

# Ausgewählte Systeme Postgres

Jennifer Wittling, Rolf Kimmelman, Jan Löffelsender

January 21, 2019



# Agenda

1. PostgreSQL Überblick
2. Postgres Architektur
3. Konzepte für rekursive Anfragen
4. pgBench
5. Messergebnisse

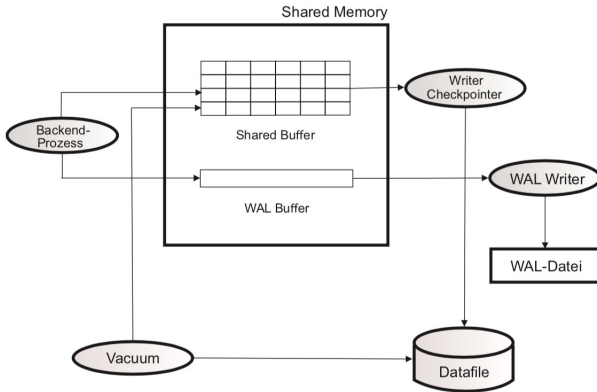


# PostgreSQL Überblick

Postgres ist ein Datenbankmanagementsystem mit folgenden Eigenschaften :

- ▶ Objektrelational
- ▶ ACID konform
- ▶ CRUD konform
- ▶ Hersteller: PostgreSQL Global Development Group, ursprünglich University of California
- ▶ Zielgruppe: Telekommunikationsunternehmen für Ordermanagement System.

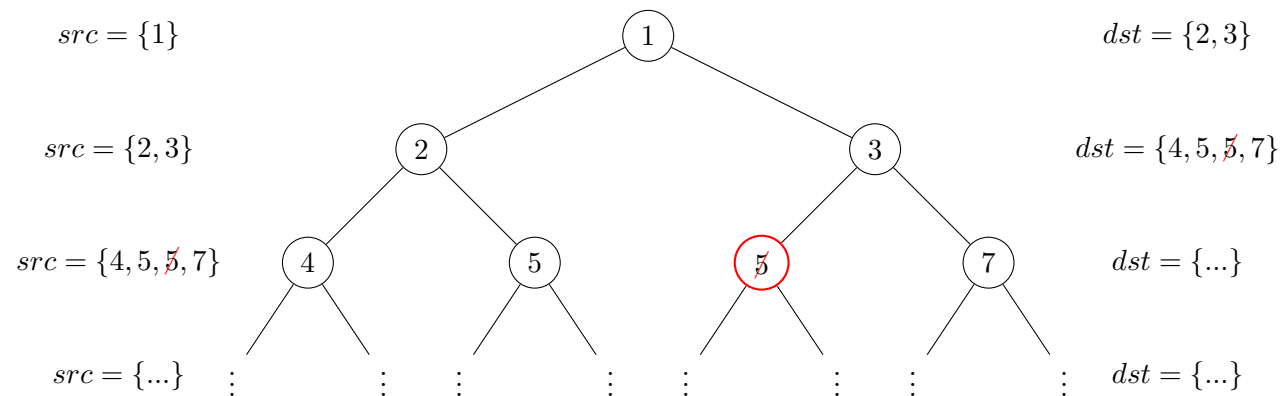
# Postgres Architektur



# Konzepte für rekursive Anfragen

Nächste Seite





Listing 1: selectRecursive

```
SELECT DISTINCT(dst) FROM team22.relation_facebook WHERE src IN(
  SELECT DISTINCT(dst) FROM team22.relation_facebook WHERE src IN(
    SELECT DISTINCT(dst) FROM team22.relation_facebook WHERE src IN(1)
  )
)
```

Listing 2: selectWithJoin

```
SELECT DISTINCT(rf3.dst)
FROM public.relation_facebook rf1,
     public.relation_facebook rf2,
     public.relation_facebook rf3
WHERE rf2.src = rf1.dst
      AND rf3.src = rf2.dst
      AND rf1.src = 765;
```

### Listing 3: StoredProcedure

```
CREATE OR REPLACE FUNCTION recursivesearch(tInput integer[], iRecursionDepth integer, sTable text) RETURNS SETOF
integer AS $$
Declare
intermDst_ integer[];
iCount integer;
BEGIN
CREATE TABLE intermDst AS SELECT * FROM unnest(tInput);
EXECUTE 'CREATE TABLE intermDst1 AS SELECT DISTINCT(dst) FROM ' || sTable || ' WHERE src IN (SELECT * FROM
intermDst)';
-- Does not return from function!
return query SELECT * FROM intermDst1;
-- Does not return from function!
intermDst_ := ARRAY(SELECT * FROM intermDst1);
raise notice 'timestamp: %', clock_timestamp();
SELECT count(*) INTO iCount FROM intermDst;
raise notice 'Count Table: %', iCount;
DROP TABLE intermDst;
DROP TABLE intermDst1;
if iRecursionDepth > 1 THEN
return query SELECT * FROM recursivesearch(intermdst_, iRecursionDepth - 1, sTable);
ELSE
RETURN;
END IF;
END;
$$ LANGUAGE plpgsql;
```

Listing 4: innerJoinSourceCodeGenerator

```

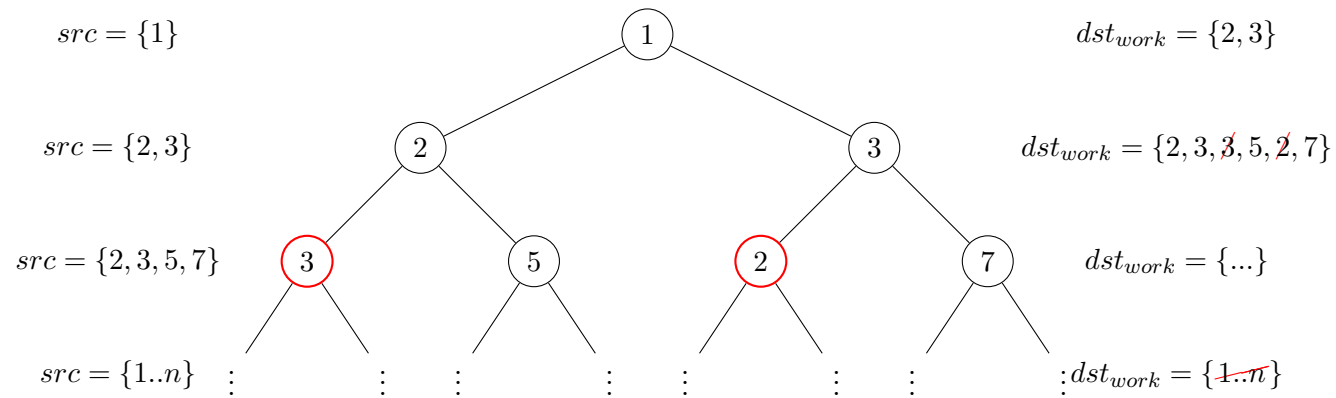
CREATE OR REPLACE FUNCTION innerJoinSourceCodeGenerator(iRecursionDepth integer, sTable text) RETURNS SETOF integer
AS $$
Declare
intermDst_ integer[];
iCount integer;
tStatement text;
tSelectStatement text;
tConcatenateStatement text;
tAlternativeStatement text;
tWhereStatement text;
tFinalStatement text;
BEGIN
iCount = 0;
tSelectStatement = '';
tWhereStatement = '';
tConcatenateStatement := 'SELECT DISTINCT(dst) FROM ' || sTable || ' WHERE src IN(';
tStatement := sTable || ' rf';
tAlternativeStatement = sTable || ' rf';
-- iCount = 0;
if iRecursionDepth = 0 THEN
raise notice 'Rekursivtiefe von 0 nicht m glich';
RETURN ;
end if;
if iRecursionDepth = 1 THEN
tConcatenateStatement = tConcatenateStatement || '765' || '));';
return query EXECUTE tConcatenateStatement;
RETURN;
end if;
WHILE iCount < iRecursionDepth LOOP
if iCount = iRecursionDepth - 1 then
tSelectStatement := tSelectStatement || tStatement || iCount || ' ';
else
tSelectStatement := tSelectStatement || tStatement || iCount || ', ';
end if;
if iCount != 0 then
if iCount = iRecursionDepth - 1 then
tWhereStatement := tWhereStatement || 'rf' || iCount || '.src = rf' || iCount - 1 || '.dst ';
else
tWhereStatement := tWhereStatement || 'rf' || iCount || '.src = rf' || iCount - 1 || '.dst AND ';
end if;
else
tWhereStatement := 'rf' || iCount || '.src = 765 AND ' ;
end if;
iCount = iCount + 1;
end loop;
tWhereStatement := 'WHERE ' || tWhereStatement;
tSelectStatement := 'FROM ' || tSelectStatement;
tFinalStatement = 'SELECT DISTINCT(rf' || iRecursionDepth - 1 || '.dst) ' || tSelectStatement || tWhereStatement;
raise notice 'FROM Statement: %', tSelectStatement;
raise notice 'Where Statement: %', tWhereStatement;
raise notice 'Finale Statement: %', tFinalStatement;
return query EXECUTE tFinalStatement;
END;
$$ LANGUAGE plpgsql;

```



Listing 5: selectSourceCodeGenerator

```
CREATE OR REPLACE FUNCTION selectSourceCodeGenerator(iRecursionDepth integer, sTable text) RETURNS SETOF integer AS
$$
Declare
intermDst_ integer[];
iCount integer;
tStatement text;
tConcatenateStatement text;
BEGIN
tConcatenateStatement := 'SELECT DISTINCT(dst) FROM ' || sTable || ' WHERE src IN(';
tStatement := 'SELECT DISTINCT(dst) FROM ' || sTable || ' WHERE src IN(765)';
-- iCount = 0;
if iRecursionDepth = 0 THEN
return query EXECUTE tStatement;
RETURN;
end if;
WHILE iRecursionDepth > 0 LOOP
tStatement := tConcatenateStatement || tStatement || ')';
iRecursionDepth = iRecursionDepth - 1;
end loop;
raise notice 'Execute String %', tStatement;
return query EXECUTE tStatement;
END;
$$ LANGUAGE plpgsql;
```



Listing 6: selectWithUnion

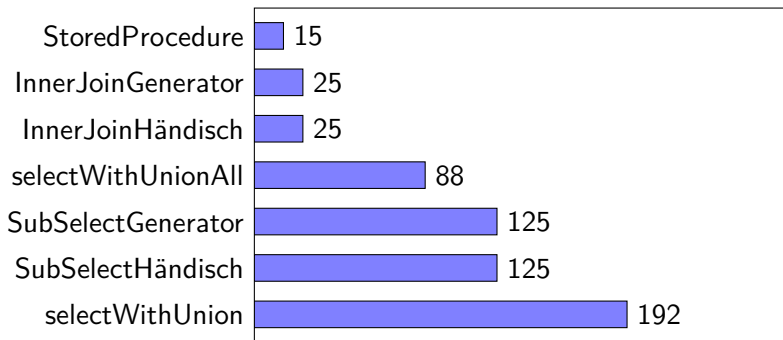
```

WITH RECURSIVE graphtraverse AS(
  SELECT DISTINCT(dst)
  FROM
    public.relation_facebook
  WHERE
    src =765
  UNION
  SELECT p.dst
  FROM
    relation_facebook p
  WHERE
    src IN ( p.src )
)
SELECT * FROM graphtraverse

```

- ▶ Tool zur Durchführung von Benchmark-Tests
- ▶ Bei einem Benchmark-Test wird eine Menge von SQL-Statements beliebig oft wiederholt.
- ▶ pgBench berechnet die Anzahl der Transaktionen pro Sekunde

# Messergebnisse pgBench



Transaktionen Pro Sekunde