

PostgreSQL 9.1 - what's new

PGConf.EU 2011

Amsterdam, The Netherlands

Magnus Hagander
magnus@hagander.net
@magnushagander

PostgreSQL 9.1

- Released September 12th
- Yes, that means production ready
- *This is what you should be using!*
 - At least if you're building new



Many new features

- **Replication and Backup**
- Security
- Performance
- SQL and application functionality



Most probable app-breaker

- *standard_conforming_strings* is now **on by default**

- This means:

```
postgres=# select '0\'Brien';  
postgres' #
```

- Make sure you use:

```
postgres=# SELECT '0' 'Brien', E'0\'Brien';  
0'Brien | 0'Brien
```



Replication and backup

- Biggest features in 9.0 were (arguably):
 - Streaming Replication
 - Hot Standby
- Many rough edges
- Management and monitoring functionality based on real world experience



Replication and backup

- Replication is now a separate permission
- Superuser not required/recommended

```
postgres=# CREATE USER replica WITH REPLICATION;  
CREATE ROLE
```

- Superusers granted replication permission by default
 - Can be revoked



Replication monitoring

- View all replication sessions from master

```
postgres=# SELECT * FROM pg_stat_replication;  
... | 2011-02-07 12:52:20.141376+01 | STREAMING | 0/13000B70
```

- View transaction replay *timestamp* on slave

```
postgres=# SELECT pg_last_xact_replay_timestamp();  
2011-02-07 12:47:36.608706+01
```



Hot Standby conflict mgmt

- Query conflicts are the big issue with HS
 - Optional feedback loop added
- Monitors for query conflicts

```
postgres=# select datname, conflicts FROM pg_stat_database;
```

```
postgres |          79
```

```
postgres=# SELECT * FROM pg_stat_database_conflicts;
```

```
datname          | postgres
```

```
confl_tablespace | 1
```

```
confl_lock       | 3
```

```
confl_snapshot   | 74
```

```
confl_bufferpin  | 1
```

```
confl_deadlock   | 0
```



Streaming base backup

- Used for backups
 - No need to set `archive_command` in small deployments
 - No need for complicated scripts
 - No need for SSH/rsync/whatever access
 - Write to directory or native tarfiles
 - Just:

```
pg_basebackup -D /some/where -x
```



Streaming base backup

- Used for deploying replicas:
 - No need to use backups/log archiving
 - Single command deployment of slave
 - Just create *recovery.conf*
- Runs over libpq protocol
- Supports all authentication and encryption options
- Requires **REPLICATION** privilege and walsender



Detailed recovery control

- Ability to pause during recovery
- With hot standby, data can be reviewed
- Addition of “named restore points” during normal operation

```
SELECT pg_create_restore_point('before_stupid')
```



Synchronous replication

- Current solutions are asynchronous
- Sync often wanted for data security
- “Semi-sync” for decent performance
- Controllable *per transaction*
- Mix of sync and async fully supported



Many new features

- Replication and Backup
- **Security**
- Performance
- SQL and application functionality



Server auth on unix sockets

- Previously, peer could only be verified from server to client
- Now we can specify

dbname=foo **requirepeer=postgres**

- Avoids local attacks
- For TCP, use SSL certificate validation



SE-PGSQL

- Integrates with SE-Linux
- Label based security
- Umm. Yeah, go try it.



Many new features

- Replication and Backup
- Security
- **Performance**
- SQL and application functionality



More monitoring points

- **pg_stat_*_tables** added counters
 - Number of vacuum
 - Number of analyze
 - Differentiated by regular and background processes
 - Helps tuning autovacuum



More monitoring points

- `pg_stat_bgwriter` counts fsync requests
 - Detect when background processes aren't keeping up
 - fsync() by backends is *very* bad



Unlogged tables

- Create tables without writing to WAL
 - Considerable performance increase for large loading or changes
- Truncate on crash recovery
- *Not* included in log based replication
- No way (yet) to convert between logged and unlogged



KNN-GiST

- “ORDER BY for GiST”
- Fast, **indexed**, “K-Next-Neighbour” search
- Full awesomeness requires PostGIS 2.0
- For example, “the 10 graphical objects closest to this point”

```
SELECT * FROM t
ORDER BY pos <-> myposition
LIMIT 10
```



Many new features

- Replication and Backup
- Security
- Performance
- **SQL and application functionality**



SQL/MED

- “Managed External Data”
- Core parts completed and included!
- Table-like access to external data
 - Other PostgreSQL servers (“dblink”)
 - CSV files (without COPY)
 - Any other data sources (“Foreign Data Wrappers”)



Serializable Snapshot Isolation

- True **SERIALIZABLE** transactions
- Low overhead
- Old behavior still there as **REPEATABLE READ**
- Not supported on Hot Standby slaves



PK functional dependencies

- Functional dependencies on **PRIMARY KEYS** are recognized for **GROUP BY**
- No more

```
postgres=# SELECT uid,first,last FROM users GROUP BY uid;
```

```
ERROR: column "users.first" must appear in the GROUP BY  
clause or be used in an aggregate function at character 12
```

- Only recognizes PRIMARY KEY, not UNIQUE constraints or indexes



Per column collation

- Before 8.4, collation was per cluster
- Since 8.4, it's per database
- Now moving to per column
- One column English, another Danish
- Controls sort order and upper/lower

```
CREATE TABLE t (  
    a text,  
    b text COLLATE "sv_SE"  
)
```



Triggers on VIEWS

- INSTEAD OF triggers only
- Can be used to implement UPDATEable views
- Much nicer to work with than RULEs
- Gets the whole modified view row, figures out the rest



Writable CTEs!

- Nicer way to write “subqueries” for DML

```
WITH del_post AS (  
    DELETE FROM posts  
        WHERE created < now() - '6 months' RETURNING *  
)  
SELECT user_id, count(*) FROM del_post  
GROUP BY user_id
```

- “Anything” supported, joins etc
- Can even be made recursive!



Writable CTEs!

- Nicer way to write “subqueries” for DML

```
WITH del_post AS (  
    DELETE FROM posts  
    WHERE created < now() - '6 months' RETURNING *  
),  
per_user AS (  
    SELECT user_id, count(*) FROM del_post  
    GROUP BY user_id  
)  
UPDATE counts c  
SET post_count = post_count - per_user.count  
FROM per_user WHERE per_user.user_id = c.user_id
```



Extensions

- Wrap extensions (contrib, postgis etc)
- Distinct objects containing schema items
- Controlled dump/reload/upgrade

```
CREATE EXTENSION pgcrypto;
```

```
ALTER EXTENSION pgcrypto  
    UPGRADE TO <newversion>;
```



Thank you!

Questions?

<http://2011.pgconf.eu/feedback/>

Twitter: @magnushagander
<http://blog.hagander.net/>
magnus@hagander.net

