PA152: Efficient Use of DB

1. Introduction

Vlastislav Dohnal

Motivation

- End-users report slow responses

```
SELECT  s.RESTAURANT_NAME, t.TABLE_SEATING, to_char(t.DATE_TIME,'Dy, Mon FMDD') AS THEDATE,
to_char(t.DATE_TIME,'HH:MI PM') AS THETIME,to_char(t.DISCOUNT,'99') || '%' AS AMOUNTVALUE,t.TABLE_ID,
s.SUPPLIER_ID, t.DATE_TIME, to_number(to_char(t.DATE_TIME,'SSSSS')) AS SORTTIME
FROM TABLES_AVAILABLE t, SUPPLIER_INFO s,
      (SELECT s.SUPPLIER_ID, t.TABLE_SEATING, t.DATE_TIME, max(t.DISCOUNT) AMOUNT, t.OFFER_TYPE
       FROM TABLES_AVAILABLE t, SUPPLIER_INFO s
       WHERE t.SUPPLIER_ID = s.SUPPLIER_ID
            and (TO_CHAR(t.DATE_TIME, 'MM/DD/YYYY') != TO_CHAR(sysdate, 'MM/DD/YYYY')
             or TO_NUMBER(TO_CHAR(sysdate, 'SSSSS')) < s.NOTIFICATION_TIME - s.TZ_OFFSET)
            and t.NUM_OFFERS  > 0 and t.DATE_TIME > SYSDATE  and s.CITY = 'SF'
            and t.TABLE_SEATING = '2’ and t.DATE_TIME between sysdate and (sysdate + 7)
            and to_number(to_char(t.DATE_TIME, 'SSSSS')) between 39600 and 82800
            and t.OFFER_TYPE = 'Discount‘
       GROUP BY s.SUPPLIER_ID, t.TABLE_SEATING, t.DATE_TIME, t.OFFER_TYPE) u
WHERE t.SUPPLIER_ID=s.SUPPLIER_ID and u.SUPPLIER_ID=s.SUPPLIER_ID and t.SUPPLIER_ID=u.SUPPLIER_ID
      and t.TABLE_SEATING = u.TABLE_SEATING and t.DATE_TIME = u.DATE_TIME
      and t.DISCOUNT = u.AMOUNT and t.OFFER_TYPE = u.OFFER_TYPE
```

```
            and (TO_CHAR(t.DATE_TIME, 'MM/DD/YYYY') != TO_CHAR(sysdate, 'MM/DD/YYYY')
          or TO_NUMBER(TO_CHAR(sysdate, 'SSSSS'))  < s.NOTIFICATION_TIME - s.TZ_OFFSET)
    and t.NUM_OFFERS > 2 and t.DATE_TIME > SYSDATE and s.CITY = 'SF'
    and t.TABLE_SEATING = '2'  and t.DATE_TIME between sysdate and (sysdate + 7)
    and to_number(to_char(t.DATE_TIME, 'SSSSS')) between 39600 and 82800 and t.OFFER_TYPE = 'Discount'
ORDER BY AMOUNTVALUE DESC,  t.TABLE_SEATING ASC, upper(s.RESTAURANT_NAME) ASC,
          SORTTIME ASC, t.DATE_TIME ASC
```

Motivation (cont.)

```
Execution Plan
----------------------------------------------------------
  0     SELECT STATEMENT Optimizer=CHOOSE (Cost=165 Card=1 Bytes=106)
  1    0   SORT (ORDER BY) (Cost=165 Card=1 Bytes=106)
  2    1     NESTED LOOPS (Cost=164 Card=1 Bytes=106)
  3    2       NESTED LOOPS (Cost=155 Card=1 Bytes=83)
  4    3         TABLE ACCESS (FULL) OF 'TABLES_AVAILABLE' (Cost=72 Card=1 Bytes=28)
  5    3         VIEW
  6    5           SORT (GROUP BY) (Cost=83 Card=1 Bytes=34)
  7    6             NESTED LOOPS (Cost=81 Card=1 Bytes=34)
  8    7               TABLE ACCESS (FULL) OF 'TABLES_AVAILABLE' (Cost=72 Card=1 Bytes=24)
  9    7               TABLE ACCESS (FULL) OF 'SUPPLIER_INFO' (Cost=9  Card=20 Bytes=200)
 10    2       TABLE ACCESS (FULL) OF 'SUPPLIER_INFO' (Cost=9 Card=20 Bytes=460)
```

Access method

Evaluation costs

Course's Objectives

- Understand query processing
  - Identify weaknesses and optimize
- Implement advanced indexing
- Design a data model for a specific query workload
- Understand and implement high availability

Course Outline

- Data storage

  Storage hierarchy, RAID, failures, …

- Query processing

  Cost estimates, joining relations, …

- Indexing

  Trees, hashing, …

- Query optimization

  Creating indexes, views, table partitioning, …

- Database tuning

  Tuning relation schema, db monitoring, …

Course Outline

- Implementation of data model
  - Row-oriented vs column-oriented
- Transaction processing

  Concurrent processing, locking, logging, deadlocks, …
- Access control

  Access rights, roles, rewrite rules, …
- High availability

  Concept, approaches, …

Recommended Reading

- Books
  - Database Systems Implementation
    - Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
    - Prentice Hall, 2000
    - Signature in FI library: D89
  - Database Systems: The Complete Book
    - Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
    - 2nd edition, Prentice Hall, 2009
    - Signature in FI library: D147

Credits

- Materials are based on presentations:
  - Courses CS245, CS345, CS345
    - Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
    - Stanford University, California

Requirements to pass

- There are two possible completion types
  - Exam (‘zk’)
  - Credit (‘z’)
  - Double-check the requirements of your study program
- Homework and exam
  - see in course's study materials

Knowledge Prerequisites

- Relational model
- Query languages
  - SQL and relational algebra
- File organizations
  - Sequential file, …
- Mostly covered in bachelor-degree courses:
  - PB154 Fundamentals of Database Systems
  - PB168 Fundamentals of Information and Database Systems

Basic Terminology

- „Database“
  - “programmed” by most of the coders
  - needed by every business
  - included in most applications
  - queried by SQL

- Relational model
  - Structure – data in relations (tables)
  - Operations – query specification (read, write, delete)
    - SQL, relational algebra
- Database system (Database management system = DBMS)
  - a collection of tools for storing and processing data
- Database
  - data that are processed, interesting for a business
  - collection of relations, integrity constraints, indexes, …
  - database schema vs. database instance

Example

```
select * from student where
  program='N-SWE' and field='DEV'
```

- Query processing:
  1. Check the query and get attributes of *student*
  2. Read rows from *student* and evaluate the condition

No	UČO	Name	Type of completion	Covid	Program	Field
1.	424370	**Martin**	zk	● ONT	N-SWE	DEV
2.	445991	**Laura**	zk	● ONT	N-SWE	DEV
3.	516803	**Martin**	zk	● ONT	N-SWE	DEV
4.	405154	**Peter**	zk	● ONT	N-SWE	DEV
5.	475723	**Martin**	zk	● ONT	N-SWE	DEV
6.	424448	**Michal**	zk	● ONT	N-SWE	DEV

Example 2

`select * from R,S where` *`condition`*

- Query processing
  1. Read dictionary and get attributes of R and S
     a. Check condition correctness
  2. Read R and for each line do:
     a. Read S and for each line do:
        i. Evaluate condition
        iii. If valid, join the lines, and add to result

Implementation of example

- Relations stored in files on disk
  - Relation *student* in `/var/db/student`

```
424370 # Martin # zk # ONT # N-SWE # DEV↵
445991 # Laura # zk # ONT # N-SWE # DEV↵
516803 # Martin # zk # ONT # N-SWE # DEV↵
405154 # Peter # zk # ONT # N-SWE # DEV↵
475723 # Martin # zk # ONT # N-SWE # DEV↵
424448 # Michal # zk # ONT # N-SWE # DEV↵
  ⋮
```

  - Data schema is not present

Implementation of example

- Relations stored in files on disk
  - Relation *student* split to multiple files
    - `/var/db/student.1`

```
424370 # 445991 # 516803 # 405154 # 475723 # 424448 # …
```

    - `/var/db/student.2`

```
Martin # Laura # Martin # Peter # Martin # Michal # …
```

    - `/var/db/student.3`

```
zk # zk # zk # zk # zk # zk # …
```

    - …
  - Data schema is not present

Implementation of example

- List of existing relations in *dictionary*
  - `/var/db/dictionary`

```
student # UČO # INT # Name # STR # Type
of completion # STR # Covid # STR #
Program # STR # Field # STR↵
R # name # STR # id # INT # dept STR … ↵
S # C # STR # A # INT … ↵
    ⋮
```

Implementation Issues

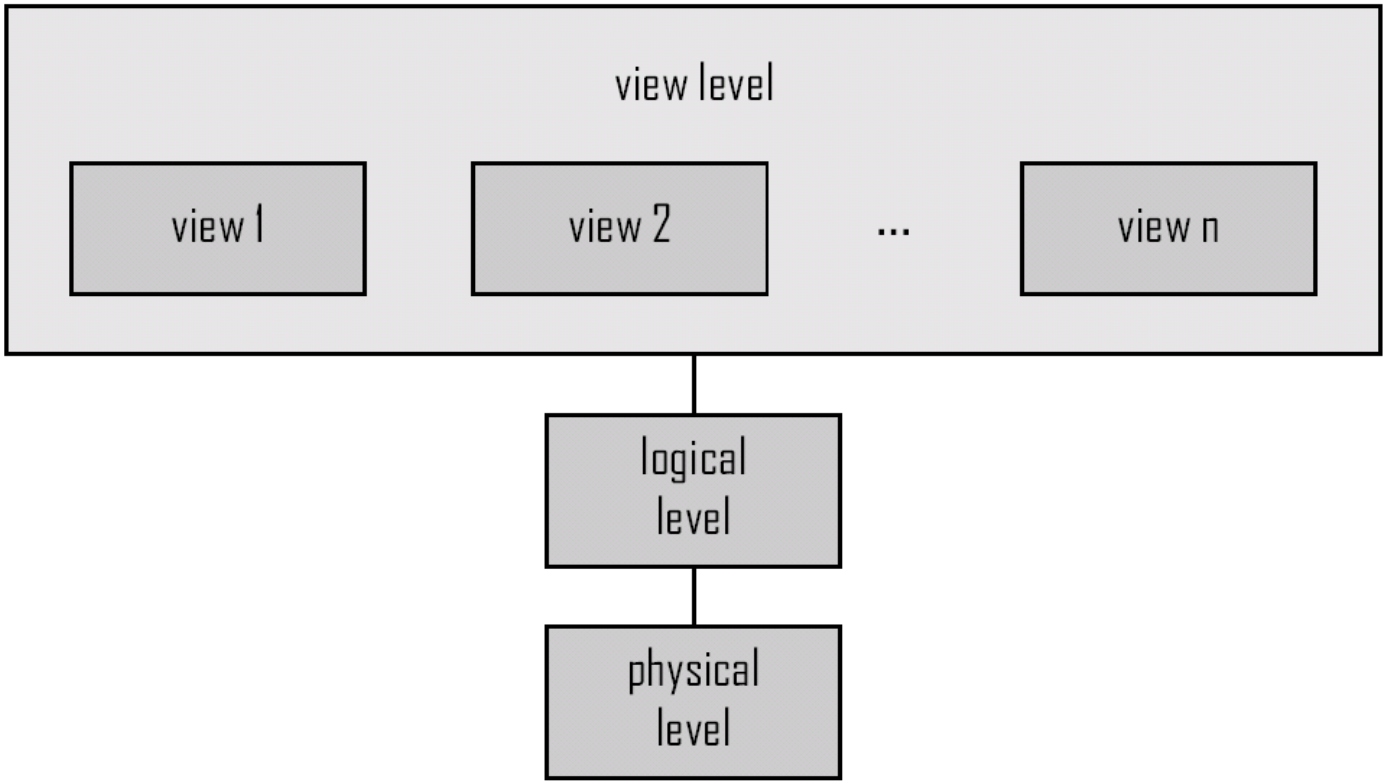
- Storage
  - Bizarre formatting – separators and new lines
    - Change in a value leads to change in whole file
  - Respecting underlying technology?
- Querying costly
  - Indexes?
- Data consistency
  - Primary keys, references, …

Implementation Issues

- Concurrent processing
- Reliability
  - Data loss can be frequent
  - Operation may not be finished completely
- Access control
  - Accessed checked on file-system level only
  - Rights might be too coarse
- API, GUI?

Database System

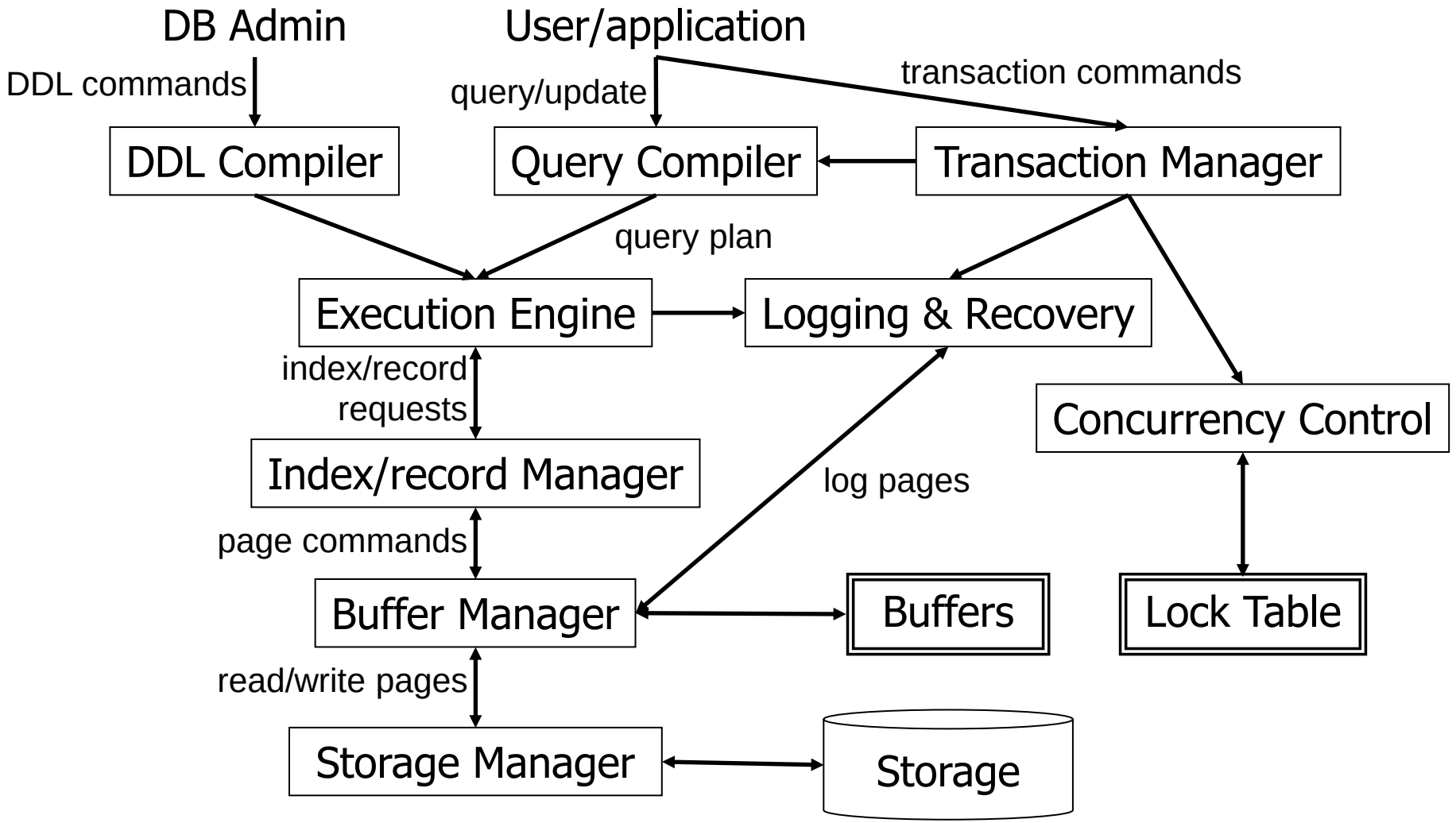
- DBMS (Database Management System)
  - Architecture:



DBMS Main Components

- Storage Manager
  - manages blocks on disks
  - manages buffers/caches
- Query Processor
  - query translation, optimization
  - query execution
- Transaction Manager
  - atomicity, consistency, isolation, durability of transaction processing

DBMS Components



Lecture's Takeaways

- Revision of
  - Terminology
  - Database system architecture and its components