# PostgreSQL for Oracle DBAs

Thuymy Tran
Database Solutions Architect

Wanda He
Principal RDS PostgreSQL Solutions Architect

# Agenda

- Terminology
- Architecture Comparison Key Differences:
  - Synonyms
  - Data Types
  - Indexes
  - Subtransactions
  - MVCC
  - Vacuum
  - Backups
  - PostgreSQL Extensions
- Key Takeaways

# Terminology

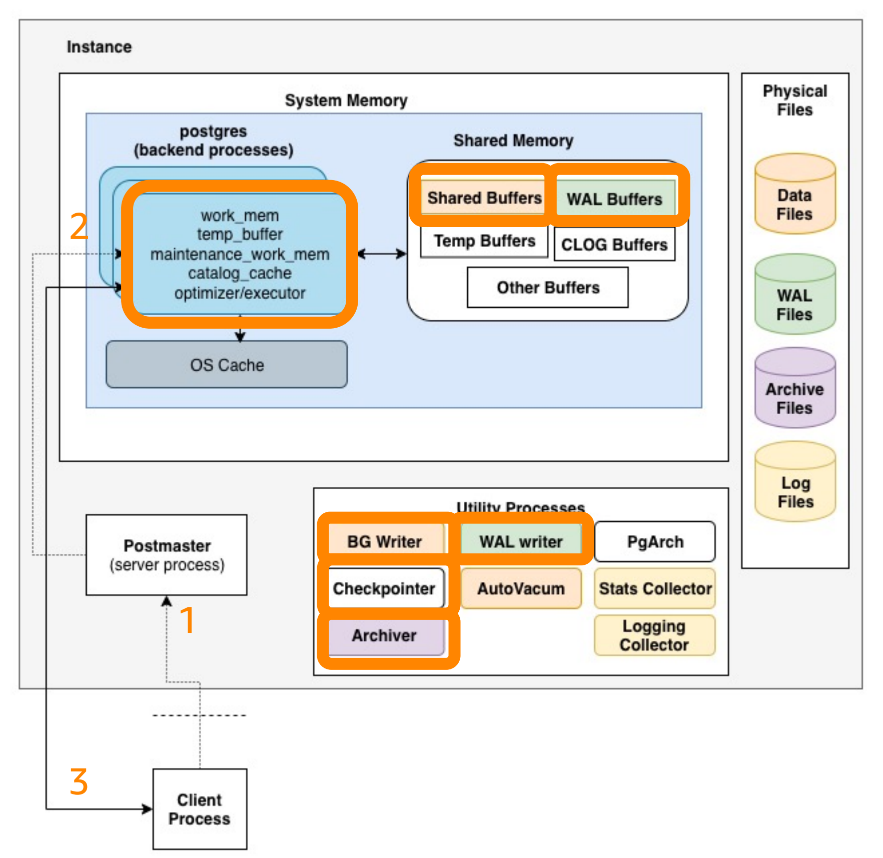
# Terminology

| Oracle | PostgreSQL |
|---|---|
| rowid | ctid |
| row | tuple |
| table | relation |
| block | page |
| redo | WAL |
| undo | MVCC |
| SCN | LSN |

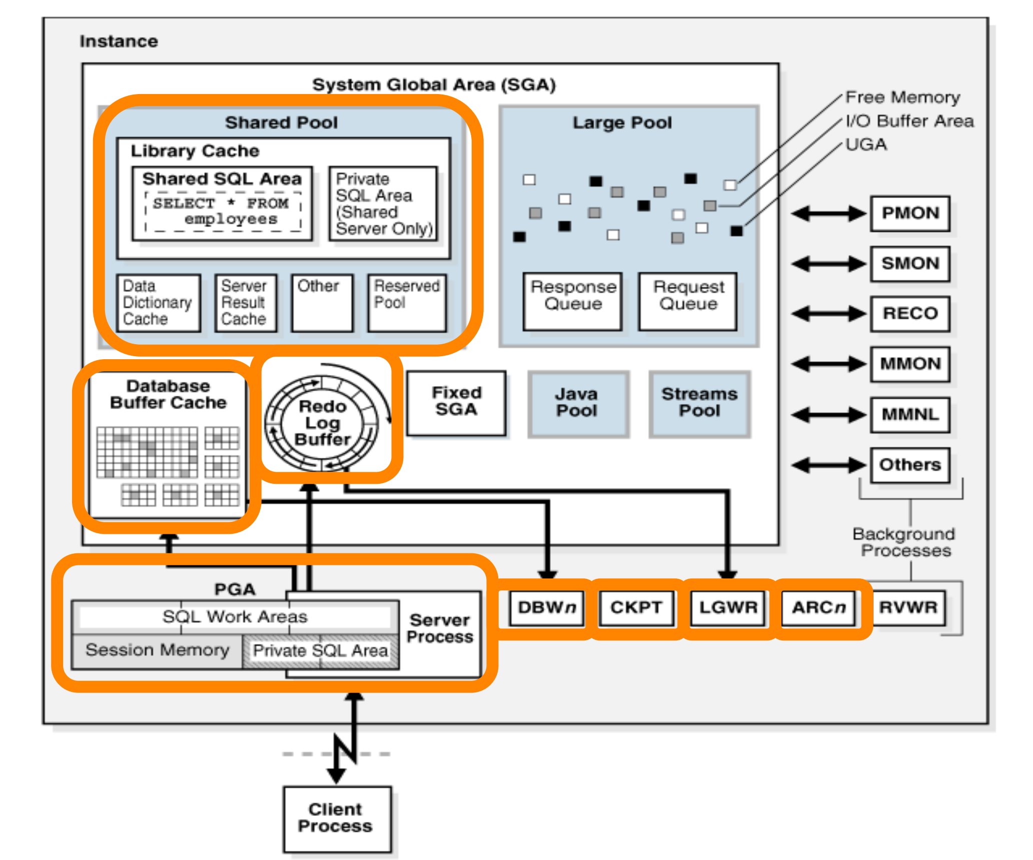
# Architecture Comparison

# Process/Memory Architecture

## PostgreSQL



## Oracle



Reference: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/process-architecture.html#GUID-4B460E97-18A0-4F5A-A62F-9608FFD43664

# Synonyms

# Synonyms

## Oracle

```
CREATE SYNONYM [schema .] synonym_name
FOR [schema .] object_name ;
```

## PostgreSQL

- No synonyms in Postgres
- Use schema search path instead, SEARCH_PATH
- To view current search path:

```
# SHOW search_path;
```

Default set up returns:

```
search_path
--------------
"$user", public
```

- To add new schema in path:

```
# SET search_path to schema1, schema2;
```

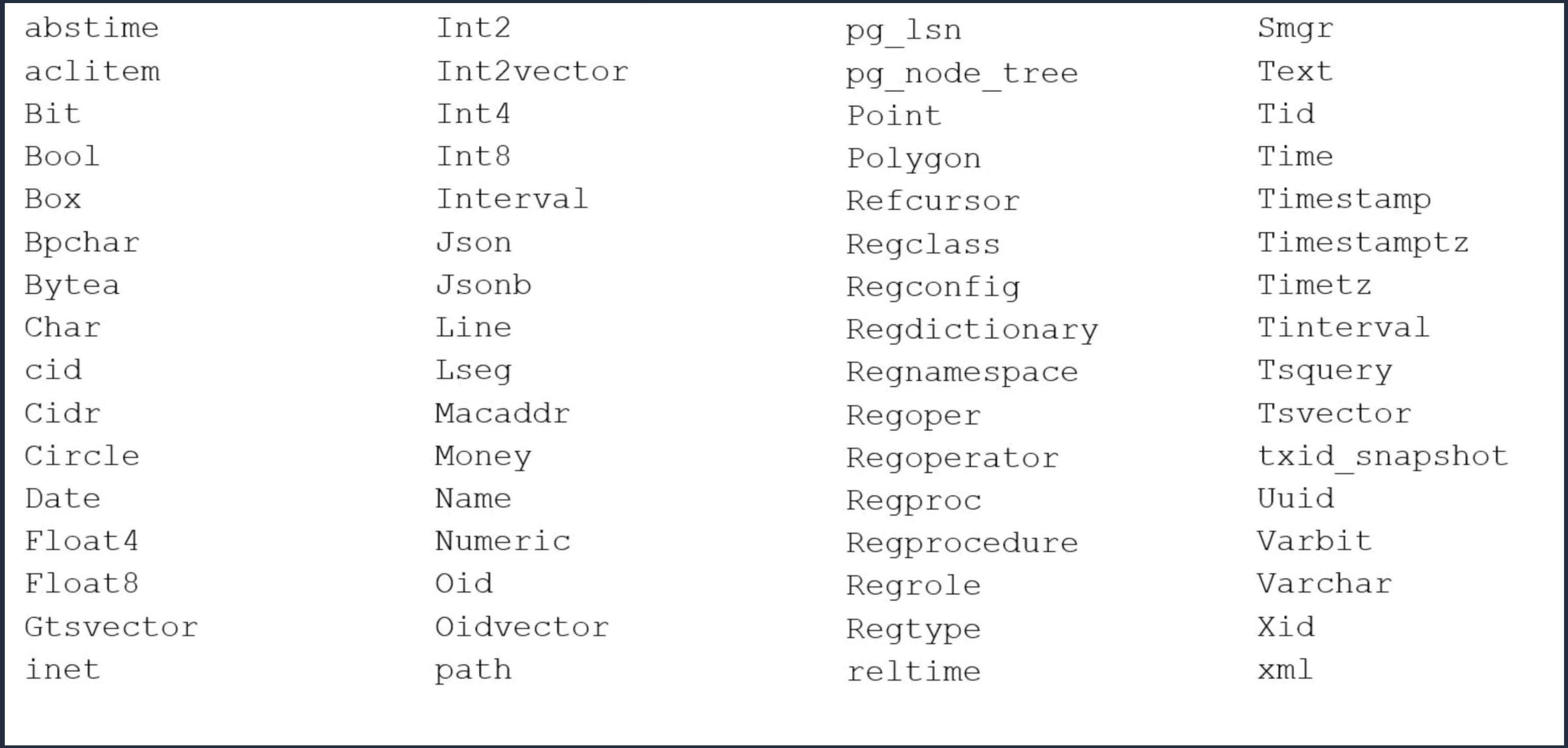
# Data Types

# Only a few data types are commonly used in Oracle

| Data Type |
|---|
| VARCHAR2 |
| DATE |
| NUMBER |
| CLOB |
| BLOB |

# PostgreSQL Data Types

## 64 BASE TYPES AND CAN BE EXTENDED FOR MORE



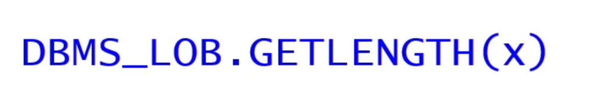
abstime
aclitem
Bit
Bool
Box
Bpchar
Bytea
Char
cid
Cidr
Circle
Date
Float4
Float8
Gtsvector
inet
Int2
Int2vector
Int4
Int8
Interval
Json
Jsonb
Line
Lseg
Macaddr
Money
Name
Numeric
Oid
Oidvector
path
pg_lsn
pg_node_tree
Point
Polygon
Refcursor
Regclass
Regconfig
Regdictionary
Regnamespace
Regoper
Regoperator
Regproc
Regprocedure
Regrole
Regtype
reltime
Smgr
Text
Tid
Time
Timestamp
Timestamptz
Timetz
Tinterval
Tsquery
Tsvector
txid_snapshot
Uuid
Varbit
Varchar
Xid
xml

# Perceived Equivalent Data Types

## DON'T ALWAYS BEHAVE THE SAME WAY

| Oracle | PostgreSQL (equivalent) |
|---|---|
| CLOB | TEXT |

- In PostgreSQL, TEXT is not a CLOB
- What data type to suggest or workaround for clob? --thuymy



```
DBMS_LOB.GETLENGTH(x)
```

# Watchout for Perceived Data Type Equivalents

## THEY DON'T ALWAYS BEHAVE THE SAME WAY

- Oracle NUMBER:

```
NUMBER(precision, scale)
```

  - Up to 38 digits *before* the decimal point
  - Up to 127 digits *after* the decimal point

- PostgreSQL NUMERIC:

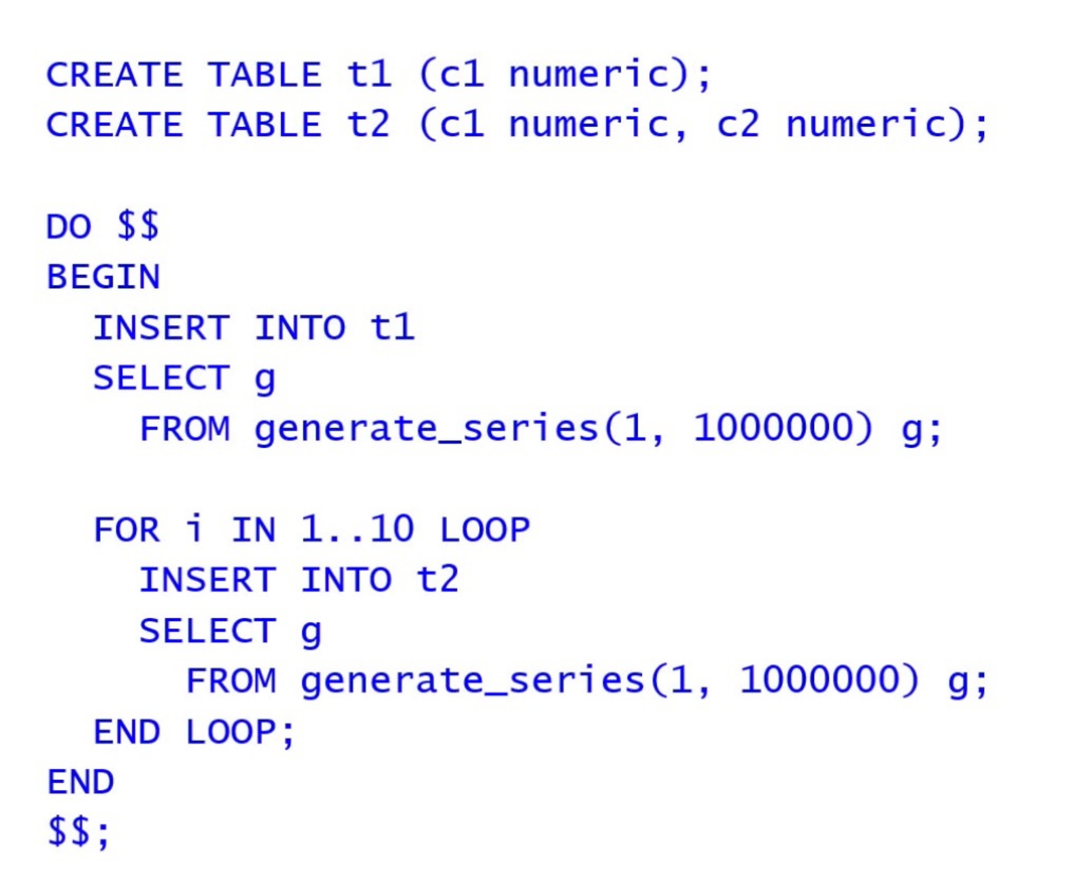
```
NUMERIC(precision, scale]
```

  - Up to 131072 digits *before* the decimal point
  - Up to 16383 digits *after* the decimal point

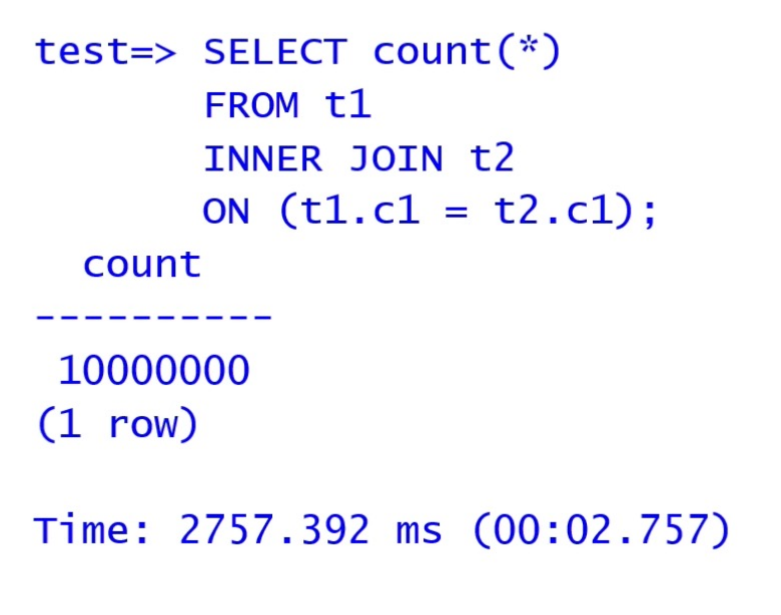
⚠ Never use NUMERIC for PKs or FKs, use BIGINT

# Data Type Performance

## NUMERIC



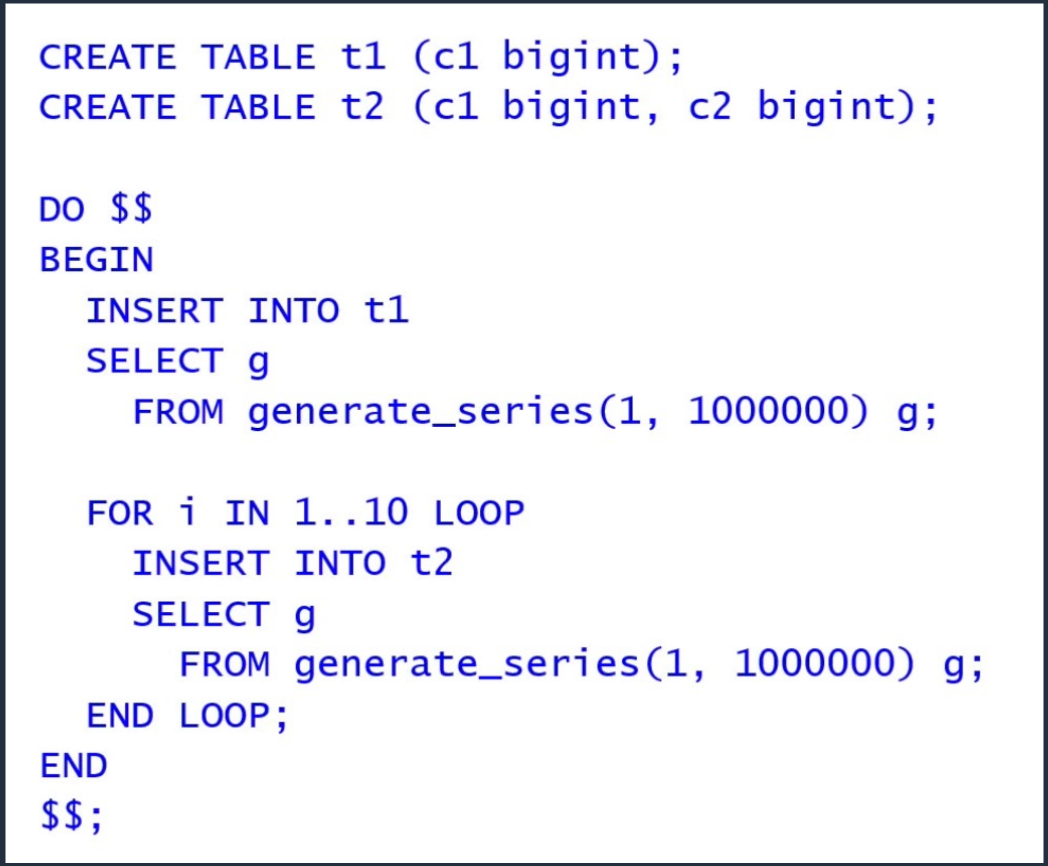
```
CREATE TABLE t1 (c1 numeric);
CREATE TABLE t2 (c1 numeric, c2 numeric);

DO $$
BEGIN
  INSERT INTO t1
  SELECT g
    FROM generate_series(1, 1000000) g;

  FOR i IN 1..10 LOOP
    INSERT INTO t2
    SELECT g
      FROM generate_series(1, 1000000) g;
  END LOOP;
END
$$;
```



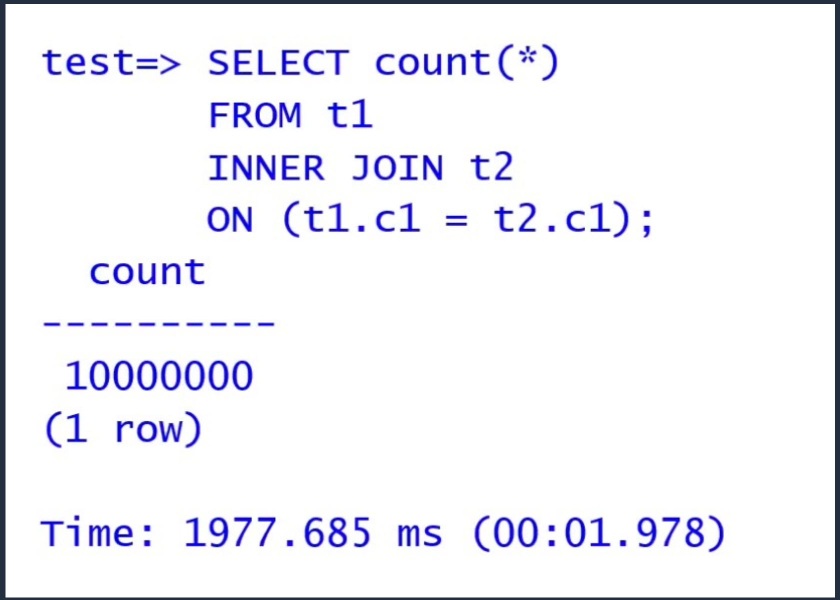
```
test=> SELECT count(*)
       FROM t1
       INNER JOIN t2
       ON (t1.c1 = t2.c1);
  count
----------
 10000000
(1 row)

Time: 2757.392 ms (00:02.757)
```

# Data Type Performance

## BIGINT



```
CREATE TABLE t1 (c1 bigint);
CREATE TABLE t2 (c1 bigint, c2 bigint);

DO $$
BEGIN
  INSERT INTO t1
  SELECT g
    FROM generate_series(1, 1000000) g;

  FOR i IN 1..10 LOOP
    INSERT INTO t2
    SELECT g
      FROM generate_series(1, 1000000) g;
  END LOOP;
END
$$;
```



```
test=> SELECT count(*)
       FROM t1
       INNER JOIN t2
       ON (t1.c1 = t2.c1);
  count
----------
 10000000
(1 row)

Time: 1977.685 ms (00:01.978)
```

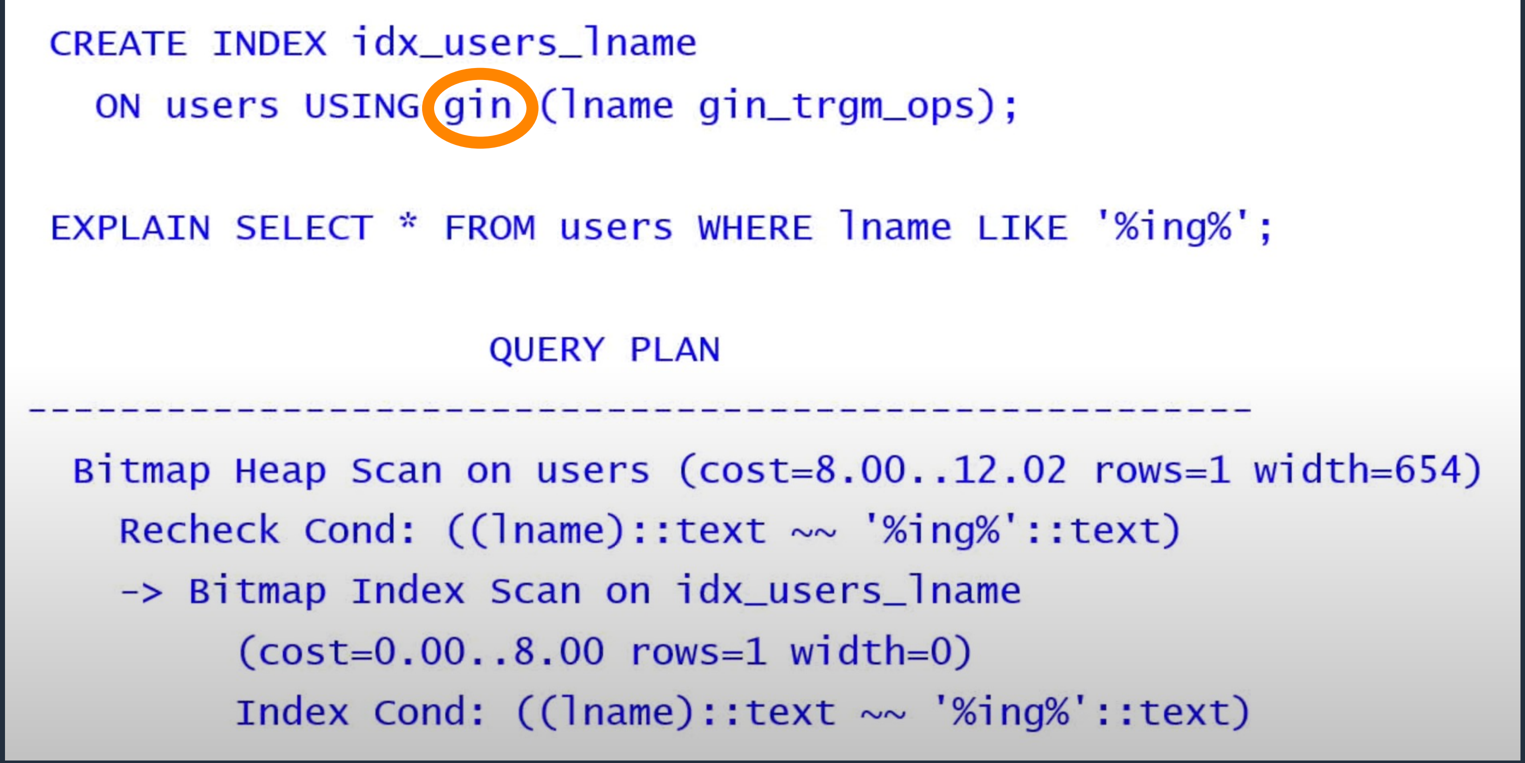
# Indexes

# PostgreSQL Indexes

## ADDITIONAL INDEXES THAT ORACLE DOESN’T HAVE

| | Use Case |
|---|---|
| GIN | • Map a large amount of values to one row<br>• Optimal for fulltext search and indexing array values |
| GiST | • Optimal for more complex comparisons (geometric data types) |
| SP-Gist | • Optimal for partitioned search trees |
| BRIN | • Stores min and max values contained in a group of database pages<br>• Optimal for time series data<br>    • Rule out certain records and therefore reduce query run time |
| BLOOM | • Test whether an element is a member of a set<br>• Optimal when a table has many attributes and queries test arbitrary combinations on them |

# GIN Index

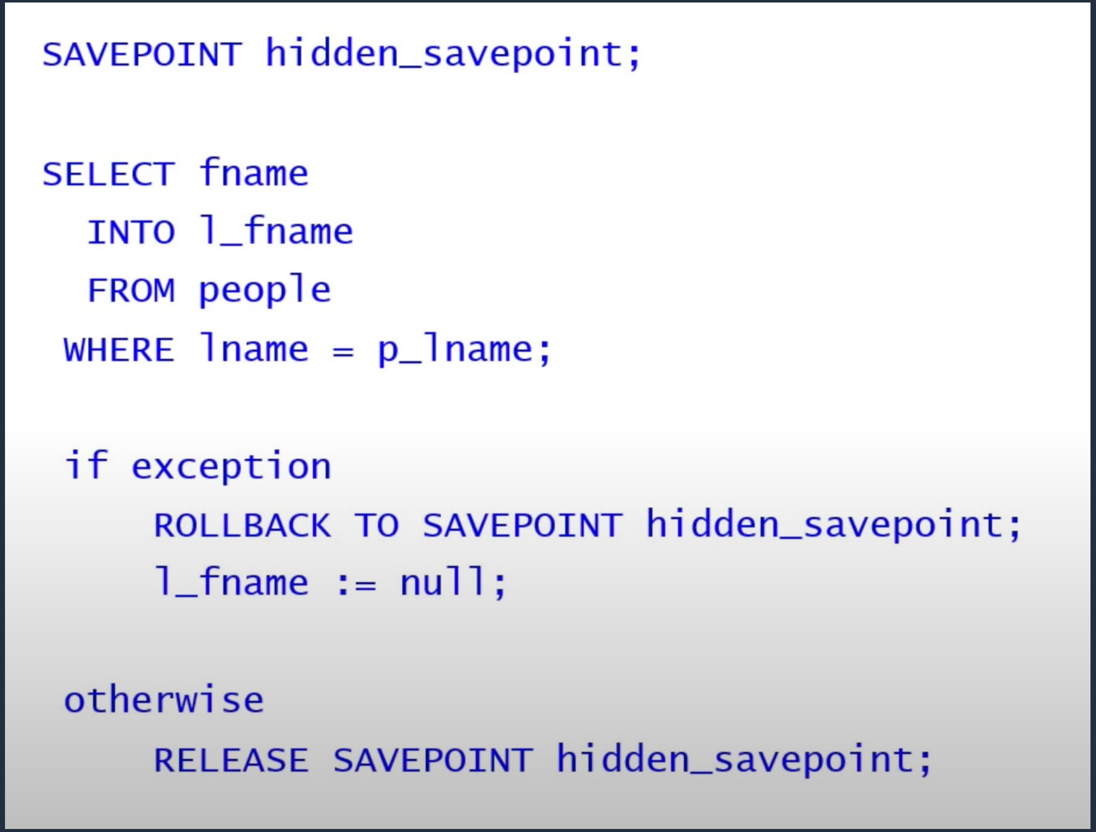


```
 CREATE INDEX idx_users_lname
   ON users USING gin (lname gin_trgm_ops);

 EXPLAIN SELECT * FROM users WHERE lname LIKE '%ing%';

                    QUERY PLAN
-----------------------------------------------------
  Bitmap Heap Scan on users (cost=8.00..12.02 rows=1 width=654)
    Recheck Cond: ((lname)::text ~~ '%ing%'::text)
    -> Bitmap Index Scan on idx_users_lname
         (cost=0.00..8.00 rows=1 width=0)
         Index Cond: ((lname)::text ~~ '%ing%'::text)
```

# Subtransactions

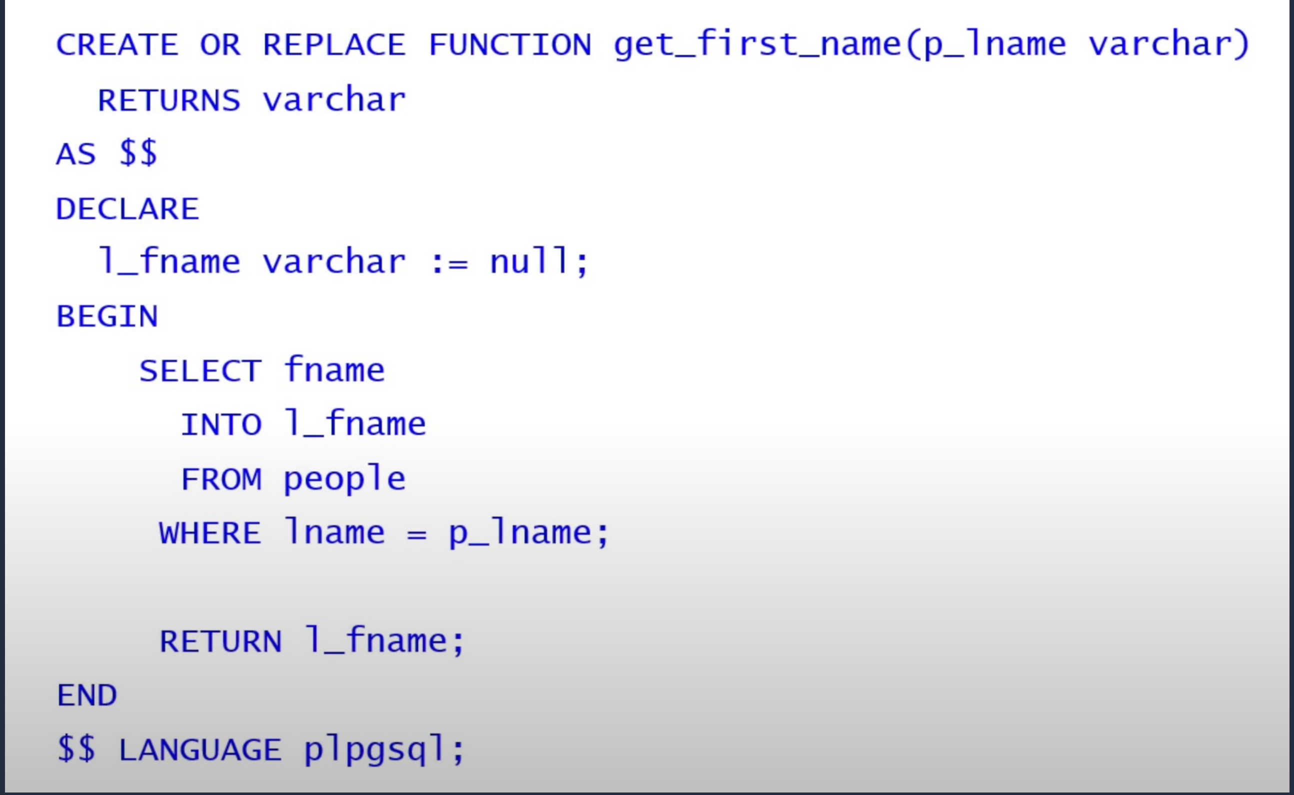
# Exceptions

## POSTGRESQL USES SUBTRANSACTIONS TO HANDLE EXCEPTIONS

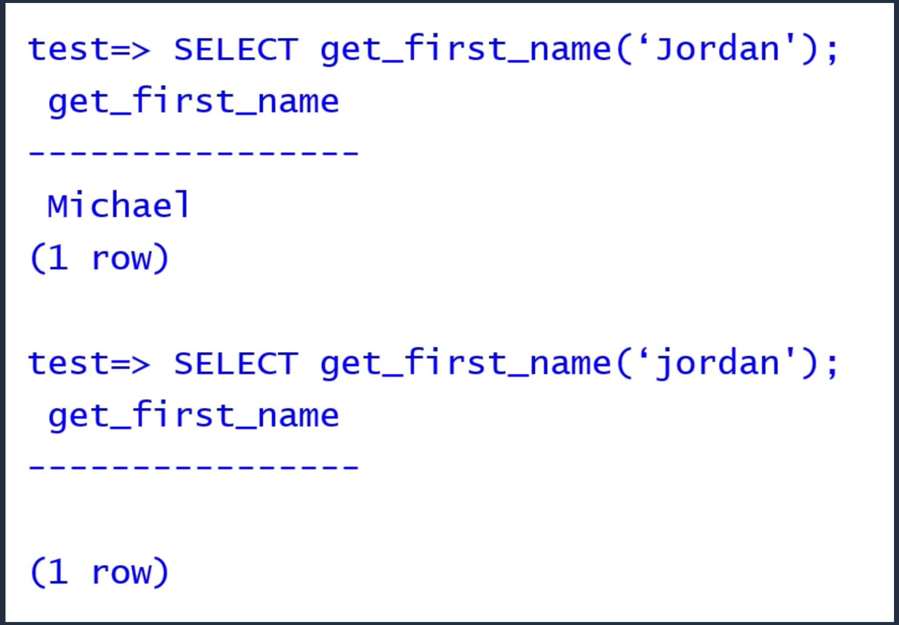


```
SAVEPOINT hidden_savepoint;

SELECT fname
  INTO l_fname
  FROM people
 WHERE lname = p_lname;

 if exception
     ROLLBACK TO SAVEPOINT hidden_savepoint;
     l_fname := null;

 otherwise
     RELEASE SAVEPOINT hidden_savepoint;
```

# Most exceptions are not necessary



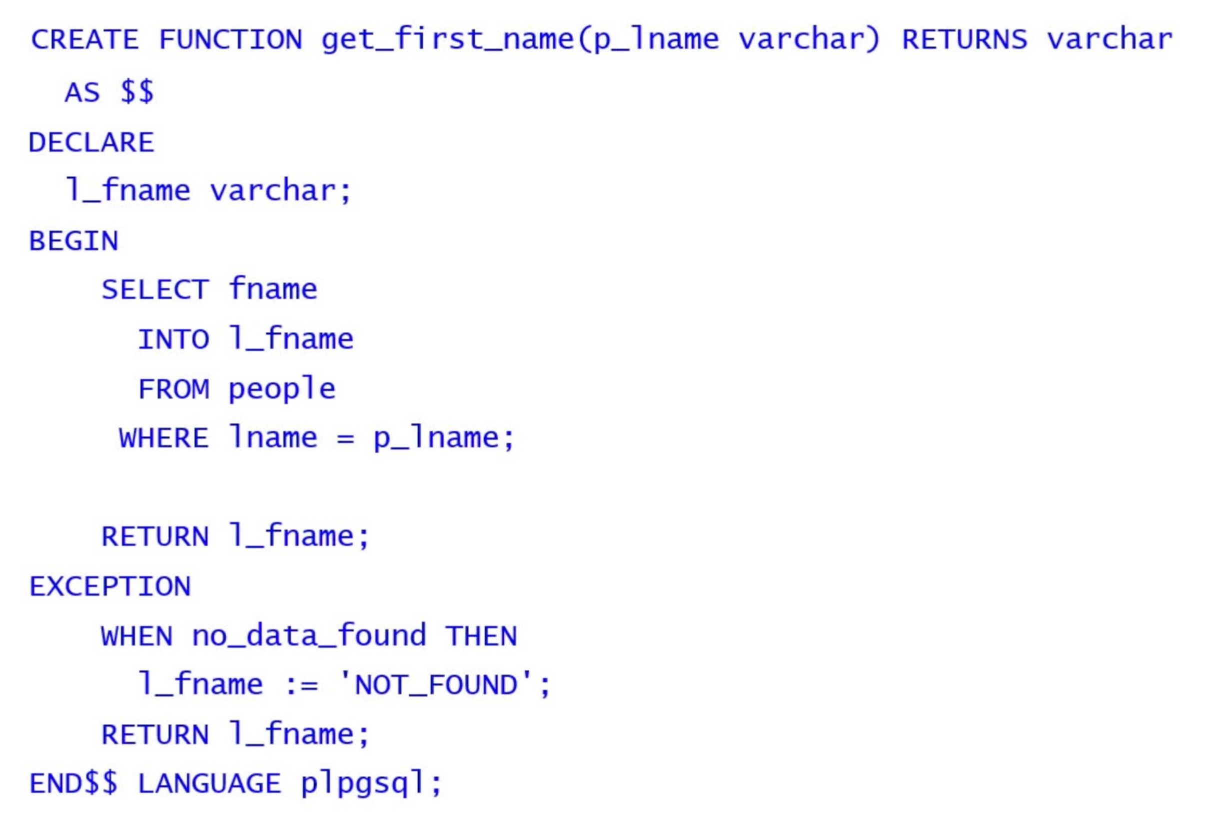
```
CREATE OR REPLACE FUNCTION get_first_name(p_lname varchar)
  RETURNS varchar
AS $$
DECLARE
  l_fname varchar := null;
BEGIN
    SELECT fname
      INTO l_fname
      FROM people
     WHERE lname = p_lname;

     RETURN l_fname;
END
$$ LANGUAGE plpgsql;
```



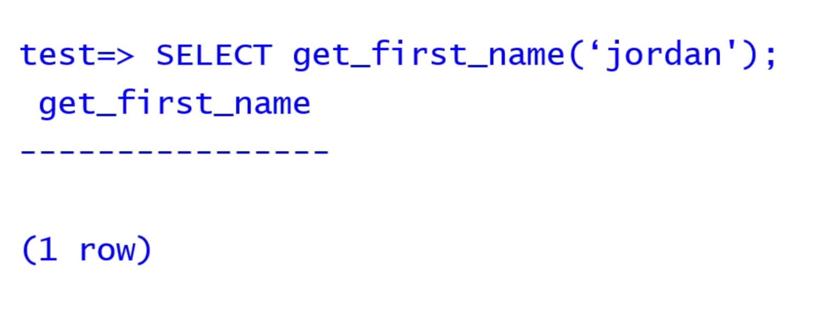
```
test=> SELECT get_first_name(‘Jordan');
 get_first_name
----------------
 Michael
(1 row)

test=> SELECT get_first_name(‘jordan');
 get_first_name
----------------

(1 row)
```

# Exceptions

## NO_DATA_FOUND AND TOO_MANY_ROWS ARE NOT EXCEPTIONS RAISED FOR A SELECT INTO STATEMENT



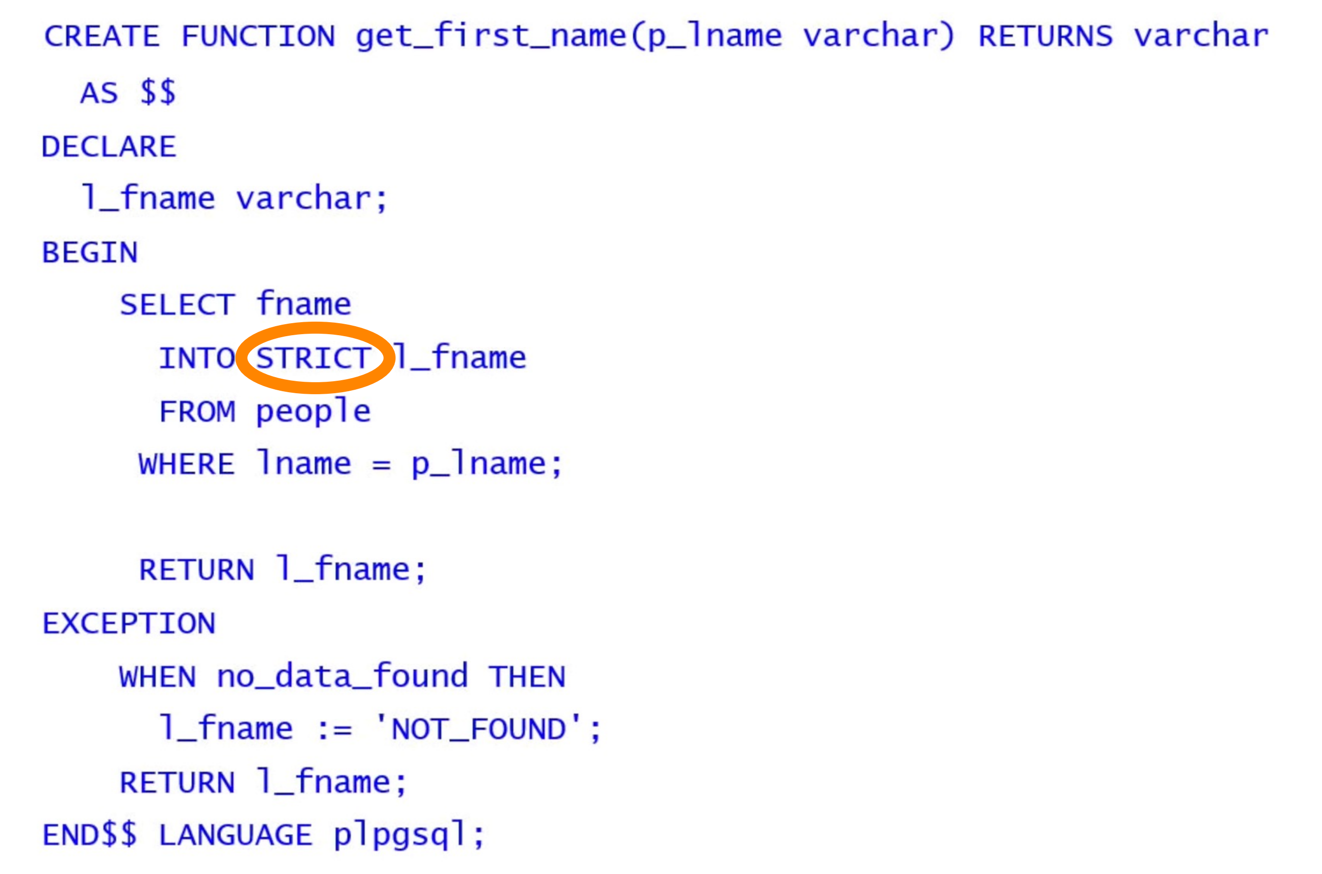
```
CREATE FUNCTION get_first_name(p_lname varchar) RETURNS varchar
  AS $$
DECLARE
  l_fname varchar;
BEGIN
    SELECT fname
      INTO l_fname
      FROM people
     WHERE lname = p_lname;

    RETURN l_fname;
EXCEPTION
    WHEN no_data_found THEN
      l_fname := 'NOT_FOUND';
    RETURN l_fname;
END$$ LANGUAGE plpgsql;
```



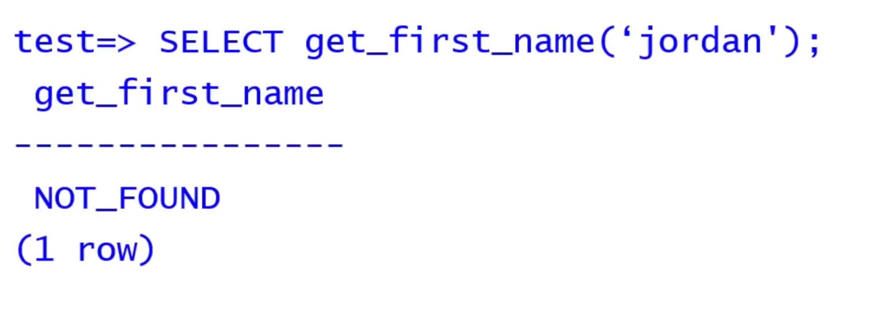
```
test=> SELECT get_first_name(‘jordan');
 get_first_name
-----------------


(1 row)
```

# Exceptions

## USE STRICT TO GET ORACLE-LIKE BEHAVIOR



```
CREATE FUNCTION get_first_name(p_lname varchar) RETURNS varchar
  AS $$
DECLARE
  l_fname varchar;
BEGIN
    SELECT fname
      INTO STRICT l_fname
      FROM people
     WHERE lname = p_lname;

     RETURN l_fname;
EXCEPTION
    WHEN no_data_found THEN
      l_fname := 'NOT_FOUND';
    RETURN l_fname;
END$$ LANGUAGE plpgsql;
```



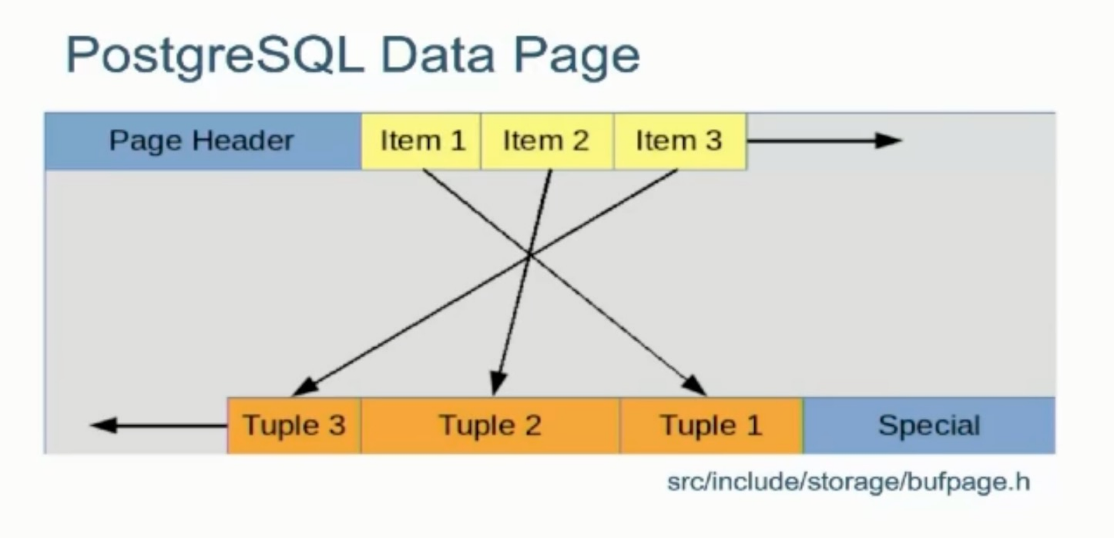
```
test=> SELECT get_first_name(‘jordan');
 get_first_name
-----------------
 NOT_FOUND
(1 row)
```

# MVCC

# MVCC

## PostgreSQL

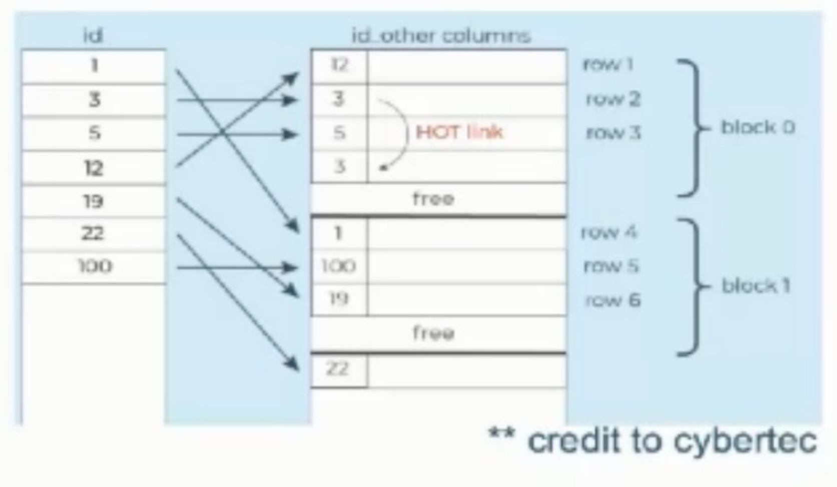
- Multi-Version Control Concurrency at the tuple level
- Old and new versions are stored within the same table itself
- New version of a tuple is created
- Pointer is used to point to an old version
- Transaction IDs are used to identify a version a query can use
- Periodic maintenance from a background process (VACCUM) is necessary to delete old versions of rows



src/include/storage/bufpage.h

# Heap-Only Tuples (HOT)

- **When a row is updated with HOT:**
  - New version is stored in the same page as the old tuple
  - Corresponding index update is skipped
  - Forwarding address is stored in the old tuple version
- **Advantages:**
  - Index change is skipped
  - Dead tuples can be removed without vacuum
- **Requirements:**
  - Enough space in block
  - No index defined on column whose value modified it

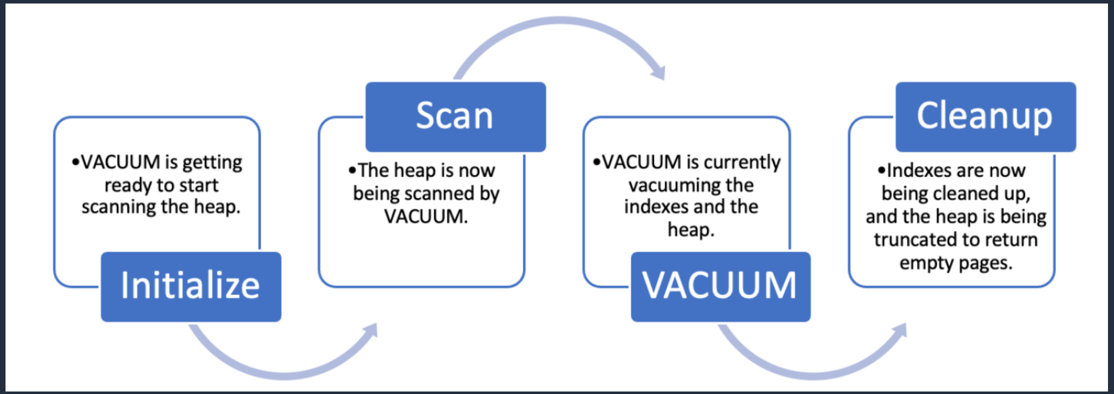


** credit to cybertec

# Vaccum

# What is a vacuum process in PostgreSQL?

- Vacuum cleans up dead tuples and updates the free space map
- Periodic vacuuming is required to:
  - Recover or reuse disk space by updated or deleted rows
  - Update data statistics
  - Update visibility map
  - Protect against transaction ID wraparound



Initialize

•VACUUM is getting ready to start scanning the heap.

Scan

•The heap is now being scanned by VACUUM.

VACUUM

•VACUUM is currently vacuuming the indexes and the heap.

Cleanup

•Indexes are now being cleaned up, and the heap is being truncated to return empty pages.

# Backups

# Logical Backups

| Oracle | Postgres |
|---|---|
| Datapump (expdp) | pg_dump |
| Datapump (impdp) | pg_restore |

# Physical Backups

| Database | Backup Tool | Incremental | Parallelism | PITR |
|---|---|---|---|---|
| Oracle | RMAN | Y | Y | Y |
| PostgreSQL | pg_basebackup | N | N | N |

# Amazon Aurora - Backup and Restore

## Automated backups:

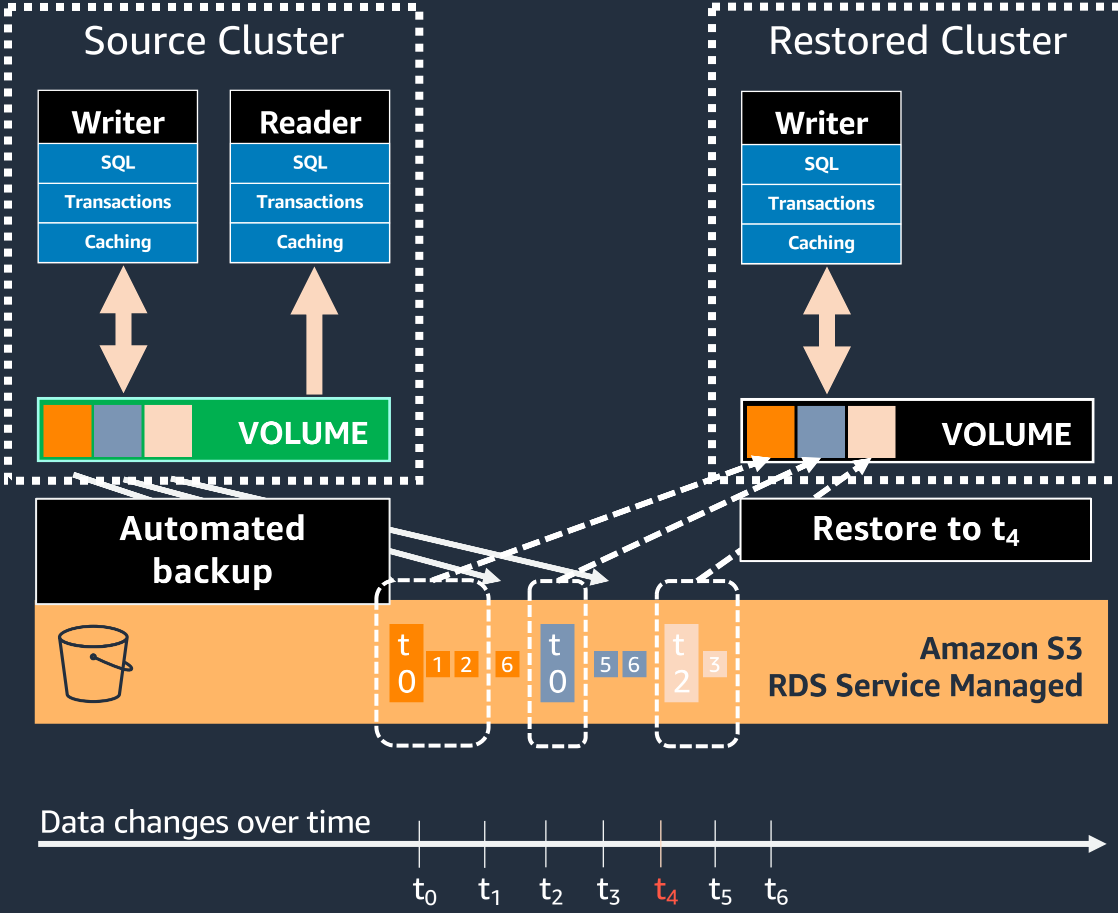
- Between 1 and 35 days retention
- Recover up to the last ~5 min point in time

## Snapshots:

- Create manual snapshots for longer retention
- No performance impact
- Copy snapshots to another region
- Share snapshots with other AWS accounts

## Restore:

- Time depends on cluster volume size
- Always creates a new DB cluster



# PostgreSQL Extensions

# PostgreSQL Extensions for added functionality

| Feature | Postgres<br>(Extensions or 3rd Party) |
|---|---|
| Auditing | pgAudit |
| Cron | pg_cron |
| Query Tuning | pg_stat_statements |
| Vector Search | pg_vector |
| Spatial Database | PostGIS |
| Database Link | Foreign Data Wrapper (FDW) |
| Invisible Index | HypoPG |

# Key Takeaways

- Get the data types right from the beginning
- Don’t be afraid to utilize Postgres’ native features
- Use Postgres extensions to add additional functionality
- WIP……

# Thank you!

Thuymy Tran
thuymyt@amazon.com

Wanda He
wanhe@amazon.com