CYBERTEC

DATA SCIENCE & POSTGRESQL

PostgreSQL:
INDEXING STRANGE THINGS

Hans-Jürgen Schönig

DATABASE SERVICES

DATA Science

- Artificial Intelligence
- Machine learning
- BIG DATA
- Business Intelligence
- Data Mining
- Etc.

POSTGRESQL Services

- 24x7 Support
- Training
- Consulting
- Performance Tuning
- Clustering
- Etc.

CYBERTEC Worldwide

Wiener Neustadt
AUSTRIA

Tallinn
ESTONIA

Zurich
SWITZERLAND

Montevideo
URUGUAY

Audi

CYBERTEC
DATA SCIENCE & POSTGRESQL

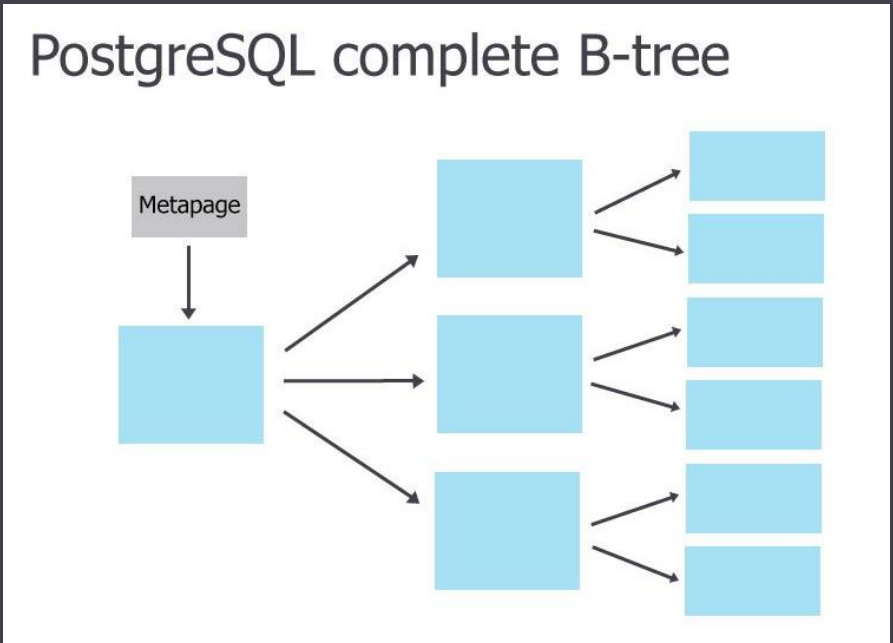
Customer **Sectors**

- University
- Automotive
- Government
- Industry
- Electronic
- Finance
- Trade
- Etc....

Btree Indexes

Btree indexes are everywhere

- A btree is the standard index in PostgreSQL
- It is basically a sorted list



https://www.cybertec-postgresql.com/wp-content/uploads/2018/01/02_PostgreSQL-Complete-B-tree.jpg

CYBERTEC
DATA SCIENCE & POSTGRESQL

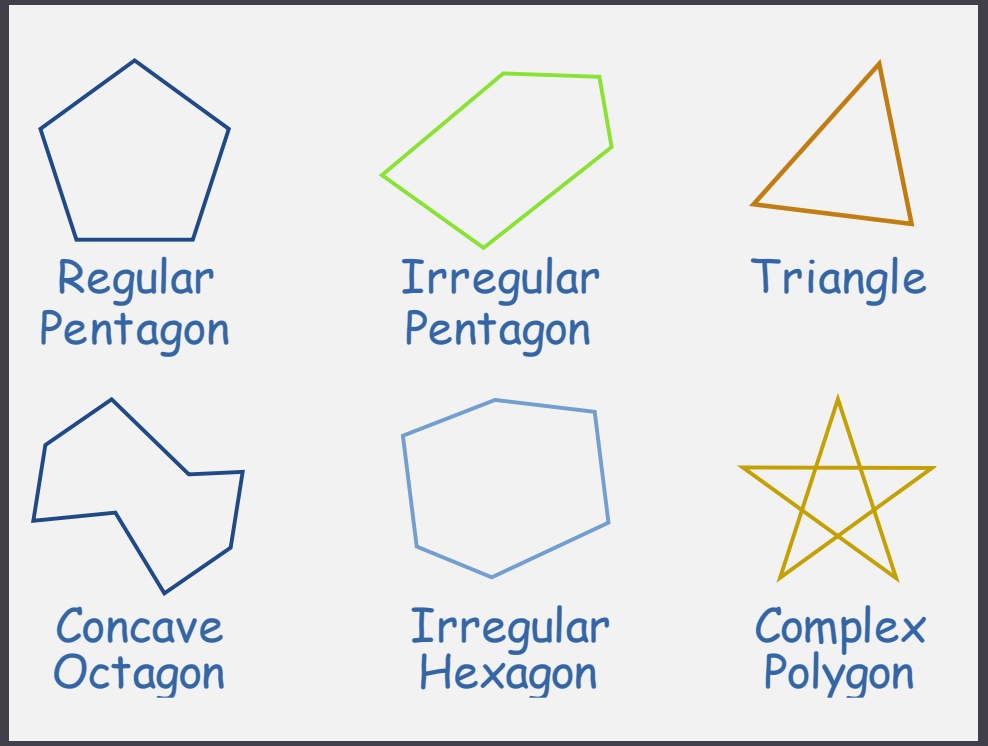
What a btree can do

- In short:
  - It can sort data

What a btree can do

- It has five “strategies”:
  - 1: <
  - 2: <=
  - 3: =
  - 4: >=
  - 5: >
- It is ideal for data that can be sorted
  - Not all data can be sorted

SPECIAL KINDS OF DATA:



https://www.mathsisfun.com/geometry/images/polygonexamples.svg

- How would you sort those objects?

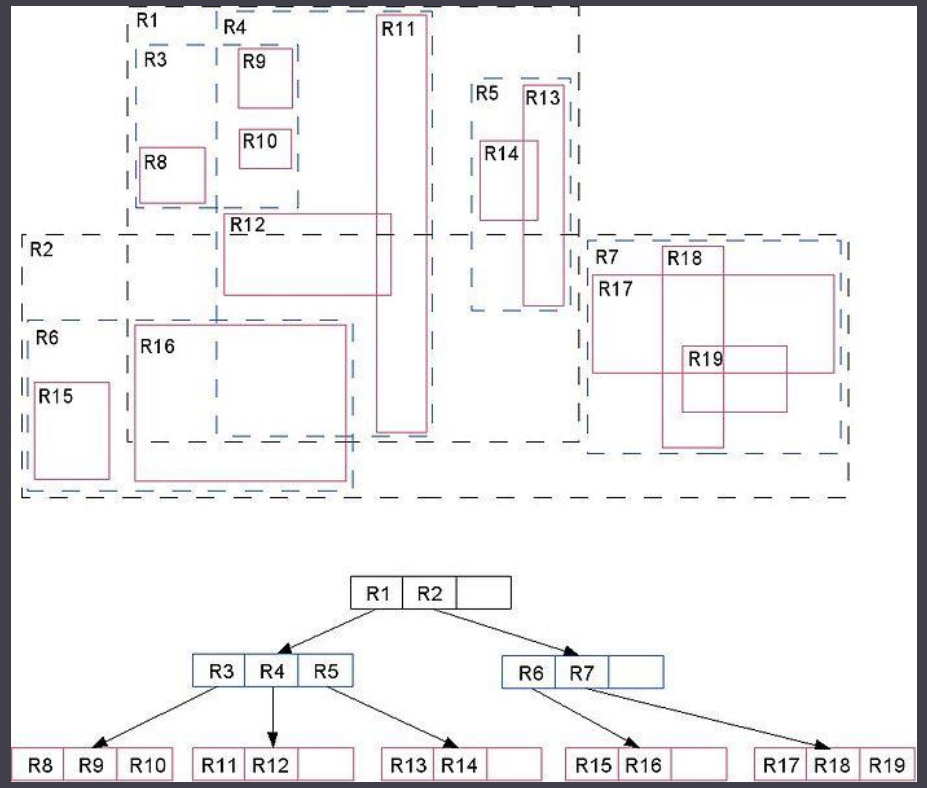
PostgreSQL INDEX TYPES

- PostgreSQL provides various types of indexes:

```
test=# SELECT amname, amtype FROM pg_am;
Amname | amtype
-------+--------
Heap   | t
btree  | i
hash   | i
gist   | i
gin    | i
Spgist | i
brin   | i
(7 rows)
```

gist: GENERALIZED INDEXING

- How Gist works



https://leopard.in.ua/assets/images/postgresql/pg_indexes/pg_indexes2.jpg

CYBERTEC
DATA SCIENCE & POSTGRESQL

gist: DIFFERENT STRATEGIES

Operation Strategy Number

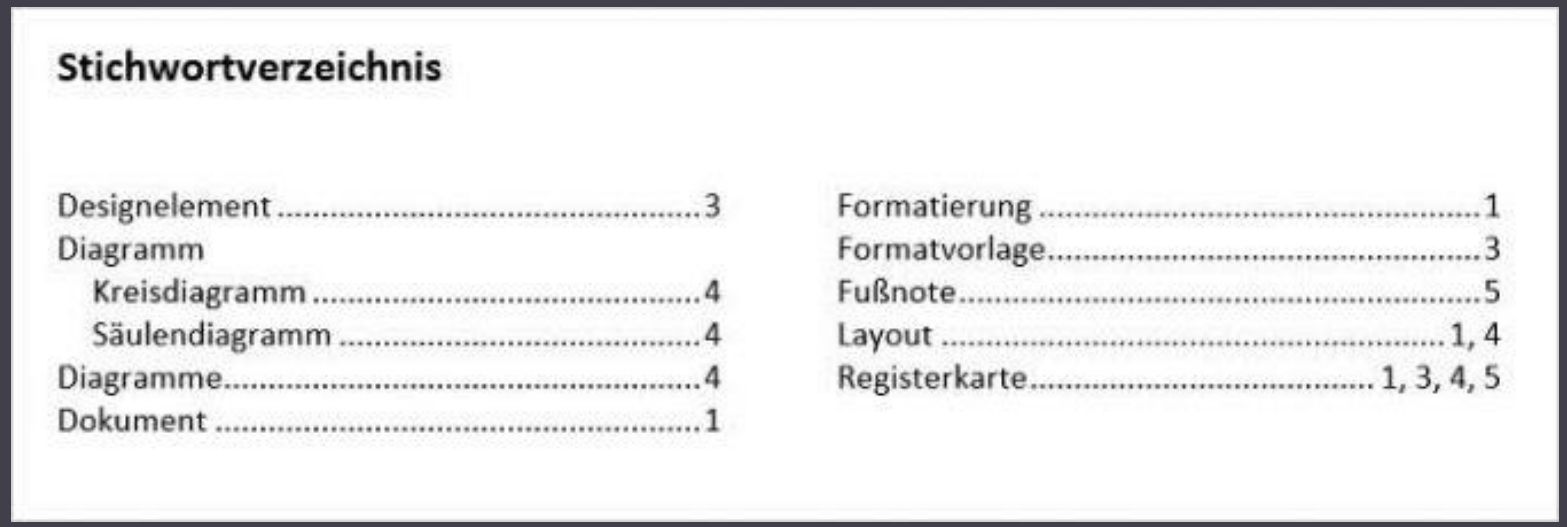
Strictly left of **1** does not extend to right of **2** overlaps **3** does not extend to left of **4** strictly right of **5** same **6** contains **7** contained by **8** does not extend above **9** strictly below **10** strictly above **11** does not extend below **12**.

gist: WHY CARE?

- Ideal for GIS data
- Ideal for similarity search ("distance")
  - pg_trgm
  - KNN search for GIS, etc.
- Ideal for generic stuff

gin: FULL TEXT INDEXING

- The "technology" has been around for a long long time



Stichwortverzeichnis

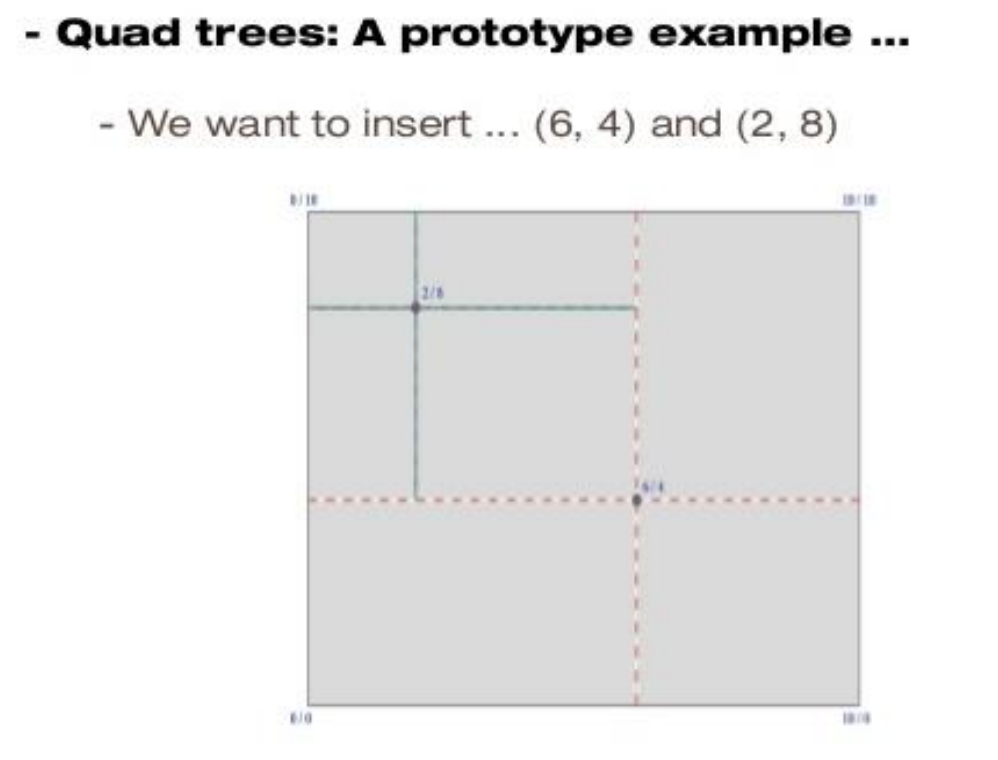
https://www.pcs-campus.de/wp-content/uploads/2013/12/word_stichwortverzeichnis_03-675x224.jpg

gin: STRATEGIES

Operation Strategy Number overlap **1** contains **2** is contained by **3** equal **4**

sp-gist: KD-trees

- Nice for prefixes and all that



https://image.slidesharecdn.com/2013dublinindexing-150312031852-conversion-gate01/95/postgresql-advanced-indexing-50-638.jpg?cb=1426130938

CYBERTEC
DATA SCIENCE & POSTGRESQL

sp-gist: STRATEGIES

Operation Strategy Number strictly left of **1** strictly right of **5** same **6** contained by **8** strictly below **10** strictly above **11**

brin:
BLOCK RANGE INDEX

- Take 128 blocks (= 1 MB)
  - Calculate minimum and maximum
- Good to handle sorted data
- Very small

Operator classes in use

Custom indexing strategy

- You can write your own operators
- Tell indexes to use your operators
- Allows to index custom stuff

Example: Social security numbers

- How would you sort that?

```
1118 09 08 78
2344 04 04 77
```

- You can only sort it if you know what it means

Indexing social security numbers

```
CREATE OR REPLACE FUNCTION normalize_si(text)
RETURNS text AS $$
        BEGIN
        RETURN substring($1, 9, 2) ||
                substring($1, 7, 2) || substring($1, 5, 2)
        END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

Indexing social security numbers

```
SELECT normalize_si('1118090878’);
 normalize_si
--------------
  7808091118
(1 row)
```

Indexing social security numbers

```
CREATE OR REPLACE FUNCTION si_lt(text, text)
RETURNS boolean AS
$$
BEGIN
            RETURN normalize_si($1) < normalize_si($2);
END;
$$ LANGUAGE 'plpgsql' IMMUTABLE;

-- lower equals
CREATE OR REPLACE FUNCTION si_le(text, text)
RETURNS boolean AS
$$
BEGIN
            RETURN normalize_si($1) <= normalize_si($2);
END;
$$ LANGUAGE 'plpgsql' IMMUTABLE;

-- greater equal
CREATE OR REPLACE FUNCTION si_ge(text, text)
RETURNS boolean AS
$$
BEGIN
            RETURN normalize_si($1) >= normalize_si($2);
END;
$$ LANGUAGE 'plpgsql' IMMUTABLE;

-- greater
CREATE OR REPLACE FUNCTION si_gt(text, text)
RETURNS boolean AS
$$
BEGIN
            RETURN normalize_si($1) > normalize_si($2);
END;
$$ LANGUAGE 'plpgsql' IMMUTABLE;
```

Indexing social security numbers

```
CREATE OPERATOR <# ( PROCEDURE=si_lt,
                     LEFTARG=text,
                     RIGHTARG=text);

CREATE OPERATOR <=# ( PROCEDURE=si_le,
                      LEFTARG=text,
                      RIGHTARG=text);

CREATE OPERATOR >=# ( PROCEDURE=si_ge,
                      LEFTARG=text,
                      RIGHTARG=text);

CREATE OPERATOR ># ( PROCEDURE=si_gt,
                     LEFTARG=text,
                     RIGHTARG=text);
```

Indexing social security numbers

```
CREATE OR REPLACE FUNCTION si_same(text, text)
        RETURNS int AS
$$
        BEGIN
                IF      normalize_si($1) < normalize_si($2)
                THEN
                        RETURN -1;
                ELSIF   normalize_si($1) > normalize_si($2)
                THEN
                        RETURN +1;
                ELSE
                        RETURN 0;
                END IF;
        END;
$$ LANGUAGE 'plpgsql' IMMUTABLE;
```

Indexing social security numbers

```
CREATE OPERATOR CLASS sva_special_ops
        FOR TYPE text USING btree
AS
        OPERATOR        1       <#   ,
        OPERATOR        2       <=# ,
        OPERATOR        3       =   ,
        OPERATOR        4       >=# ,
        OPERATOR        5       >#   ,
        FUNCTION        1       si_same(text, text)
;
```

Indexing social security numbers

```
CREATE INDEX idx_special ON t_sva (sva sva_special_ops);

SET enable_seqscan TO off;

explain SELECT * FROM t_sva WHERE sva = '0000112276';
                    QUERY PLAN
-----------------------------------------------
 Index Only Scan using idx_special on t_sva
       (cost=0.13..8.14 rows=1 width=32)
   Index Cond: (sva = '0000112276'::text)
(2 rows)
```

CEO
Hans-Jürgen Schönig

MAIL hs@cybertec.at
PHONE +43 2622 930 22-2
TWITTER @PostgresSupport

CYBERTEC
DATA SCIENCE & POSTGRESQL