EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

DB2
Sommersemester 2022
Prof. Torsten Grust, Denis Hirn
WSI — Database Systems Research Group

Assignment 9

Handin until: Friday, 15.07.2022, 09:00

1. [20 Points] **Performance Impact of Indexes**

   Indexes have impact on the performance of data modifying operations. But can an additional index also **decrease** the performance of a `SELECT`-query? Consider the following table:

```
CREATE TABLE skewed (
  sort        INTEGER NOT NULL,
  category    INTEGER NOT NULL,
  interesting BOOLEAN NOT NULL
);

INSERT INTO skewed
  SELECT i AS sort, i % 1000 AS category, i > 50000 AS interesting
  FROM   generate_series(1, 1000000) i;

CREATE INDEX skewed_category ON skewed (category);
ANALYZE skewed;
```

   We now want to query the first twenty interesting rows in category 42:

```
SELECT *
FROM   skewed
WHERE  interesting AND category = 42
ORDER  BY sort
LIMIT  20;
```

   (a) Use `EXPLAIN ANALYZE` to get the query plan and describe in your own words how query evaluation proceeds and how this is supported by index `skewed_category`.

   In general, it is a good idea to support top-N queries (`ORDER BY`/`LIMIT`) with an index on the sort criteria. Let's test this!

   (b) Create an additional B$^+$Tree index `skewed_sort` on column `sort` of table `skewed`.

   (c) Index `skewed_sort` will decrease the performance of the query. **But why?**
   Again, print the query plan, explain the evaluation strategy in your own words, and compare the estimated costs to the plan of (a). Answer the following questions:

   - Why is it — in general — a good idea to use the index on `sort` to support this query?
   - What is the cause of the drop in performance in the specific example?

   (d) Is there another index that incorporates the idea of using the sort order, but does not suffer the problem of (c)? Find the index that best matches the given query. Measure and explain its benefit compared to (a).

2. [10 Points] **UB-Trees: B⁺Tree on Z-order values**

   One way to index a table on multiple columns is to use *composite indexes*. However, these do not support both dimensions equally. Instead, the first dimension dominates the order of entries.

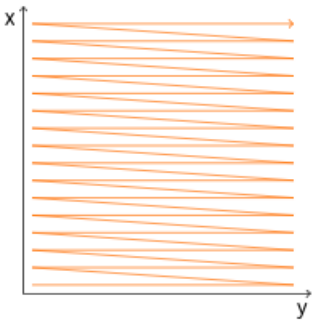
   Consider a table `points(x INT, y INT)` representing points in a two-dimensional space:

```
-- A two dimensional space
CREATE TABLE points (x INT, y INT);

-- Populate space with points
INSERT INTO points
  (SELECT x, y
   FROM   generate_series(0,99) AS x,  generate_series(0,99) AS y);

-- Create primary key index
CREATE UNIQUE INDEX points_x_y ON points USING btree (x,y);
```

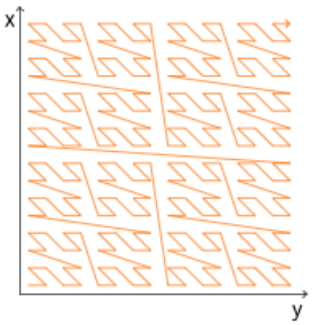
   The *composite index* reflects spacial locality of points in a two-dimensional space only on the first dimension `x`. Locality regarding *both* dimensions can be achieved with an *expression index* that combines both dimensions in a single locality-preserving value, called *Z-order value*:

```
CREATE INDEX points_z_value_x_y ON points USING btree (z_order_value(x,y));
ANALYZE;
```

   The value of the Z-order of a point `(x,y)` is calculated using the bit-interleaving function provided in `zorder.sql`[1]. This reflects locality for both input dimensions:



$\langle x,y \rangle$ (concatenated)



Z-Order

   For both B⁺Tree indexes, `points_x_y` and `points_z_value_x_y`:

   - Choose an arbitrary leaf page. Use function `bt_page_stats(indexname, pageno)` provided by extension `pageinspect` to identify a leaf page.
   - Collect all points (x,y) referred by this leaf page. Use function `bt_page_items(indexname TEXT, pageno INT)` provided by `pageinspect` to access the *RID*s of all items on the B⁺Tree page.
   - Illustrate the (x,y) location of these points in the two-dimensional space by plotting them on a grid of size 100×100 with *Gnuplot*.

     You can access a web-based version of Gnuplot at http://gnuplot.respawned.com.[2]
   - Compare the two plots: describe and explain your findings **briefly**.

[1] See Wikipedia for more information on Z-order values: https://en.wikipedia.org/wiki/Z-order_curve

[2] Examples for *"Data"* and *"Plot script"* can be found in files `data.txt` and `points.plot`.