

## Aufgabe 1: Page Reference String (PRS) Analysis (1 P.)

a) Given the following PRS:

$\langle (T2,D) (T2,G) (T1,C) (T1,D) (T1,B) (T1,C) (T2,C) (T1,C) (T1,B) (T1,C) (T2,E) (T1,B) (T1,C) (T1,D) (T2,A) (T2,G) (T2,F) \rangle$

Calculate the sequentiality and locality. Use the working-set-method for locality and calculate the average over both transactions. Assume a window size of  $\tau = 3$ .

b) Given the following stack depth distribution:

Stack depth	Frequency	Stack depth	Frequency
1	521	6	121
2	413	7	89
3	311	8	56
4	265	9	37
5	143	10	17

Answer the following questions:

- How large should the buffer be, if we want no more than 7.5% of cache-misses?
- How many cache hits (in %) will there be, given a buffer size of 5?
- How many cache misses are avoided, if the buffer size is increased from 5 to 7?

## Aufgabe 2: Replacement Strategies (1 P.)

Given the following sequence of page accesses:

**C, B, D, D, A, A, C, B, A, B, C, A, D, B, D, B, A**

- Given a buffer size of 3, calculate the amount of cache misses for the strategies CLOCK, LRU- $k$  with  $k = 2$  and GCLOCK. For GLOCK, assume an initial value  $E_i = 2$  for every page and increment of  $W_i = 1$  on reference.
- Which replacements would happen for the given sequence if an optimal replacement strategy, which knows all future page accesses beforehand<sup>1</sup>, was used?
- Given a buffer size of 2: Find one example of a PRS for which FIFO has less cache **misses** than LRU, and one for which FIFO has less **hits** than LRU.

<sup>1</sup>Such a strategy (clairvoyant) can, of course, not be implemented, but it can be compared to other replacement strategies to benchmark their performance.

## Aufgabe 3: Performance in PostgreSQL and pgAdmin (1 P.)

In this and the following exercise sheets we will use the relational database system *PostgreSQL*. PostgreSQL is available free of charge under an open source license, for example at <http://www.postgresql.org/> as installable packages or for Linux distributions via the common package managers (`apt-get`, `yum`, `pacman`, ...). Install PostgreSQL (version 10 or higher) in a way appropriate to your system.

*pgAdmin* is a graphical user interface for PostgreSQL, which simplifies its use and administration. Also install this program (<http://www.pgadmin.org/> or package manager).

Load the two data records linked on the website into the database. Take the opportunity to refresh your SQL knowledge and make a few queries against the database. First, run `VACUUM ANALYZE`. This request has no result, however, it immediately creates statistics about the imported tables.

### Indices and their effects

You can retrieve the query plan, which is created by the database by using “Explain” (German “Analysieren”) in the query tool. The actual query results are not needed for this exercise. Use the TPC-H dataset for the following queries.

- a) Which **query plan** is generated for the following query? Please describe the plan as detailed as possible. HINT: If you hover your mouse over an element in the query plan, a tool tip, with additional information, will be shown.

```
SELECT * FROM lineitem WHERE l_suppkey < 10
```

- b) How will the plan change, if you create the following index?

```
CREATE INDEX index1 ON lineitem(l_suppkey);
```

- c) By prefixing your query with `EXPLAIN (ANALYZE, BUFFERS)`, PostgreSQL will explain give additional information on **DB buffer usage**. How does the buffer usage change if you run the following query three times? Give the output row of `Buffers` ... each time.

```
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM orders WHERE o_totalprice < 100
```

- d) PostgreSQL ships with an extension which allows “**prewarming**” a table, which means filling the buffer with relevant pages. Execute the following queries and again give the `Buffers` ... row. Explain what the output means and why there may still pages be read from the disk.

```
CREATE EXTENSION pg_prewarm;  
SELECT pg_prewarm('orders');  
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM orders WHERE o_totalprice < 100
```