

Conceptual Architecture of LEAP Database Management System

Analyzed by

BASEM SHIHADA

University of Waterloo
Dept. of Computer Science
200 University Avenue West
Waterloo, Ontario, Canada
(519) 8851211 ext. 5387

CS798 Assignment #1

29th January 2002

Table of Contents

1 ABSTRACT

2 INTRODUCTION

3 CONCEPTUAL ARCHITECTURE

3.1 Architecture view for general relational database model

3.2 User view of LEAP's Architecture

3.2.1 User Command Interface

3.2.2 Query Language Operation

3.2.3 Low-level Operation

3.3 Conceptual Architecture of LEAP

3.3.1 User I/O Command Interface

3.3.2 Error Handler

3.3.3 Parser

3.3.4 Expression Evaluation

3.3.5 Relational Language

3.3.6 Hashing

3.3.7 Tuple Attribute

3.3.8 Stack

3.3.9 Memory Buffer

3.3.10 Utility Unit

3.3.11 Database

4 SCENARIOS

4.1 First Scenario

3.4 Two Scenario

5 DATA DICTIONARY

6 EVOLVABILITY

7 REFERENCES

1 Abstract:

Open source software requires high-level component description, inter-component relation description and a detailed component description. Most of the current open source implementations suffer from the availability of a comprehensive description of the software component. This results a miss-concept in the architecture of the software or a future mistakes in determining how the software components communicate between each other. Furthermore, by the growth of the software, the developer becomes unable to identify or to locate the software components relations, which will make the future modification sometimes very expensive. This results that some open source software goes to end.

This assignment presents a conceptual architecture for the open source LEAP Database Management System. “LEAP is a relational database management system (RDBMS). It is used as an educational tool around the world to help students, and assist both researchers and teachers as they study or teach databases”. LEAP is written in C with an approximate 30,100 line of code (without counting the .h files).

The goal of this assignment is to go through the functionality of LEAP and build a conceptual architecture to this DBMS. This assignment discusses the software major components, the component relations and the whole picture of the software architecture.

2 Introduction

LEAP is a relational database management system, designed as an interpreter for the relational algebra. It represents data as being stored in tables also called *relations*. A row in the relation is called *tuple*. Columns in a relation are called *attributes*. As a certain tuple is needed from the relation, a tuple is selected depends on an identifier value, a new operation is created and a tuple is extracted from the relation. This operation is called the *select* operation, which is based on a Structured Query Language (SQL). The general conceptual components of the relational database are, the user application software, the database management system and the actual database (file system stored in the physical device).

Within LEAP, the various relations are divided into separate databases. A database is a collection of logically related relations. The relations within a database are usually related to one another. LEAP's databases are relatively simple, and there are two special databases: master and user. Master contains the data dictionary. User is the default database the user connects to. Also the standard LEAP distribution software contains three additional databases: date, stanczyk and korth.

This assignment is outlined as follows; section 3 surveys the traditional relational database management systems, with a summary of the basic characteristics of the software architecture. It also addresses the parameters required to build a relational database management system. Moving to cover the user prospective view of the system. Then focuses on LEAP's modules, with a description of each module and draw a conceptual architecture of LEAP. Section 4 specifically illustrates two different scenarios to show how LEAP's components interact with each other. Section 5 lists some word abbreviations and important definitions. Section 6 mainly shows what LEAP's developers are planning to do to enhance LEAP for the future use and shows the level of flexibility for future updates.

3 Conceptual Architecture

3.1 Architecture view for general relational database model

This section will provide an overview of the relational database management systems platform. For a database management system to be considered relational, the system is required to provide the basic relational model structure, the data management unit (add, delete, modify), and a relational algebra language such as SQL. In other words, a RDBMS is built on three-schema structure, external, conceptual, and internal. The external schema is the user interface with RDBMS, keeping the user away from how the information is physically stored or retrieved. The conceptual schema is the interface to define the different database relations. The internal schema consists of physical organization of the information. Figure 1 illustrates the three RDBMS layers.

In the coming sections LEAP will be analyzed depending on the basic RDBMS structure. Every future stage of the analysis will take each layer and split it down to detailed fictional modules. Then finally a conceptual architecture for the software will be constructed.

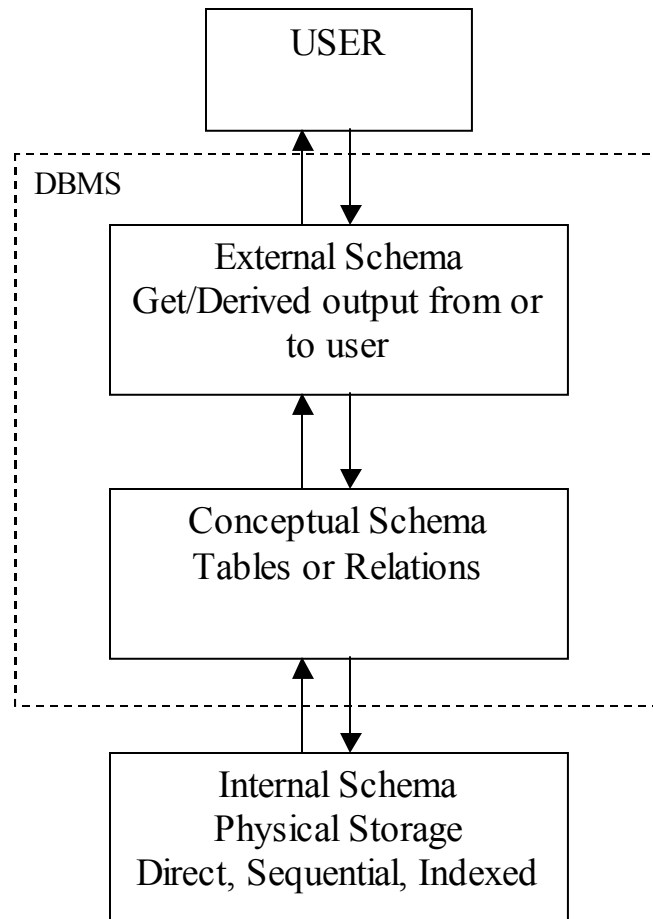


Fig. 1 Conceptual Architecture for a RDBMS

3.2 User's view of LEAP's Architecture:

From the user prospective, the system basically consist of three layers, figure 2 illustrates the software user's view:

3.2.1 User command interface

The user interface enables the user to interact with the database system in a command driven manner. This interface tells the user what's running, which options have been set, and what operation is being executed.

Within this interface the user can view different databases, for example open stanczyh database and view its relations, and print a particular relation.

The user enters the command through the command line interface. The commands are interpreted and a result is translated to a query language mode. The query language operation uses the low-level data operation with an assistant from the operating system to access the physical representation of the data, in this case the database files.

3.2.2 Query language operation

It's the middle layer, which uses the low-level operations provided from the operating system to locate a relation and extract the required data out from it or perform the resultant query operation on it.

There are five operations on the stand-alone database structure. Create new database, a database with a specified name is created, and will contain specific relations at a later stage. Remove database the specified database is removed. Add relation a specified relation should be inserted into the database relation. Delete relation the specified relation should be removed from the database relation structure, and the structure of the list maintained. And search relation a given relation should be located within the database.

3.2.3 Low level operations

The low-level operation unit has a direct interface with the operating system. This layer sees the real representation of the data over the physical devices and provides the level of abstraction on how this relation is implemented and how data can be extracted.

LEAP contains several structures each structure contains all relations that have been inserted into the structure and facilitate searching for a given relation. Trees and hash tables are both methods used to represent the data structures.

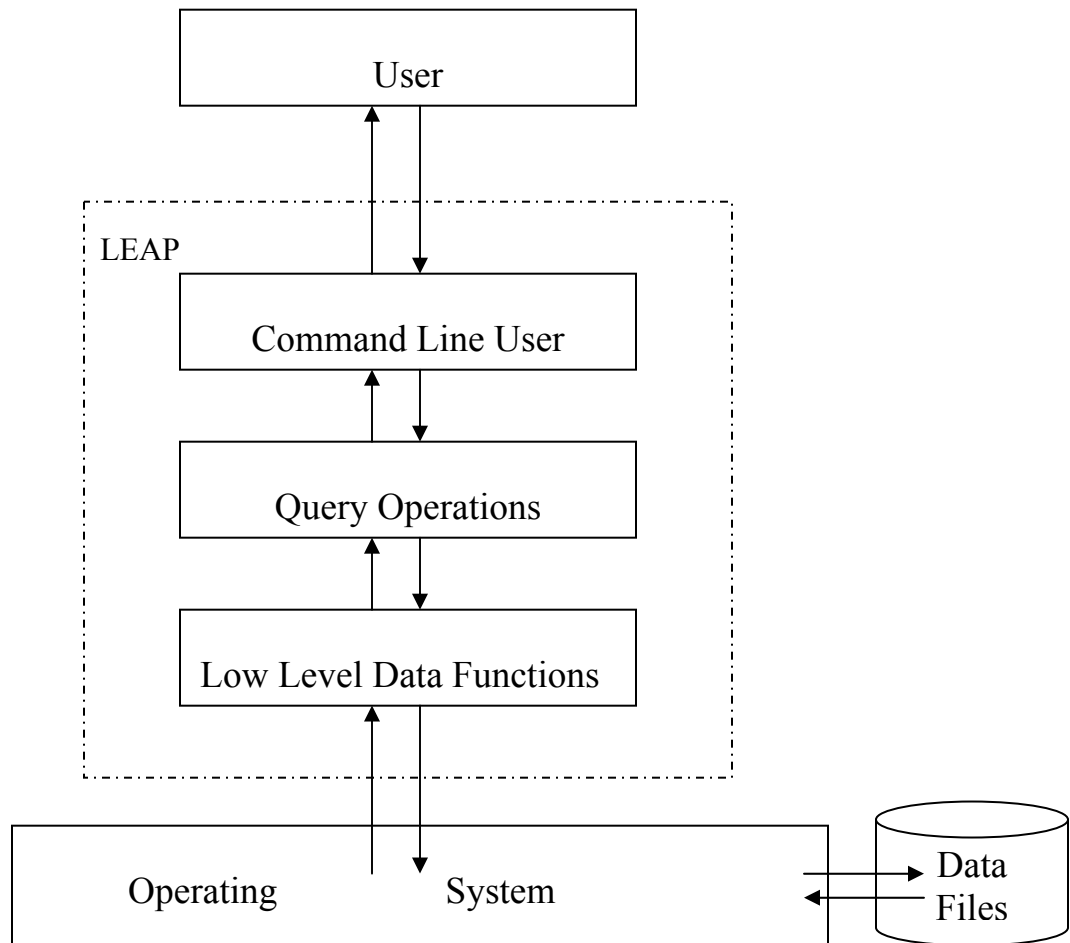


Fig. 2 User Prospective of LEAP's Architecture

3.3 Conceptual Architecture of LEAP:

From the conceptual architecture prospective, the system consists of several Modules; figure 3 visualize LEAP modules:

3.3.1 User I/O Command Interface

This is the main LEAP program interface, which handles the user I/O commands, in other words it's the LEAP's command line processing unit. The user is able to do the following operations:

- Update or delete tuples, for deleting tuple use: delete (relation). For updating tuple use: update (relation) (condition) (update_statements)
LEAP's Conditions are of the following format
([value | attribute] <, <=, =, >=, >, <>, ~ [value | attribute])
- Create a relation by using: relation (name) ((attribute1, type1, length1), (a2,t2,l2), ..., (aN,tN,lN))
- Add an element to a relation by using: add (relation) (value1, value2, ..., valueN)
- Delete a relation by using: delrel (relation)

3.3.2 Errors Handler

LEAP error handling module defines the user text messages associated with an error ID number. LEAP's errors are classified into event errors, messages, and syntax error.

3.3.3 Parser

This parser is mainly used to parse the user command line strings. This parser is similar in work to any parse tree implementation. For example, "The execution phase moves through the tree, up in each level of the parse tree with a relation. Each level's result relation is used as input to the next level, until the root is reached. The relation produced at this level is the result relation for the command"[14].

3.3.4 Expression Evaluation

This module provides the functions, which are built to evaluate an expression. LEAP's uses different datatypes such as, string, integer and

Boolean. On the other hand, conditional relational operators are also used. For example, ([value | attribute] <,<=,=,>=,>,<> [value | attribute])

3.3.5 Relational Language

LEAP's relational language operations are defined and activated from this module. This module is similar to structure query language. This module is considered the core of LEAP because it implements the relational algebra language. For example, query language.

3.3.6 Hashing

LEAP's uses hashing techniques for data storage. Hashing operations and routines makes sure that there is no data missing or data redundancy in the relations. Any new hashing value is compared for unique, in case of redundancy a new values or hashing procedures are used.

3.3.7 Tuple Attributes

The attributes module is responsible for providing the attribute structure of a tuple and link to the stored descriptive information of the tuple.

3.3.8 Stack

LEAP's stack stores the parse tree pointers. The stack is activated when a processing of a LEAP parse tree is required.

3.3.9 Memory Buffer

LEAP's uses a linked list structure to maintain the memory buffer. This module is directly interfaced with the database data representation for insertion or retrieval operations.

3.3.10 Utility Unit

This unit has all the modules or the functionalities, which work independently from others, usually the routines that interface with the operating system platform.

3.3.11 Database

This module supports a number of operations on the LEAP database. Get keys, change database, and other database functions. Database functions basically locate cretin key or locate a database for modifications such as, insert new element, remove existed element or modify existed element.

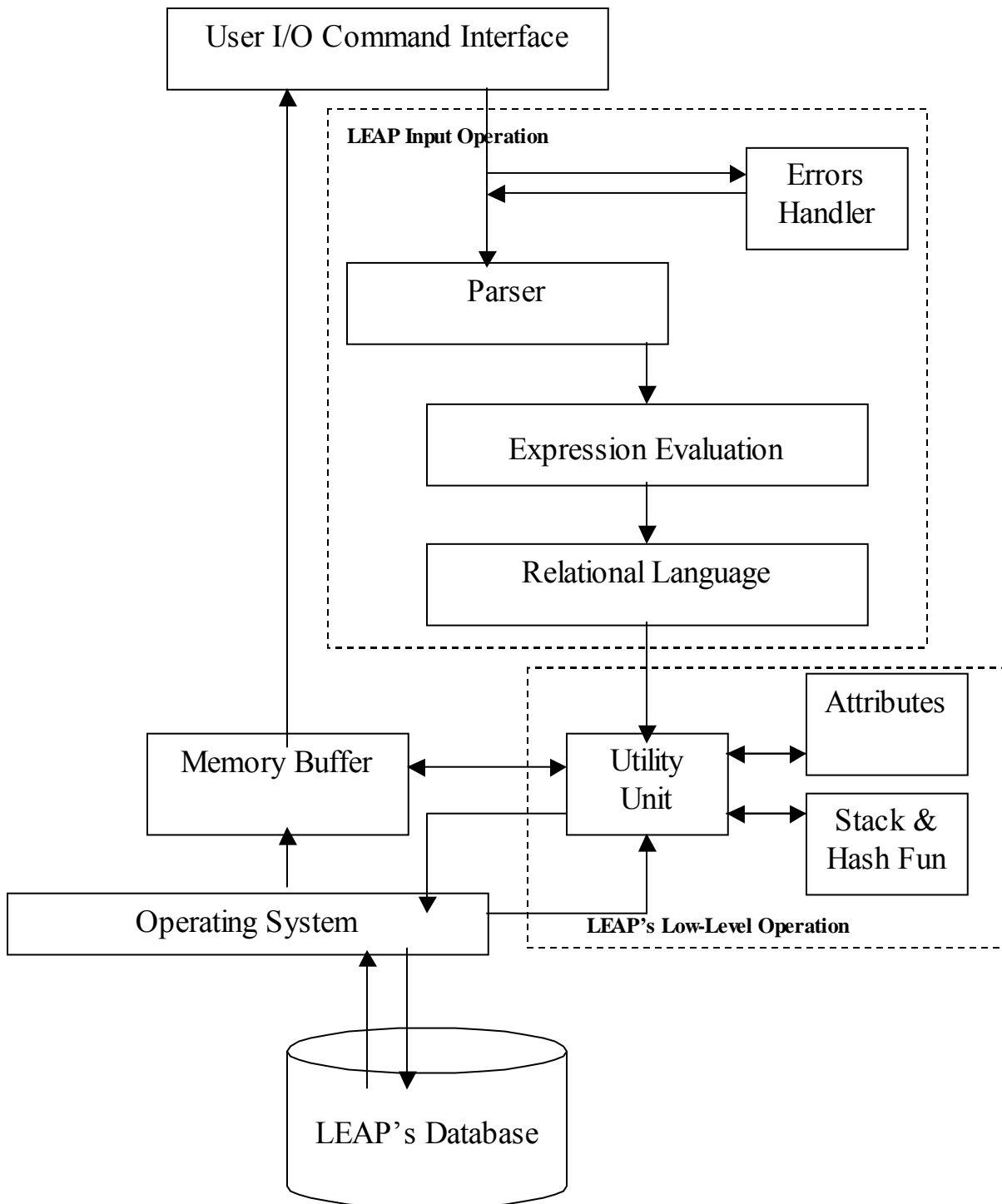


Figure 3: Conceptual Architecture of LEAP

4 Scenarios

4.1 First Scenario:

In the following scenario it's assumed that the user selected a database and listed the available data and wants to retrieve a tuple:

1. The user types the command in the User I/O Command Line Interface, for example, get (relation) (condition) (element).
2. The command is first parsed. It moves through the parse tree levels. "Each level result relation is used as input to the next level, until the root is reached" [14]. The relation produced at this level is the result relation for the command. If an error is occurred an error ID number is returned and the Error Handler unit will send back the user the correspondent message to this ID.
3. The command is now translated to a relational language command. The relational unit interfaced with a utility unit. The utility unit starts identifying the attributed for the tuple. Using stack, the utility unit is able to locate stored data structure pointers (for example a search trees or hash tables).
4. The utility unit controls the stored files through the operating system, a request is sent to the operating system to allocate the tuple.
5. The operating system with the utility unit control buffers back the selected tuple to the memory.
6. The user gets back the selected tuple.

4.2 Second Scenario:

In the following scenario it's assumed that the user selected a database and wants to insert a tuple:

1. The user types the command in the User I/O Command Line Interface, for example, add (relation) (value1, value2, ..., valueN).
2. The command is parsed, and checked for correctness.
3. The relational language with the utility unit set the attribute of the coming tuple and uses the parse tree to locate an empty space for the tuple. The tuple is hashed and a unique key is created.
4. The operating system allocates the file and inserts the tuple to the file.
5. The operation result it buffered and returned back to the user.

5 Data Dictionary

DBMS	: Database Management System
RDBMS	: Relational Database Management System
Relation	: It represents data as being stored in tables
Tuple	: A row in the relation
Attribute	: a Column in a relation.
SQL	: Structured Query Language
Hashing	: Technique that provides a direct access to records.
Stacks	: List, in which all operations are performed from one end.
Parser	: Process of identifying the grammatical structure of an input.
Parse Tree	: The data structure which a string can be represented in a pictorial form.
Query Lang.:	Language used by DBMS to perform operations on data.

6 Evolvability

This section discusses how LEAP is built to handle any future changes. LEAP parsers could be re-written with different tools such as, Lex/Yacc to enable real complex query language operations. The design of a new SQL language should be re-defined by Lex/Yacc and re-written in parse.c file. The user interface could also be modified from a command line to a graphical user interface using C/C++ or JAVA. The LEAP's group is planning to extend the functionality of LEAP by the following:

1. Update the operator: the operator functionality is being updated to handle more than one tuple functions at the same time.
2. Debugging: a new variable is added to handle the levels of debug in the program. The debug levels are vary from 0 to 9
3. LEAP Server: This server software needs some modeling to run over different platforms. This server will work as a centralized database workstation.

7 References

1. LEAP's Main page <http://leap.sourceforge.net/>
2. Information Page <http://www.dogbert.demon.co.uk/info.htm#newfeatures>
3. □□□□Publications <http://www.leyton.org/publications.html>
4. LEAP's development analysis
<http://www.dogbert.demon.co.uk/development/notes.htm>
5. "Theory and Practice of Relational Databases" by Stefan Stanczyk, Bob Champion and Richard Leyton <http://www.theorypractice.org/examples.htm>
6. An Introduction to Database Systems, 5th Edition, Edison Wesley, 1990
7. LEAP's User Guide <http://leap.sourceforge.net/user.htm>
8. Database Systems Laboratory, Department of Computer Science
University of Massachusetts, Amherst <http://www-ccs.cs.umass.edu/db.html>
9. G.M. Sacco and M. Schkolnick. *Buffer management in relational database systems*. acm Transactions on Database Systems, 11(4):473--98, 1986.
10. Michael Stonebraker, Heidi Stettner, Nadene Lynn, Joseph Kalash, and Antonin Guttman. *Document Processing in a Relational Database System*. ACM Transactions on Office Information Systems, 1(2):143--158, April 1983.
11. P. Atzeni and V. De Antonellis. *Relational Database Theory*. Benjamin /Cummings Publishing Company, Inc., (1993).
12. G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989.
13. K. Youssefi and E. Wong. *Query Processing in a Relational Database Management System*. In Proceedings of the Fifth International Conference on Very Large Databases, pp 409--417, 1979.

14. Richard Leyton “*The Design and implementation of a relational database management system*”, part 1.