Migrations to PostgreSQL

(from Oracle)

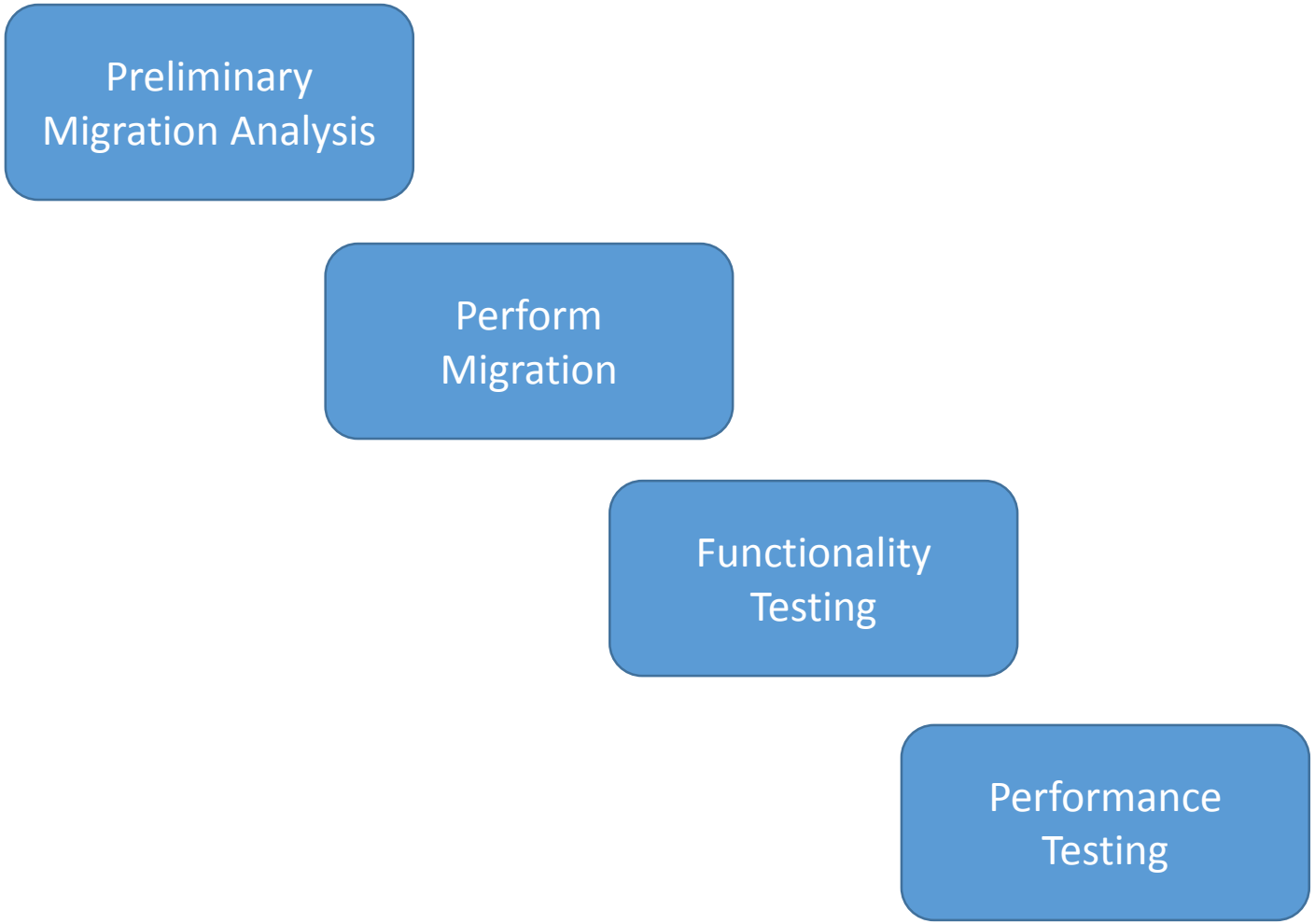
I am : Venkata B Nagothi

A PostgreSQL consultant for the past 6+ years

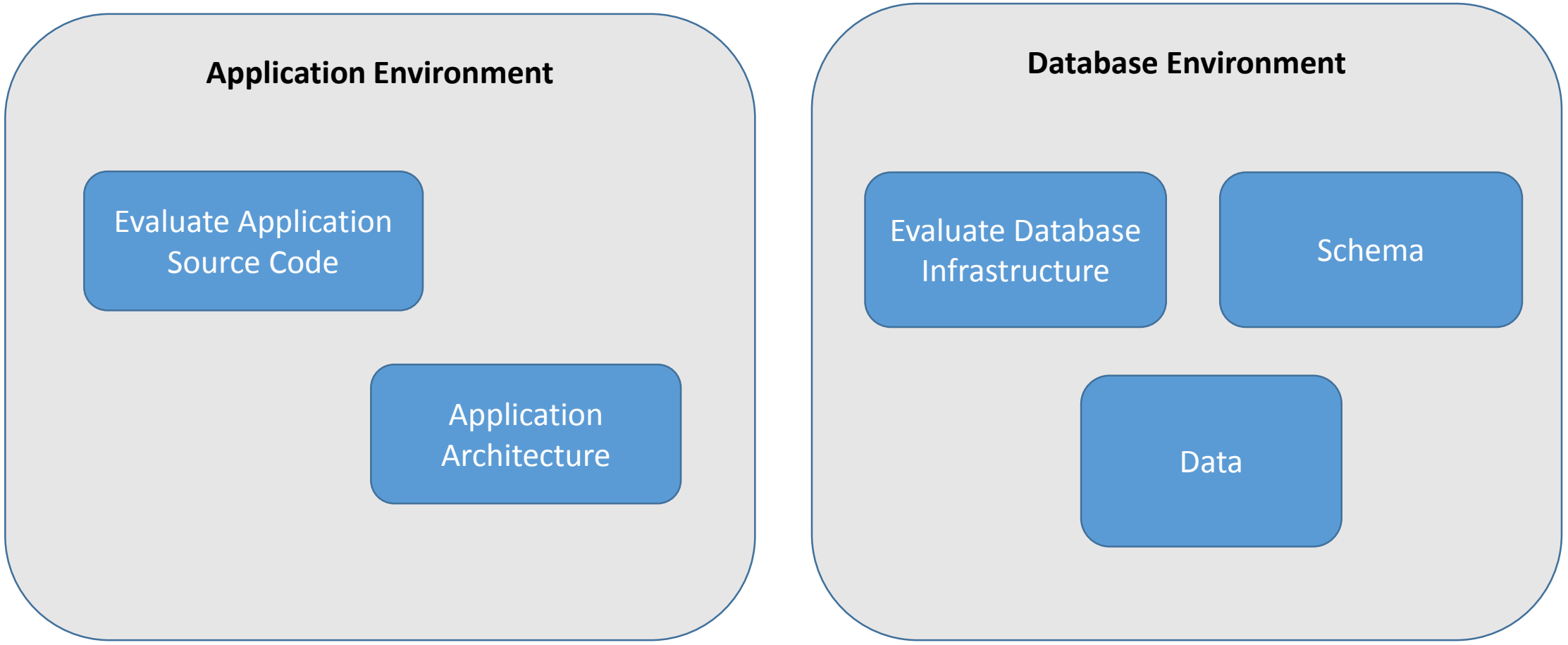
Why migrate to PostgreSQL ?

- Cost-effective and feature-rich open-source database
- PostgreSQL Community
- Integration capabilities
- Can replace any commercial database
- Cloud adoption

Migration process - overview



Preliminary Migration Analysis



Preliminary Migration Analysis

- Major roadblocks for migration
  - Heavy PL/SQL usage
  - Heavy usage of Large Objects
  - Propriety application with its own schema
  - Application compatibility issues
  - No application source code
- Evaluate database migration effort
  - Amount of manual migration effort needed
- Evaluate application migration effort
  - This is critical as the Application code is change is mostly manual

Migration challenges

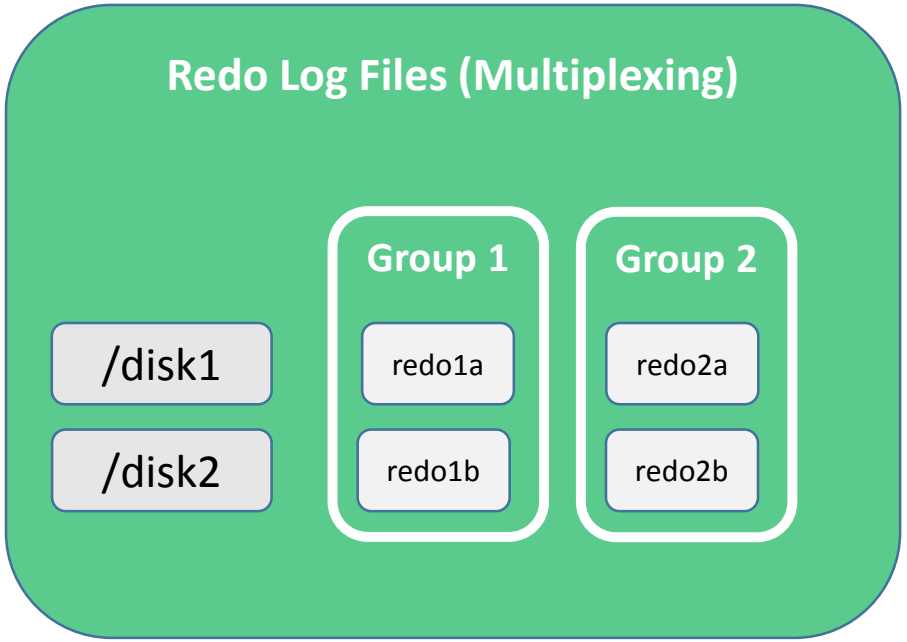
- Database Design / Architecture challenges
- High Availability challenges
- Development Challenges
- Data Migration challenges

Database Design / Architectural challenges

Database Design / Architecture

Transaction Log Files

Oracle



PostgreSQL

WAL Files (No Multiplexing)

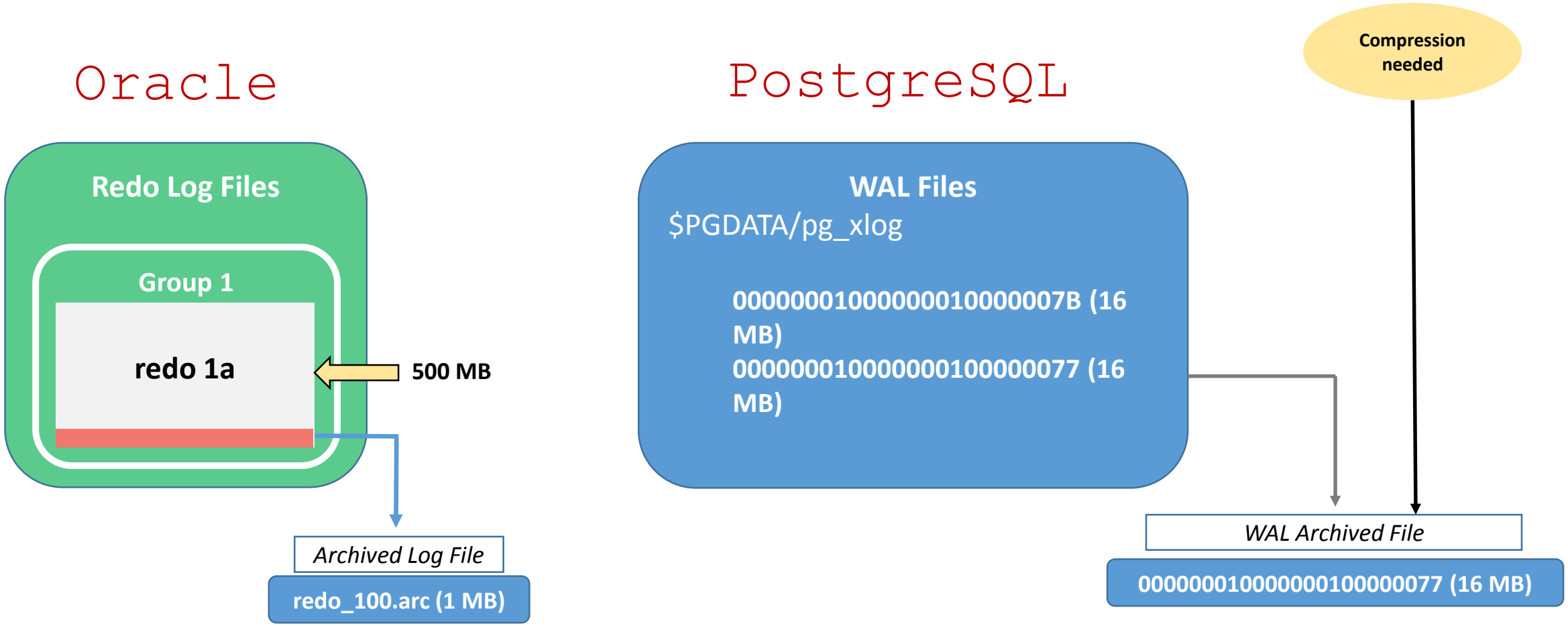
$PGDATA/pg_xlog

000000010000000100000078
000000010000000100000077
000000010000000100000078
000000010000000100000072
000000010000000100000079

- Database will hang if the pg_xlog Disk space is full
- Each WAL file size is 16 MB
- No multiplexing
- No hard limit on the number of files
- I/O balancing is needed

Database Design / Architecture

Archived Log files



Database Design / Architecture

Data files

- Storage is Directory Bound
- Data files are auto generated
- DBA has no control over data file management

Oracle

Data Files

/disk1/datafile1
/disk2/datafile2
/disk3/datafile3
/disk4/datafile4

PostgreSQL

Data Files
(directory)

13157_fsm	13172_fsm	13189
13157_vm	13172_vm	13191
13159	13174	1417
13161	13176	1417_vm
13162	13177	1418

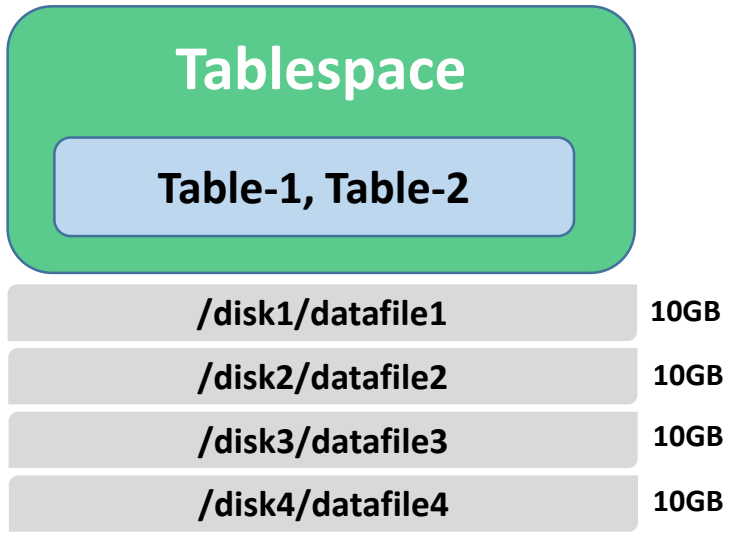
Database Design / Architecture

Data File structure

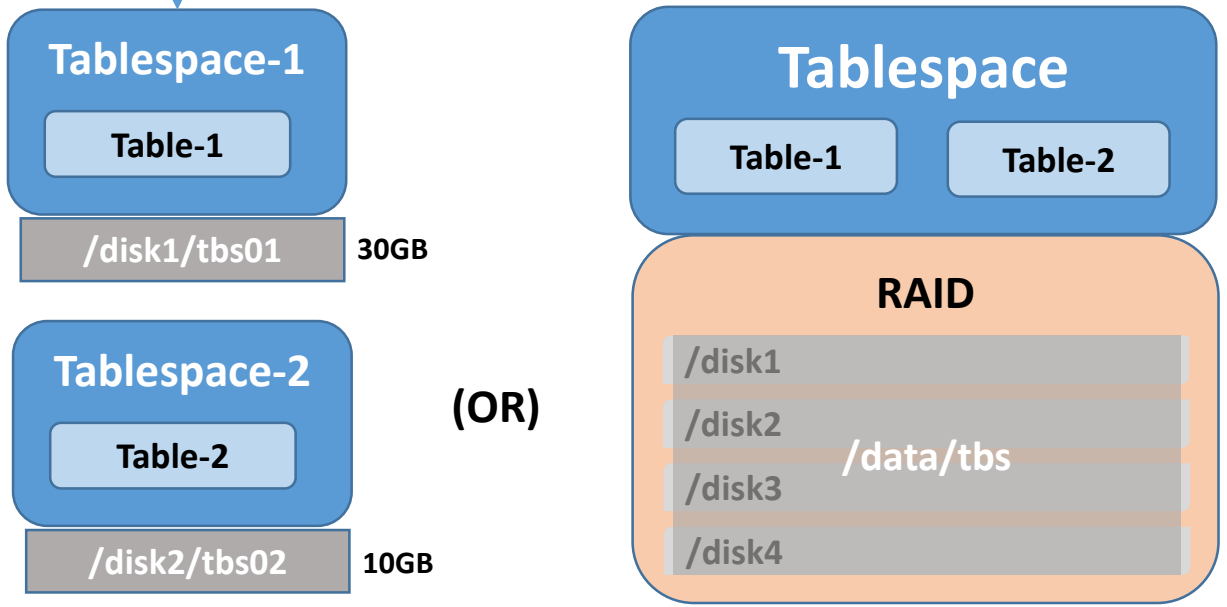
Table-1 = 30 GB
Table-2 = 10 GB

I/O Balancing is a challenge

Oracle



PostgreSQL



Database Design / Architecture

Control File

Oracle

Control File (Multiplexing)

/disk1/oradata/control01.ctl
/disk2/oradata/control02.ctl

PostgreSQL

pg_control file (No Multiplexing)

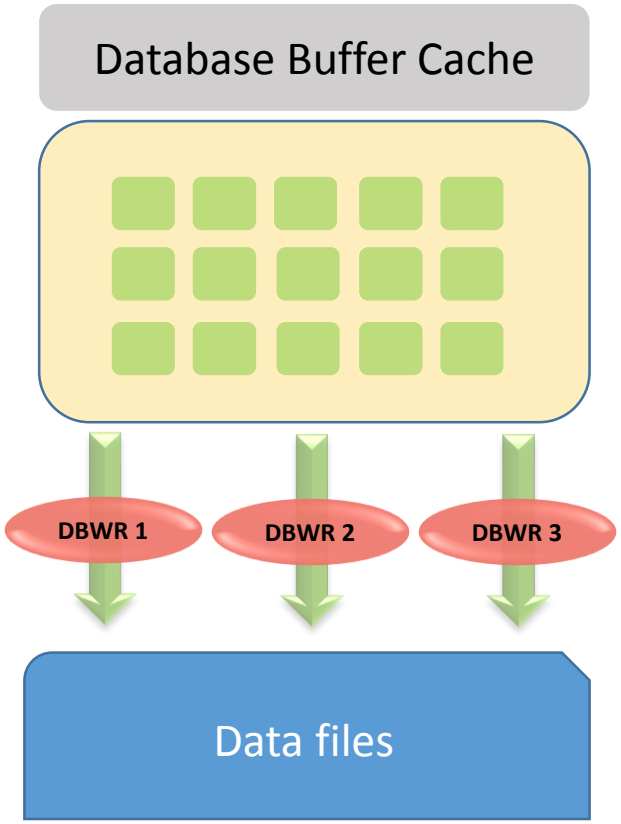
$PGDATA/global/pg_control

- Loosing pg_control file will result in incomplete recovery

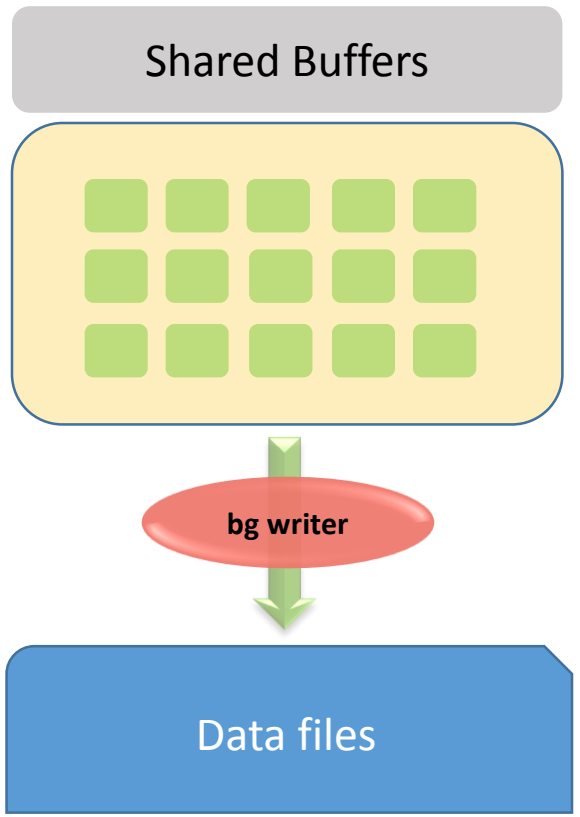
Database Design / Architecture

bg-writer

Oracle



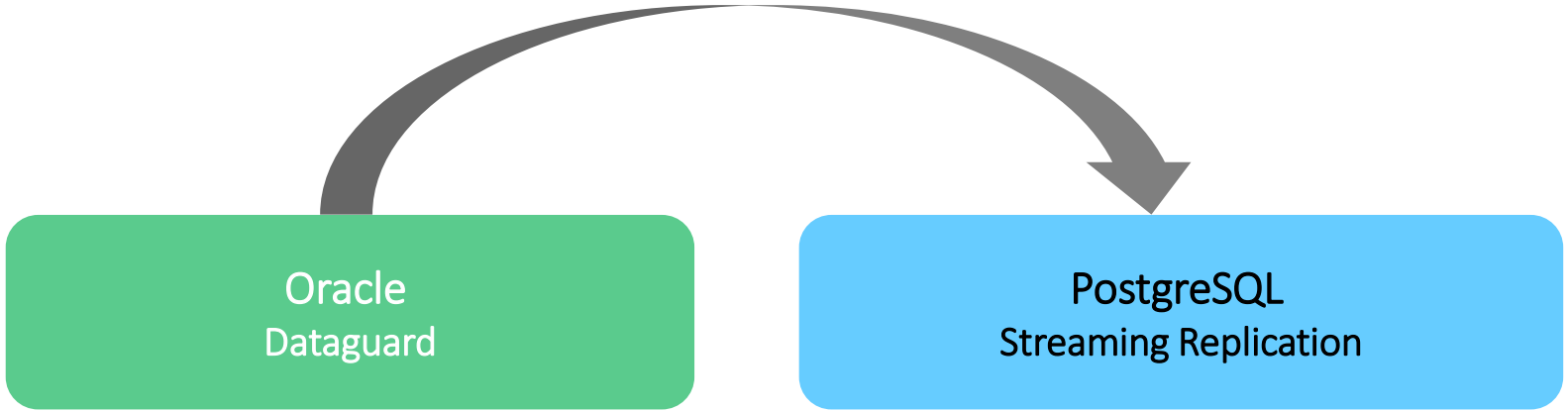
PostgreSQL



It would be good to have multiple bg-writers for better write performance and scalability in an high-transaction multi-CPU environment

High Availability challenges

High Availability

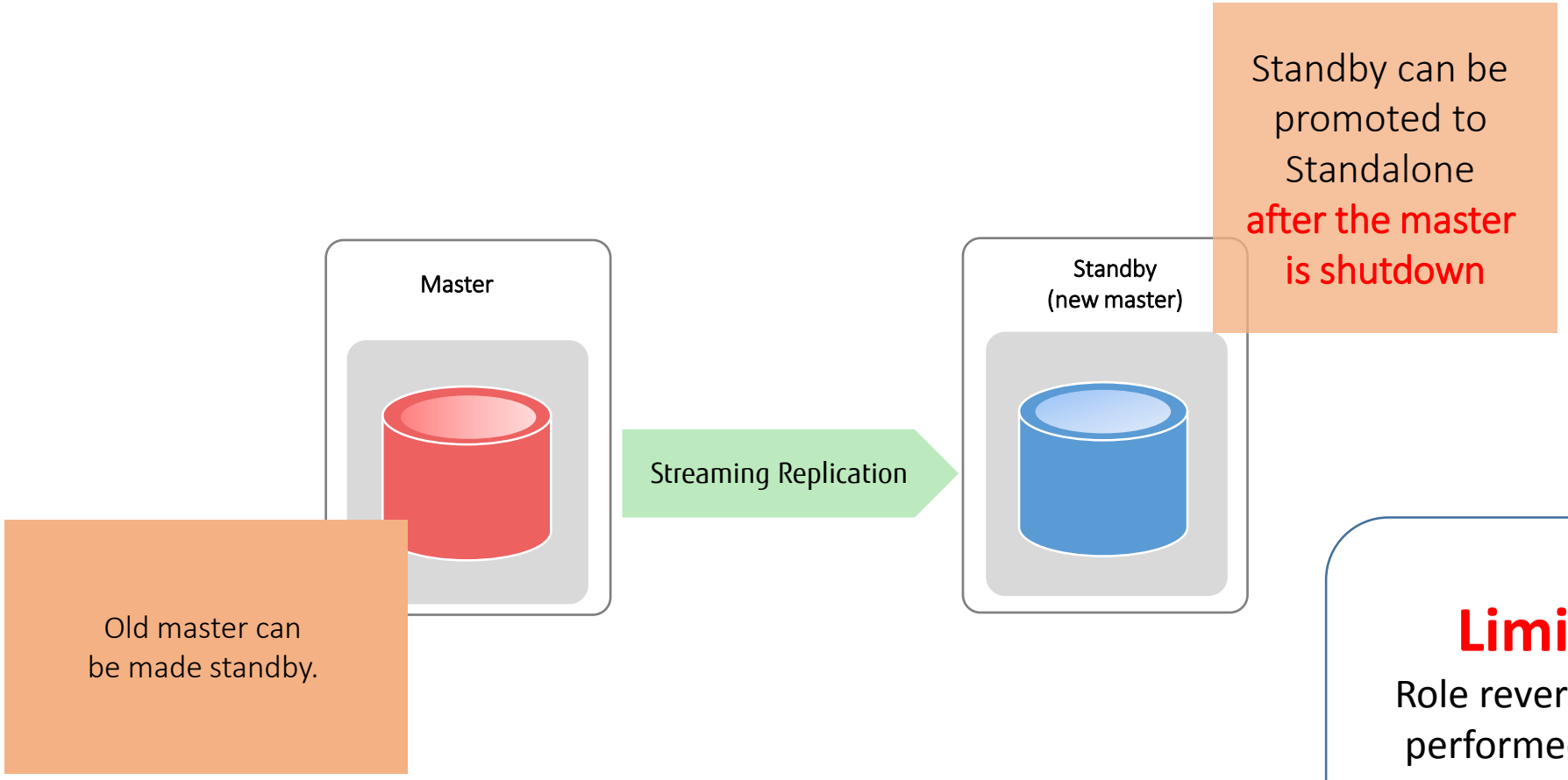


Migration Impact (Streaming Replication)

- More simpler to implement
- Supports all the protection modes in Oracle
- Supports cascading replication with some limitations

High Availability

Role reversal (for disaster recovery) – Limitations in PostgreSQL



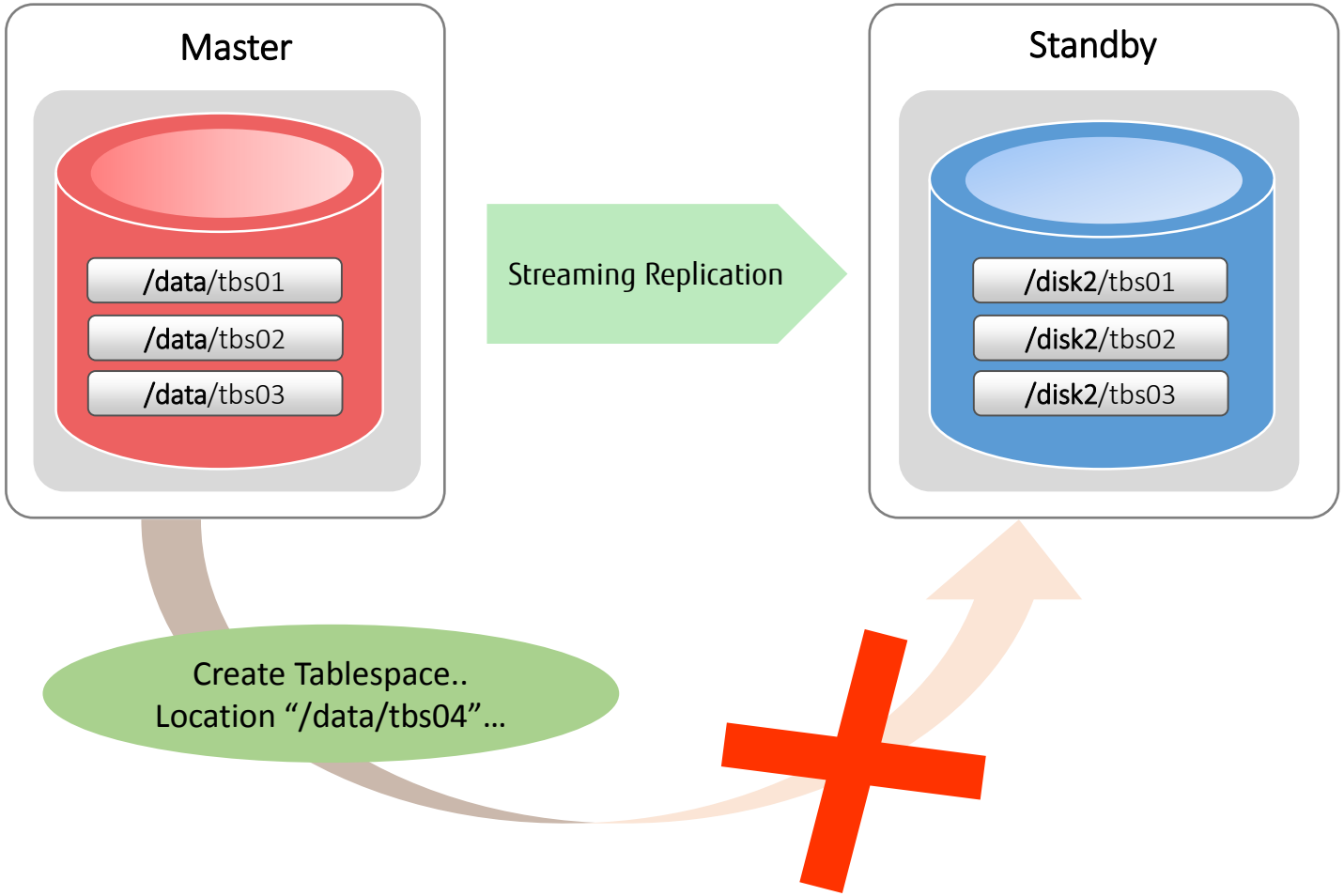
Limitation

Role reversal cannot be performed without an Outage to the master.

High Availability

PostgreSQL Streaming Replication – Limitations

- Standby can be built on a different filesystem using pg_basebackup



Development challenges (database)

Database migration

ora2pg

- ora2pg is the most open-source tool used for migrating the Oracle database to PostgreSQL.
- Database migration time and cost can be evaluated by identifying the manual migration effort needed
- Database migration can be in three phases
  - Schema migration
  - PL/SQL migration
  - Data migration

Schema Migration

- Most of the Schema migration can be done automatically using ora2pg.
- The Oracle database objects not supported by PostgreSQL must be identified and must be migrated manually.

Database migration

Schema Migration

Unsupported Object	PostgreSQL Alternative solution
Materialized Views	Auto refresh and query re-writing is not supported
Index-Organized-Tables	This requirement can be partially fulfilled by Clustering a Table.
Public Synonyms	Public Synonyms are not supported in PostgreSQL. “search_path” can be used as an alternative
Global Temporary Tables	There is no support for Global Temporary Tables. Unlogged Tables can be used instead.
Partitioned tables	Partitioned Tables in PostgreSQL are not quite similar to Oracle style partitioning and cannot be used as an alternative. • Child tables are more like individual tables • Constraints are not inherited to child tables • No support for Global and Local indexes

PL/SQL Migration

- Ora2pg partially migrated PL/SQL objects
- As PostgreSQL does not support objects like Packages, most of the PL/SQL objects must be migrated manually

Database migration

PL/SQL Migration

Unsupported PL/SQL Object	Alternative PostgreSQL Solution
Packages	Packages are unsupported in PostgreSQL.
DBMS* Packages	Some of the DBMS packages are supported by orafce external contrib module. The unsupported packages cannot be migrated. A custom function must be built if required.

PACKAGES	PostgreSQL Alternative
PACKAGE DEFINITION	SCHEMA must be used as an alternative for Package definitions
GLOBAL VARIABLES	TEMP TABLES must be used as an alternative for GLOBAL VARIABLES
PACKAGE BODY	Package Body must be converted to FUNCTION(S)

- Application functionality testing is critical and can poses challenges
- Reaching the expected Performance benchmark can be a challenge too.
- There could be a need to re-write the whole business logic to full fill the application requirements.

Database migration

PL/SQL Cursors

PostgreSQL Cursors	Limitations
ROWCOUNT attribute is not supported	MOVE LAST must be used to fetch the count of the rows in a cursor. This could be a significant performance penalty when cursor has millions of rows.

Cursor position

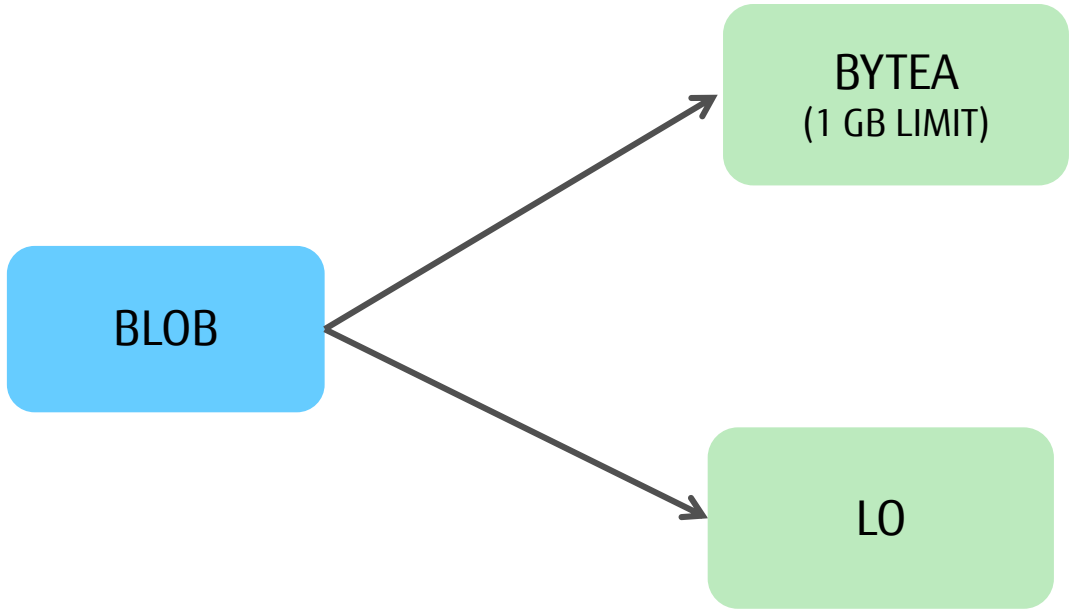
1
2
3
4
5 <- Current cursor position
.
.
.
.
.
1 billion <- MOVE ALL

Cursor position must be moved to last row of the cursor to fetch the count.

Data Migration

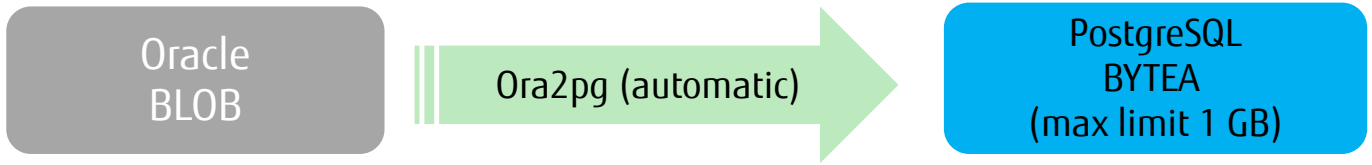
- PostgreSQL is rich in data-type
- Data migration can be performed using Ora2pg which is very efficient
- However, significant challenges can be encountered while migrating **Large objects** and **JSON** data

Migration Large Objects



Database migration

Migrating Large Objects



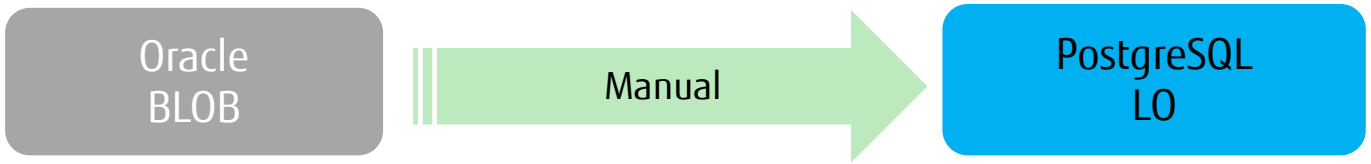
Migration Impact

- Oracle BLOBs are automatically migrated to PostgreSQL bytea by ora2pg
- BYTEA cannot be streamed. This can lead to unstable Application performance which would have a negative impact on the benchmarking metrics
- Can accommodate more than 1 GB due to PostgreSQL compression algorithm
- Triggers an Application code change
- BYTEA into the memory at a time which will impose performance problems resulting in excessive usage of Memory and Network bandwidth

BYTEA Column	Size
Text File	10K
Text File	10K
Image	20M
Image	300M
Text file	100K

Database migration

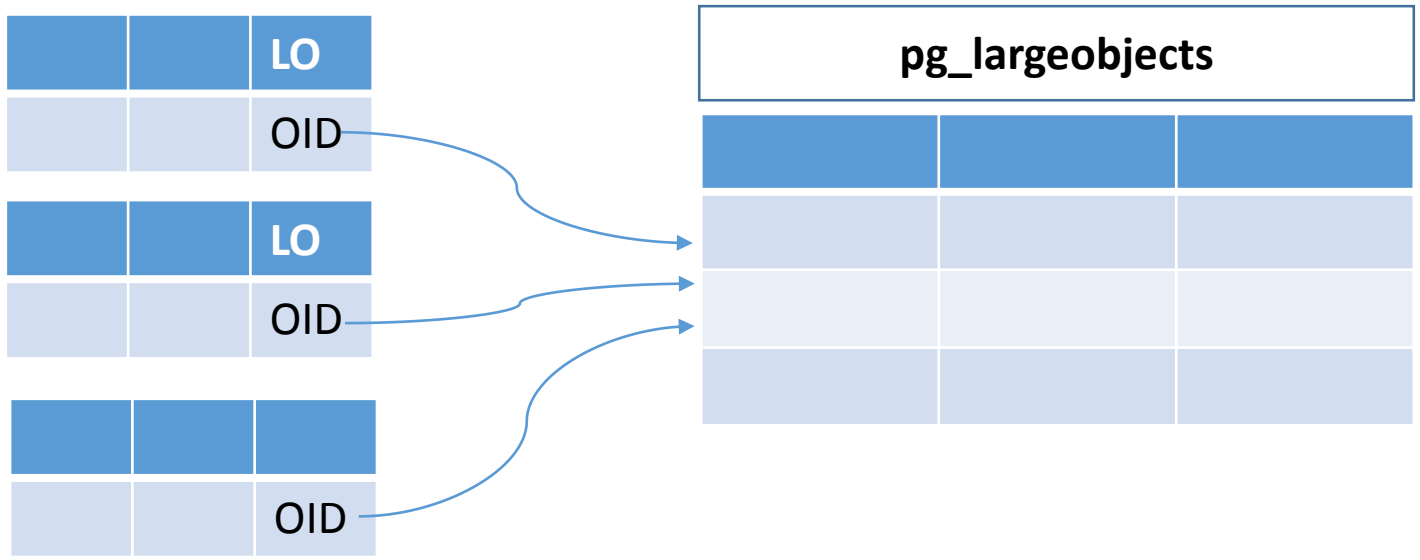
Migration Large Objects



Migration Impact
• LO objects can be streamed in multiple chunks resulting in effective performance • Custom ETL scripts must be built • Migration effort and time can be a challenge • Application code change is expected • Application functionality and Performance testing

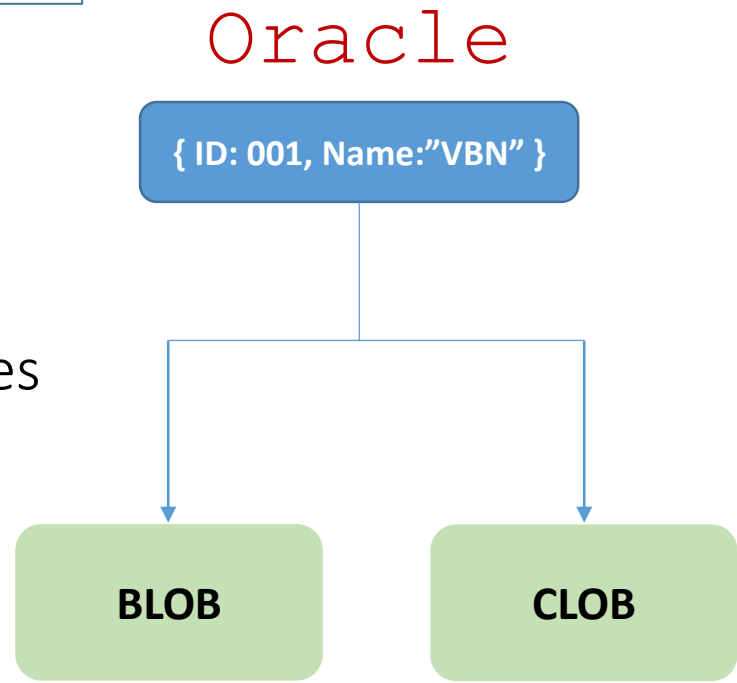
Pg_largeobjects - Limitation

- pg_largeobjects is used to store the large objects in PostgreSQL
- There is a serious limitation around this.
- All the Large objects of all the tables in the database are stored in a single table called pg_largeobjects. This can turn out to be a serious design flaw and can be a significant performance bottleneck.



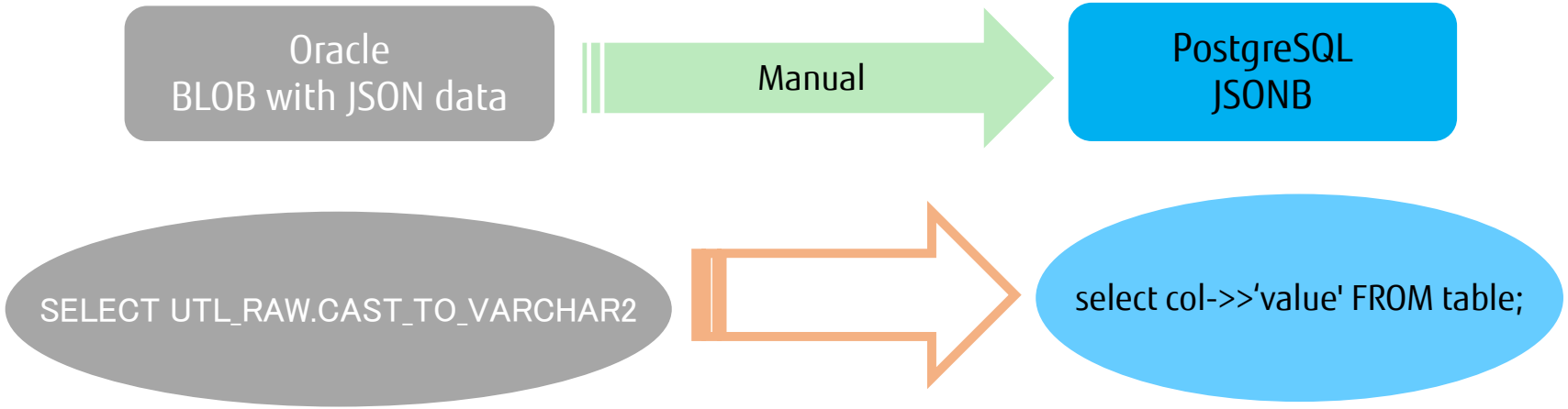
Migrating JSON Data

- Generally, Oracle uses BLOB or CLOB to store JSON data
- Technically, JSON is treated as a Large object
- As PostgreSQL's support towards JSON is powerful and efficient, JSON data can be migrated to PostgreSQL's JSON/JSONB data types
- Unfortunately, this is a manual and time consuming process. Significant application code change is also required.



Database migration

Migrating JSON Data

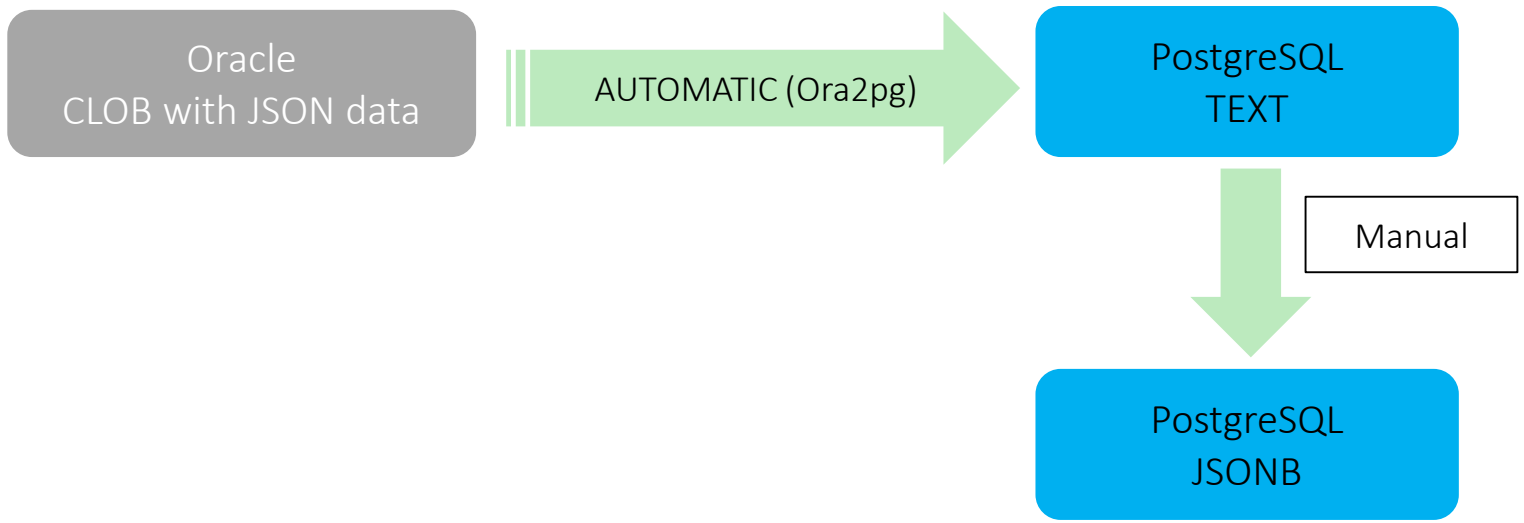


Migration Impact

- BLOB migration to JSONB cannot be done directly and no ETL can make this possible
- Can result in an heavy Application design and code change
- Migration effort and time can be a challenge
- Heavy development can be on the cards

Database migration

Migrating JSON Data



Migration Impact
• Can result in an heavy Application design and code change • Migration effort and time can be a challenge

Development challenges
(Migrating Oracle SQLs for Application)

SQLs Migration for application

- Application migration is the most critical and challenge episode of whole migration project
- Migration SQLs in the Application source code is very critical and can take up majority of the migration time.
- Application source code must be analysed to identify the SQLs that needs to be changed which is not an straight forward way unfortunately.
- A single SQL syntax change can trigger a change at hundreds of places in the application source code files

SQLs Migration for application

Hierarchical queries

Oracle	PostgreSQL
CONNECT BY START WITH SYS_CONNECT_BY_PATH CONNECT_BY_ROOT	• WITH RECURSIVE is a straight alternative for migrating Hierarchical queries • Depending on the complexity of the query logic, connectby() function part of tablefunc contrib module can be of use • **pl-pgsql functions is another alternative** if the Hierarchical queries cannot be converted using WITH RECURSIVE alone

SQLs Migration for application

Migration Impact

AUTOCOMMIT OFF

- Application behaviour is different in **autocommit off** mode
- Multiple transactions will be automatically part of a transaction block, which means COMMIT ALL or NONE
- Heavy application code change may be required
- Functionality testing
- Performance testing
- Changing legacy code might impose further more challenges

SQLs Migration for application

- Implicit type casting can be highly beneficial and can help reduced the need to change the application code to a greater extent
- Increases the possibility usage of Indexes
- Avoids the need to create function-based Indexes

Table

varchar	varchar
2	Sydney
3	London
4	Singapore

Oracle

```
select * from table where col1 > 2;
```

3	London
4	Singapore

PostgreSQL

```
select * from table where col1::int > 2;

create cast(varchar as integer) with inout as implicit;

select * from table where col1 > 2;
```

3	London
4	Singapore

Questions ?