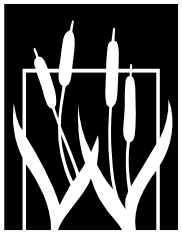


Whitemarsh
Information Systems Corporation

Database Management Systems: Understanding and Applying Database Technology
Chapter 2

DBMS Applications and Components

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

DBMS Applications and Components

- Application Classifications
- Static and Dynamic Relationships
- The Nature of a Static Relationship Application
- The Nature of a Dynamic Relationship Application
- Problems and Benefits of Static Relationship DBMS
- Problems and Benefits of Dynamic Relationship DBMS
- Implementing the Static Relationship Application
- Implementing the Dynamic Relationship Application
- DBMS Components and Subcomponents
- The DBMS
- DBMS Issues

- Application Components and DBMS Components
- DBMS Requirements Summary

First: A Database Management Systems (DBMS) is a large collection of computer software processes to:

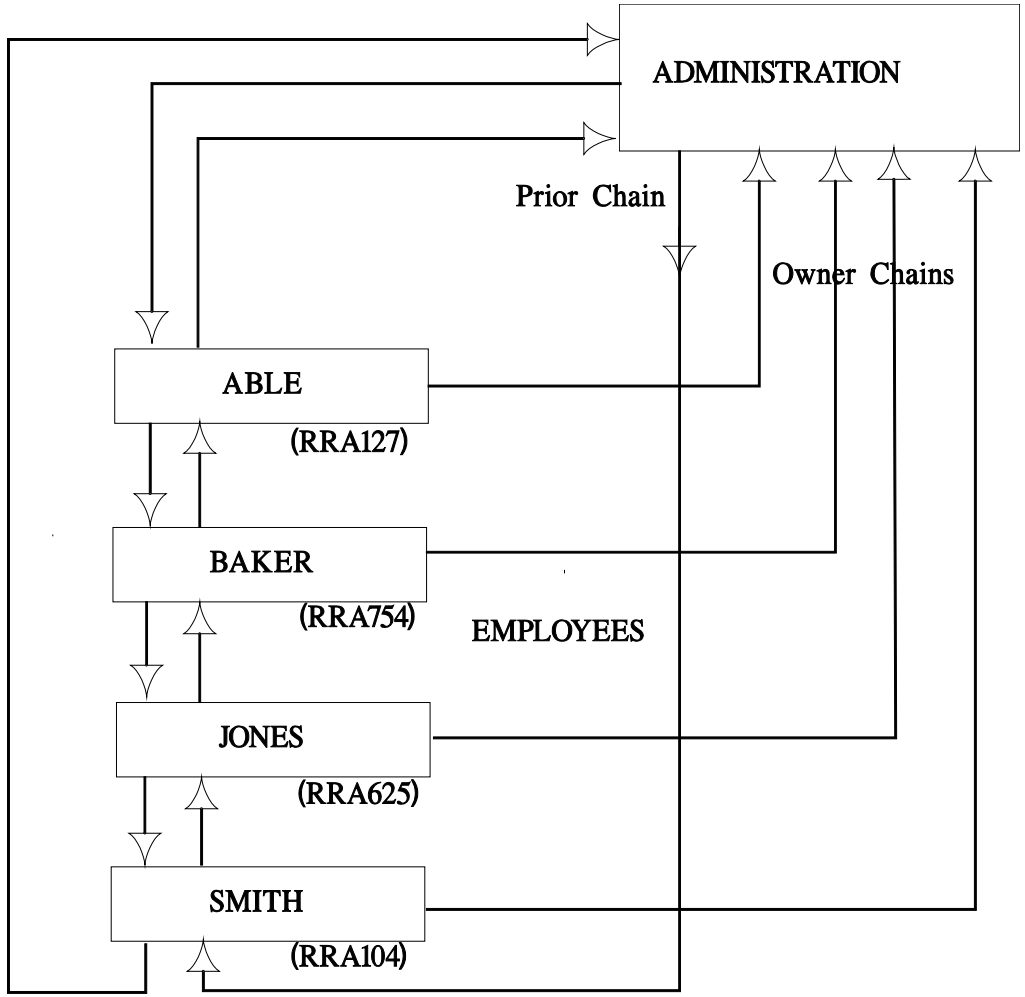
- Define the logical database component of a specific database that is, the tables, columns, and relationships
- Define the physical component of a specific database, that is, the storage structure definition, and the techniques for access strategies, data loading and updates, and database backup.
- Specify interrogations that access data through one or more types of languages such as host languages like COBOL or FORTRAN, and/or natural languages like query-update, report writers, or procedure-oriented.
- Specify system control facilities, that is, audit trails, backup and recovery, concurrent operations, and the like.

1.0 Application Classifications

- Applications can be classified in many ways.
  - Their major use, such as accounting, engineering, or personnel.
  - Or, they can be classified by the types of audiences they serve: operations, support and control, or high-level MIS.
  - But, for DBMS purposes, the real difference in applications is whether their design is constant or is variable.
- DBMS facilities too can be classified in many ways
  - Host language or
  - Self-contained,
  - Network, hierarchical, independent logical file or relational, or
  - Embedded pointers or value based relationships
- The real difference in DBMSs is how they handle the differences in applications which are either:
  - constant and unchanging
  - variable and changing.

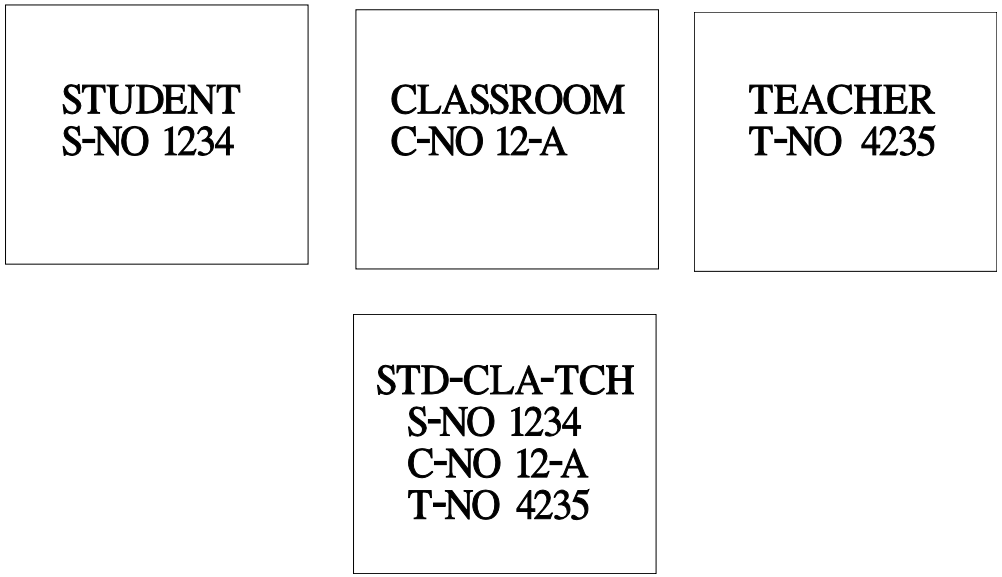
2.0 Static and Dynamic Relationships

- Relationships between an owner and a member
  - Static: "members" made known to "owners" at load or update type



SET: DEPT-EMPL
OWNER: DEPARTMENT
MEMBER: EMPLOYEES
ORDER: SORTED ON LAST NAME

- Dynamic: “members” and “owners” can know of each other only at retrieval time



Relationships Type Impact on Applications		
Application Characteristic	**Relationships**	
	Static	**Dynamic**
Fundamental Orientation	Corporate	Project
Required DBA Orientation	Centralized	Decentralized
Database Design Effort	Significant	Casual
Frequency Of Design Changes	Low	High
Design Change Control	Strict	Lax
Design Change Effects	Profound	Trivial
Typical Data Volume	Large	Small
Application And Database Design Interdependence	High	Low
Predominant Interrogation Language	COBOL	Query-Like

- Either static or dynamic relationships dominate a database application. Seldom both.

With Static Applications...

- Static applications have relationships that are normally:
  - Well-defined,
  - Stable, and
  - Seldom-changing.

- Static relationship applications tend to be
  - Production-oriented,
  - Large transaction volumes
  - High velocities.

- The principal access languages for static relationship applications are languages like COBOL.
  - Highly proceduralized
  - Highly configurable
  - Highly controls end-user actions and capabilities

While Dynamic Applications...

- A dynamic applications have relationship that are normally:
  - ♦ Casually defined
  - ♦ Dynamic
  - ♦ Change often with data changes.

- Dynamic relationship applications tend to be:
  - ♦ MIS (management information system), prototypes, or research.
  - ♦ Lower transaction volumes, and
  - ♦ Lower velocities than the static relationship applications.

- The principal access languages for dynamic relationship applications are natural languages like query-update, procedure-oriented, and report writer.
  - ♦ Lightly proceduralized
  - ♦ Simply configured, straightforward
  - ♦ Provides great controls and capabilities to end-users

- **A static relationship DBMS's DDL contains formally defined relationship clauses with**
  - Attendant integrity,
  - Row storage, and
  - Retrieval subclauses.

- Rows are often:
  - Stored in the order loaded,
  - Maintained in DDL defined ordering, or
  - Stored and maintained in primary key value order

- Rows are selected and presented:
  - According to primary key value order or
  - As specified in the DDL.

- A dynamic relationship DBMS's DDL usually does not contain formally defined relationship clauses.

- Rows are often
  - ♦ Stored in the order presented and
  - ♦ Maintained in any order, or
  - ♦ Stored and maintained in primary key value order

- The user of a dynamic database has sort clauses as a fundamental part of every access language.

- Rows are selected and sorted before presentation.

- Once presented, the rows can be re-sorted and re-presented.

- In most DBMSs, the rows are not actually sorted, their address identifiers are.

DBMSs Reflect the key difference: Static vs Dynamic Relationships

DBMS Type (Inter-table relationships)	
Static	**Dynamic**
Defined through DDL	Optionally defined
Created through DBMS	Created through user defined field values
Bound at load/update	Bound at retrieval
Changed through delete and re-add	Changed through field update

- The effect of having a dynamically related set of rows is the inverse of having a statically related set of rows.
- With dynamic relationships,
  - Each interrogation has an equal chance of retrieval and update efficiency.
  - There is no bias to the processing.
  - Often there are no restrictive relationship integrity clauses.
  - The burden of maintaining relationship integrity is placed on the user
  - Thus it follows that the integrity clauses governing the interrelationships may also be undefined.

- With static relationships,
  - Each interrogation is very biased towards the sorting and/or sequencing of rows bound into the relationship
  - There are restrictive relationship integrity clauses.
  - The burden of maintaining relationship integrity is managed by the DBMS
  - Integrity clauses governing the interrelationships are defined during database design time.

- This lack of relationship integrity has prompted the builders of some dynamic relationship DBMSs to borrow a concept, in static relationship DBMS concept: called referential integrity.
  - ♦ First, it is the specification of the rules that relate two rows from two different types, and
  - ♦ Second, it is the automatic enforcement of the rules by the DBMS.
  - ♦ This simple concept is, however, fraught with pitfalls, and must be carefully implemented. Referential integrity is covered in Chapter 3.
- Knowledge of a DBMS's data model is important, but not as important as knowing whether the DBMS creates static or dynamic relationships.
- If the DBMS's relationships are static and the application is dynamic, application failure is almost always the outcome.
- If the DBMS's relationships are dynamic and the application is static, then the application will run, but the production performance will normally fall far below achievable levels.

3.0 The Nature of a Static Relationship Application

- How many times since the creation of data processing has a bill of materials (BOM) system's design been substantively changed?
  - Few times if ever.
  - The reason is simple: the process's design is static.

- The software is written once and runs for years.

- And because it runs for years, and with voluminous transactions, the software written to process BOMs is *very close to the machine.*

- This software must be well designed and expertly implemented.

- A very large effort is justified because the design changes are few, and because the through-put required is very high.

- Static relationship database applications are similar to BOM
  - Less prone to change
  - Require very high through-put over a static set of relationships.
  - These applications thrive on a certain kind of DBMS.

- Database benchmarks principally based on static relationship processing have, over the years, shown this assertion to be true.

Static Application Characteristics	
Fundamental Orientation	Corporate
Required DBA Orientation	Centralized
Database Design Effort	Significant
Frequency Of Design Changes	Low
Design Change Control	Strict
Design Change Effects	Profound
Typical Data Volume	Large
Application And Database Design Interdependence	High
Predominant Interrogation Language	COBOL

4.0 The Nature of a Dynamic Relationship Application

- In contrast to static relationship applications, there is a class of applications for which design change seems to be a way of life.

- Changes arise from two sources.
  - First, there are changes that exist because the application is young and not yet completely evolved.
  - Second, there are changes that are due to the very nature of the application.

- Dynamic relationship applications tend to require many changes to the tables and the relationship types among rows in order to keep the applications relevant to the user's changing needs.

- These applications tend to
  - Contain smaller amounts of data than do static relationship applications,
  - Be better suited to a natural language environment than COBOL environment as changes are needed faster than can be programmed.

Dynamic Application Characteristics	
Fundamental Orientation	Project
Required DBA Orientation	Decentralized
Database Design Effort	Casual
Frequency Of Design Changes	High
Design Change Control	Lax
Design Change Effects	Trivial
Typical Data Volume	Small
Application And Database Design Interdependence	Low
Predominant Interrogation Language	Query-Like

Static & Dynamic Application Impact or DBMS			
DBMS Component		**Relationship Type & Effect**	
		Static	**Dynamic**
Logical	Tables per data base	Many	Few
	Relationship Mechanism	Pointers	Data Values
Physical	Relationship Building	Load/Update	Retrieval
	Row Keys	Single	Multiple
	Data Loading	Via Structure	Row by Row
	Relationship Change	Delete and Re-Add	Data Value Change
	Fundamental Bias	Toward Database design or Production	Neutral or ad hoc MIS-like

Static & Dynamic Application Impact or DBMS			
DBMS Component		**Relationship Type & Effect**	
		Static	**Dynamic**
Interrogation	Host Language	Excellent	Good
	Natural Languages	Average	Excellent
System Control	Multiple Table Audit Trails	Easy	Hard
	Reorganization Logical Physical	 Hard Expensive	 Easy Reasonable
	Multiple DB Processing	Acceptable	Excellent
	Multiple Table Locks	Easy	Hard

5.0 Problems and Benefits of Static Relationship DBMS

Problems

- Applications implemented with static relationships can have problems in two areas.
  - ♦ Produce a report that does not mirror the database's structural definitions.
  - ♦ The second problem pertains to the database itself.

- Redesigning a static database usually involves:

  - ♦ Actual redesign of the database's individual tables
  - ♦ Formal relationships among the tables
  - ♦ Programs that process data from the databases
  - ♦ Programs are often written with the database's structure in mind.
  - ♦ Once the database design is changed, the programs that process data from it often require changes

Benefits

- All tables are under the control of a single schema.
- The DBMS performs relationship maintenance from centralized point of view.
- Opportunity to build well-engineered and controlled databases.
- This results naturally from the data loading and maintenance process, which is performed principally through COBOL programs.
- These two benefits greatly increase database integrity.
- Reports that mirror the database structure are highly efficient.

6.0 Problems and Benefits of Dynamic Relationship DBMS

- Applications implemented with a dynamic relationship DBMS have almost the inverse set of problems and benefits of a static relationship DBMS.

- If no referential integrity, database integrity can be a significant problem with a dynamic relationship DBMS.

  - ♦ If the value in an owner column is accidentally changed, then the owner's corresponding member rows are lost with respect to that relationship.

  - ♦ Members can be placed in a wrong relationship if the data value in the member is updated incorrectly.

- DBMS benchmarks have shown that production-oriented data-processing-intensive operations perform more slowly than with static relationship DBMSs.

  - ♦ Dynamic relationship DBMSs often have to access the owner row to get the data value used to select--via secondary index searches--the member rows.

  - ♦ The static relationship DBMS, on the other hand, simply follows a predefined *pointer-based* road map to another row that may be on the same physical page.

- Other difficulties center around the implementation of multiple table applications

- It is the very independence of the tables that causes the problems.

  - Whenever multiple table updates,
    - Elaborate schemes must be implemented to ensure that all affected rows are updated in unison.

    - Compounding the lock problem, all well-designed dynamic database applications must have coordinated backup and recovery, and audit trails.

  - While these problems can be overcome by good user code and sophisticated techniques, they can cause concern to the application's users if not dealt with effectively.

- The benefits are just as significant as the problems.
  - Flexibility,
  - Speed in some settings, and
  - Distributed processing
  - Tables associated with a dynamic relationship database may be separate physical files
  - Data loading and maintenance can be carried out without fear of disturbance by processes accessing other table instances.
  - Physically independent tables can often be processed sequentially--at great speeds--since all rows in the table's file are of the same type and format.
- Finally, the greatest benefit of the dynamic relationship DBMS is distributed database.
  - Multiple applications might operate independently throughout the year, and then
  - Brought together at year's end under the control of a *virtual* schema for corporate-wide reporting.

7.0 Implementing the Static Relationship Application

The fundamental goals of a static relationship database can be only to:

- Create a data organization that mirrors the fundamental processes in the business organization
- Impose maximum DBA control over user updating and reporting of data in the database

There cannot be different goals, because the static relationship DBMSs will not permit them to be implemented.

7.1 Static Relationship Application The Logical Database

- The database design process should take considerable time
- Designer should determine the *natural* organizations of data.
- Existing data processing systems may only throw the designer off track
  - Typically implemented to produce ad hoc reports.
  - Ad hoc data collections
  - Ad hoc programs to transform the collected data into the report.
  - When another report was needed, another system was created.
  - Finally, all the month-end systems were combined into one to reduce the multitude of file extracts, sorts, and prints.
  - These systems were set for data processing efficiency, not the organization's efficiency.

But organizational efficiency is what database is all about.

- Once information requirements analysis is complete, the real database design effort can begin
  - ♦ High-level design: serves the needs of the whole organization.
  - ♦ Second level set of database designs: addressing a specific database problem within overall scope
  - ♦ The combined scope of the second level designs should approximate all the high level design.
  - ♦ Finally, the lowest level of design should be constructed in the DBMS's data definition language syntax.

- Products generated during the first two design levels are all part of the business model.
  - ♦ First Level: Organization as a whole, and requirement for the middle level.
  - ♦ Mid Level: specific application or business-unit. Mid-level is requirement for low-level.
  - ♦ Lowest level design applies both to a specific application and needs of a specific DBMS.

7.2 Static Relationship Application Physical Database

- Static relationship DBMSs the storage structure and access strategy are normally complex.
  - Included are different kinds of pointers, indexes, relationships, storage organizations, overflow tables, and access techniques.
  - These give a static relationship database its speed.
  - These things bind static relationship database designs to their applications.
- There are two other significant parts of the physical database:
  - Data loading subsystem
  - Data update subsystem

Data Loading Subsystems

- The data loading subsystem is a one-time-only application.
  - Database integrity begins with this subsystem.
  - A static relationship database is best suited to be a non-redundant data store for a large collection of related applications.
  - Before database, related applications often had data in different formats, lengths, code tables, and so forth.
  - All this data had to be brought together into one common format before loading into a database.
  - The data loading subsystem becomes the single melting pot for diverse data.
  - The data loading subsystems' design and implementation Takes a great deal of time;
  - Conversion of existing and running applications to the single database takes talent, skill, and cleverness.

Data Update Subsystems

- The data update subsystem is typically required for both batch and on-line updating.

- Most row access is through pointers that
  - Lead owner to owner,
  - Owner to member,
  - Member to member, and
  - Member to owner.

- This traversal process must be very carefully designed to make it efficient.

- Efficiency is required because the many different applications must have their formerly separate update and retrieval transactions coordinated and engineered to keep transaction response times acceptable.

7.3 Static Relationship Application Interrogation

- Most static relationship database interrogations are through host languages like COBOL or FORTRAN. This is for two reasons:
  - ♦ The database structure is often so complex that the sophistication of a COBOL-like language is typically required to make best use of it.
  - ♦ A sophisticated natural language, which would allow any kind of interrogation to be formulated, traditionally operates too slowly.
- A static relationship database derives its efficiency from:
  - Strict structure and
  - Pre-planned road maps of pointers.

Query-Update Languages, Procedure Oriented Languages & Report Writers

- Sophisticated query-update language enables users to easily formulate interrogations that do not map to the database's design.
- Most static relationship DBMS vendors do not have very sophisticated query-update languages.
- They have procedure-oriented languages, report writers, and elementary query-update languages.

7.4 Static Relationship Application System Control

- System control activities
  - ♦ Control database,
  - ♦ The DBMS,
  - ♦ The application.

- Most significant to static relationship applications is reorganization.

- Since almost all programs are created with the database structure in mind, changing that structure is a complicated job.

- The process of reorganization must be very carefully planned so that the affected programs are identified and researched for the required reprogramming effort.

- Whenever data is needed from several static relationship organized databases, the normal method of extraction is through host language interfaces (HLI).

- Most static relationship DBMSs do not permit multiple database processing capabilities for either their query-update or report writing languages.

7.5 Static Relationship Application Summary

- The most important facts:
  - ♦ All aspects of its design should remain unchanged for as long as possible.
  - ♦ Application designers should spend as much time as possible on analysis and design.
  - ♦ If cut short, then there will be many expensive database reorganizations.
  - ♦ If the database is designed correctly from the start, then it will run efficiently for a long time.

8.0 Implementing the Dynamic Relationship Application

- The fundamental goals of the dynamic database environment are:
  - ♦ To create databases from existing collections of data processing files or single-purpose applications
  - ♦ To allow the users maximum control over the updating and reporting from their databases
- User control over updating and reporting cannot be restricted as the facilities enabling user control permeate dynamic relationship DBMSs.

8.1 Dynamic Relationship Application Logical Database

- Database design takes place very informally.
- Simple flat file-like structures can be created by coding up the DDL and submitting it to the DBMS.
- For dynamic relationship DBMSs, *file* and *table* have almost the same meaning.
- Complex structure of several interrelated tables, can also be created.
- Interrelationships are encoded into
  - Views or
  - DML language expressions,
  - Executed by the DBMS when the interrogation is submitted.
- Some dynamic relationship DBMSs require connecting columns to be same data type
- Other than that, the database design process is quite simple.

8.2 Dynamic Relationship Application Physical Database

- Table & O/S File relationship.
  - ♦ Some dynamic relationship DBMSs implement each table as a distinct operating system (O/S) file.
  - ♦ Some allow the combination of several of these tables into one O/S file.
  - ♦ This is usually done to increase performance.
  - ♦ When this is done, though, some of the flexibility of the independent O/S file approach is lost.

Relationship Processing

- Most of the dynamic relationship DBMSs allow for the creation of indexes, both primary (unique value required) and secondary (non-unique values allowed).
    - They use primary indexes to locate individual rows
    - Secondary indexes to locate sets of rows.
    - Dynamic relationship DBMSs use primary index from one table as a secondary index of another table as a way to implement a parent-child relationship.
    - Dynamic relationship DBMS may allow the user to state which two columns relate the tables.

Data Loading Subsystems

- The data loading subsystem for a dynamic relationship application is also informal.
- Dynamic relationship DBMSs often provide data loading utilities that transform a data processing file's records into a database's table's rows.
  - If the transformation is simple, the utility works.
  - If the transformation isn't simple, the user must accomplish data loading through a host language interface program.

Data Update Subsystems

- Data update is performed by the users, one table at a time.
- Since each table is independent of any other, it is of little concern when other users update rows from other tables.
- Simple collections of tables can be updated by one of the DBMS's natural languages.
- Complex updates are normally done through host languages.

8.3 Dynamic Relationship Application Interrogation

- Most dynamic relationship DBMSs are supported by very well-developed natural languages.
- Some are actually very sophisticated programming languages that allow multiple table access, automatic report formatting, code table look-up, branching, looping, terminal prompting, definition, and invocation of stored procedures.
- Dynamic relationship DBMSs that do not employ the ANSI/SQL language often have under developed host language interfaces.
- Few users ever need to use COBOL or FORTRAN given sophisticated natural languages.

8.4 Dynamic Relationship Application System Control

Reorganization:

- Simple process with a dynamically organized database.
- A utility is invoked, the column types are added, deleted, or changed, and the DBMS automatically reorganizes the table.
- Whenever there is a new relationship between two tables, only one table is changed by incorporating the new column type.
- Following the addition of the column type, the user creates the update program to store values required by this new column for the relationship.
- Once valued, the basis of the relationship is in place, waiting to be used.

Concurrent, Multi-Table Operations

- One drawback of a dynamically organized database application occurs when multiple table updates are attempted.
- The user or program attempting the update must acquire locks over all the tables involved.
- This is difficult, because a user is normally allowed to *open-for-update* only one table at a time, and a row must be opened before it can be locked.

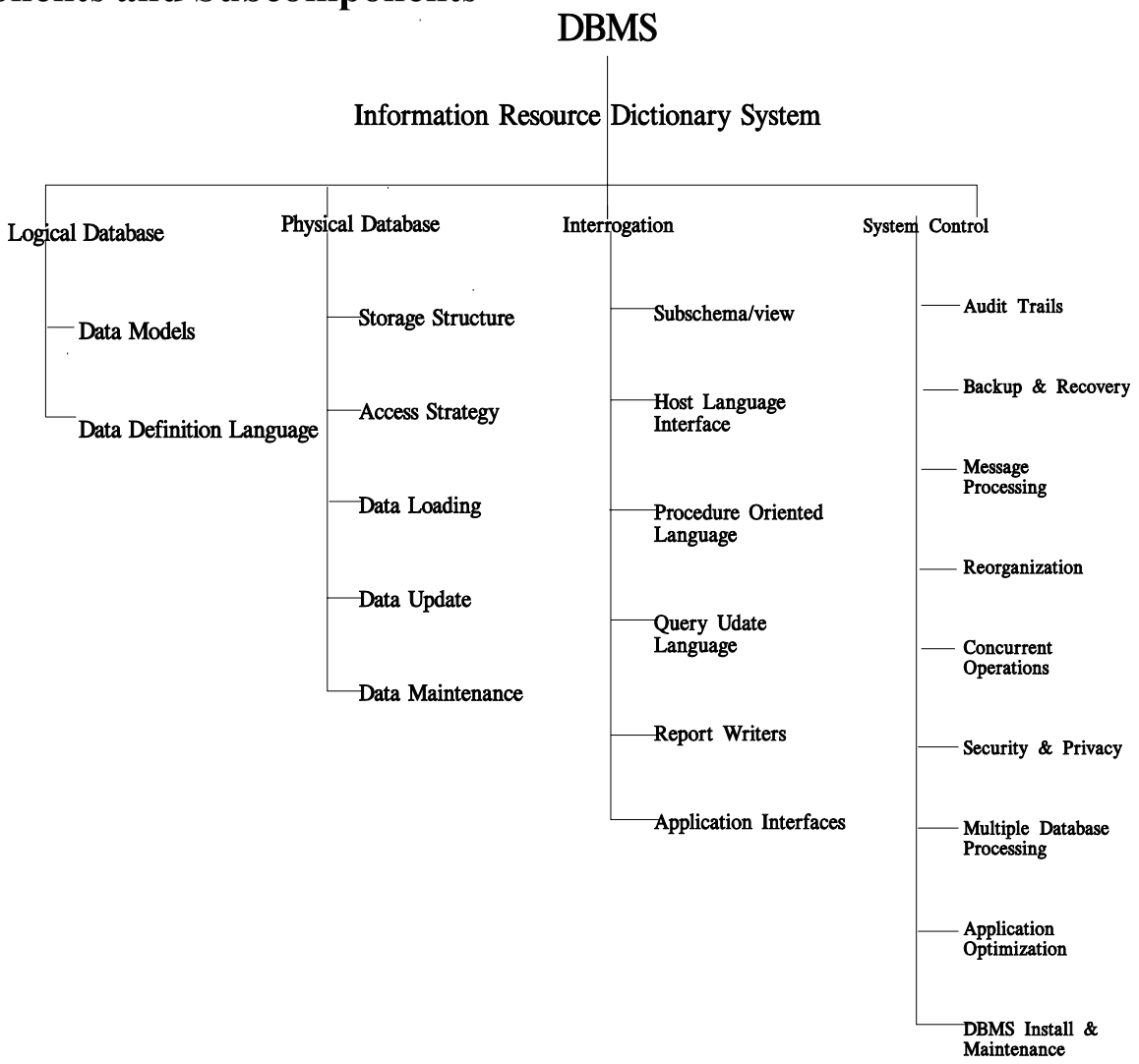
Audit Trails

- Construction of a multiple table audit trail is also difficult.
- Audit trail are usually made on a table basis.
- Integrated audit trail can only be created through the update program.
- Host language program should be used.
- All other languages must be very sophisticated to construct multi-table (multiple rows across multiple tables) audit-trail transactions.

8.5 Dynamic Relationship Application Summary

- Key fact: User is in control of database definition, updating, reporting, and many of the system control activities.
- Dynamic relationship DBMS facility is a *do your own thing DBMS*.
- Disastrous for payroll, accounts payable, and the like
- For some applications, like marketing research, it is ideal.

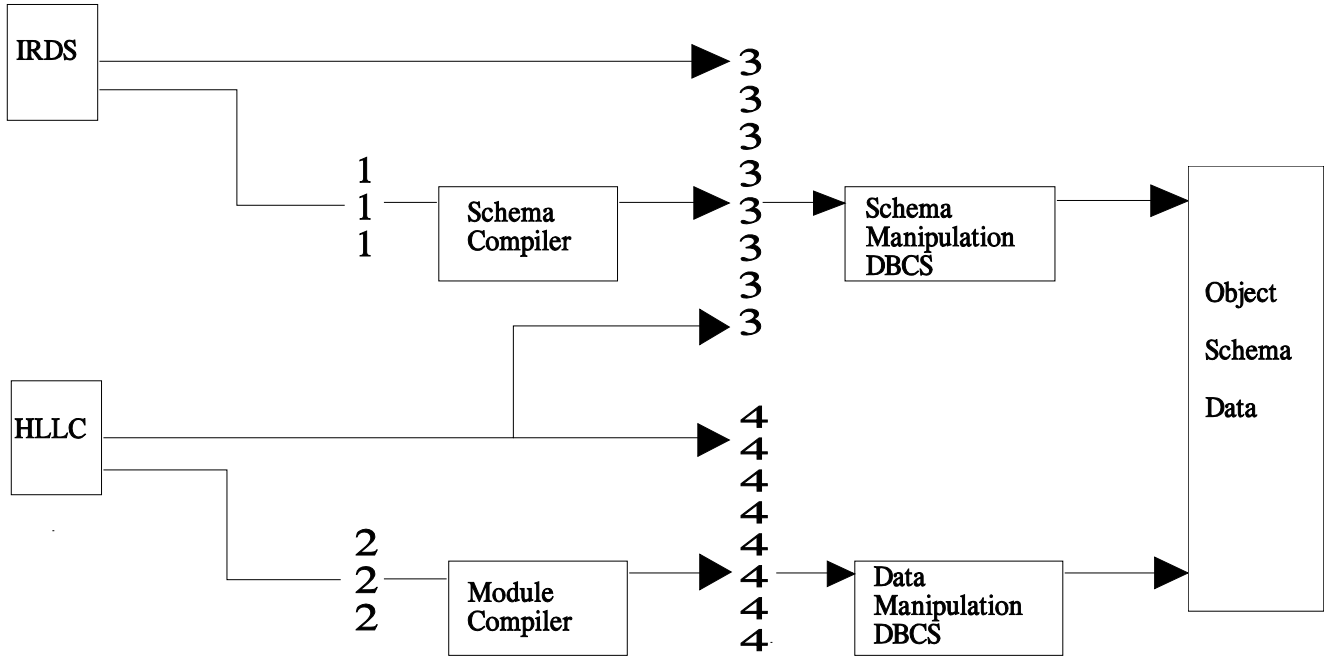
9.0 DBMS Components and Subcomponents



ANSI/SPARC, in their 1975 draft reference model report, recognized the critical importance of interfaces by stating:

> In the course of the early discussions [about a reference model], it emerged that what any standardization should treat is interfaces. There is potential disaster and little merit in developing standards that specify how components are to work. What is proper for standards specification is how the components are meshed; in other words, the interfaces. (1)

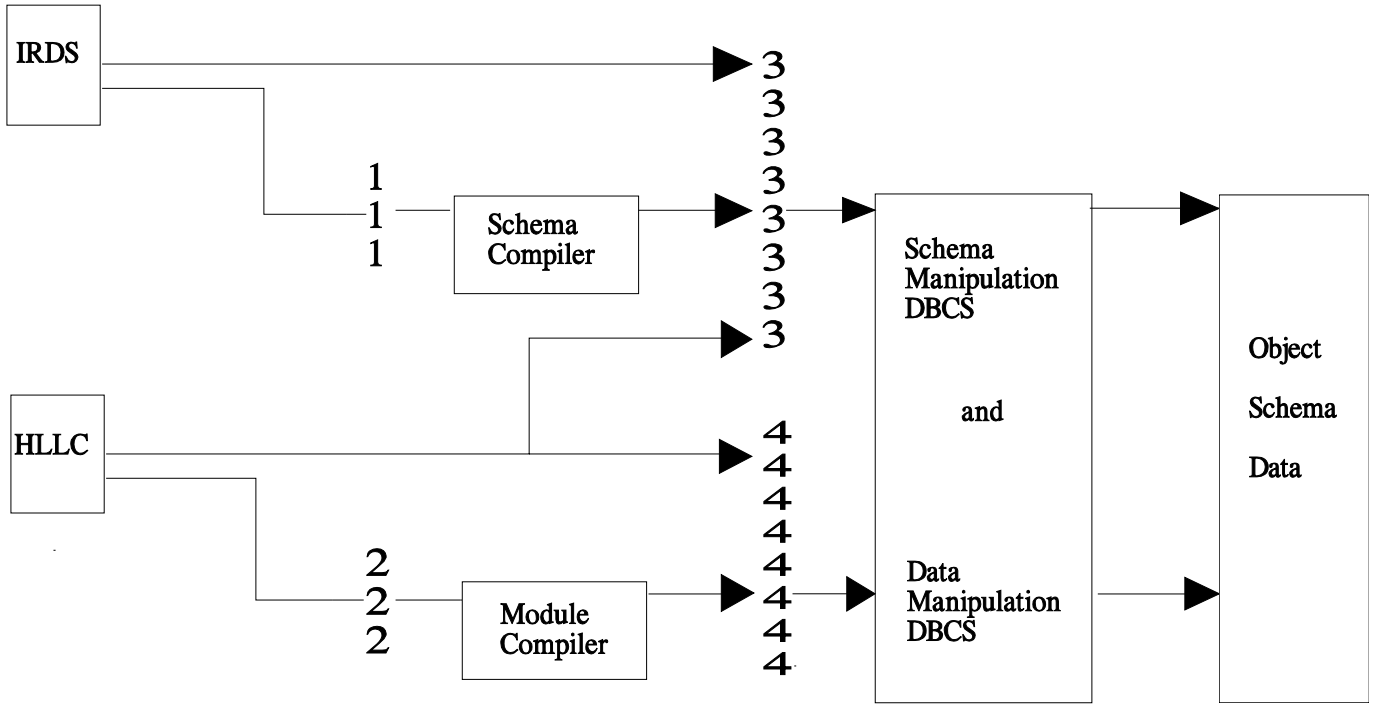
Network Data Language (NDL) Interface Architecture



Legend:

1 = Schema Definition Language
2 = Procedural Language
3 = Schema Manipulation Language
4 = Data Manipulation Language
IRDS = Information Resource Dictionary System
HLLC = High Level Language Compiler, e.g.,
Cobol, Query, POL

SQL Interface Architecture



Legend:

1 = Schema Definition Language
2 = Procedural Language
3 = Schema Manipulation Language
4 = Data Manipulation Language

IRDS = Information Resource Dictionary System
HLLC = High Level Language Compiler, e.g., Cobol, Query, POL

10.0 The DBMS

All fully functional DBMSs contain four distinct sets of facilities:

- **Logical database**: Structuring of a database according to one or more of the four popular data models

- **Physical database**: Enable data loading, data update, various access strategies that affect storage structure design, and facilities to maintain the database, i.e., backup. Data update and database backup are included within physical database to understand their physical effect on the database

- **Interrogation**: Retrieval and update of the database's rows through a number of interrogation languages. These range from simple-to-use non-programmer facilities to sophisticated languages for professional programmers

- **System control**: Audit trails, backup and recovery, message processing, reorganization, concurrent operations, security and privacy, multiple database processing, DBMS installation and maintenance, and database application optimization.

10.1 The Logical Database

The logical database component of a DBMS enables user definition of

- Tables
- Columns
- Explicitly defined relationships
- Operations

10.1.1 Tables

Tables: Name, Description, Primary Key, Secondary Key, Alternate Unique Keys

10.1.2 Columns

Column Types	
Column Data Type	**Definition**
Single Value	Each component represents a single value such as Birthdate with the value 11/11/1987
Multi-value	Each component represents multiple values such as Nicknames with values "Buddy, Guy, Mac"
Groups	Each component has subcomponents to represent single-set of values such as Address with Street-1, Street-2, City, State, Zip
Repeating Groups	Each component has subcomponents to represent multi-sets of values such as Dependents that contains subcomponents, Dependent Name, Dependent Birth date, Dependent SSN.
Nested Repeating Groups	Employee (Dependents (Hobbies))

Column Types by Data Model				
Column Type	**Network**	**Hierarchical**	**Independent Logical File**	**Relational**
Single Valued	✔	✔	✔	✔
Multi-Valued	✔		✔	
Multi-Dimension	✔			
Group	✔			
Repeating Group	✔		✔	

Column Types by SQL Version				
Column Type	**SQL 1986**	**SQL 1989**	**SQL1992**	**SQL 1999, 2003**
Single Valued	✔	✔	✔	✔
Multi-Valued				✔
Multi-Dimension				✔
Group				✔
Repeating Group				✔

10.1.3 Relationship Types

Name	Example
One-to-many	Employee to dependents
Owner-multiple-member	Territory contains salesmen and customers
Singular-one-member	Top performing employees
Singular-multiple-member	Top performing current, former, part-time, and retired employees
Recursive	Organization contains organization
Many-to-many	Automobiles and owners
One-to-one	Table and its primary key
Inferential	Many houses each with a location, and then buyer with desired location

Relationship Types by Data Model				
Relationship Type	**Network**	**Hierarchical**	**Independent Logical File**	**Relational**
Owner & one member	✔	✔	✔	✔
Owner & multiple member	✔			
Singular & one member	✔			
Singular & multiple member	✔			
Recursive	✔			
Many-to-many			✔	
One to One	✔		✔	✔
Inferential			✔	

Relationship Types by SQL Version				
Relationship	**SQL 1986**	**SQL 1989**	**SQL1992**	**SQL 1999, 2003**
Owner & one member	✔	✔	✔	✔
Owner & multiple member				✔
Singular & one member				✔
Singular & multiple member				✔
Recursive				✔
Many-to-many				✔
One to One				✔
Inferential				✔

10.1.4 Operations

Record Operation Type	
Operation	**Description**
Find	SELECT According to STORED Order
Get	Obtain Record From Find
Add	Install a New Row Into Database
Delete	Remove an Existing Row From Database
Modify	Change Some Data Column Values in Existing Row

Record Operation Types by Data Model				
Operation Type	**Network**	**Hierarchical**	**Independent Logical File**	**Relational**
Find	✔	✔	✔	✔
Get	✔	✔	✔	✔
Add	✔	✔	✔	✔
Delete	✔	✔	✔	✔
Modify	✔	✔	✔	✔

Record Operation Types by SQL Version				
Relationship	**SQL 1986**	**SQL 1989**	**SQL1992**	**SQL 1999, 2003**
Find	✔	✔	✔	✔
Get	✔	✔	✔	✔
Add	✔	✔	✔	✔
Delete	✔	✔	✔	✔
Modify	✔	✔	✔	✔

Relationship Operation Types	
Operation	**Description**
Connect	Add to a Named RELATIONSHIP in Specific Order
Disconnect	Delete From RELATIONSHIP
Get Owner	Obtains The Parent of the Row That is Current
Get Member	Obtains the First Child of the Owner For the Named Relationship
Get next	Obtains the Next Row Within The Named Relationship
Intersect	Find and Keep Only the Common
Difference	Find and Keep Only the Not Common
Join	"Append" Relations to Each Other
Divide	Subset
Product	Cross-Product
Union	Merge and Drop Duplicates

Relationships Operations by Data Model				
Operation Type	**Network**	**Hierarchical**	**Independent Logical File**	**Relational**
Connect	✔			
Disconnect	✔			
Get Owner	✔	✔		
Get Member	✔	✔		
Get Next	✔	✔		
Intersect				✔
Difference				✔
Join			✔	✔
Divide				✔
Product				✔
Union				✔

Relationship Operations by SQL Version				
Relationship	**SQL 1986**	**SQL 1989**	**SQL1992**	**SQL 1999, 2003**
Intersect	✔	✔	✔	✔
Difference	✔	✔	✔	✔
Join	✔	✔	✔	✔
Divide	✔	✔	✔	✔
Product	✔	✔	✔	✔
Union	✔	✔	✔	✔

10.2 The Physical Database

- DBMSs allow an index to be defined for any column.
- These indexes are used in selection clauses to enhance processing.
- Sophisticated access strategies optimize the access of rows by using all the indexed columns referenced, or by ignoring the fact that a column is indexed when list processing is slower than row processing.
- Variable physical "data" designs such as including
  - All of a table's data on one O/S file,
  - Storing the data from several tables on one O/S file, or
  - Spreading the data from one table across multiple O/S files.
  - Variable physical designs enable portions of a very large database to be moved off-line when there is no need for on-line access.
- Alternative O/S access methods,
- Blocking factors
- DBMS row sizes to achieve even more highly tuned performances.

10.3 Interrogation

Language Type	Description
Host Language Interface	A programming language that acts as a "host" for DBMS language commands. Most commonly the Host Language is able to compiled and executed. These languages are commonly standardized by ANSI committees. Examples are COBOL, Fortran, C, C++
Procedure Oriented Language	A programming language that is also often referred to as a 4GL (fourth generation language). This language type may either be executed, once compiled or directly interpreted, line by line. No POL language, save SQL/PSM has been standardized by an ANSI Committee. Most often these languages are "privately owned," vs public domain. Access to DBMSs and in turn databases is commonly either through inclusion of DBMS language commands (like Host Languages) or through Open Database Connectivity (ODBC). The ANSI Standard SQL Persistent Stored Module (PSM) falls into the category of a POL.

Language Type	Description
Query-update Language	A single statement language that accomplishes a single purpose such as adding, deleting, or modifying a row of data, or in some cases, sets of rows of data. The ANSI standard SQL language falls into this category.
Report-writer	A programming language is geared towards creating reports. This language is similar to a POL language but has special verbs and operations geared to reports.

Language Selection Criteria	INTERROGATION LANGUAGE TYPE			
	Host Language Interface	**Procedure Oriented Language**	**Report Writer**	**Query Update Language**
Task development effort	High	Medium	Medium	Low
Relative work units	100	10	10	1
Level of user control over database interaction	Low	Medium	Medium	High
Range of portability from one DBMS to another of the same data model	Medium to High	Low	Low	Low

10.4 System Control

System Control Type	Description
Audit Trails	The Capture of Database Update Information
Message Processing	Exception Condition Processing
Backup and Recovery	Copying & Restoring Databases
Reorganization	Logical & Physical to Restore Optimum Organization
Security and Privacy	Protecting Data from Theft
Multiple Database Processing	Mechanisms to Access Several Physical Databases
Concurrent Operations	The Specification of Conflicting Operations
Application Optimization	The Discipline of Reducing Resource Consumption
Installation and Maintenance	Procedures to Install, Validate, and Maintain DBMS Software

11.0 DBMS Issues

- Use of some DBMS facilities provokes argument. Each side of the argument usually has sound logic behind its position.

- For example, the issue of sorting.

  - For dynamic DBMS, sorting data before it is presented to the user is a critical operation.

  - For the DBMS maintain rows in a predetermined order is a waste of time, as users will always want rows in a different order.

  - For the static relationship DBMS, if database is properly designed it's a waste computer time to sort rows because the DBMS would have them into a *proper* order.

  - Besides, if sorting does have to be done, the user should retrieve the rows and let the operating system utility sort them, relieving the DBMS of such a ridiculous and unimportant task.

12.0 Application Components and DBMS Components

- There are significant differences between the components of a database application and the components of a DBMS.

- Despite the differences, the application and the DBMS are inextricably linked. The differences are related to perceptions and use sequencing.

Issue and Consequences	DBMS Maintained Sorting	User Maintained Sorting
Flexibility	Reduces Ability to Create Different Sort Orders on Reports	Users Can Develop Reports or Updates in Different Order
Performance	Slows Updates But Speeds Reports Using DBMS Provided Record Ordering	Speeds Updates, But Slows Reports as Each Run Unit Must Perform its Own Record Sorting

Managing Database Viewpoint	Interrelationships Among Database Viewpoints			
	Logical	**Physical**	**Interrogation**	**System Control**
Technology (2)	Data model	Database creation and maintenance	Data selection and reporting	Audit trails, protection and evolution etc.
Staffing (3)	Database specialist	DBMS specialist	Interrogation specialist	System control specialist
Project (4-8)	Conceptual specification phase Implementation phase Production and administration phase			
DBMS (9)	Schema and sub-schema	Access methods, data loading, update, and maintenance	Query, host language, report writers	Utilities, languages and techniques

13.0 DBMS Requirements Summary

- Sophisticated DBMSs support multiple data models.
- The two portable data models are ANSI/NDL (network) and ANSI/SQL (relational).
- Multiple data model supports enable databases to be built that are network only, hierarchical only, independent logical file only, or relational only.
- Today's popular combinations are network and relational (IDMS/R and SUPRA), hierarchical and relational (FOCUS), and independent logical file and relational (ADABAS and Model 204).
- The interface language of choice for the network and relational data models is NDL and SQL respectively.
- Regardless of the data model, all DBMSs must have sophisticated editing, validation, and referential integrity.
- Sophisticated DBMSs also support multiple access methods for the files.
- Depending on the kind of performance needed, the relationships supported between tables and files should, at a minimum, be one-to-one, one-to-many, or many-to-many.

- Modern DBMSs have a variety of languages, such as query-update languages, procedure-oriented languages, and report writers, so that whole applications can be implemented without resorting to the use of host languages.

- Finally, the complete DBMS has a rich set of system control facilities for backup and recovery, and audit trails.

- Critical to an effective environment is multiple database processing, multiple threaded operations, and the ability to manipulate the DBMS software and buffers to achieve different performance characteristics.