

0. Ability to change executor and planner
1. Data types, operators, access methods, indexes without core developers, deploy online!
2. FDW, procedural languages

Need more:

1. API to pluggable storages
2. API to TM — XTM, eXtensible transaction manager
3. Type-specific compression

.....more...
Case 1: Web

1996: Start using Postgres on Web, no 8-bit support — introduced locale
1999: World’s top-5 portal. We start with PostgreSQL 6.5.
Requirement: on the hardware ~ my smartphone to support > 1 mln. users/day

- Quickly run out of resources
- Denormalize, use arrays -> slow -> discover GiST → improve GiST - intarray with indexes
- Need FTS, made tsearch using intarray and GiST indexes
- Need fast search on hierarchical data — ltree — GiST indexes
- Need flexible schema — hstore — GiST index
- Need faster FTS — GIN index for tsearch, hstore
- Need misprint search — pg_trgm — GiST/GIN indexes
- Compete to NoSQL - better/binary json - jsonb — GIN index
- Need faster FTS — RUM access methods

2017: STILL USING PostgreSQL!
Extensibility makes PostgreSQL Universal Database!
PostgreSQL Forks (we love forks!)

- Commercial
- Open Source

- TelegraQ CQ
- TruQ CQ Cisco
- ParAccel
- Vertica HP
- Greenplum
- Enterprise Postgres Fujitsu
- Aster Data Terradata
- Netezza IBM
- PowerGres
- TelegraQ CQ
- RedShift Amazon
- GresCube
- Vitesse DB
- Greenplum
- CitusDB
- CitusDB
- RecDB
- AgensGraph
- Postgres-XC
- Postgres-X2
- TimescaleDB
- Postgres-XL
- 2ndQPostgres
- Postgres Pro
- Enterprise

PostgreSQL Forks (we love forks!)

- PostgreSQL - OLTP, Web, Science extensions: GIS (PostGIS), time-series (TimeScaleDB), Stream data (PipelineDB), interactive analytics (Citus), vector ops, async mpp on GPU (PG-Strom)
- AgensGraph — Graph model, graph query language
- Postgres Pro Enterprise - Multimaster, block compression, 64-bit TX, partitioning, adaptive planning, scheduler, incremental backup
- Redshift — Cloud, column storage, compression, OLAP
- Greenplum - MPP, OLAP (complex reports)
- Postgres-XL — MPP OLAP (ACID)
- Vitesse DB — OLAP (JIT, column storage, threads)
- Aurora PostgreSQL — OLTP, S3 storage
- EDB Postgres Advanced Server - Oracle compatibility
Case 2: NoSQL Postgres

- **SQL/JSON**
  - 2018
  - Custom types support
  - SQL 2016 support
  - Jsonb compression
  - Subscripting syntax

- **JSON**
  - 2012

- **JSONB**
  - 2014
  - Advanced indexing
  - Update, delete

- **HSTORE**
  - 2003-2006

- **SQL/JSON++**
  - 2019
  - Custom types support
  - Advanced indexing
  - Update, delete

SQL/JSON++ supports SQL 2016, Jsonb compression, and subscripting syntax.
PostgreSQL users

+BIG RUSSIAN Enterprise!
Open source vs commercial DBMS

Popularity trend

- Commercial License
- Open Source License

© 2017, DB-Engines.com
PostgreSQL is #4!

<table>
<thead>
<tr>
<th>Rank</th>
<th>DBMS</th>
<th>Database Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Oracle +</td>
<td>Relational DBMS</td>
<td>1360.05 +11.25 -52.96</td>
</tr>
<tr>
<td>2.</td>
<td>MySQL +</td>
<td>Relational DBMS</td>
<td>1322.03 +23.20 -51.53</td>
</tr>
<tr>
<td>3.</td>
<td>Microsoft SQL Server +</td>
<td>Relational DBMS</td>
<td>1215.08 +4.76 +1.27</td>
</tr>
<tr>
<td>4.</td>
<td>PostgreSQL +</td>
<td>Relational DBMS</td>
<td>379.92  +6.64 +54.10</td>
</tr>
<tr>
<td>5.</td>
<td>MongoDB +</td>
<td>Document store</td>
<td>330.47  +1.07 +5.00</td>
</tr>
<tr>
<td>6.</td>
<td>DB2 +</td>
<td>Relational DBMS</td>
<td>194.06  -0.53 +12.61</td>
</tr>
<tr>
<td>7.</td>
<td>Microsoft Access</td>
<td>Relational DBMS</td>
<td>133.31  +3.86 +7.34</td>
</tr>
<tr>
<td>8.</td>
<td>Cassandra +</td>
<td>Wide column store</td>
<td>124.21  -0.58 -9.76</td>
</tr>
<tr>
<td>9.</td>
<td>Redis +</td>
<td>Key-value store</td>
<td>121.18  -0.87 +5.64</td>
</tr>
<tr>
<td>10.</td>
<td>Elasticsearch +</td>
<td>Search engine</td>
<td>119.41  -0.82 +16.84</td>
</tr>
</tbody>
</table>

https://db-engines.com/en/ranking
PostgreSQL in Russia is #1
Case 3: PostgreSQL and Supernovae

SN 1987a, Type II, +2.9, Tarantula Nebulae in LMC, 168 000 ly, progenitor: Sanduleak -69° 202,
The Scale of the Universe

- Supernovae (Ia) - «standard candles»
- Used to measure the distance to the host galaxy
Figure 1. The world is growing. The expansion of the Universe began with the Big Bang 14 billion years ago, but slowed down during the first several billion years. Eventually it started to accelerate. The acceleration is believed to be driven by dark energy, which in the beginning constituted only a small part of the Universe. But as matter got diluted by the expansion, the dark energy became more dominant.
<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>The rate of supernovae from the combined sample of five searches</td>
<td>1996</td>
</tr>
<tr>
<td>Radiation hydrodynamics of SN 1987A. I. Global analysis of the light curve for the first 4 months</td>
<td>2000</td>
</tr>
<tr>
<td>The rate of supernovae. II. The selection effects and the frequencies per unit blue luminosity</td>
<td>1993</td>
</tr>
<tr>
<td>Non-equilibrium radiative transfer in supernova theory-models of linear Type-II supernovae</td>
<td>1993</td>
</tr>
<tr>
<td>Distribution of supernovae relative to spiral arms and H II regions</td>
<td>1994</td>
</tr>
</tbody>
</table>
M31 (Andromeda), AZT-5 telescope

SN 2008fv in NGC 3147, Draco
Dmitry Tsvetkov, SAI MSU
Blink Comparator (manual discovery)
Many hours of hard work!
M31 (Andromeda), AZT-5 telescope
**Postgres**  
**Spatial Join (Machine Discovery, < 1s)**

<table>
<thead>
<tr>
<th>Observations: $10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Observations Table" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Catalog(s): $10^9$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Catalog Table" /></td>
</tr>
</tbody>
</table>
Astronomy meets database

Indexing the SKY with PostgreSQL

HEALPIX                      HTM                      Q3C (PG)

PGSphere (PG)

The sphere segmentation in Q3C
Database of astronomic catalogues in Cambridge University

~5 dbs, ~40 users, up to ~ $10^7$ queries per day, size 40Tb

pg 9.4 + q3c + hstore

Real time Detection of alerts in the Gaia
~10 dbs, 10 users, up to ~ $10^6$ queries per day, size 30Tb

pg 9.3 + synchronous replication + q3c

Robotic network telescopes by SAI MSU
8 observatories (5 in Russia, 3 outside)
total size ~100TB
pg 9.0-9.4 + pgsphere + replication
See:
PostgreSQL scalability on 4x18 x86

pgbench -s 1000 -j $n -c $n -M prepared -S on 4 x 18 cores Intel Xeon E7-8890 processors median of 3 5-minute runs with shared_buffers = 32GB, max_connections = 300
When I started using Postgres (Today)

Excellent scalability on 4-socket X86 machines
Support a wide range of workloads
  - High security data, relational, NoSQL, OLAP

Postgres-centric companies
  - 2ndQuadrant, EDB, Postgres Professional (roadmaps)

Postgres groups
  - NTT, Fujitsu, Amazon, Alibaba, Tencent,…

Postgres Development from
  - Community driven → Business driven

Postgres is used in mission-critical enterprise systems
There are many ways to help Postgres.
Core development
Development, review, testing, reporting bugs

Ecosystem
Extensions, drivers, ORM, monitoring tools... Postgres support in applications
Distributions, packages

Documentation
Improvement, translations, writing books, papers, ...blogging!

Meetings, Education
Creating of local communities, Conference, meetups, seminars, hackatons, educational and training courses

Use PostgreSQL!
Use Postgres in your company!

Sponsorship
Help development, sponsor community events.
For example: Improve documentation

- PG 11 — XML build!
- Why we don’t have figures yet?!  
  - Need community standards for diagrams, tools, draw diagrams
- Help beginners to learn PostgreSQL
  - Need friendly «PostgreSQL for beginners» with demo database
Chapter 65. GIN Indexes

65.1. Introduction

GIN stands for Generalized Inverted Index. GIN is designed for handling cases where the items to be indexed are composite values, and the queries to be handled by the index need to search for element values that appear within the composite items. For example, the items could be documents, and the queries could be searches for documents containing specific words.

We use the word *item* to refer to a composite value that is to be indexed, and the word *key* to refer to an element value. GIN always stores and searches for keys, not item values per se.

A GIN index stores a set of (key, posting list) pairs, where a *posting list* is a set of row IDs in which the key occurs. The same row ID can appear in multiple posting lists, since an item can contain more than one key. Each key value is stored only once, so a GIN index is very compact for cases where the same key appears many times.

Figure 65.1. GIN index
Appendix L. Demo Database “Airlines”

This is an overview of a demo database for PostgreSQL. This appendix describes the database schema which consists of eight tables and several views. The subject field of this database is airline flights in Russia. You can download the database from our website. See Section L.1 for details.

Figure L.1. Airlines in Russia

You can use this database for various purposes, such as:
- learning SQL language on your own
- preparing books, manuals, and courses on SQL
- showing features in stories and articles

When developing this demo database, we pursued several goals:
- Database schema must be simple enough to be understood without extra explanations.
- At the same time, database schema must be complex enough to allow writing meaningful queries.
- The database must contain true-to-life data that will be interesting to work with.

This demo database is distributed under the PostgreSQL license.

You can send us your feedback to edu@postgresqlpro.ru.

L.1. Installation

The demo database is available at edu.postgresqlpro.ru in three flavors, which differ only in the data size:
- demo-small.zip (21 MB) — flight data for one month (DB size is about 300 MB)
- demo-large.zip (30 MB) — flight data for two months (DB size is about 500 MB)
- demo-complete.zip (30 MB) — flight data for six months (DB size is about 800 MB)

L.2. Schema Diagram

Figure L.2. Bookings Schema Diagram

L.3. Schema Description

The main entity is a booking (bookings).

One booking can include several passengers, with a separate ticket (tickets) issued to each passenger. A ticket has a unique number and includes information about the passenger. As such, the passenger is not a separate entity. Both the passenger’s name and identity document number can change over time, so it is impossible to uniquely identify all the tickets of a particular person; for simplicity, we can assume that all passengers are unique.

The ticket includes one or more flight segments (ticket_flights). Several flight segments can be included into a single ticket if there are no non-stop flights between the points of departure and destination (connecting flights), or if it is a round-trip ticket. Although there is no constraint in the schema, it is assumed that all tickets in the booking have the same flight segments.
Books about PostgreSQL
BSD License, translate them!
Case 4: Trusted PostgreSQL

How to know your database is not compromised?

Credereum uses digital signature to sign transactions and Blockchain to store signatures to solve this problem.
Let we have functions:

\[ h(\text{private key, previous signature, data}) \rightarrow \text{signature} \]

\[ f(\text{public key, signature, previous signature, data}) \rightarrow \text{TRUE or FALSE} \]

Roles

- Owner DB
- Write User
- Read User
Verification

1. if (signature != sgnBC) = verification fails
2. verification = f(WriteUser.PublicKey, sgnBC, sgnBCP, row)
Optimize blockchain traffic

TrustedDB

\[
\text{BlockSignature} = \text{H(OwnerDB.PrivateKey, SignaturePrevious, signature[i] as data)}
\]
1. Check row <-> RowSignature (verifyicate row satisfied by RowSignature, i.e. they are both wrong or both truth, but choice of wrong/truth will done on next step)
   1. f(WriteUser.PublicKey, RowSignature, RowSignaturePrevious, row)
2. Block Verification (verification correctness of RowSignature)
   1. BlockSignature == sgnBC
   2. F(OwnerDB.PublicKey, sgnBC, sgnBCP, RowSignatures[i])
Several Postgres groups are working on

Postgres Distributed
Postgres Parallel
Postgres Asynchronous
Postgres Extendable+
Postgres NoSQL
Postgres Scalable
Postgres Trusted
Welcome to PGConf.ru 2018
February 5-7, Moscow
Lomonosov Moscow State University
ご清聴ありがとうございました