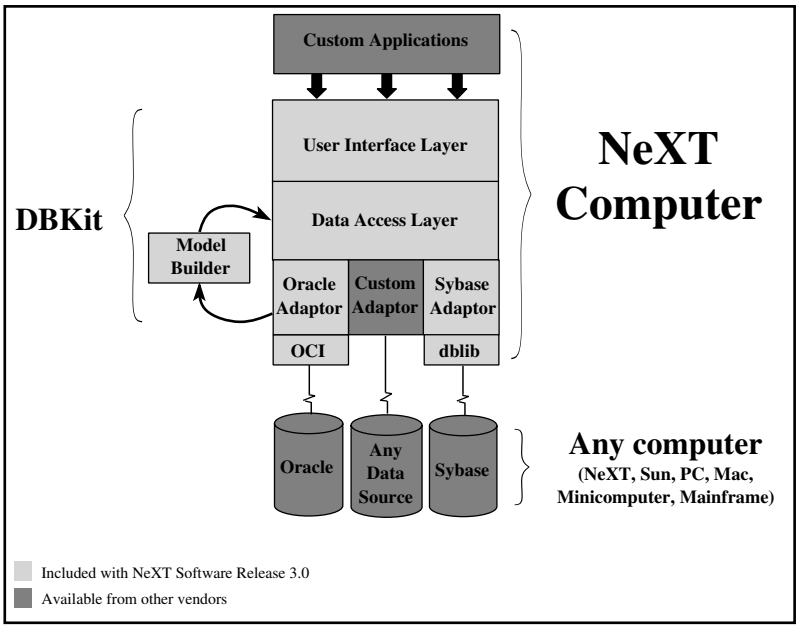


Figure 1 *Database Kit Architecture*

DBKIT: Developing Object-Oriented Database Applications

Executive Summary

For businesses to compete in the 1990s, requires effective management of one of their most valuable assets, their information. More than ever, organizations are realizing that distributed access to accurate and timely information is a strategic competitive weapon and vital to organizational success.

But as an organization's information requirements increase, MIS departments face a growing backlog of applications. And, often a bottleneck occurs as they try to provide decision-makers with what they desperately need—the right data, on time, presented in an intuitive fashion. All too frequently, the "right" data exists, but in different databases dispersed throughout the organization, and designed with experts, not users, in mind.

NeXT designed the Database Kit™ (DBKit) so organizations can have flexible access to data which is fully integrated with their custom applications. DBKit provides transparent access to data throughout an organization, exploits the NeXT™ price/performance advantage, and reduces time and cost of the application development cycle.

I. What is the Database Kit?

The Database Kit (*DBKit*) is a set of tools used to create database-oriented applications with NeXTstep™. Its robust suite of object classes and methods extend the scope of NeXTstep's object-oriented programming environment to databases. DBKit is designed for developers with a working knowledge of object-oriented application development and database systems.

DBKit radically shortens the time required to design and implement database applications that have graphical user interfaces (GUIs) and simplifies management and control of database transactions within an application. DBKit's layered architecture allows developers to design applications independently of the applications they will be used with. The resulting applications can be migrated to a different database as they are, without recoding the user interface or application logic. This enables a single, easy-to-use application to integrate data from multiple sources.

Figure 1 shows the relationship of DBKit's four major components: the User Interface Layer, the Data Access Layer, the Adaptors, and the Model

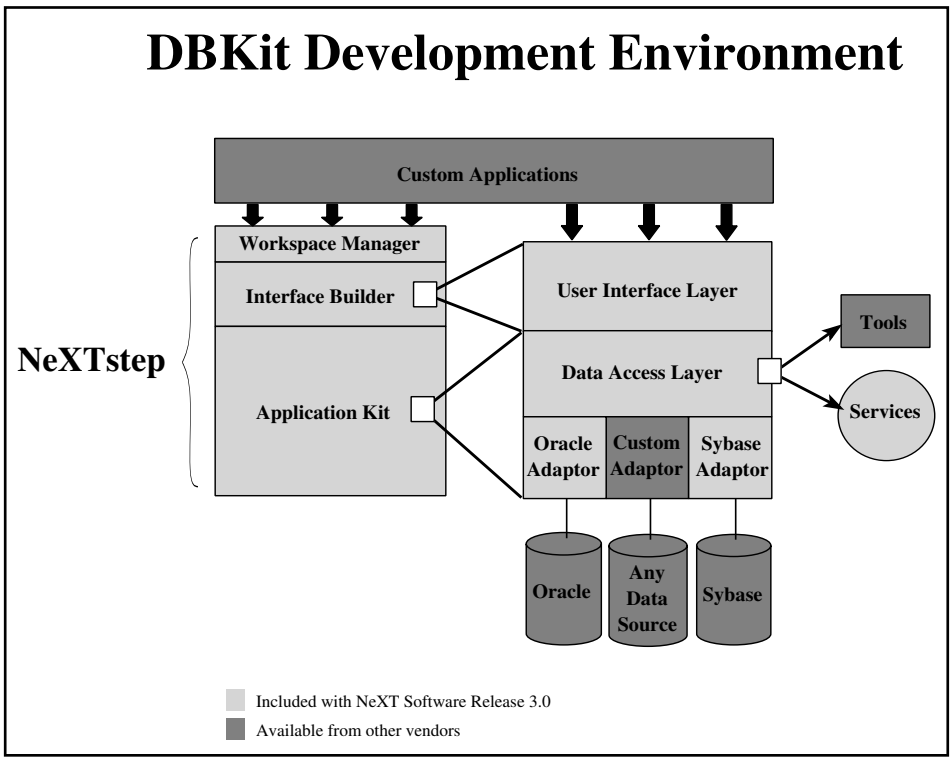


Figure 2 *DBKit Development Environment*

Builder. This paper begins by describing how DBKit integrates with the NeXTstep development environment. It describes the overall DBKit architecture and the features provided by each of the four major DBKit components. Finally, it discusses the benefits of DBKit both from an application developer's perspective and from an end-user's point of view. The Appendix contains detailed information on the functionality offered by specific DBKit object classes.

This paper assumes that the reader has a general familiarity with NeXTstep's object-oriented development environment, including Interface Builder™ and the Application Kit™. Those unfamiliar with NeXTstep should read a companion paper, "The NeXTstep Decision: Object-Oriented Applications Development with NeXTstep," before reading on about DBKit.

II. Database Kit and NeXTstep

DBKit is an integral part of NeXTstep's application development environment, as shown in Figure 2, and as such, works optimally and seamlessly with all of the objects in NeXTstep's Interface Builder and Application Kit.

Just as Interface Builder and the Application Kit provide objects for all of the common elements of general applications (such as buttons, windows, panels, and sliders), DBKit provides a rich set of objects that incorporate the core functionality of database applications. Essentially, DBKit's object classes extend the power of object-oriented programming to database application development.

Furthermore, DBKit-based applications can easily incorporate all the functionality offered by NeXTstep, including interprocess communication, support for graphics and sound within an application, and the ability to cut/copy/paste between applications.

The resulting development environment significantly reduces the effort required to design, implement, and maintain GUI applications that incorporate strategic corporate data.

III. Database Kit's Architecture

DBKit's layered architecture provides a bridge between application code and data sources. The DBKit architecture allows applications access to

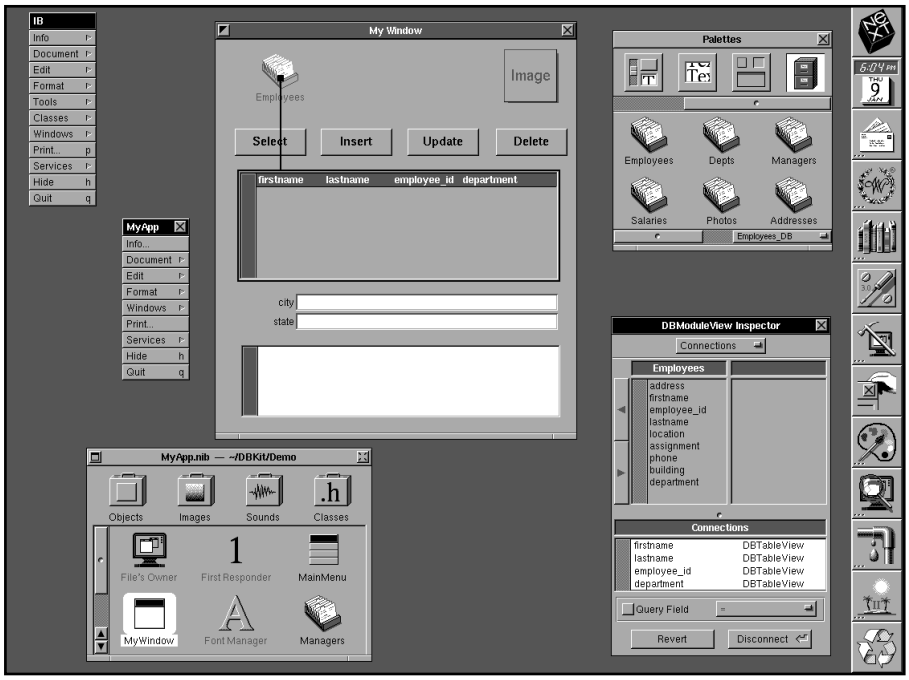


Figure 3 *Developing Applications with DBKit and Interface Builder*

many different types of data sources including relational and hierarchical databases, on-line news feeds, and more.

User Interface Layer

The object classes in the User Interface Layer simplify the design and construction of database applications which have friendly, window-based user interfaces. They provide a graphical interface for selecting data from a data source and displaying elements returned from it, such as tabular data, images, rich text formatted information, and other binary large objects.

Some of these user interface objects appear on custom palettes within Interface Builder. Developers can graphically select one or more data sources, and then simply drag and drop objects from the palettes to create sophisticated database applications.

Figure 3 illustrates how DBKit's user interface objects can be graphically connected to specific data elements. And, database operations (e.g., select, insert, save, new, delete, and others) can be graphically selected using the target/action paradigm of Interface Builder.

Another object in the User Interface Layer allows end-users to graphically construct queries within an application, without knowing any details or syntax of a particular vendor's query language.

All DBKit user interface objects may be used in conjunction with all the data display objects and protocols in Interface Builder and the Application Kit.

Data Access Layer

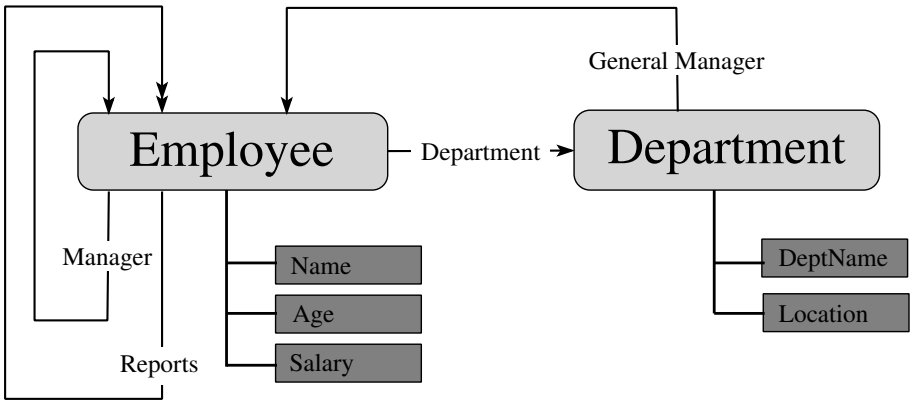
The Data Access Layer is a database independent, programmatic interface which gives high level access to data sources. It allows data manipulation with limited knowledge of the underlying data source and its particular query language.

The main function of the Data Access Layer is to move data to and from a data source. A data buffering mechanism that is controlled from within the application supports efficient record processing.

In addition, objects in this layer manage database connections, generate database-independent queries, and bind data elements with user interface objects.

Adaptors

Adaptors translate database-independent queries into specific function calls for a given data source. They pass these function calls to the interface library belonging to the specific data source to which they are connected. When data elements are returned from the data source, the Adaptors present them as Objective C objects.



- Employee.Department.DeptName
- Employee.Manager.Salary
- avg (Employee.Manager.Reports.Age)

***Figure** 4 Database Kit Expressions*

Adaptors can be extended to support specific features of the data source (such as stored procedures in SYBASE®), so all of the functions available in a given data source can be fully utilized rather than taking a "lowest-common-denominator" approach.

Finally, Adaptors are responsible for gathering information about a data source's structure. In the case of a relational database, the Adaptor queries the database's data dictionary for details about its schema (such as tables, columns, primary and foreign keys, and indexes).

DBKit comes bundled with the client libraries and Adaptors for both SYBASE and Oracle®, permitting connections to any SYBASE SQL Server™ or ORACLE RDBMS™. Since DBKit is based on a well-defined set of protocols, new adaptors can be easily created to access additional databases. Adaptors for other databases are currently under development in conjunction with other database vendors.

Model Builder

With the Model Builder application, developers can quickly and easily build data models. These models contain high-level information specifying relationships between data elements (e.g., "joins" across tables within a relational database).

The Model Builder gleans information about a database's structure or schema via an Adaptor, and then uses it to generate a default model. This structural foundation can then be enhanced to specify more complex relationships between data elements and between tables (called *expressions*). In essence, the Model Builder translates a database's table/column view of the data elements into logical hierarchies of related elements.

The resulting model of the data structure more closely represents the "real world" relationships that exist between data elements. It also allows navigational access across data source entities and supports computed field values. Figure 4 illustrates this concept with an example that shows how expressions are derived from a set of relational database tables with the Model Builder.

By incorporating models into the development process, the developer no longer has to worry about the semantics of accessing specific data elements, or constantly reconstructing these relationships for every new application as it is built.

The Model Builder also creates a table containing database login information. A default table is generated based on information from NeXT's NetInfo™ network management software. The application developer can alter this table for added security or special permissions.

Benefits for the Application Developer

- **Faster Application Development**
- **Integrated Development Environment**
- **Increased Flexibility and Extensibility**
- **Robust Applications**
- **Database Independent Implementation**
- **High Level Data Modeling**
- **Improved Maintainability**
- **Manipulation of a Wide Range of Data Types**

IV. Database Kit and the Application Developer

Faster Application Development

Object-oriented programming allows the developer to take a building-block approach to an application, and thus develop applications much more quickly and efficiently.

DBKit extends the scope of the building blocks available in NeXTstep by adding a toolkit of database-related objects, dramatically reducing the time required to develop and deploy database-oriented applications. By eliminating the necessity to continually reimplement functionality common to most database applications, such as managing data source connections, or specifying relationships between data elements, application developers can focus instead on developing the code that makes their application *unique*, and gives their enterprise a competitive advantage.

Simple forms-oriented database applications can be developed with minimal programming effort, while even large database systems can be constructed in a fraction of the time required by traditional database programming techniques.

Integrated Development Environment

All of NeXTstep's tools, including the DBKit, are designed to work together to provide a complete, integrated development environment. NeXTstep gives the development team access to a collection of class libraries, windowing systems, and programming languages maintained by one vendor, released at one time. With NeXTstep and DBKit, it all works together because it was *designed* to work that way.

Increased Flexibility and Extensibility

With DBKit, the developer is no longer restricted by the functionality of a database vendor's specific language or forms package. All DBKit data display and data access objects can be used "as is" or customized for a particular need. Even the Adaptors can be customized so they incorporate new functionality offered by a particular data source or by the requirements of a specific application.

Since the full suite of NeXTstep Application Kit objects is available to the developer, they too can be subclassed and incorporated into the database application giving it added functionality.

And NeXTstep's object-oriented architecture enables developers to add their own custom objects, which can then be shared across numerous applications.

The ability to modify and add objects easily (and hence, optimize one portion of the application without affecting other portions) creates tremendous programming flexibility.

Robust Applications

Applications created using DBKit can incorporate extensive data verification and complex logic. It's easy to develop this type of robust, mission-critical application using the tools within the NeXTstep environment. DBKit-based applications can be fur-

ther enhanced by integrating the flexibility and creativity of the C, Objective C, and C++ programming languages.

DBKit's object-oriented architecture lets developers work at a higher level of abstraction—they care less about how given objects work internally, concentrating instead on which objects provide what overall functionality, and how they communicate with each other. Developers are able to easily manage applications—even as they become more complex. They are able to ensure the robustness which is paramount to the mission-critical applications they are developing.

Database Independent Implementation

DBKit allows developers to design applications independently of the database itself. DBKit's layered architecture effectively isolates all of the application's basic logic from the semantics of how data is actually stored in and retrieved from a particular vendor's database. Thus applications can migrate to a different data source without recoding the user interface and application logic.

DBKit's open architecture enables developers to quickly and easily implement new Adaptors to interface to additional data sources.

Optionally, function calls specific to one data source may be included at the application level for possible performance or flexibility gains (with the trade-off of reduced application portability).

High-Level Data Modeling

Because DBKit-based applications are built on high-level data models, the developer concentrates on the application logic, rather than the semantics of accessing specific data elements—focusing on the flow and presentation of data, instead of manually constructing every query and dealing with a variety of database library interfaces. DBKit isolates the developer from the complexities of the underlying database by packaging explicit data relationships and attributes into easy-to-manage objects.

DBKit's high-level programmatic interface for identifying and manipulating data also allows navigational access across data source entities, and supports computed field values.

For example, when accessing a relational database, the data model contains all the relevant information to construct a "*join*" across tables. It can also automatically construct the proper sequence of SQL commands so a complex value can be computed from various data elements.

Improved Maintainability

Because DBKit is based on object-oriented programming, and because it offers a high level of abstraction from the underlying data storage model, it provides for vast improvements in application maintenance—application developers do not have to worry whether changes to one portion of the code will have unintended consequences that could propagate throughout the application.

Whether data element specifications in the database are being updated, or changes are required for performance reasons, the developer has the flexibility to make changes in the underlying database configuration *without* having to rewrite the entire application; only the data models need to be updated to reflect changes in the database.

Manipulation of a Wide Range of Data Types

Using DBKit, a developer can build applications that manipulate a wide variety of nontraditional data types—from simple text and numeric fields to rich text, images, and sound—in an object-oriented environment.

Each of these *multimedia* data elements are retrieved from the data source and bound to objects just as any other data type would be. They are then seamlessly displayed using NeXTstep's Display PostScript® capability.

Bundled Support for Oracle and SYBASE

The Oracle and SYBASE client libraries are bundled with DBKit, enabling connections to any Oracle or SYBASE database without any additional software! Not only can developers become productive right away, but there is no incremental cost for deploying their DBKit-based applications.

V. Database Kit and the End-User

Consistent Look and Feel

Applications developed with DBKit maintain both the ease-of-use and consistent look and feel that are the hallmarks of NeXTstep applications. When users learn one NeXTstep application, they have learned the key elements of most applications, including DBKit-based applications. End-users can easily and intuitively access their data, or use a set of related applications, in a manner familiar to them.

This translates into immediate savings for the corporation in application training and support costs.

Benefits for the End-User

- **Consistent Look and Feel**
- **Integrated End-User Environment**
- **Integration of Data from Multiple Sources**
- **Integration of Sound and Image Data Types**
- **Supports Various Data Selection Methods**
- **Performance and Scalability**

Integrated End-User Environment

DBKit applications are not isolated applications. Any application properly designed under NeXTstep can cut, copy, and paste data from other applications, and request Services from other applications, such as Webster's Ninth New Collegiate Dictionary®, NeXT's Digital Librarian™, or NeXTMail™. More sophisticated database applications can be easily designed to access mathematical models in *Mathematica*®, or send numerical information to Lotus Improv™ for charting or analysis. In this way, users have a completely integrated set of tools, and each new application adds value to all others present on the desktop.

Integration of Data from Multiple Sources

DBKit's adaptor-based architecture allows multiple data sources to be integrated within one application. These data sources can be relational databases from a single vendor or from multiple vendors, and can include on-line news feeds and hierarchical databases.

Applications developed with DBKit enable the end-user to access all information relevant to a particular decision from within one, easy-to-use application. In the financial services arena, for example, this might mean incorporating data on stock market prices coming from a real-time stock ticker with historical price information on a particular security in order to make a buy/sell decision.

The ability to incorporate data from multiple data sources within one application improves data reliability, eliminates the need for reentering data, and leads to faster, more efficient decision-making.

Integration of Sound and Image Data Types

DBKit seamlessly integrates image, sound, and rich text information within database applications. This *multimedia* information adds tremendous richness to numeric and textual data, and can be a significant competitive advantage for an organization. See Figure 5 for an example of a simple DBKit-based application that incorporates images and rich text information.

Supports Various Data Selection Methods

Database applications built with DBKit support three standard data selection methods:

- *Fixed queries*, where the parameters of the search are predefined by the application developer. These are essentially "canned" queries and cannot be modified by the end-user in any way.
- *Query-by-Example*, which allows users to vary the values of the search criteria within a static set of data fields. For example, in a customer information application, one user may query for all customers in the state of California; another user may wish to select customers who have purchased a specific product.
- *Ad-hoc queries* of arbitrary complexity, built with minimal restraint on the scope of data field values or data ranges involved. This would be comparable to constructing an SQL statement in a relational database environment.

Performance and Scalability

DBKit's integral support of the client/server model makes it an ideal mechanism for integrating local, easy-to-use NeXTstep interfaces with com-

Figure 5 *Seamless Integration of "Multimedia" Data Types*

pute-intensive searches that run on remote cycle-servers, all within an application that's easy-to-maintain. Now end-users can access the data they need more quickly and reliably.

In addition, the client/server model gives an organization a great deal of flexibility in matching its server investment to its increasing needs. For example, a department or company might use an application on a NeXT computer as a database server, and eventually migrate to a Sequent®, Pyramid®, Teradata™, or other large database server as the number of users and size of the database increase.

The Bottom Line

NeXT is the only general-purpose computer vendor to integrate the object-oriented paradigm into the entire development environment. DBKit builds upon this foundation, greatly reducing the time and effort required to create graphical, intuitive, and consistent database applications.

Welcome to NeXTstep, and the world of database application development using the NeXT Database Kit.

APPENDIX
NeXT Database Kit Object Classes

User Interface Objects

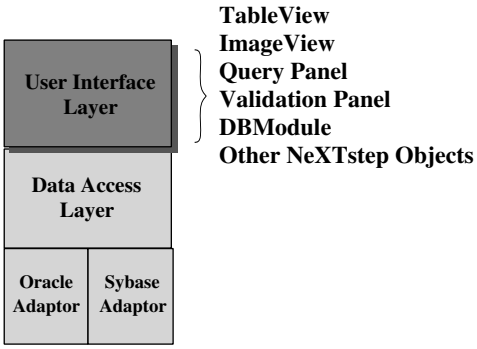


Table View

- Displays rows of data returned from a data source and presents them in tabular form
- Functions as a "pick list," allowing the user to select a specific row or rows for additional processing

Image View

- Allows images extracted from data source to be displayed
- Displays images represented by NXImage objects
- "Drag and drop" of images from the Workspace Manager™ for data entry is allowed if the application supports it

Query Panel

- Allows end-users to graphically construct queries within an application without needing to know any details or syntax of a particular vendor's query language

Validation Panel

- Displays queries before they are sent to the database for processing, enabling query statements to be validated before they are executed

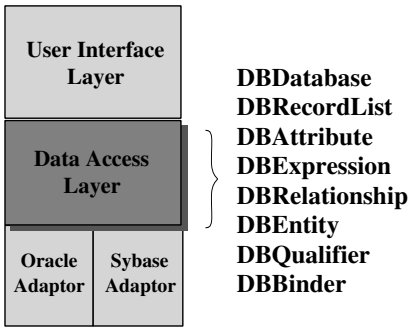
DBModule

- Represents an entity in an available database
- Displays as an object in an Interface Builder palette
- Uses information from a data model to specify relationships with other entities and data elements
- Supports graphical connections between data elements and display objects to establish data flow
- Supports graphical specification of master/detail relationships
- Supports navigational access to data elements across database entities
- Supports Interface Builder's target/action paradigm to perform database functions (e.g., Select, Insert, Delete, Update)
- Collects expressions before sending them to the Access Layer

Other NeXTstep Objects

- Database Kit objects work seamlessly with all Interface Builder and Application Kit user interface objects

Access Layer Objects



DBDatabase

- Manages access to a given data source, including data source login and transaction handling
- Stores data models that specify the structure of the data source and the relationships between data elements

DBRecordList

- Stores the results of a query. It is a coherent "snapshot" of retrieved data elements
- Allows sequential or random access to rows of data within it

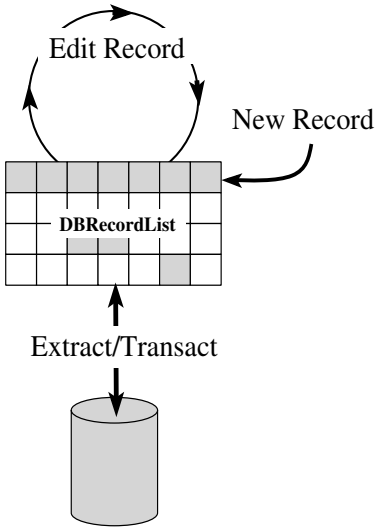


Figure 6 *Buffer Management*

- Collects record modifications, insertions, and deletions before they are committed to the data source
- Provides a data buffering mechanism to support efficient record processing (see Figure 6). Specific buffer management capabilities include:
  - -Random or sequential access to rows of data, as required for accessing extremely large data sets
  - -Shadow values that ensure consistency in multiuser editing
  - -Update parameters can be optimized to be lazy, or immediate for recoverability or efficiency

DBAttribute

- Maps to a SQL column or a structure field. It is typed and represents a unit of state information.
- Contains meta-information such as whether the data element is part of a primary key or a foreign key, or whether it should appear in a structure browser.

DBExpression

- Behaves as a virtual attribute
- Its type is a reference type that points to either a single DBAttribute or a computed value
- In relational databases, DBExpression instances can contain navigational references across "joins"

DBRelationship

- Contains information specifying the relationship between data elements in different tables. Corresponds to "join" information in the relational model.

DBEntity

- Analogous to a table in a relational database or, a structure type in a hierarchical database

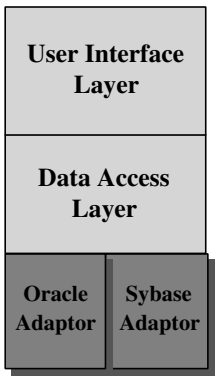
DBQualifier

- Contains qualifiers to constrict data element selection from a data source (the "Where" clause in SQL)
- Multiple DBQualifier objects may be combined with logical operators

DBBinder

- Flexibly moves data between NeXTstep objects and external data sources
- Holds information on what data is to be retrieved and where it should go

Adaptors



- Translate DBKit structures into calls to the client libraries
- Query the data dictionary for information on database structure
- Can be extended to support features specific to one data source (e.g., stored procedures)
- Support multiple connections to a single data source
- Accept data source specific language commands (e.g., SQL) to perform complex operations such as schema modifications or optimized query statements
- Support string tables for internationalization
- Based on a well-defined set of protocols so that new Adaptors may be easily created