KODA - The Architecture And Interface Of A Data Model Independent Kernel

Gopalan Arun, Ashok Joshi
Oracle Corporation
{garun, ajoshi}@us.oracle.com

Abstract

In this paper we describe the architecture and interface of KODA, a production strength database kernel. KODA is unique in the industry in its ability to support two different data models viz. Oracle Rdb (a relational database system) and Oracle CODASYL DBMS (a CODASYL database system). Our experience in designing and implementing KODA demonstrates

- the feasibility of implementing multiple data models on top of a common kernel,
- the benefits of leveraging performance and feature enhancements for multiple products,
- the benefits of maintainability of a common code base,
- ease of migration and interoperability between the two products without customers having to re-learn common kernel utilities like backups, data file organization and analysis tools.

Over the years, KODA based products have performed exceptionally well in industry standard benchmarks. This clearly demonstrates that including data model dependent functionality in the kernel is not a pre-requisite for high performance.

A data model independent kernel like KODA can thus be used, without much change, as the database industry moves from CODASYL to Relational to Object-Relational data models and beyond.

Proceedings of the 24th VLDB Conference
New York, USA, 1998

1.0 Introduction

KODA was consciously designed to support multiple data models in a shared disk environment. It provides a simple, powerful interface that carefully isolates the data model dependent aspects from the data model independent aspects of the functionality. This makes the kernel easier to maintain and also makes it possible for multiple clients to leverage the benefits of the functionality in the kernel. The focus of this paper is to describe the interfaces that a full function database kernel can provide and yet be data model independent.

The shared disk architecture of KODA means that special care was taken in areas like message passing and buffer cache coherency algorithms (implemented using underlying distributed lock manager functionality). In addition, by encapsulating most of the operating system dependencies within KODA, it is possible to easily port the KODA client products to multiple platforms.

The rest of the paper is organized as follows. We begin with a description of the common components in KODA that all clients share. In each component, we also describe the interfaces that KODA makes available to clients. This is followed by a description of the client specific portions in KODA. It is interesting to note that less than 5% (in lines of code) of KODA is client specific.

2.0 Common Components

KODA follows a layered architecture with each major component being independent of the others. This makes it possible to evolve each component separately. During execution, only the components that are needed are invoked. This aspect of KODA makes it easy to extend its support to different data models. Some of the major components of KODA and their layering is shown in Figure 1.

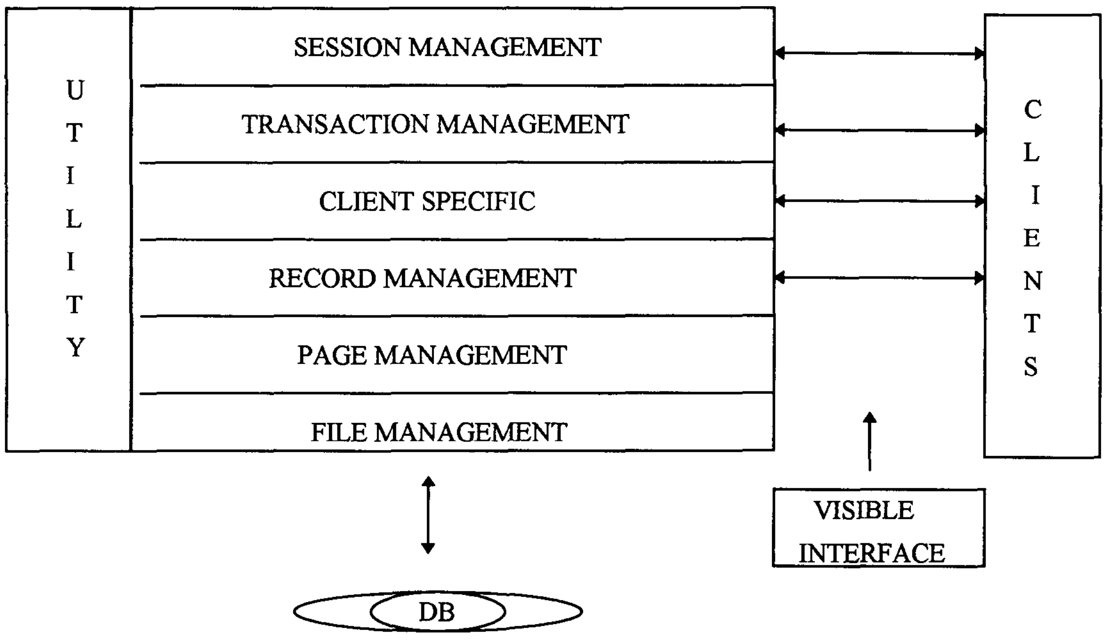

Figure 1 : Components in KODA

2.1 Session Management Component

This component deals with the task of managing a database user context. A client can establish a new session context with a database by performing a "database bind sequence" with KODA. As part of this bind sequence, KODA creates a user session context, sets up on-disk and in-memory data structures to handle the user's context and also initiates a deadman-lock termination protocol to detect the abnormal termination of a user while still bound to the database.

The client-visible interface to KODA for this component is a call to *BIND* (which specifies the database name) and the call to *UNBIND* (to terminate a session).

2.2 Transaction Management Component

Once a client has established a session with the database, a transaction can be initiated with options to declare the desired isolation levels, the read-write or read-only nature of the transaction and intention lock modes. Within a transaction, KODA maintains a markpoint. This markpoint can be set and reset under the client's control, and can be used to support verb-rollback. A verb is simply a unit of work that can be rolled back (e.g. a single SQL or CODASYL statement). The client has an option of committing or aborting a transaction. Aborting a transaction when a client has terminated abnormally or in the event of a system failure is handled completely within KODA. A two phase commit protocol is also supported with the option to call PREPARE before a call to COMMIT.

The client visible interface to KODA for this component are the calls to *START* a transaction, *ABORT* a verb, *PREPARE* a transaction, *COMMIT* a transaction and *ROLLBACK* a transaction.

2.3 Record Management Component

The record manager handles data at the record and segment (portion of a record) level. A KODA record is simply a stream of bytes. The record manager is responsible for fetching, modifying, storing and deleting records specified by the caller, one record at a time. In addition, the record layer controls and performs:

- Generation and application of before and after image journal records
- Supporting record level locking protocols and isolation levels.
- Supporting versioned read capability.

The interfaces in the record layer that are visible to the client are *FETCH* a record, *MODIFY* a record, *STORE* a record, *ERASE* a record, and *FETCH* the next record in a table (relational sequential scans).

2.4 Page Management Component

The page manager controls data at the page level. It reads

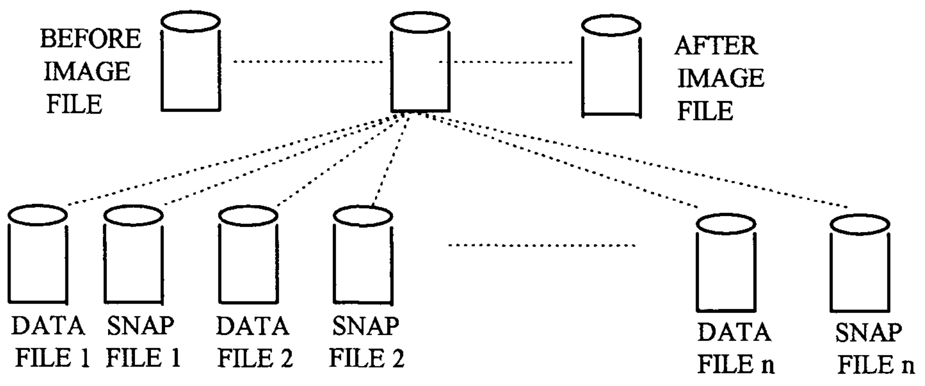

Figure 2 : Database File Structure

blocks from disk using the operating system primitives, and constructs database pages for presentation to the record layer. It is also responsible for writing changed pages back to disk. In addition, the page layer handles physical correctness requirements using global page locks. A variety of caching policies and replacement algorithms are also implemented in this component. See [JOSHI91] for more details on the distributed lock protocols in KODA.

In general, page layer interfaces are hidden from the client and are accessed only internally by KODA. There are some client entry points for diagnostic support (described in a later section).

2.5 File Management Component

The structure of the on-disk database files is common to all KODA clients. This aspect is essential if the goal is have ease of migration and interoperability between the clients. A typical KODA file structure for a database consists of a root file, one or more data files and files for the before and after image of transactional changes. The root file contains information about the current status of all database operations. Data files contain user data and database metadata. Each data file has a corresponding snapshot file which is used to support versioned reads. Figure 2 shows the different database files.

The KODA interfaces for file management is internally used by the page layer. These interfaces are not visible to the clients.

2.6 Utility Support Component

This section deals with KODA support for database utilities. KODA provides interfaces to *CREATE*, *MODIFY*, *VERIFY*, *ANALYZE*, and *DELETE* databases. Clients can also *BACKUP* and *RESTORE* databases to and from stable storage.

In addition, KODA has diagnostic interfaces to dump into files the contents of various user, page and record layer data structures. Contents of database pages and before and after image files can also be displayed. A *SHOW STATISTIC* entry point into KODA provides an interface for a comprehensive statistic collection utility.

3.0 Oracle CODASYL DBMS Specific Component

Oracle DBMS is a multi-user, general purpose database management system fully compliant with the CODASYL standard. Oracle DBMS can be used to access and administer databases ranging in complexity from simple hierarchies to complex networks with multi-level relationships.

An Oracle DBMS database consists of records and sets. The smallest entity in a Oracle DBMS database is called a data item. Data items are grouped together to form a record. The relationships among records is defined by a set. A typical set includes an owner record and one or more member records. Physical data files called areas, are used to store records and sets. Refer to [OracleDBMS] for a more detailed explanation of the terms.

Most of the Oracle DBMS specific logic in KODA is encapsulated in a thin access layer that interfaces to the record manager. The access layer uses set owner and membership information to direct the record manager to store and retrieve records. KODA supports the notions of parent record id, next record id and previous record id (within a set), to facilitate set traversal.

The KODA interfaces that are available to Oracle DBMS in addition to those mentioned in the record manager are; to *READY* an area (acquire appropriate lock on the data file), to *FIND MEMBER(S)* of a particular set, and to *FIND OWNER* of a particular set. The KODA interface for creating an Oracle DBMS has options to create sets and records.

4.0 Oracle Rdb Specific Component

Oracle Rdb is a fully functional relational database system that is ANSI SQL compliant. An Oracle Rdb database consists of records that belong to logical entities called tables. Each record consists of one or more columns. KODA supports partitioning of a table into horizontal and vertical fragments. Each of these fragments is called a logical area. One or more logical areas can be assigned to a physical area. A physical area is a physical data file. See [OracleRdb] for more information on physical and logical area definition and assignment.

The KODA interface that is available to Oracle Rdb in addition to those mentioned in the record manager are; *READY* logical and physical areas in the desired lock mode. The KODA interface for creating an Oracle Rdb database has options for creating logical areas (partitions) for tables and assigning them to physical files.

5.0 Summary

As is demonstrated in the preceding sections, with just a few client specific interfaces, KODA is able to satisfy the requirements of two client products based on different data models. The clear and powerful client visible interface in the different layered components in KODA is the single biggest contributor to achieving data model independence. The full functionality and high performance of KODA further asserts that this independence can be achieved without compromises. We believe KODA in the future can seamlessly adapt to the next generation data models.

5.1 Acknowledgements

Steve Klein gets the credit for many of KODA's early algorithms. Rick Anderson, Jeff Arnold, Jay Banerjee, Keith Brefczynski, Ananth Raghavan, S. Raghupathy, T.K Rengarajan, Mike Rubino, Peter Spiro, Bill Wright, and Craig Zastera have all made valuable contributions to KODA over the years.

5.2 References

[Joshi91] Joshi, A.M, Adaptive Locking Strategies in a Multi-node Data Sharing Environment, VLDB, 1991.

[OracleRdb] Oracle Rdb, Guide to Database Design and Definition, Release 7.0, Oracle Corporation, Part No. A41749-1.

[OracleDBMS] Oracle CODASYL DBMS, Introduction to Oracle CODASYL DBMS, Release 6.0, Oracle Corporation, Part No. A24850-1.