Multi-tenancy with PostgreSQL

Robert Ivens, ROCLASI Software Solutions

JBS Group, Partner

Overview

- Data partitioning strategies
- Sequence vs UUID
- Backup/Restore
- Managing Schema changes
- How can PostgreSQL help

Why Multi-tenancy

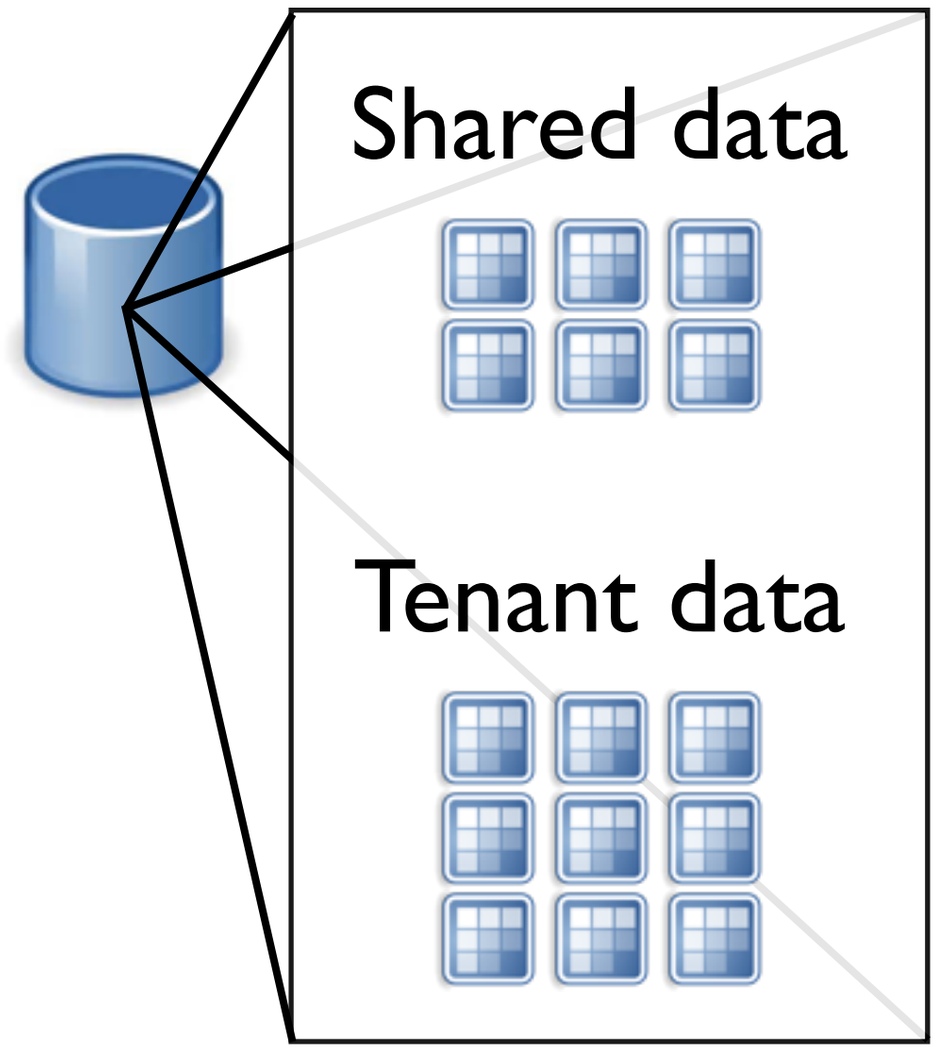
- Separation of customer data (SaaS)
- Separating data of divisions/departments within a company
- Single code base on a single server

Data partitioning strategies

Data partitioning strategies

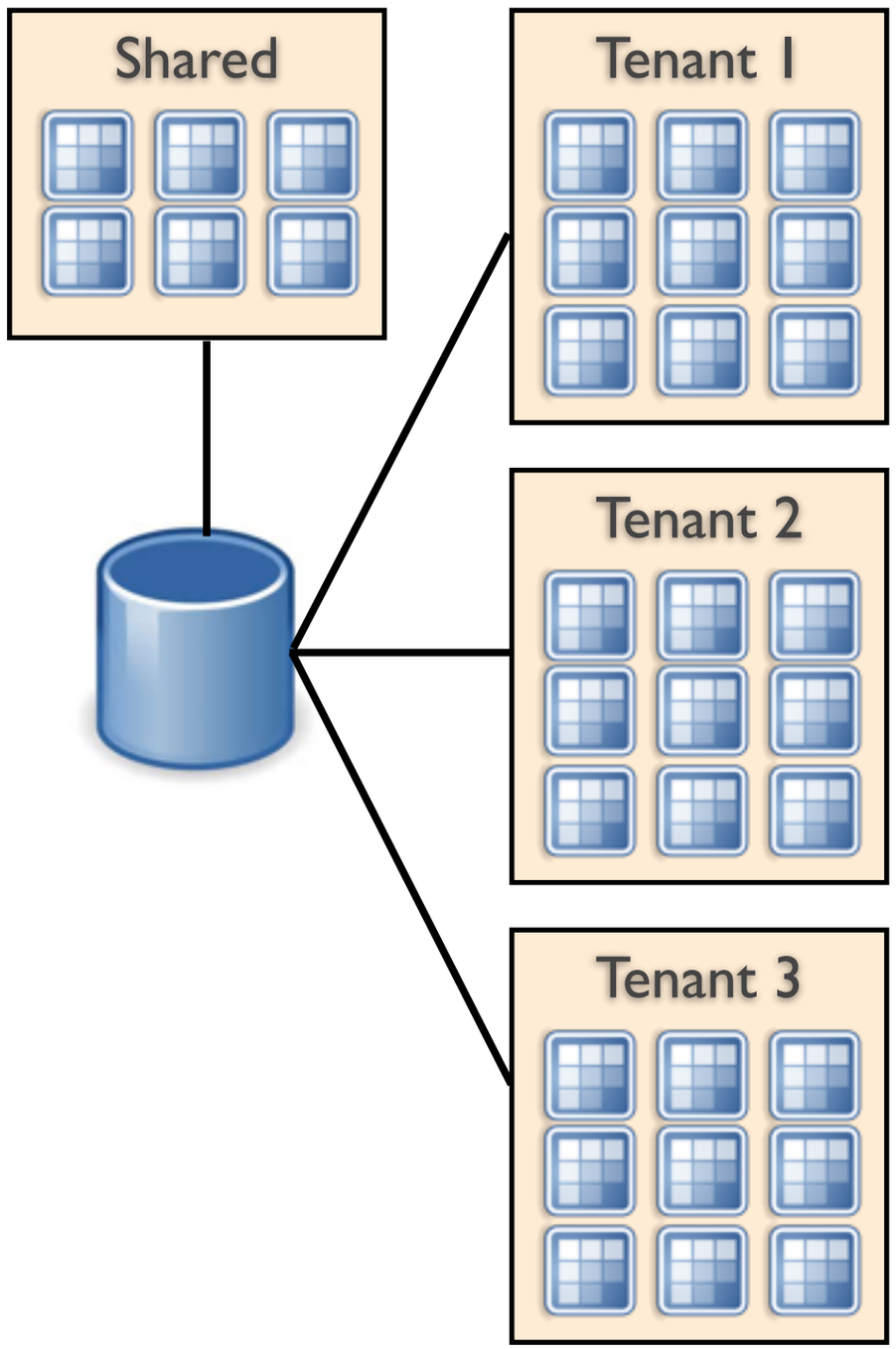
- Single database
- Single database with multiple schemas
  - Schema is a logical database
- Multiple databases
- Single database with partitioned tables
- Hybrid of the above options

Single database



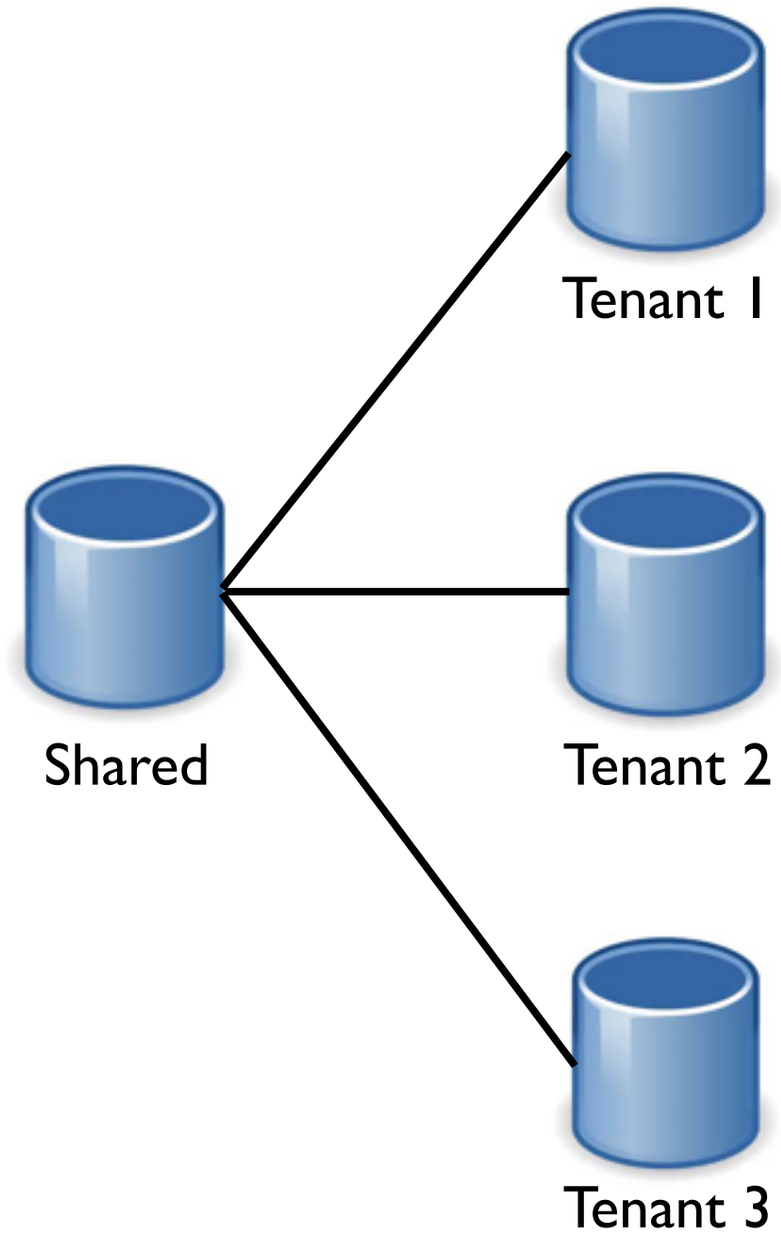
- Pros
  - Instant adding of new tenants
  - Quick schema changes
  - Easy relating shared- with tenant data
- Cons
  - Large Index sizes
  - Restoring data of single tenant
  - No real separation of data
  - Data access by tenant

Multiple schemas



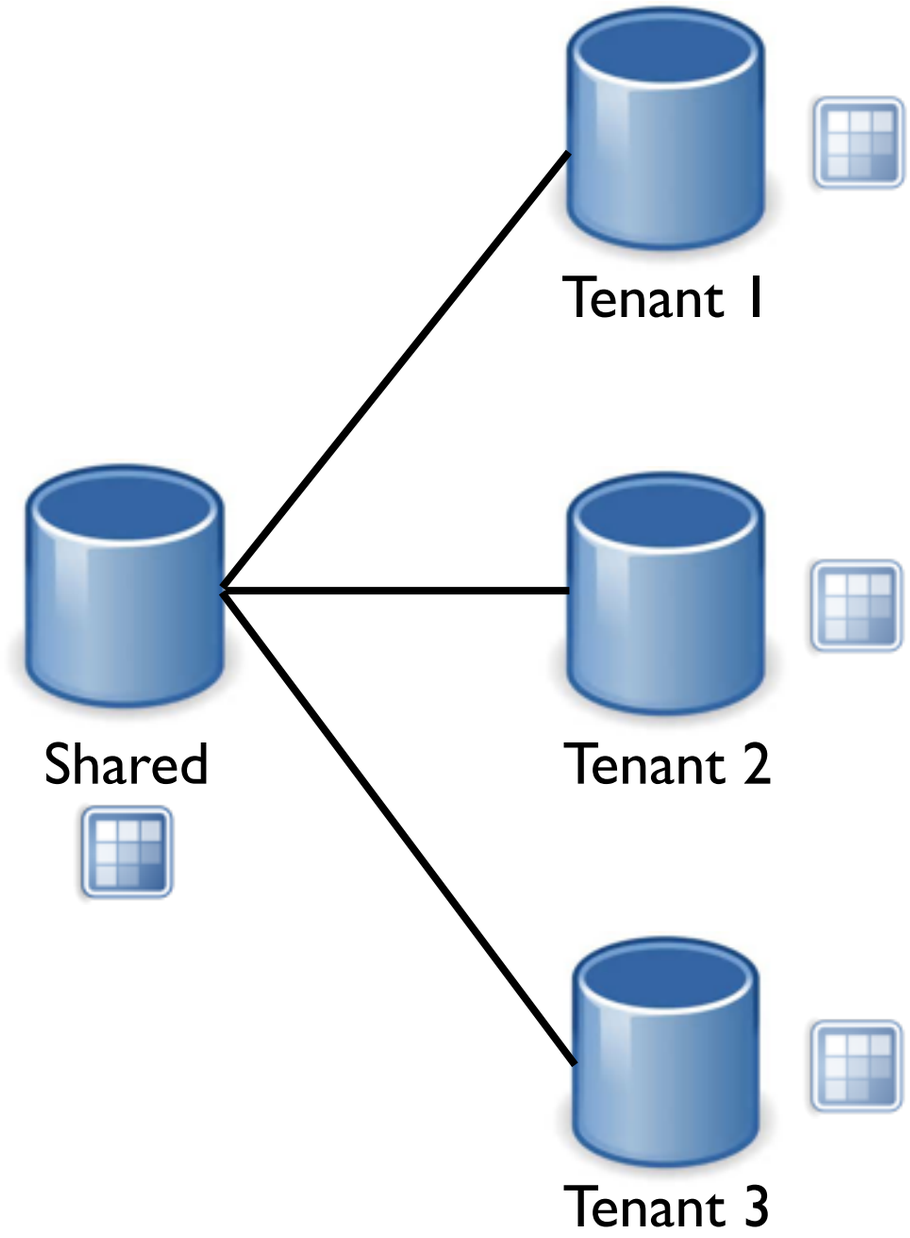
- Pros
  - True separation of data
  - Small index sizes
  - Easy relating tenant- with shared data
  - Easy restoring of data
  - Data access by tenant (DB-level security)
- Cons
  - Schema changes
  - Adding new tenants

Multiple databases



- Pros
  - True separation of data
  - Small index sizes
  - Easy restoring of data
  - Data access by tenant (DB-level security)
- Cons
  - Schema changes
  - Adding new tenants

Multiple databases cont.

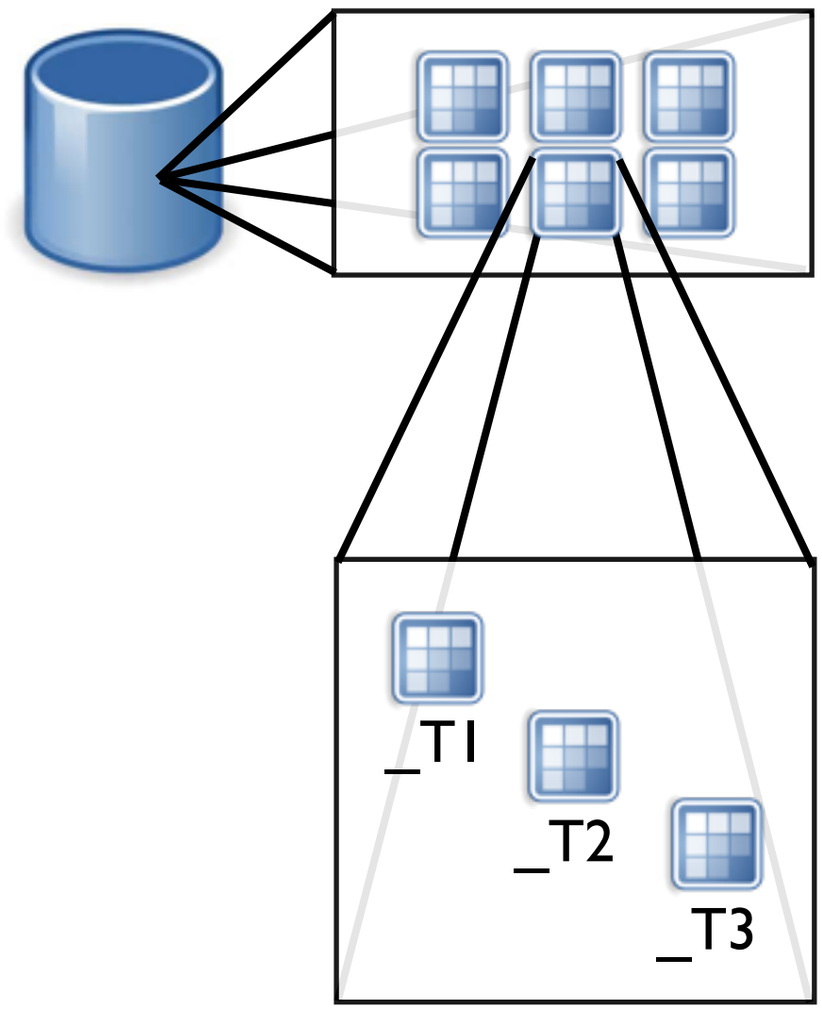


Relating shared data

- Foreign Data Wrappers (FDW)
  - PostgreSQL extension
- One direction only

http://www.postgresql.org/docs/current/static/postgres-fdw.html

Multiple tables (sharding)



- Pros
  - all benefits of a single DB
  - Small index sizes
  - Easy restoring of data
- Cons
  - schema changes
  - solutionModel to re-target forms/relations/valuelist (when needed)

Data partitioning strategies

- foundset.**addFoundsetFilterParam**(*dataprovider*, *operator*, *value*)
- databaseManager.**addTableFilterParam**(*server*, *table*, *dataprovider*, *operator*, *value*, *filterName*)
  - set table param to match ALL tables within connection
- databaseManager.**switchServer**(*originalServer*, *tenantServer*)
  - requires to be the same DB type

Data partitioning strategies

- Schema changes
  - Mark connections as ‘clone from...’ on the tenant connections
  - Servoy will push all changes to all clones connections
  - Using your own schema change script can be faster
  - Servoy won’t create indexes/foreign key constraints/ etc. for you
  - When changed outside of Servoy it requires a restart of the Servoy server

Data partitioning strategies

Sequence vs UUID

Sequence vs UUID

- UUID is globally unique
- Sequence (Big Integer) is locally unique
- UUID's are 'more expensive'
  - CHAR(36) / VARCHAR(36) *(36 bytes)*
  - Big integer (*8 bytes)*
  - Integer *(4 bytes)*
  - PostgreSQL: Native UUID datatype *(16 bytes)*

Sequence vs UUID

```
   Column    |          Type
-------------+------------------------
 id          | uuid
 id_varchar  | character varying(36)
 id_char     | character(36)
 id_int      | bigint
```

Shows in Servoy as:

Name	Type	Length
id	TEXT	36
id_varchar	TEXT	36
id_char	TEXT	36
id_int	INTEGER	0

Sequence vs UUID

Column	Type
id	uuid
id_varchar	character varying(36)
id_char	character(36)
id_int	bigint

Index sizes over 100K records

Name	Type	Table	Size
id_varchar_idx	index	uuid_test	5792 kB
id_char_idx	index	uuid_test	5792 kB
id_uuid_idx	index	uuid_test	3104 kB
id_int_idx	index	uuid_test	2208 kB

Sequence vs UUID

Questions