PostgreSQL

Introduction for Oracle DBAs

Mathias Zarick, Vienna, 22.02.2019

@Trivadis

Introduction – Mathias Zarick

- Principal Consultant at Trivadis Delphi GmbH in Vienna
- Graduated from University of Rostock / Computer Science
- Trainer
  - Data Guard, Architecture and Internals for advanced DBAs, GoldenGate, Grid Infrastructure
- Main focus:
  - Oracle database … NoSQL DBs: Mongo DB
  - Oracle high availability projects (Real Application Clusters, Data Guard, Maximum Availability Architecture, Replication with Streams and GoldenGate)
  - Backup/Recovery
  - Trivadis Toolbox Architect
  - Developer of TVD-Standby
  - Research projects in Trivadis Technology Center (TTC)

ORACLE Certified Master
Oracle Database 11*g* Administrator

ORACLE Certified Master
Oracle Database 12*c* Administrator

PostgreSQL newbie

trivadis makes IT easier.

Agenda – PostgreSQL

1. **Introduction**
2. **Support, License, Cost**
3. **Architecture**
4. **Some Comparisons with Oracle by Example**
5. **Conclusion**

Introduction

What is PostgreSQL?

- MySQL is most popular open source database
- PostgreSQL is most advanced open source database
- Developed for more than 20 years
  - University Postgres (1986 – 1993)
  - Postgres95 (1994 – 1995)
  - PostgreSQL (1995 – today)

Major Features of PostgreSQL (1)

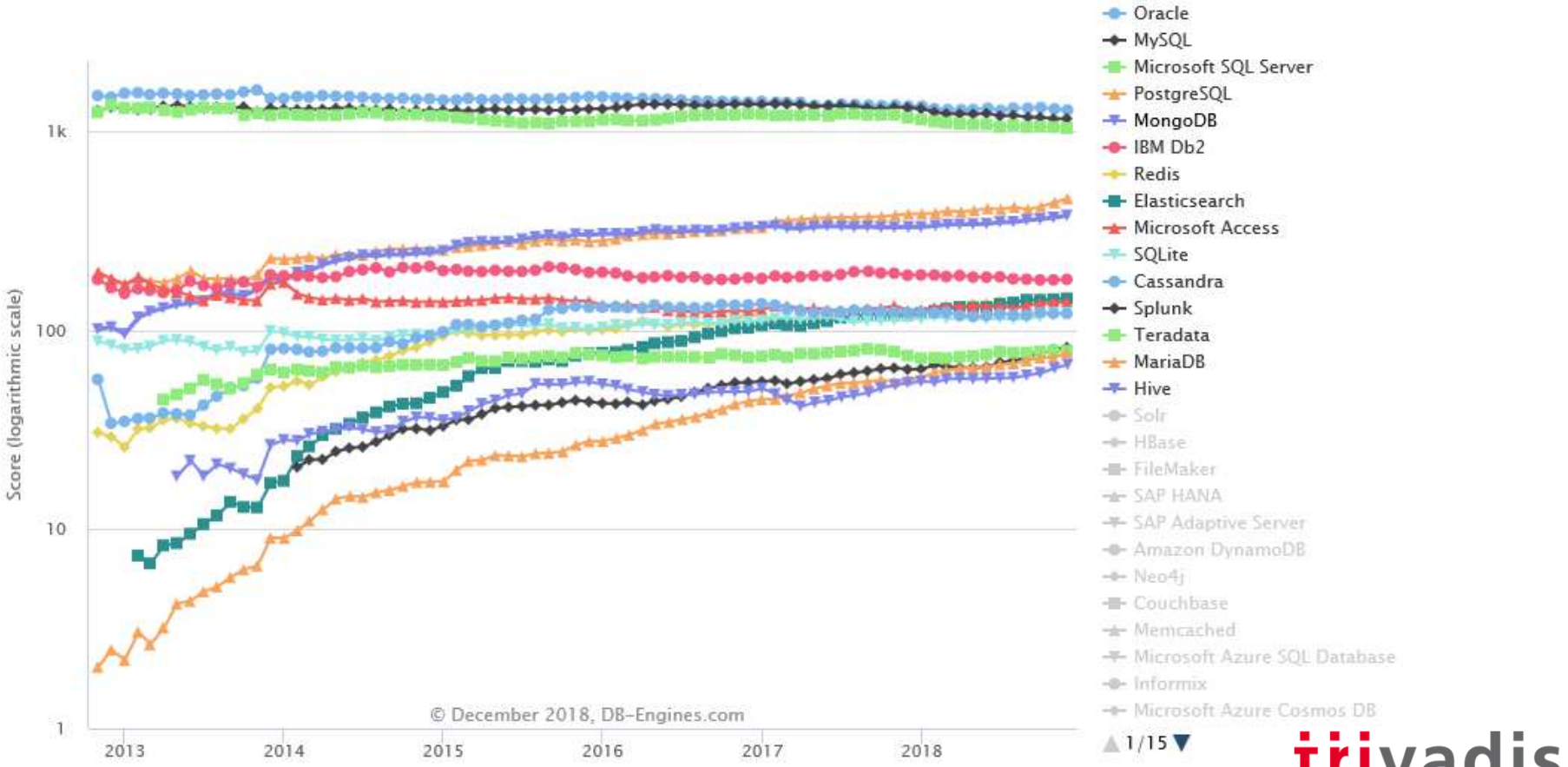
- Portable (Windows, Linux, Mac OS/X and UNIX)
- ACID compliant relational database
- Is scalable because it has important performance features like
  - Table partitioning
  - Parallel execution
- Has advanced security features:
  - Host based access control
  - Various authentication methods (Kerberos, password, certificates, ldap, …)
  - Row level security
  - SSL connectivity

Major Features of PostgreSQL (2)

- Triggers / functions
  - PL/pgSQL similar to Oracle PL/SQL
  - PL/Tcl, PL/Perl, PL/Python
  - C
- Backup and Recovery (see later)
  - Point-in-time
  - Hot backups
- High availability
  - Log-shipping standby server (async, sync, hot standby (read-only while applying))
  - Logical replication

DB-Engines Trend

Source: https://db-engines.com/en/ranking_trend



DB-Engines Ranking

Source: https://db-engines.com/en/ranking

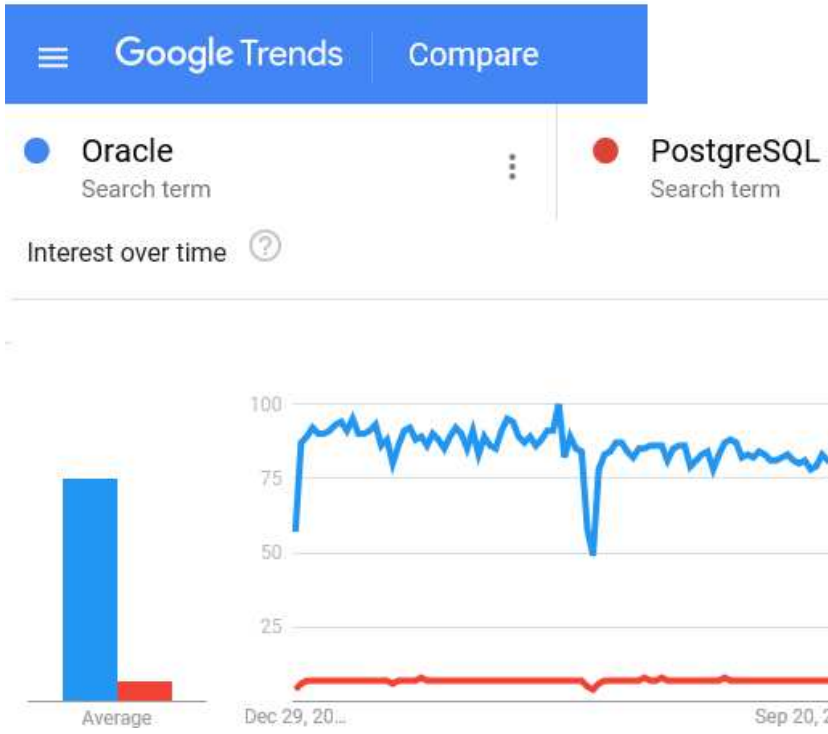
The DB-Engines Ranking ranks database management systems according to their popularity.
The ranking is updated monthly.

341 systems in ranking, December 2018

Rank			DBMS	Database Model	Score		
Dec 2018	Nov 2018	Dec 2017			Dec 2018	Nov 2018	Dec 2017
1.	1.	1.	Oracle	Relational DBMS	1283.22	-17.89	-58.32
2.	2.	2.	MySQL	Relational DBMS	1161.25	+1.36	-156.82
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1040.34	-11.21	-132.14
4.	4.	4.	PostgreSQL	Relational DBMS	460.64	+20.39	+75.21
5.	5.	5.	MongoDB	Document store	378.62	+9.14	+47.85
6.	6.	6.	IBM Db2	Relational DBMS	180.75	+0.87	-8.83
7.	7.	↑ 8.	Redis	Key-value store	146.83	+2.66	+23.59
8.	8.	↑ 10.	Elasticsearch	Search engine	144.70	+1.24	+24.92
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	139.51	+1.08	+13.63
10.	10.	↑ 11.	SQLite	Relational DBMS	123.02	+0.31	+7.82

The world is changing fast

Google Trends



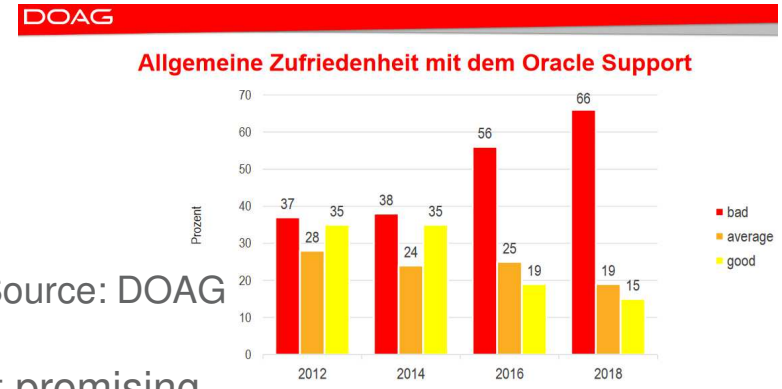
- Oracle falls since years
- PostgreSQL is constant
- Oracle is still searched much more

PostgreSQL awareness

- It is an open source, advanced RDBMS
- It supports modern SQL constructs
- But … it is much less broaden than Oracle and MySQL
  - **So why we care about it?**

More and more customers are upset with Oracle

- High costs (http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf)
- Lower support quality over the years



Source: DOAG

- Aggressive license auditing
- On-premises strategy (and roadmap) is not promising
- Unfriendly with 3rd party clouds https://oracle-base.com/blog/2017/01/28/oracles-cloud-licensing-change-be-warned/

Why PostgreSQL?

- PostgreSQL is the RDBMS closest to Oracle
  - Feature set
  - SQL dialect
  - Procedural language (PL/pgSQL)
- For a fraction of the price! (or completely free)
- Many respected developers endorse it:
  - Markus Winand
  - Lukas Eder
  - Felix Geisendörfer

SQL Standards adherence at a glance

	MYSQL 5.7	PostgreSQL 9.6	SQLite	DB2 11.1	Oracle 12.1	SQL Server 2016
SQL Standards						
SQL:1999 LATERAL	✗	✓	✗	✓	✓	!
SQL:1999 GROUPING SETS	✗	✓	✗	✓	✓	✓
SQL:1999 WITH	✗	✓	✓	✓	✓	✓
SQL:1999 WITH RECURSIVE	✗	✓	✓	✓	✓	✓
SQL:2003 FILTER	✗	✓	✗	✗	✗	✗
SQL:2003 OVER	✗	✓	✗	✓	✓	✓
SQL:2003 WITHIN GROUP	✗	✓	✗	✗	✓	✗
SQL:2003 TABLESAMPLE	✗	✓	✗	✓	✓	✓
SQL:2008 FETCH FIRST	✗	✓	!	✓	✓	✓
SQL:2011 OFFSET	✓	✓	✓	✓	✓	✓
SQL:2011 OVER	✗	✓	✗	✓	✓	✓
SQL:2011 TEMPORAL TABLES	✗	✗	✗	✓	✓	✗
SQL:2016 ROW PATTERN MATHING	✗	✗	✗	✗	✓	✗

Source: https://www.slideshare.net/MarkusWinand/modern-sql

Support, License, Cost

Support

- Many companies provide regional or worldwide support (https://www.postgresql.org/support/professional_support/)
  - Here appear 3 companies from Austria:
    Conova in Salzburg, Cybertec in Wiener Neustadt, Gekko in Vienna
- A few companies are also very active in the development
  - EnterpriseDB (https://www.enterprisedb.com/)
  - 2ndQuadrant (https://2ndquadrant.com/en/)
- As for other OSS projects, they usually provide extra-features only for subscribers

EnterpriseDB Standard

- https://www.enterprisedb.com/products/subscriptions
- Standard PostgreSQL Subscription
  - "Almost" vanilla version but pre-packaged
  - EDB Postgres Enterprise Manager (comparable to OEM)
  - EDB Postgres Failover Manager (comparable to Oracle Data Guard with FSFO)
  - EDB Postgres Replication Server (logical replication comparable to Streams or Oracle GoldenGate)
  - EDB Postgres Migration toolkit
  - Etc.
- Price: **1225€ per core** (or vcore) / year (offer 2016)
- Trivadis is EDB partner

trivadis
makes IT easier.

EnterpriseDB Advanced Server

- https://www.enterprisedb.com/products/subscriptions
- Commercial Subscription

  All Standard features plus:

  – Enterprise Security (password profiles, PL Wrapping, session auditing)

  – Enterprise Performance (Partition by, optimizer hints)

  – Enterprise Development
    (see https://www.enterprisedb.com/products/compare-postgres-databases#securityfeatures for the full list)

  – Enterprise Management (Resource manager, extended catalog views)

  – Compatibility with Oracle (EDB*OCI, PL/SQL support, PL/SQL debugger)
- Price: **1750€ per core** (or vcore) / year (offer 2016)

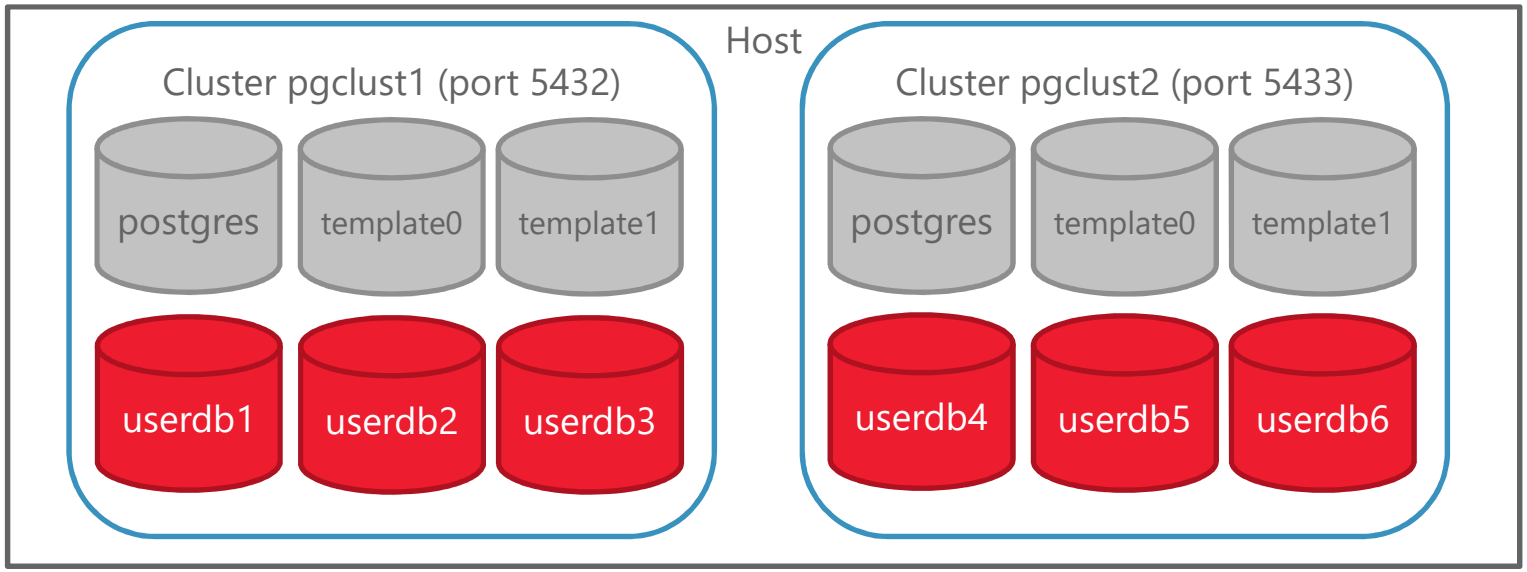
2ndQuadrant

- 2ndQuadrant maintains and supports additional tools
  - Pglogical (replication)
  - Postgres-BDR (bi-directional replication)
  - Postgres-XL (parallel server scale-out)
  - Barman (backup and recovery)
  - Repmpr (replication failover)
  - Other tools for analytics

Architecture

Cluster, instance, database, schema … what?

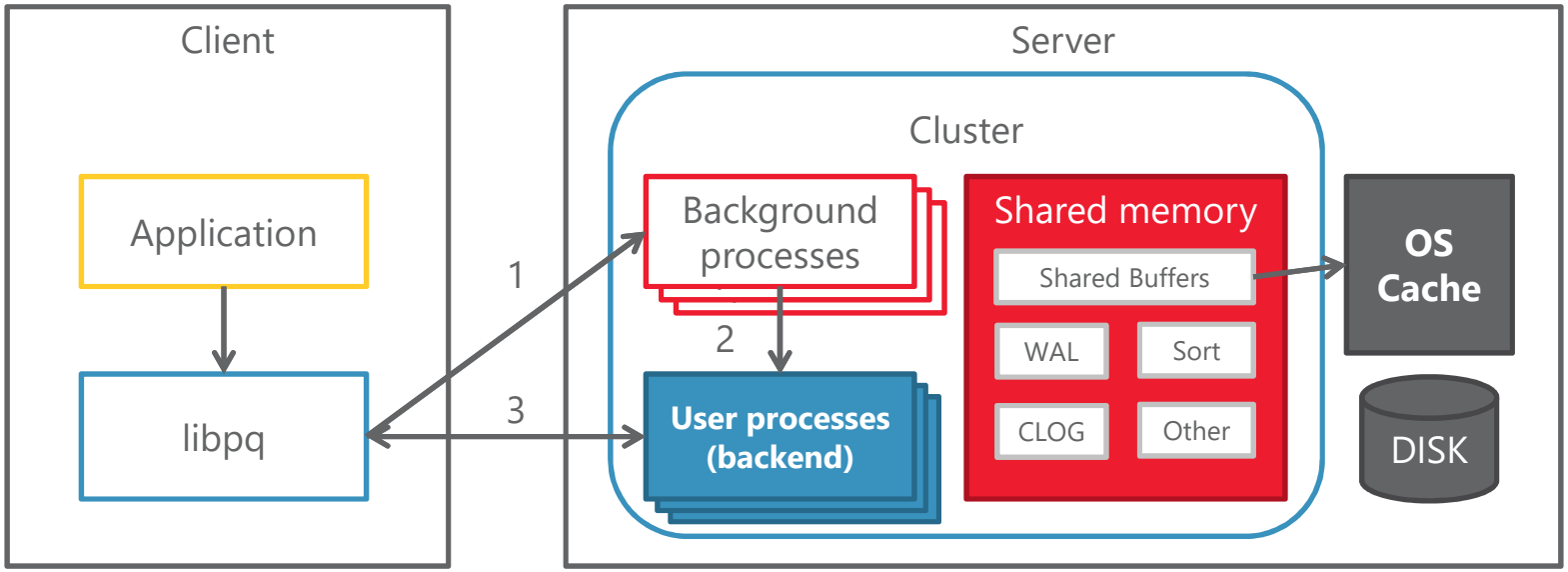
- A cluster is an instance of postgreSQL containing many databases
  - Conceptually similar to MySQL, SQL Server and Oracle Pluggable Databases



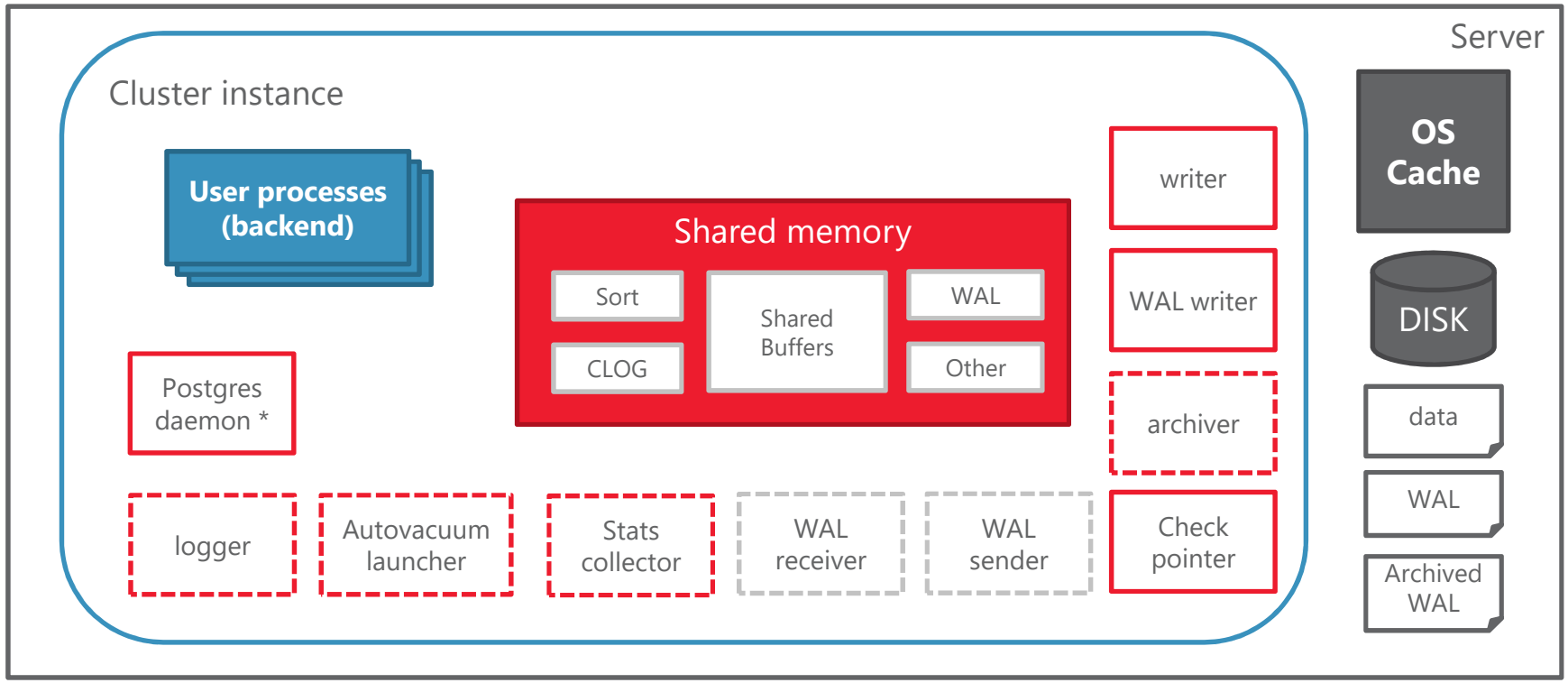
Default Databases

- Postgres
  - Is the “SYSTEM” or “master” database
  - It contains system tables, views, procedures, metadata, definition of users and roles, etc.
- Template0
  - Read-only, empty database used for creating VERY EMPTY databases (like PDS$SEED in Oracle)
- Template1
  - Read-write database, generally empty, might be modified and it is the default template for new databases (like model in SQL Server)

Overall Architecture: client-server interaction



Overall Architecture: background processes



* Once called "postmaster process"

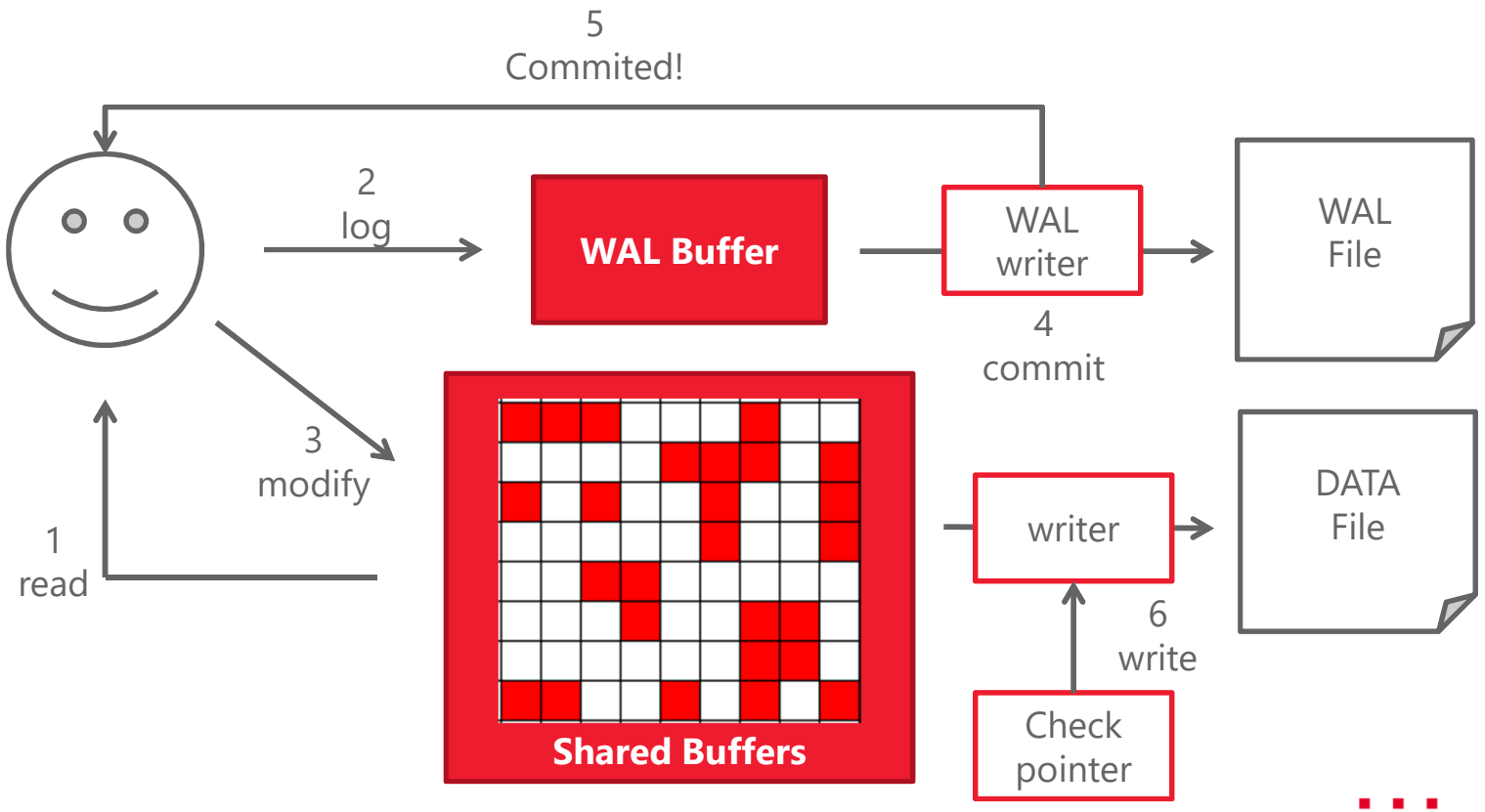
Mandatory background processes

- **Postgres** (A.K.A. postmaster)
  - Gets connection requests, authenticates them and spawns backend processes
- **Checkpointer**
  - Writes dirty pages to disk and registers the checkpoint
- **Writer**
  - The “Lazy writer” that writes pages to disk in small increments
- **WAL Writer**
  - Flushes the WAL buffer to disk (in WAL files) at every commit

Optional background processes

- **Logger**
  - Writes to the “alert log” or whatever log is configured
- **Autovacuum Launcher Process**
  - Automates VACUUM and ANALYZE commands
- **Stats Collector (do not confuse with table statistics gathering)**
  - Collects information about the cluster/server activity
- **WAL Sender** and **Receiver**
  - Respectively sends WAL to a standby and receives WAL from primary

After Image: Write Ahead Logs



Before Image: more complicated!

- **Oracle** has Rollback Segments (UNDO mechanism)
  - Provides consistent reads while writes happen
  - The block is overwritten and the old image is saved in the UNDO tablespace
- **SQL Server** has NO rollback segments for read consistency
  - Writes lock reads and viceversa
  - Unless a special mechanism is implemented (ALLOW_SNAPSHOT_ISOLATION)
- **PostgreSQL** has NO rollback segments
  - But provides consistent reads while writes happen. How?
  - The new image is written in a new location while the previous image is left intact

Before Image: explanation

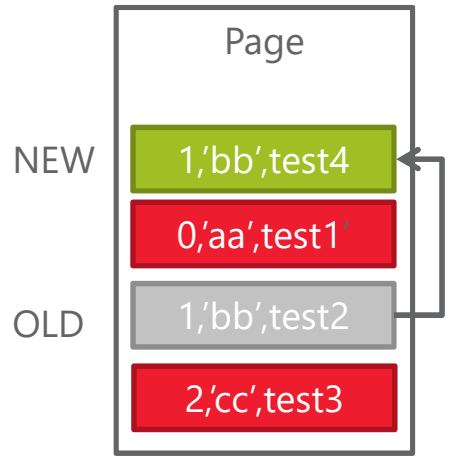
- Every time a row is updated, the new version is stored in a new location
- Case 1: same page

- Case 2: different page

Before Image: explanation

- Every time a row is updated, the new version is stored in a new location
- Case 1: same page

UPDATE t SET col2='test4' WHERE col0=1;



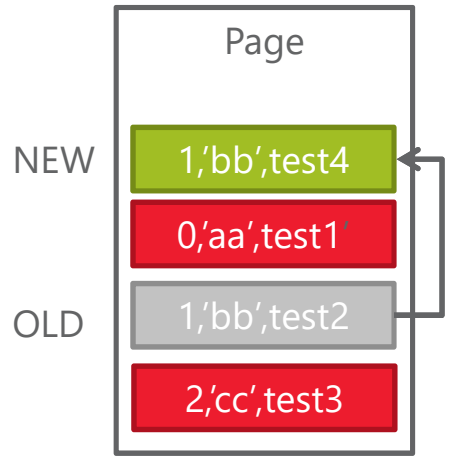
- Case 2: different page

Before Image: explanation

- Every time a row is updated, the new version is stored in a new location

- Case 1: same page

UPDATE t SET col2='test4' WHERE col0=1;



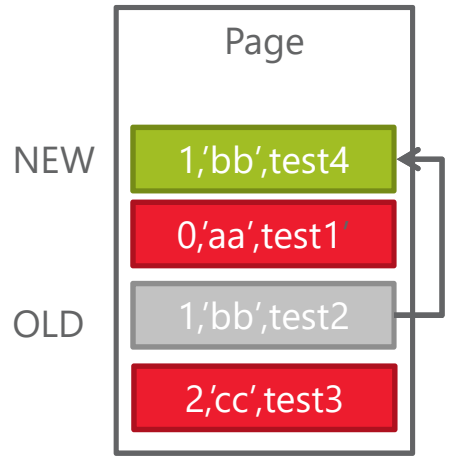
- Case 2: different page

Before Image: explanation

- Every time a row is updated, the new version is stored in a new location

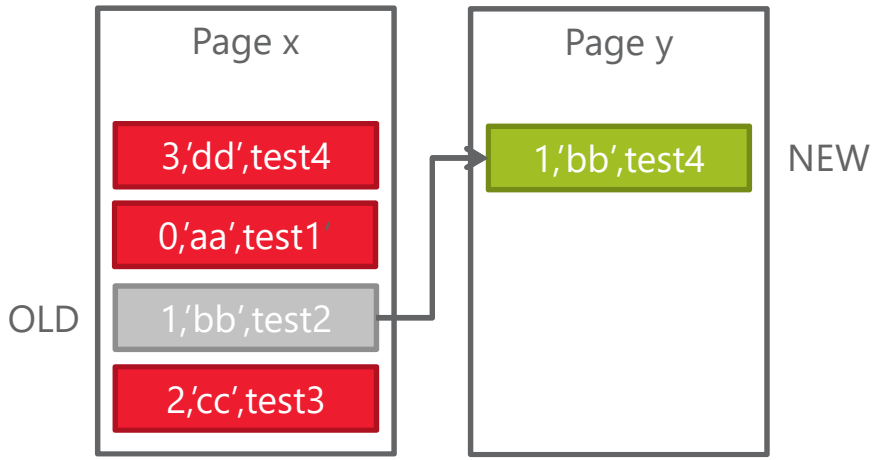
- Case 1: same page

UPDATE t SET col2='test4' WHERE col0=1;

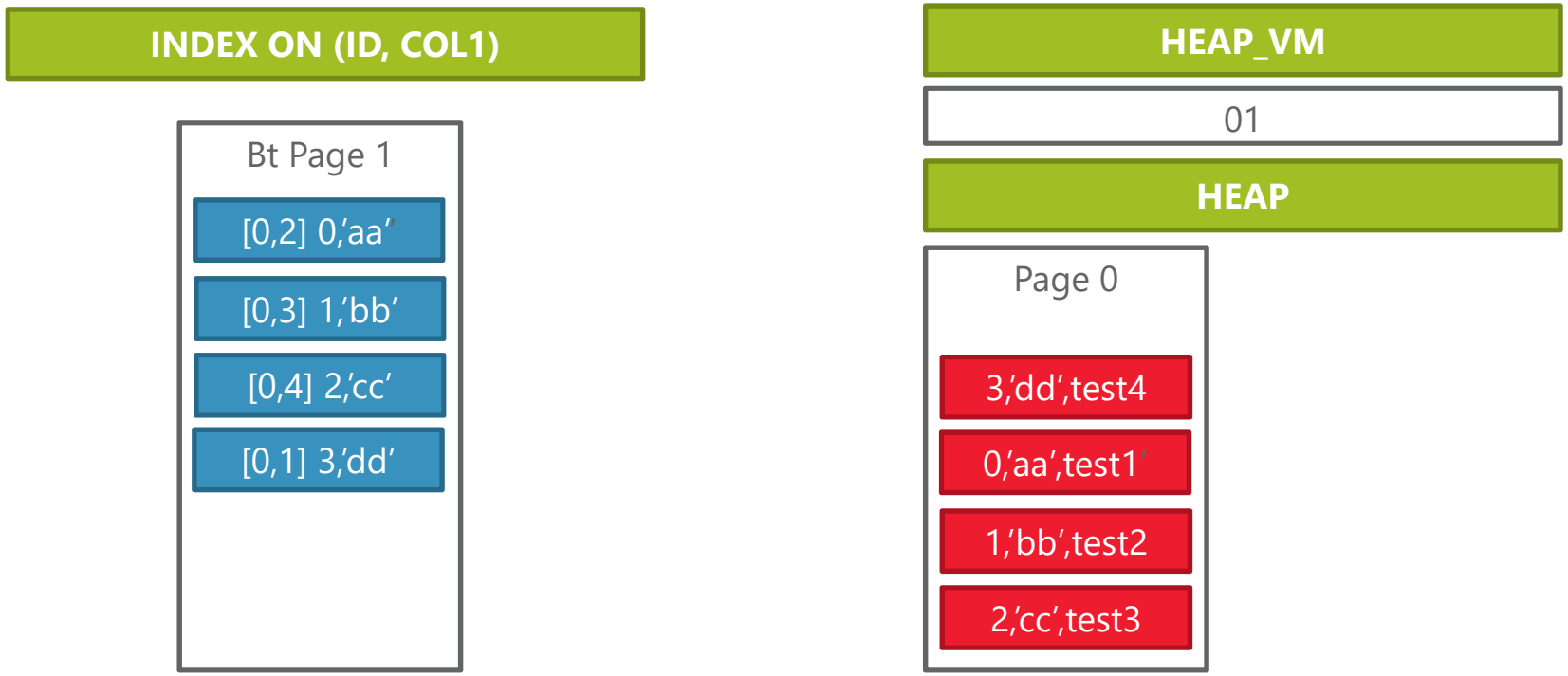


- Case 2: different page

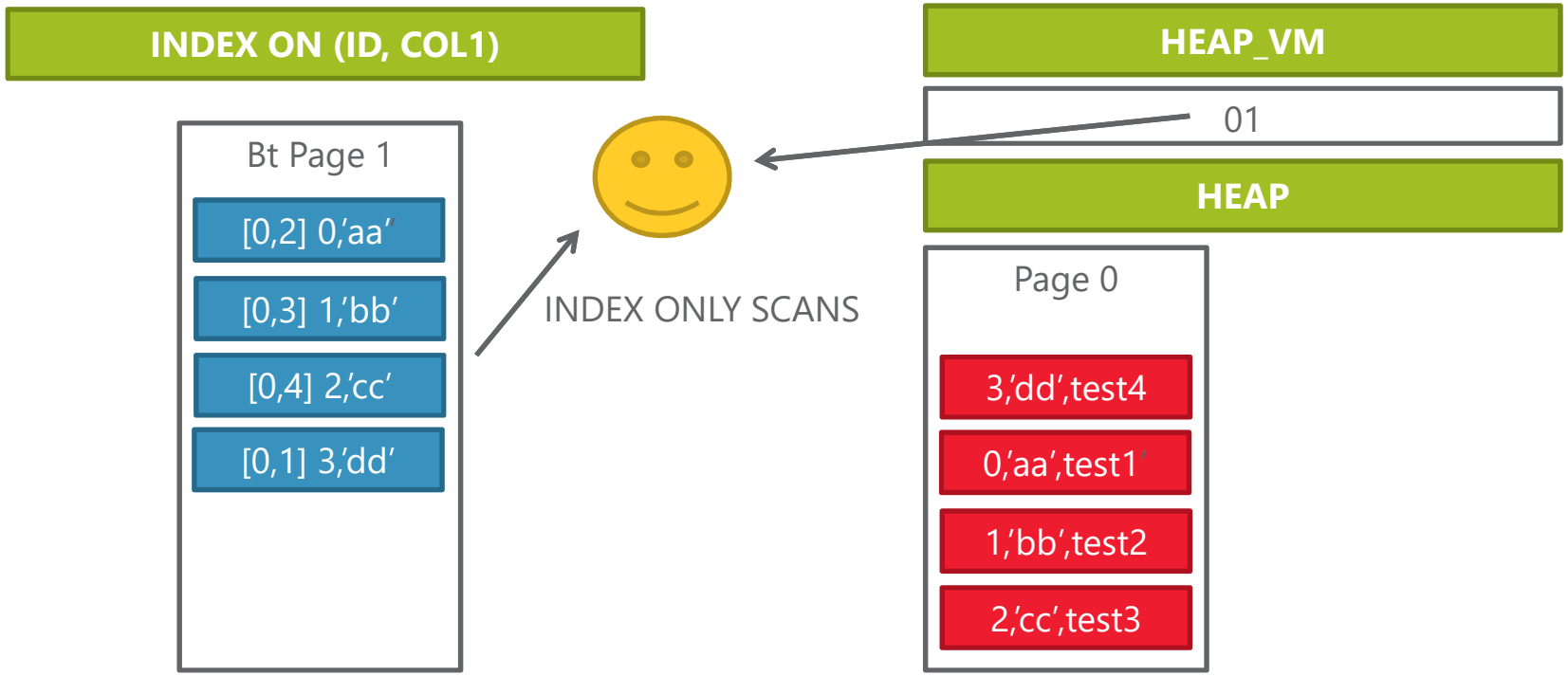
UPDATE t SET col2='test4' WHERE col0=1;



Before Images, Indexes and Visibility Maps



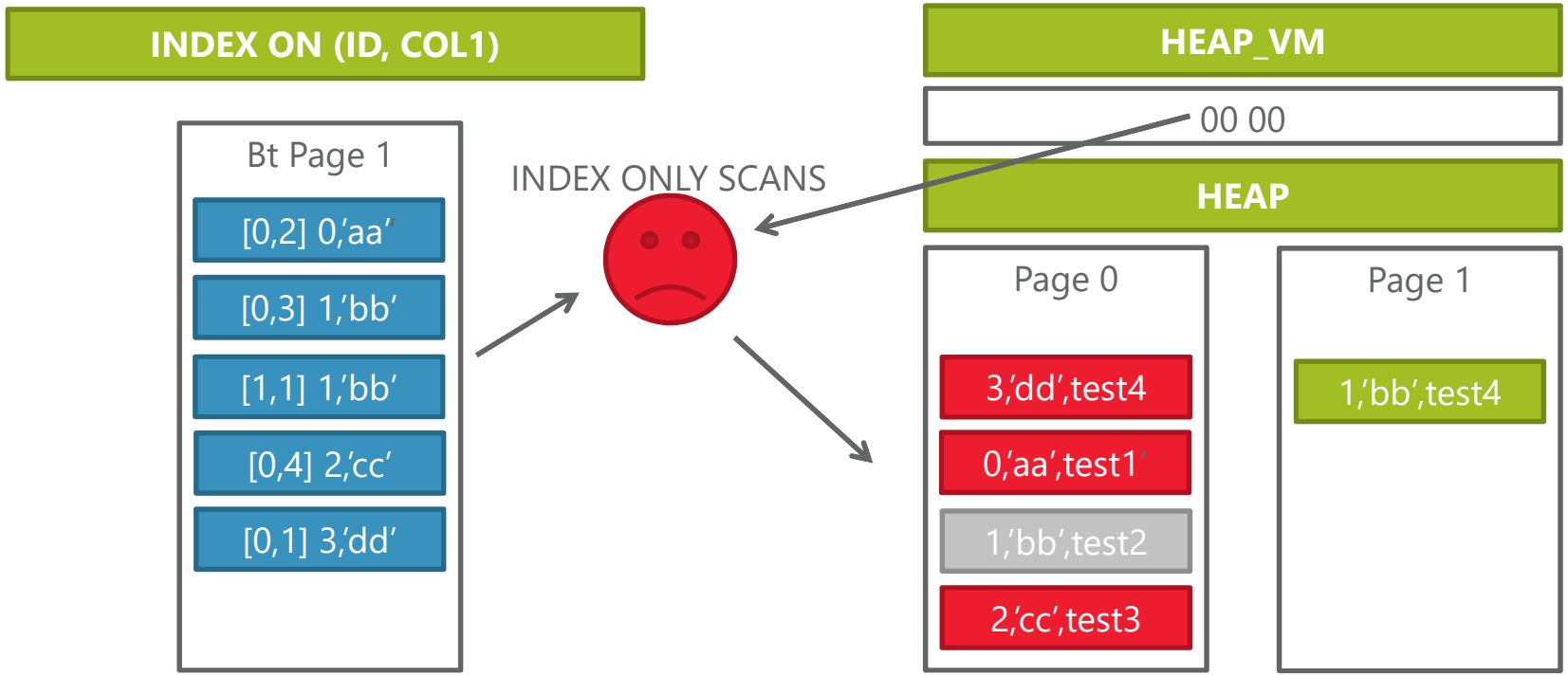
Before Images, Indexes and Visibility Maps



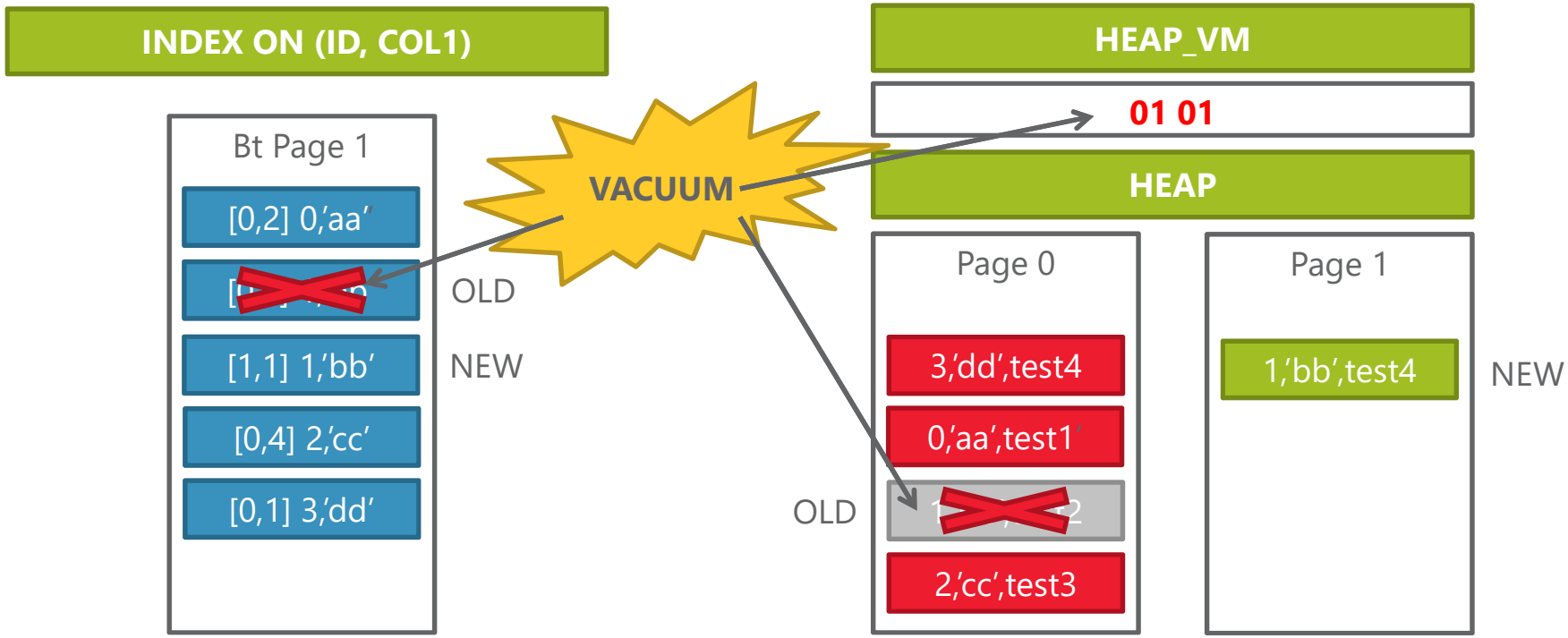
Before Images, Indexes and Visibility Maps

trivadis
makes IT easier.

Before Images, Indexes and Visibility Maps



Before Images, Indexes and Visibility Maps



trivadis makes IT easier.

Pros and Cons of this implementation

- Pros (compared to Oracle)
  - No need for Rollback Segments, no ORA-01555 for rollback segments too small ☺
- Cons
  - High modification == High Fragmentation! VACUUM required
  - **When a tuple is modified AND migrates from page, its address changes!**
    - Need to update ALL the indexes that point to it, regardless if the modified column is indexed
    - This is the famous «write amplification» problem in PostgreSQL
    - The table fillfactor (equivalent of pctused) mitigates the problem
    - Vacuum becomes critical for performance

VACUUM

- Must be performed regularly in order to
  - Reclaim space occupied by old tuple images
  - Update data statistics used by the query planner
  - Update the visibility map
  - Reset the transaction ID of old blocks to prevent wraparound
- A default VACUUM is executed regularly by default
  - Many parameters help in the VACUUM fine-tuning
- Manual VACUUMing is possible

Some Comparisons with Oracle by Example

How To Use Sequences

- Oracle

```
SQL> CREATE SEQUENCE s;
SQL> SELECT s.nextval
     FROM dual;
```

- PostgreSQL

```
SQL> CREATE SEQUENCE s;
SQL> SELECT nextval('s');
```

Some oddities in PostgreSQL in comparison to Oracle

- Constraints are enforced per row not per statement (unless defined as deferrable)

Oracle

```
SQL> INSERT INTO demo VALUES (1);
1 row created.
SQL> INSERT INTO demo VALUES (2);
1 row created.
SQL> UPDATE demo SET n=n+1;
2 rows updated.
```

PostgreSQL

```
INSERT INTO demo VALUES (1);
INSERT 0 1
INSERT INTO demo VALUES (2);
INSERT 0 1
UPDATE demo SET n=n+1;
ERROR:  duplicate key value violates
unique constraint "demo_pk"
DETAIL:  Key (n)=(2) already exists.
```

- Statement error:
  - Oracle: erroneous statement is discarded, TX is still open
  - PostgreSQL: rollback of transaction to beginning or last savepoint

Check for correct IP addresses in a database table

- Oracle

```
CREATE TABLE ips
(ip VARCHAR2(30));

CREATE OR REPLACE FUNCTION
is_valid_ip(ip VARCHAR2) RETURN
BOOLEAN ... <and so on and so on>

CREATE OR REPLACE TRIGGER ip_check
BEFORE INSERT OR UPDATE
ON ips
FOR EACH ROW
BEGIN
  IF NOT is_valid_ip(:new.ip) THEN
    raise_application_error(-20100,
'Invalid IP!');
  END IF;
END;
/
```

- PostgreSQL

```
CREATE TABLE ips (ip inet);
```

PL/SQL vs. PL/pgsql (1)

- Oracle
  - Create custom packages and use a lot of supplied packages
- PostgreSQL
  - No packages can be created
  - For most Oracle packages there are equivalents (e.g. functions)
  - For some you can use EDBs Oracle-compatibility support
  - Cannot be used directly in a trigger (execution of function or procedure is mandatory)
  - Procedures exist as of version 11

PL/SQL vs. PL/pgsql (2)

Oracle

```
CREATE OR REPLACE FUNCTION
get_bal(acc_no IN NUMBER)
RETURN NUMBER
IS
  acc_bal NUMBER(11,2);
BEGIN
SELECT balance
INTO acc_bal
FROM accounts
WHERE account_id = acc_no;
RETURN acc_bal;
END;
/
```

PostgreSQL

```
CREATE OR REPLACE FUNCTION
get_bal(acc_no IN INTEGER)
RETURNS INTEGER
AS $$
DECLARE acc_bal INTEGER;
BEGIN
SELECT balance
INTO acc_bal
FROM accounts
WHERE account_id = acc_no;
RETURN acc_bal;
END;
$$ LANGUAGE plpgsql;
```

Conclusion

- **Oracle database**
  - Sophisticated technology
  - Rather expensive
  - Dissatisfaction with support

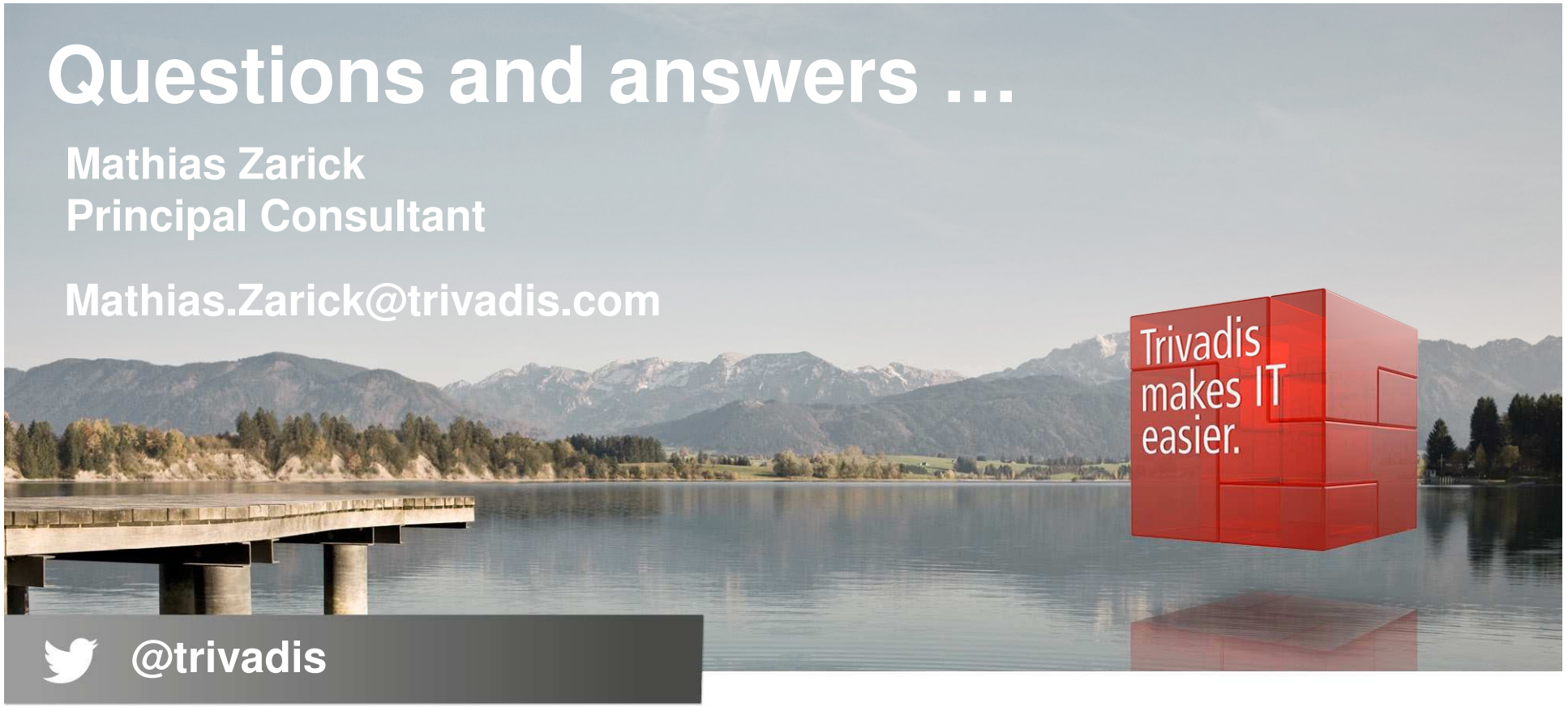
- **PostgreSQL**
  - Technology is quite advanced
  - Costs can be dramatically cut
  - Support is given by technical enthusiasts

- **Migration?**
  - “Leave as is” will work for the least applications
  - But there are ways and even tools to achieve a successful migration

Questions and answers …

Mathias Zarick
Principal Consultant

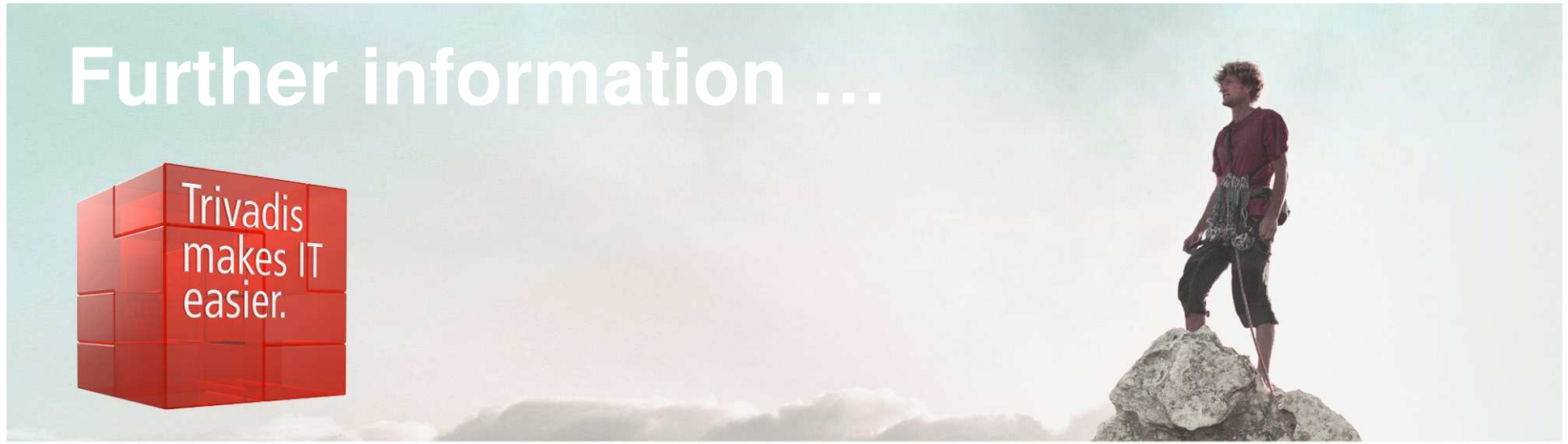
Mathias.Zarick@trivadis.com



@trivadis

trivadis
makes IT easier.

Further information …



https://wiki.postgresql.org/wiki/PostgreSQL_for_Oracle_DBAs
https://www.enterprisedb.com/products/edb-postgres-subscription-plans
https://www.enterprisedb.com/compare-postgres-databases
https://www.postgresql.org/support/professional_support/europe/
https://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion
https://oracle-base.com/blog/2017/01/28/oracles-cloud-licensing-change-be-warned/
https://www.doag.org/formes/pubfiles/10833356/Support-Umfrage_2018.pdf
https://www.doag.org/de/home/news/ergebnisse-der-oracle-support-umfrage-2018/detail/
http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf
https://de.wikipedia.org/wiki/PL/pgSQL
https://www.postgresql.org/docs/11/index.html
https://www.techrepublic.com/article/theres-one-big-reason-that-postgres-cant-kill-oracle-and-its-not-the-technology/
https://hashrocket.com/blog/posts/faster-json-generation-with-postgresql
https://www.ntchosting.com/encyclopedia/databases/postgresql/sequence/
http://www.ludovicocaldara.net/dba/pgsql-lo-space-usage-part-3/
https://www.slideshare.net/MarkusWinand/modern-sql
https://db-engines.com/en/

trivadis
makes IT easier.