RWTH AACHEN UNIVERSITY

Implementation of Databases
Assignment #1

Due on November 1, 2016

Universitätsprofessor Dr. rer. pol. Matthias Jarke

Dr. rer. nat. Christoph Quix

Submitted by:

Sanchit Alekh, Idil Esen Zülfikar, Nihal Hegde

MSc. Software Systems Engineering

Problem 1

Part 1

Name each of the five layers in the database architecture specified in the lecture, explain the concepts handled in each layer, and the interfaces between layers.

Solution

Following are the components (in top-down order) of the five-layer architecture of Databases:

1. **Logical Data Structures:** The Logical Data Structures layer is the highest layer of abstraction in the layered database architecture. On this layer, concepts such as relations, tuples and views have a primary role. Being the functional layer closest to the end-user, this layer is also responsible for receiving SQL queries, translating and optimising them. It also contains the external schema description and integrity rules for the database. It uses the Set-oriented Interface.

2. **Logical Access Structures:** The Logical Access Structures is the second layer in the database architecture, and it primarily deals with concepts such as records, keys, sets, and access paths. Its main role is cursor management, which enables rows in a result set to be processed sequentially. It also performs sorting of components and data dictionary. The auxiliary structures within its scope are access path data and internal schema description. It interfaces itself via the Record-oriented interface.

3. **Storage Structure:** Storage Structures is the next layer in the five-layer database architecture. It is entrusted with the management of records and indices, therefore it deals with internal records and their actual B* tree representations. It has access to auxiliary structures namely the Database Key Translation Table (DBTT), Free Place Administration (FPA) and page indices. It uses the Internal Record Interface to communicate with its adjacent layers.

4. **Page Assignment:** The Page Assignment layer moves a conceptual layer closer towards the physical storage volume. Its job is to manage buffers and segments, which are used to temporary move data between the storage disk and main memory for processing by the DBMS. Therefore, it operates on a page and segment level, making use of auxiliary structures such as page and block tables. It uses the System Buffer Interface.

5. **Memory Assignment Structures:** Memory Assignment Structures is the lowermost layer in the database architecture and closest to the physical volume. Being responsible for managing the external memory and files located on the disk, it works on the conceptual level of blocks and files. The auxiliary structures within its scope are extremely low-level data structures, namely the Volume Table of Contents (VTOC), extent tables and system catalogues. It interfaces itself via the File Interface.

Part 2

The following tasks belong to different layers, sort them so that they match the architecture top-down

Solution
Following is the correct top-down order of the tasks according to the layers of the Database Architecture where they are operated upon.

(a) **View Formulation and Management**

(b) **Logical Relation and Cursor Management**

(c) **Access Path Management**

(d) **Buffering**

(e) **Media Access**

Part 3

(a) What does data independence mean?

Solution
Data Independence refers to the ability to modify the physical or logical scheme of data without having to make changes in the application program itself. On the other hand, it also makes sure that databases provide a separate layer of abstraction, whereby changes in the application program will insulate the data from any modifications. The aim of data independence is that each higher level of the data architecture should be immune to changes at the next lower level. We can change the conceptual schema at one level without affecting the data at another level. It also means we can change the structure of a database without affecting the data required by users and programs.

(b) Why is it an important feature of database systems?

Solution
Approaches used to store data before the advent of databases, e.g. files, were not able to tackle the problem of Data Independence. By not having an effective abstract separation between the application program and the data, storing data in files made the application program and the files extremely volatile when changes were made to any one of them. E.g. the change of the logical arrangement structure of the data stored in the file would mean that the application program would have to be re-written for data retrieval. Same would hold true, if any physical change would be made to the filesystem. Databases, from their inception, were envisioned to overcome these problems, and instead be based on ACID, i.e. **Atomicity, Consistency, Isolation and Durability**. Therefore, Data Independence is an important feature of Database Systems.

(c) How is data independence achieved in the five-layered architecture?

Solution
Data Independence in the five-layered architecture is achieved by providing different layers of abstraction that helps to effectively concentrate on the substantial tasks at each level.

Moreover, it leads to localisation of procedures and data, and a virtual black box due to information hiding between the layers. The **Information Hiding Principle** is extremely important in this regard, each layer hides some information from the other layer, making them independent of each other, and therefore, changes at one level do not actuate changes in the other levels. Eg. at the Logical data structures level, position indicator and explicit relations in the schema are hidden. At the Logical access paths level, the number and kind of the physical access paths, internal representation of records etc. are hidden, and so on. These aforementioned techniques aid to realise the data independence property in Database Systems.

Problem 2

Part 1
Formulate the following queries as expressions in relational algebra.

(a) Codes of all African countries (the entire country must belong to Africa)

Solution

$\Pi_{country}(\sigma_{(continent='Africa' \wedge percentage=100)} \ encompasses)$

(b) Give the names of all lakes through which a french river rows (at least one french city must be located at the river; country code for France is F)

Solution

$\Pi_{name}(\sigma_{country='F'} \ (geo\_River \bowtie_{(geo\_River.River=Lake.River)} Lake))$

(c) The name of the deepest sea

Solution

$\Pi_{name}(Sea) - \Pi_{sea1.name} \ (\rho_{sea1}(Sea) \bowtie_{(sea2.depth>sea1.depth)} \rho_{sea2}(Sea))$

(d) The names of all countries to which the highest mountain belongs. Name the result relation as CountriesWithTheHighestMountain

Solution

$temp1 \leftarrow \Pi_{height} Mountain - \Pi_{m1.height} \ (\rho_{m1}(Mountain) \bowtie_{(m2.height>m1.height)} \rho_{m2}(Mountain))$
$temp2 \leftarrow \Pi_{country}(\sigma_{Mountain.height=temp1} \ (Mountain \bowtie_{(Mountain.name=geo\_Mountain.Mountain)} geo\_Mountain))$
$\Pi_{name}(\sigma_{(code \in temp2)} \ Country)$

Part 2
Now for the further set of tasks as stated below, formulate the tuple relational calculus and SQL queries. **(Note: The SQL Queries are based on *MySQL* syntax)**

(a) Give the codes of the countries where people speak German or English.

Solution

Tuple Relational Calculus:
$(< l.country > | \ l \in Language \wedge (l.Name = `English' \vee l.Name = `German'))$

Problem 2 continued on next page...

Structured Query Language:
SELECT Country FROM Language WHERE Name= "English" OR Name= "German";

(b) Give the list of all languages spoken in countries where Buddhists live.

Solution

Tuple Relational Calculus:
$(< l.Name >| \ l \in Language \wedge \exists r \in Religion \wedge (l.Country = r.Country) \wedge r.Name = `Buddhist')$
Structured Query Language:
SELECT Name FROM Language WHERE Country IN (SELECT Country FROM Religion WHERE Name = "Buddhist");

(c) Give the list of all rivers, not sourcing in Europe

Solution

Tuple Relational Calculus:
$(< gs.River >| \ gs \in geo\_Source \wedge \exists e \in encompasses \wedge (gs.Country = e.Country) \wedge e.Continent \neq `Europe')$
Structured Query Language:
SELECT River FROM geo_Source WHERE Country IN (SELECT Country FROM encompasses WHERE Continent != "Europe");

(d) Give the list of countries, the name of their lakes and the name of their mountains. Include countries that have mountains but no lakes and vice versa, but do not include countries that have neither mountains nor lakes.

Solution

Tuple Relational Calculus:
$(< c.Name, gl.Lake, gm.Mountain >| \ c \in Country \wedge \exists gl \in geo\_Lake \wedge c.Code = gl.Country) \wedge \exists gm \in geo\_Mountain \wedge c.Code = gm.Country \wedge \neg(gl.Lake = \emptyset \wedge gm.Mountain = \emptyset)$
Structured Query Language:
SELECT DISTINCT c.Name, gl.Lake, gm.Mountain FROM Country as c LEFT JOIN geo_Lake as gl ON c.Code=gl.Country LEFT JOIN geo_Mountain as gm ON c.Code = gm.Country WHERE gl.Lake IS NOT NULL OR gm.Mountain IS NOT NULL;