EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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

Assignment 8

Handin until: Friday, 08.07.2022, 09:00

1. [5 Points] **Page Organization**

   Load file `page-organization.sql` to create table `ternary(a INT, b TEXT, c NUMERIC(3,2))`. Note that the tuples of table `ternary` are inserted sorted by columns `a` and `c` (in this order).

   Consider the following two queries:

```sql
SELECT *
FROM   ternary AS i
WHERE  i.a < 4242;
```

```sql
SELECT *
FROM   ternary AS i
WHERE  i.c = 0.42
```

   (a) Modify both queries to determine the number of pages over which the rows of the result are distributed. Use the function `page_of(rid tid)` to get the page numbers.

   (b) Briefly explain the different results of the two queries in task (a). Can PostgreSQL benefit from fact that the tuples are inserted sorted?

2. [5 Points] **B⁺Tree Prefix Compression**

```sql
DROP TABLE IF EXISTS t;
CREATE TABLE t (a text, b int);

INSERT INTO t(a, b)
...
;

CREATE INDEX t_a ON t USING btree (a);
ANALYZE t;
```

   Fill table `t` with *(two)* different sets of rows that allow you to deduce whether PostgreSQL implements *B⁺Tree Prefix Compression* on `text` columns. Hand in *all* SQL commands that you have used to perform this experiment.

3. [5 Points] **B⁺Tree - Variation**

   In this task, we assume that all key values inserted into the B⁺Tree are unique and the `Insert`-operation does **not implement** ***redistribution***.

   Find five *additional* key values $a \ldots e$ such that after inserting the entries in order $a \ldots e$ and then deleting them in reverse order $e \ldots a$, the following holds:

   (a) The resulting tree is the **same** as in Figure 1.

   (b) The resulting tree is **different** from the one in Figure 1.

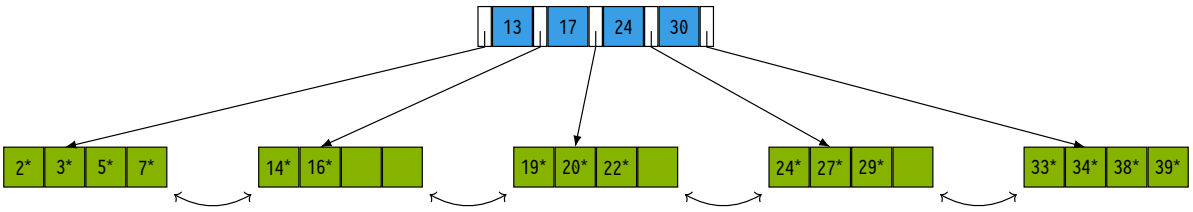

Figure 1: A B⁺Tree

4. [15 Points] **Solving Predicate Equations**

   Create and populate table `unary(a INTEGER PRIMARY KEY)` as provided in `unary.sql`.

   Examine the following four queries. Assume that variable `:n` is not `NULL` and that it is set to an *arbitrary* integer value (for example: `\set n 64`).

   Q_1:

```sql
SELECT u.a
FROM unary AS u
WHERE u.a - 1 = :n;
```

   Q_2:

```sql
SELECT u.a
FROM unary AS u
WHERE (u.a::double precision)^2 = :n;
```

   Q_3:

```sql
SELECT u.a
FROM unary AS u
WHERE u.a % :n = 1;
```

   Q_4:

```sql
SELECT u.a
FROM unary AS u
WHERE :n / u.a = 1;
```

   For each of the queries, answer the following questions and briefly explain your answer:

   (a) Is the query evaluation supported by the `unary_a` index?

   (b) If not, try to find an **equivalent** query that is supported by the `unary_a` index. If there is no such query, explain why. Be careful not to change the query semantics. This means that any rewritten query must return exactly the same results as the original query for all *possible* instances of table `unary`, and all *possible* values of `:n`.

   (c) Instead of rewriting the query, an *expression index* can be used to support the query. Define this index and check if it is actually used. What is the disadvantage of defining individual expression indices with respect to all four queries?