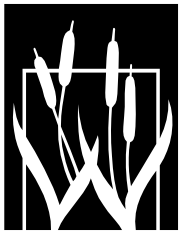


Whitemarsh
Information Systems Corporation

Database Management Systems: Understanding and Applying Database Technology

Chapter 1

ANSI Database Standards

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

ANSI Database Standards

- In the Beginning
- Reference Models
- Generations of Database Management Systems
- De Jure Database Standards Organizations
- The Battle over Data Models
- DBMS Standards
- ANSI/NDL
- ANSI/SQL
- ANSI/SQL (1999)
- THE ANSI/IRDS
- Database Standards Summary

1.0 In the Beginning...

- **Mid 1960s** the Systems Committee of CODASYL (Committee On Data Systems and Languages) undertook two surveys of existing DBMSs.

- The CODASYL organization then created several committees to develop specifications for a network DBMS.

- The American National Standards Institute (ANSI) reviewed the CODASYL work and in the early 1970s began the database standards process by developing a **reference model**.

- The process to standardize syntax and semantics of DBMS began with the establishment of the committee X3H2 in 1978

  - NDL is for network databases,
  - SQL for relational databases.

- Another ANSI committee, X3H4, developed the Information Resource Dictionary System (IRDS) standard for storing the metadata

2.0 Reference Models

- A reference model is a mechanism or framework for describing a database management system in terms of its interfaces, processing functions, and the flow of data through the DBMS.
- The ANSI Standards Planning and Requirements Committee (SPARC) published in 1975 and became known as the ANSI/SPARC reference model.
- The complete reference model was never finished, and the ad hoc study committee was disbanded in 1975.
- The most significant contribution of the ANSI/SPARC architecture was the recognition of the need for an organization to have a conceptual schema.

1975 ANSI Reference Model

There are three realms of interest in the philosophy of information.

- Real world,
- Ideas about the real world existing in the minds of men, and
- Symbols on paper or other (storage) medium representing these ideas.

Regarding, symbols on paper or other (storage) medium representing these ideas, .

- **External**, including a simplified model of the real world as seen by one or more applications;
- **Conceptual**, including the limited model of the real world maintained for all applications; and
- **Internal**, including the data in computer storage representing the limited model of the real world.

More Specifically,

- An external schema that contains the specifications of the external objects, the associations and structures in which they are related, operations permitted upon these objects, and administrative matters.

- A conceptual schema that includes the specification of the conceptual objects, and their properties and relationships, operations permitted on these objects, consistency, integrity, security, recovery, and administrative matters.

- An internal schema that includes the specifications of the internal objects, indices, pointers, and other implementation mechanisms, other parameters affecting (optimizing) the economics of internal data storage, integrity, security, recovery, and administrative matters.

1982 ANSI/SPARC Ad Hoc Database Study Group Issues Studied

- Is metadata different from data?
- Are metadata and data stored separately?
- Are metadata and data described in terms of different data models?
- Is there a schema for metadata--a meta schema?
- Are there also external and internal meta schema?
- Are the interfaces used to retrieve and change metadata different from those used to retrieve and change data?
- Can a schema be changed on-line?
- How would on-line schema changes affect the data?

The 1975 Draft Data Management **Reference Model Was Too Simplistic as it Did Not Address:**

- Distributed schemas that may result from distributed database processing
- Open systems architecture
- Establishment of a unified concurrency model (teleprocessing and database interactions)
- Operating system (O/S) based security and privacy

ISO Data Management **Reference Model**

Objectives:

- The identification of interfaces
- The positioning of all such interfaces relative to each other
- The identification of facilities provided at each interface
- The identification of the process and, where appropriate, specific data which supports each interface
- The positioning of the use of the interfaces in terms of an information systems life cycle
- The identification of the binding alternatives associated with each appropriate identified interface

ISO/IEC Data Management Reference Model

- **Dictionary Definition**: the specification of the types of data that an organization requires in its dictionary system. This provides the schema for the dictionary database, referred to as the dictionary schema.

- **Dictionary Use**: the storage and retrieval of dictionary data relating to all aspects of other information systems with special dictionary capabilities, such as keeping many versions of data. The dictionary database includes the data definitions for application databases.

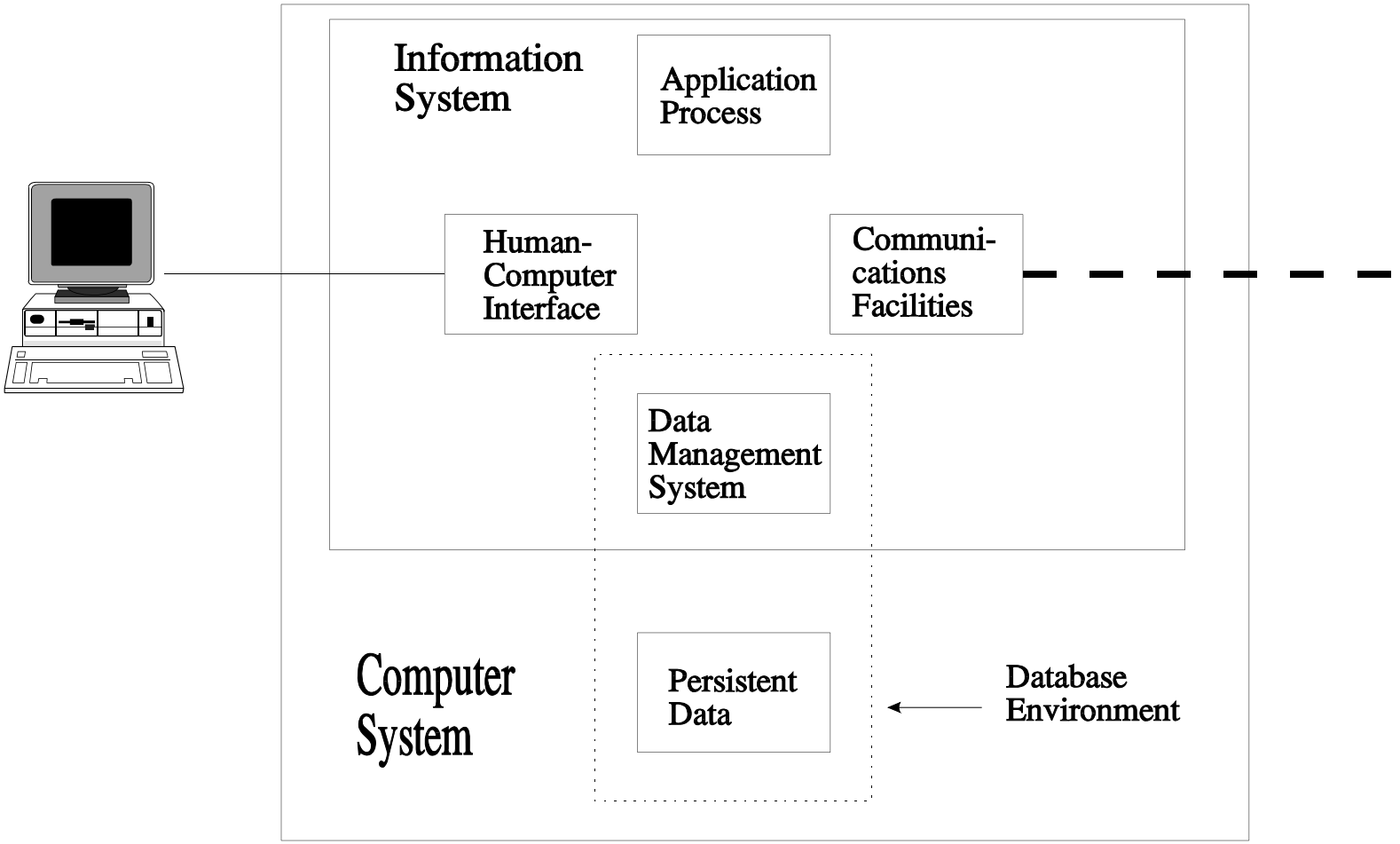
- **Application Schema Definition**: the specification of the schema in the form appropriate to the DBMS. Depending on support for the DBMS provided by an IRDS, the dictionary system may provide the schema based on its data descriptions.

- **Application Database Use**: the storage and retrieval of data from an arbitrary database structure.

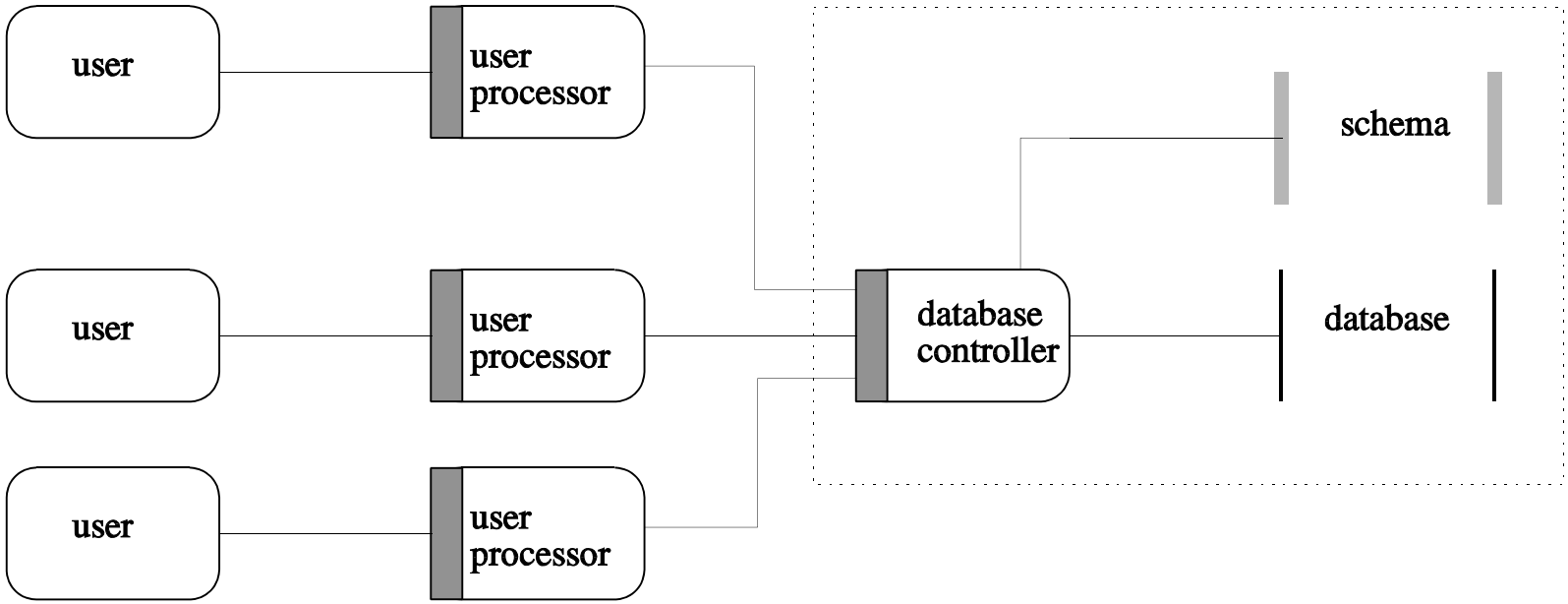
- **Database Creation**: the establishment of a database.

- **Database Maintenance**: the changes to a database required as a result of changes to its schema.

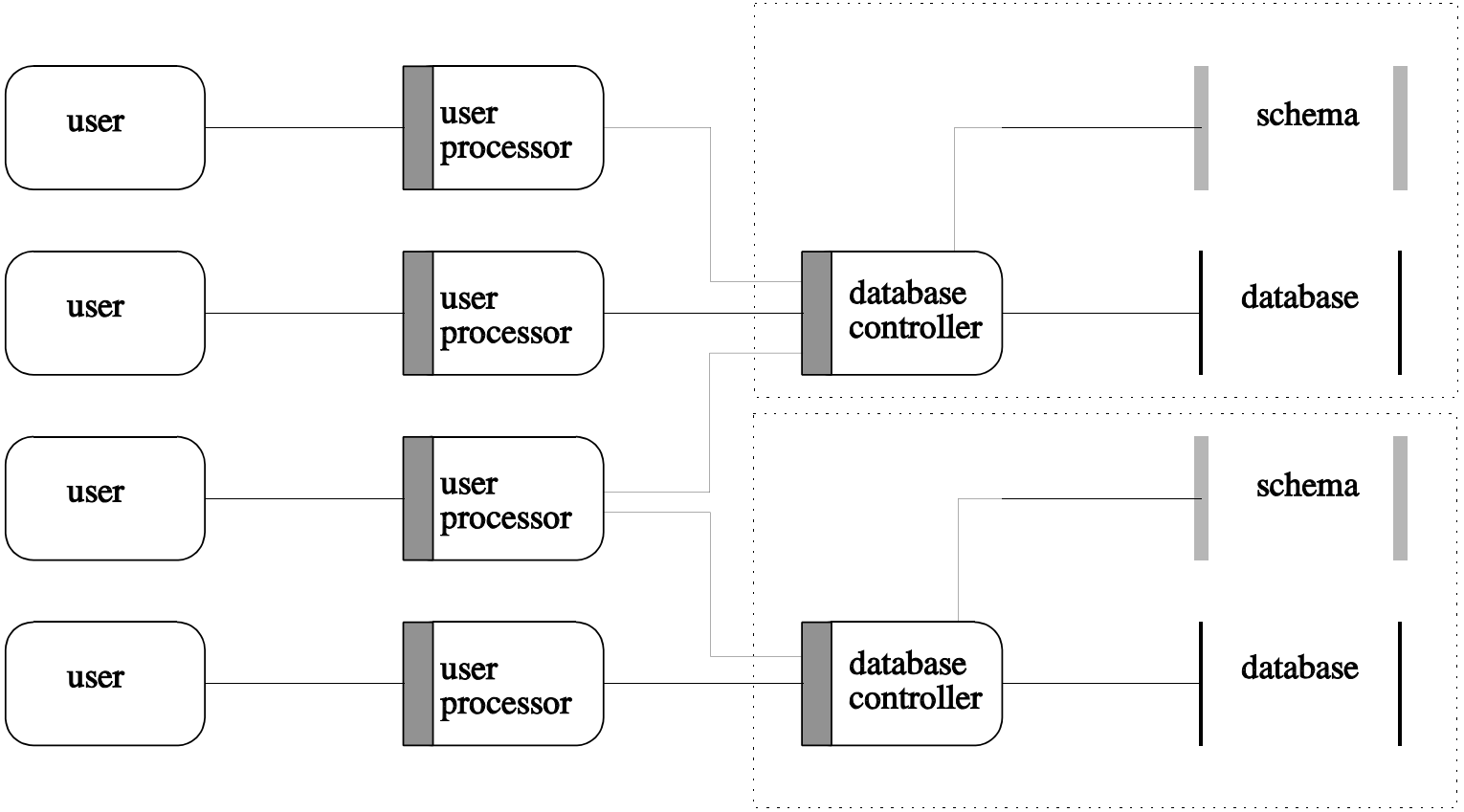
Data Management within an Information Systems Architecture



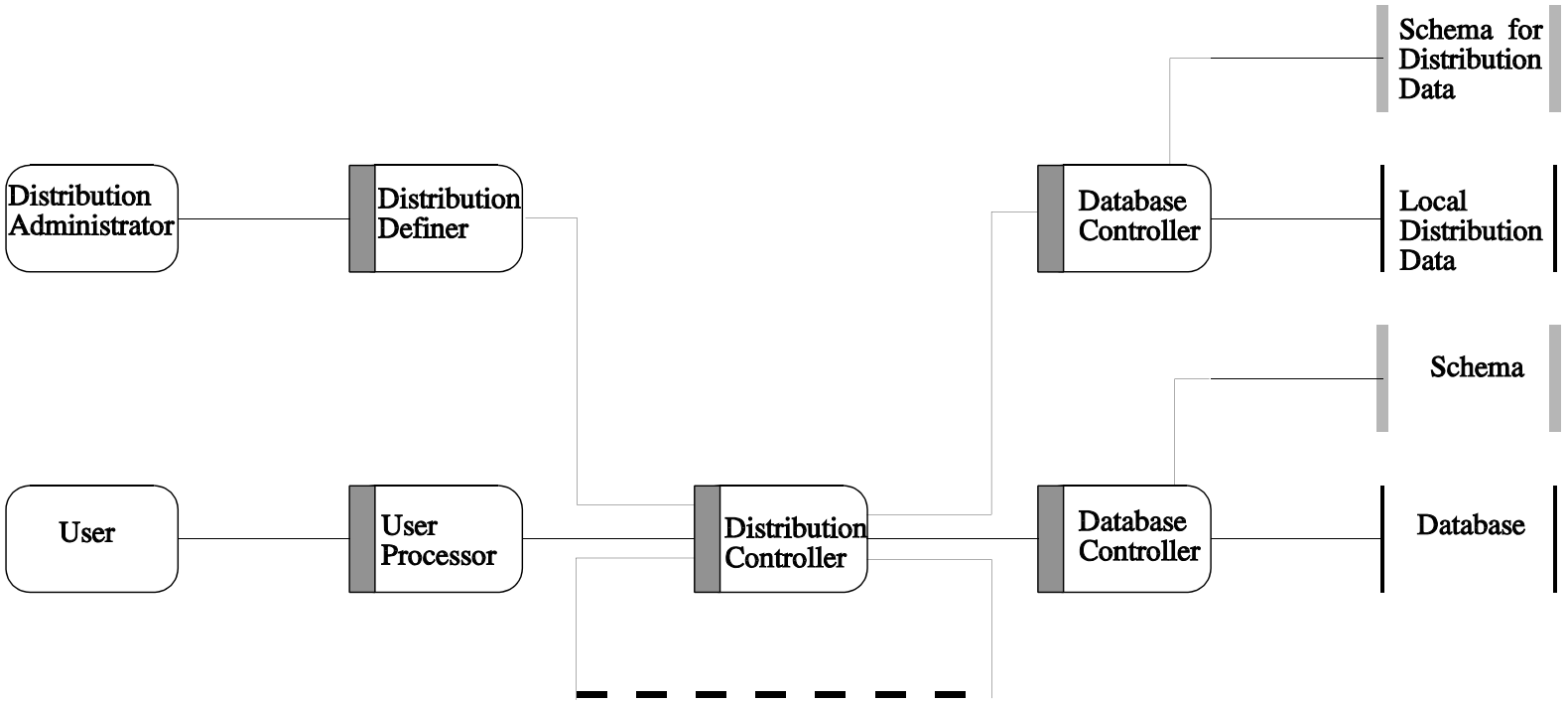
Multi-user Access to a Single Database



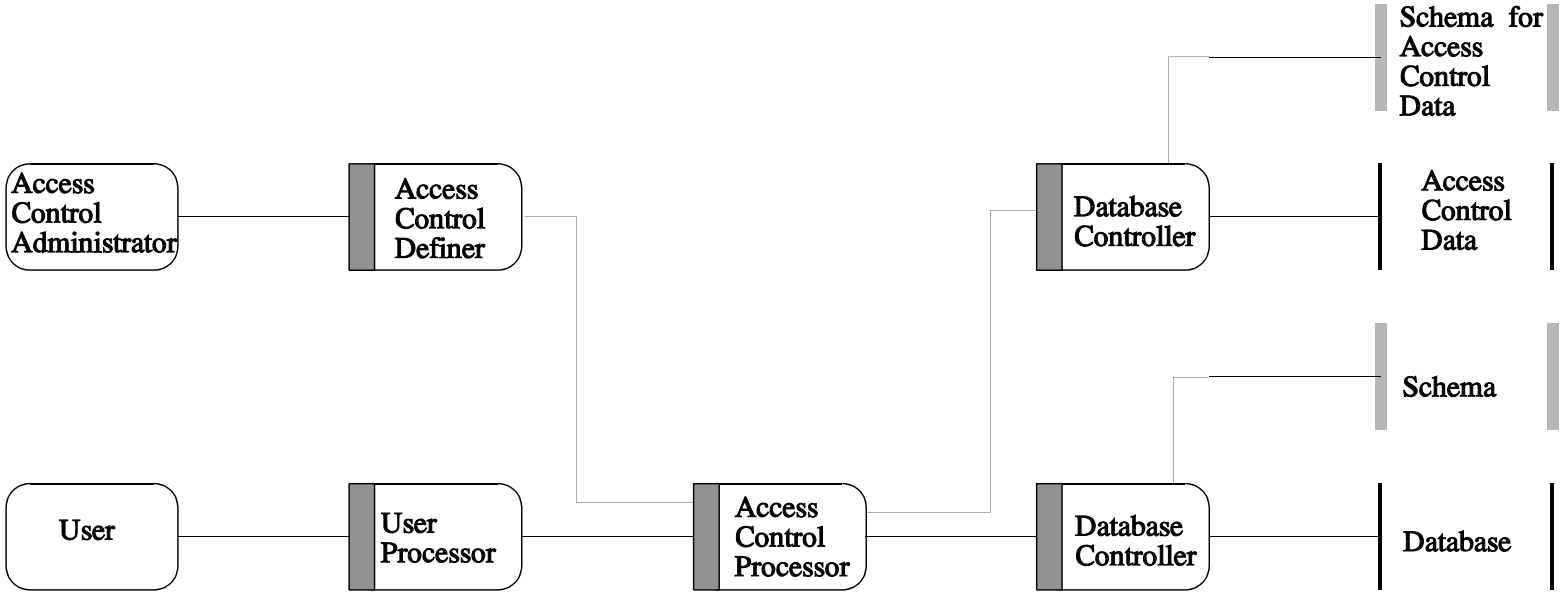
Multi-user Access to Multi-Databases



Distributed Database



Access Control within Database



3.0 Generations of Database Management Systems

3.1 First Generation of DBMS Technology: (1955-1970)

- National defense systems
- Commercial systems
- File oriented systems

3.1.1 National Defense Systems (1955-1970)

- Critical requirements
  - ♦ Threat identification
  - ♦ Threat analysis
  - ♦ Threat response
- Required human characteristics
  - ♦ Hierarchical data structures
  - ♦ Natural language for access
  - ♦ Inverted access

National Defense Systems (cont.)

- Typical systems
  - Adam (Advanced data management)
  - Madam (More advanced data management)
  - TDMS (Time/shared data management system)
  - System 2000
- All systems were self-contained. There were natural languages for
  - Data structure definition
  - Data loading
  - Data update
  - Reporting
  - No access through COBOL or FORTRAN

3.1.2 Commercial Systems (1960-1970)

- Critical requirements
  - ♦ Preserve the investment in COBOL code
  - ♦ Interact with complex record structures
  - ♦ Rapid record processing for standard reports
- Required computer characteristics
  - ♦ Network data structures
  - ♦ Interface through COBOL
  - ♦ Traditional record then value access

Commercial Systems (Cont.)

- Typical systems
  - Honeywell's IDS-2 (network data model, record processing)
  - Cullinet's IDMS (network data model, record processing)
  - Univac's DMS-1100 (network data model, record processing)
  - IBM's IMS (hierarchical data model, record processing))
- All systems were host-language based. There were natural languages for:
  - Data structure definition and modification
    - Host (COBOL) language for all other access
    - Data loading
    - Data update
    - Reporting
  - No access through natural languages

3.1.3 File Oriented Systems (1965-1970)

- Critical requirements
  - Easy access to established files
  - Interact with simple structures
  - Access for standard and ad hoc reports
- Required computer characteristics
  - Simple file structures
  - Interface through COBOL or a natural language
  - Traditional record then value access or
  - Access to record through indexes

File Oriented Systems (Cont.)

- Typical systems
  - Inquire
  - Model-204
  - Adabas
  - Ramis, Nomad, Focus
- All systems were mixed language access
  - Host language interface for COBOL, C, etc access
  - Natural languages for:
    - Data structure definition and modification
    - Data loading
    - Data update
    - Reporting

3.2 Second Generation of DBMS–Relational DBMS (1970-1998)

- Critical Requirements
  - ♦ Independent, two-dimensional file structures
  - ♦ Relationships based exclusively on pre-defined, shared values
  - ♦ Set processing versus record-at-a-time processing
  - ♦ Access through simple structured query-language (SQL) style commands

- Required computer/human characteristics
  - ♦ Very fast disk drives and large computer memory
  - ♦ End-users need hi-level of database design knowledge for record selection, and relationship processing.
  - ♦ Transformation of set processing to record-at-a-time processing for common computer languages

Second Generation of DBMS–Relational DBMS (cont.)

- Typical Systems
  - Oracle Corporation's Oracle
  - Informix Corporation's Informix
  - Sybase Corporation's Sybase
  - IBM Corporation's DB/2
- Natural languages for data structure definition and modification
- Natural language for
  - Data loading
  - Data update
  - Reporting

3.3 Third Generation of DBMS (1998—)

- Critical requirements
  - ♦ Independent table structures with nested substructures of self-contained data
  - ♦ Value based and "pointer-based" relationships between "records" from different tables
  - ♦ Robust integrity constraint language
  - ♦ Rich data types (blobs, clobs, abstract-data, multi-media, text processing, mathematics and spatial data processing)
  - ♦ Self contained programming language
- Required computer/human characteristics
  - ♦ Very fast disk drives and large computer memory
  - ♦ End-users need hi-level of database design knowledge for record selection, and relationship processing.
  - ♦ Transformation of set processing to record-at-a-time processing for common computer languages

Third Generation of DBMS (1998—)

- Probable Conforming Systems
  - Oracle Corporation's Oracle
  - IBM Corporation's DB/2
  - Microsoft SQL/server
  - CA's Ingress

- Natural Languages for:
  - Data structure definition and modification
  - Data loading
  - Data update
  - Reporting

4.0 De Jure Database Standards Organizations

Key Groups

- CODASYL
- ANSI
- ISO

CODASYL (Conference on Data Systems Languages)

- Founded in 1959, dissolved in 1989
- Goal: Development of data systems languages which are hardware independent
- Organization
  - Executive Committee
  - Standing Committees
  - Products: Journals of Development (JOD)

Key Committees

- COBOL
- FORTRAN
- Data Definition Language Committee (DDLC)

ANSI

- Founded in 1918 as American Engineering Standards Committee
- Became American Standards Association (1928)
- Became United States of America Standards Institute (1966)
- Became American National Standards Institute (1969)

ANSI Goals

- National standards coordination
- Approve & promulgate voluntary standards
- Focal point for government & industry coordination
- Represent USA to ISO

- **ANSI Organization**

  - Key Standing Committees

  - SMC (Standards Management Committee)

  - Key Technical Committees from INCITS (ANSI's International Committee for Information Technology Standards
    - H1--operating Systems
    - H2--database languages
    - H4–IRDS (this committee and the standard has now been dissolved)
    - J1--PL/1
    - J2--BASIC
    - J3--FORTRAN
    - J4--COBOL

ISO

- Started in 1926. Disbanded During WW-II and was restarted in 1946
- Goals
  - Coordinate National Standards
  - Set up International Standards
  - Exchange Information
- Organized by Technical Committees
  - TC-97 Computers and Information Processing
  - TC-97/SC32 Data Management and Interchange: [Develop] Standards for data management within and among local and distributed information systems environments. Provide enabling technologies to promote harmonization of data management facilities across sector-specific areas. Specifically, SC 32 standards include:
    - Reference models and frameworks for the coordination of existing and emerging standards;

- Definition of data domains, data types and data structures, and their associated semantics;

- Languages, services and protocols for persistent storage, concurrent access, concurrent update and interchange of data;

- Methods, languages, services and protocols to structure, organize and register metadata and other information resources associated with sharing and interoperability, including electronic commerce.

ISO (cont.)

- TC-97/SC32/ Database Committee Technical Subcommittees
  - WG-1: Open EDI – electronic data interchange
  - WG-2: Metadata – conceptual schemas, data dictionaries, metadata repositories, and CASE tools
  - WG-3: Database – SQL/99 main parts for core and package components of SQL/99
  - WG-4: SQL Multimedia and Application Packages – text and spatial data type definitions and processing rules
  - WG-5: Remote Database Access – specification of packages and rules for distributed database processing

Database Centric Working Groups and Projects

JTC1/SC32/WG3 Projects:

- Part 1: Framework
- Part 2: Foundation
- Part 3: Call-Level Interface
- Part 4: Persistent Stored Modules
- Part 9: Management of External Data
- Part 10: Object Language Bindings
- Part 11: Schemata
- Part 13: JRT
- Part 14: SQL-XML

JTC1/SC32/WG4 Projects:

- Part 1: SQL/MM Framework
- Part 2: SQL/MM Full-Text
- Part 3: SQL/MM Spatial
- Part 4: SQL/MM General Purpose Facilities
- Part 5: SQL/MM Still Image

Federal Information Processing Standards (FIPS)

- Established in 1965 by Brooks Bill
- Goals: Establish standards for all agencies
- Developed and administered conformance tests in support of Federal agency procurements
  - Cobol
  - Fortran
  - C
  - Posix
  - SQL
- Organization
  - Administered by National Institute of Standards and Technology (NIST)
  - Technical committees from agencies supporting NIST
- Effectively dissolved in 1996.
  - NIST terminated support of federal agency standards work
  - NIST terminated conformance testing

Flow of Information/interaction

Theory

- CODASYL Develops JODS (Journals of Development)
- ANSI adopts (Accept or Delete)
- FIPS adopts ANSI Standards (to regulate Federal Government procurement)

Actual Practice (through 1988)

- CODASYL develops COBOL and ANSI adopts
- CODASYL and ANSI Are the same for FORTRAN
- ANSI develops and standardizes database languages
- FIPS adopts ANSI standards and performs conformance testing

Today

- ANSI develops and standardizes languages (e.g., Cobol, Fortran, C, and SQL)

ANSI NCITS H2 Technical Committee on Database

SQL Projects:

- Part 1: Framework
- Part 2: Foundation
- Part 3: Call-Level Interface
- Part 4: Persistent Stored Modules
- Part 9: Management of External Data
- Part 10: Object Language Bindings
- Part 11: Schemata
- Part 13: JRT
- Part 14: SQL-XML

SQL MM Projects:

- Part 1: SQL/MM Framework
- Part 2: SQL/MM Full-Text
- Part 3: SQL/MM Spatial
- Part 4: SQL/MM General Purpose Facilities
- Part 5: SQL/MM Still Image

5.0 The Battle over Data Models

The ANSI/SPARC committee in 1972 addressed the battle concerning data models with the following statement:

> There is continuing argument on the appropriate data model: e.g., relational, network, hierarchical. If, indeed, this debate is as it seems, then it follows that the correct answer to this question of which data model to use is necessarily *all of the above*.

A data model is merely a formalized method of defining

- Columns
- Tables
- Relationships among Tables, and
- Operations on rows of tables and relationships between tables

Data Models (1965 - 1998)

- Network
- Hierarchical
- Independent Logical File
- Relational

But Before the Data Models,

- Record Processing
- Set Processing

Type of relationship processing	Characteristic	Example	Typical DBMSs
Record Processing	Records contain "pointers" to their: owners, children, and siblings. You must first obtain the record to then obtain the pointer to then process the relationship.	Get Parent. Obtain address of Child 1 Get Child 1 Get pointer to Child 2 Get Child 2 Get pointer to Parent of Child 2 Get Parent of Child 2	DMS 1100 IDMS IDS-2 Total IMS

Type of relationship processing	Characteristic	Example	Typical DBMSs
Set processing	Records are devoid of pointers that express relationships among record instances.	Use Primary key index to get record from Owner Table Use Primary key index value to get record set from Child Table based on value of Primary Key column Process the records from the record-set one at a time off of a queue (called a cursor)	Adabas Model 204 Ramis Focus DB-2 Oracle Sybase MS/SQL

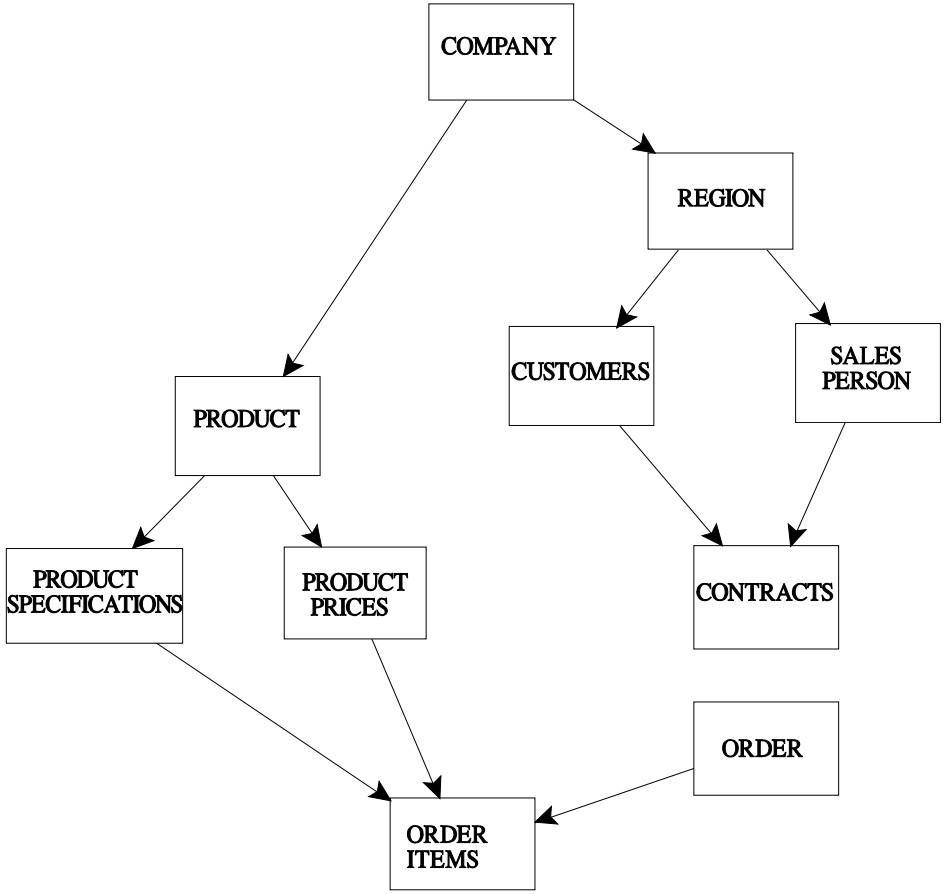
Network Data Model

What makes this a network? Simple:

- Order Items has >1 parents. Three to be exact.
- Contracts has >1 parents. Two to be exact.

Key Stats:

- 1 Schema
- 10 tables
- 11 formally defined relationships



1 to Many

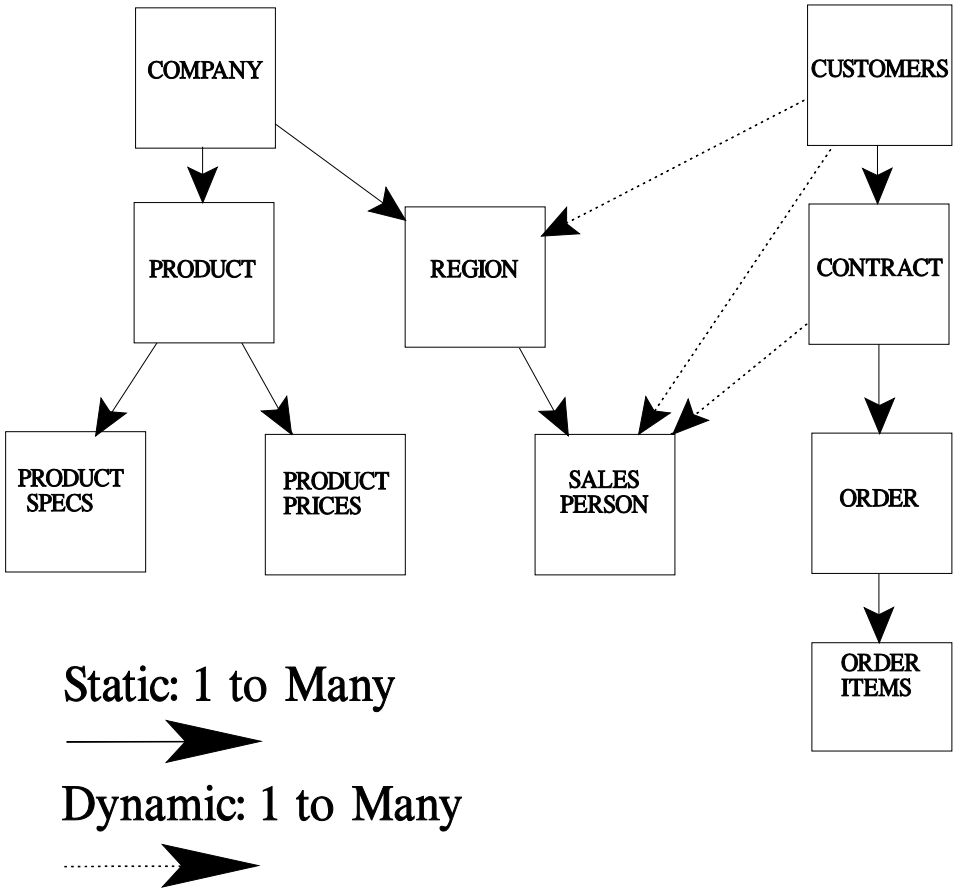
Hierarchical

What makes this Hierarchy? Simple:

- All tables have 1 to many relationships.

Key Stats:

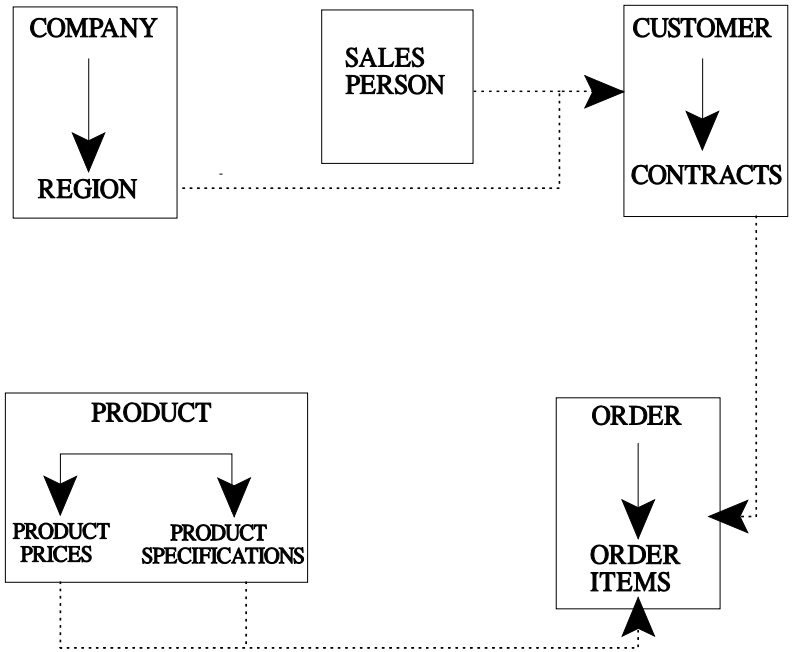
- 2 Schemas
- 6 tables in 1 schema, and 4 tables in the second schema
- 5 formally defined relationships in 1, and 3 formally define relationships in another. Requirement for inter-database access for "dotted" relationships



Independent Logical File

What makes this Independent Logical File?
Simple:

- All "files" have single or hierarchically organized sets of tables.



Key Stats:

- 5 "Files"
- 2 tables in three, 1 table in one, and 3 tables in one.
- 1 formally defined relationships in three, 2 formally defined relationship in on3, and five "value-based" relationships to inter-connect the five "Files."

Static: 1 to Many

Dynamic: 1 to Many

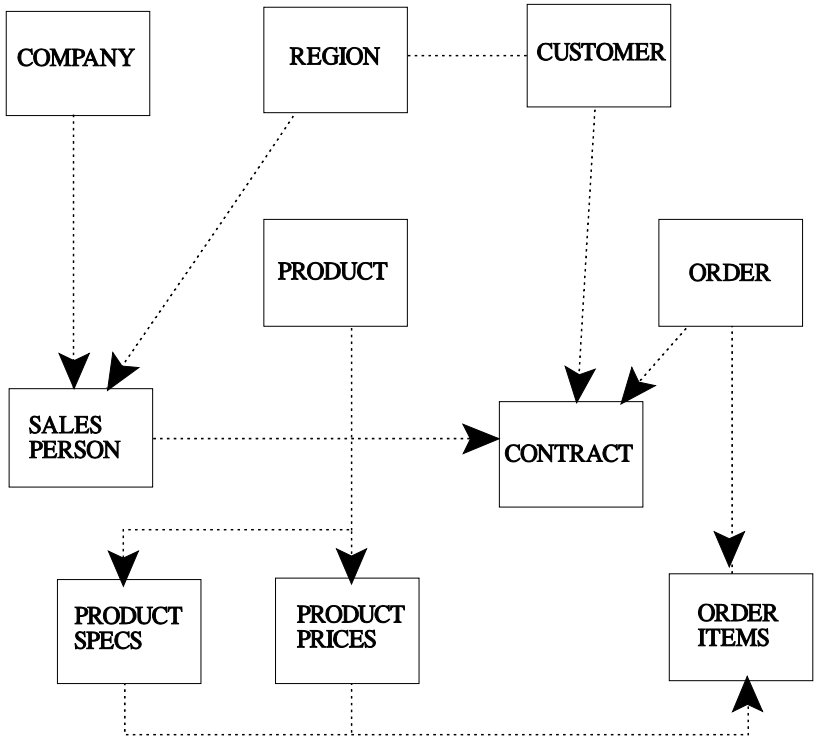
Relational Data Model

What makes this Relational? Simple:

- All tables consist of single-valued columns.

Key Stats:

- 10 tables
- All relationships are "value based;" defined through primary and foreign keys.



Static: 1 to Many

Dynamic: 1 to Many

What's a DBMS Data Model

Record Organization

- Type types of data (e.g., character, integer, decimal)
- Data Structures for each named component

Relationships

- Static: "members" known to "owners" at load or update type
- Dynamic: "members" and "owners" know of each other only at retrieval time
- Referential integrity and referential actions: a DBMS facility for both static and dynamic relationships

Operations

- Record actions (add, delete, or modify)
- Relationship actions on Owner modify and delete, and Member insert and modify

Fundamental Data Model Alternatives

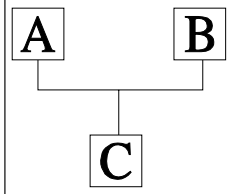
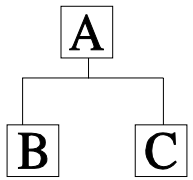
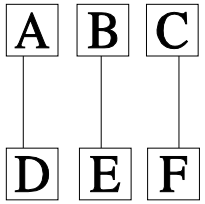
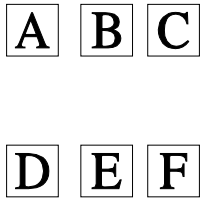
Data Model Type	Data Definition Language Components	Data Manipulation Language Components
Static	Record Structures and Relationships	Operations
Dynamic	Record Structures	Relationships and Operations

Data Model Components	{ Record Organization } + { Inter-record Relationships } + { Operations }
Data Model Languages	{Data Definition Language } + { Data Manipulation Language }
Dynamic Data Model	{ DDL { RO} } + { DML { REL + OPS } }
Static Data Model	{ DDL {RO + REL} } + { DML { OPS } }

WHERE:

DDL: Data Definition Model
DML: Data Manipulation Language
RO: Record Organization
REL: Relationships
OPS: Operations

Data Model Alternatives

Characteristics		Data Model: Network	Hierarchic	Independant Logical File	Relational
Record Organization		SV, MV, MD, G, RG	SV segments	SV, MV, RG	SV
Relationship (REL)					
Operations (OPS)	Record	A, D, F, M	A, D, F, M	A, D, F, M	A, D, F, M, P
	Relation-ships	C, D, GO, GM, GN	GO, GM, GN	J, INT DIV, UN DIF	J, INT, DIV, UN PR, DIV
DDL		REL, RO	REL, RO	RO	RO
DML		OPS	OPS	REL & OPS	REL & OPS

DBMSs and Data Models

Interrecord Relationship Mechanism				
Data Model	**Static**		**Dynamic**	
	Network	**Hierarchy**	**Independent Logical File**	**Relational**
Typical systems	Supra IDMS/R IDS DMS-2 DMS-1100 VAX/DBMS	System 2000 IMS	Inquire Adabas GIM family Nomad Focus Ramis CA/Datacom Model 204	DB/2 CA/ingress Oracle Sybase Informix SQL/server

6.0 DBMS Standards

- 1986: ANSI ratified two database standards
  - NDL (Network [data model] Data Language)
  - SQL (formerly Structured Query Language [relational data model])

- NDL

  - NDL was derived from the CODASYL [network] data model
  - Implemented by Unisys, Honeywell, Cullinet, Digital Equipment Corporation
  - The CODASYL 1978 specification used as base document
  - NDL relationships in NDL are not required to be value-based.
  - NDL relationships are defined in the data definition language
  - NDL can explicitly declare almost all the types of relationships.
  - NDL has referential integrity to control insertion and retention.

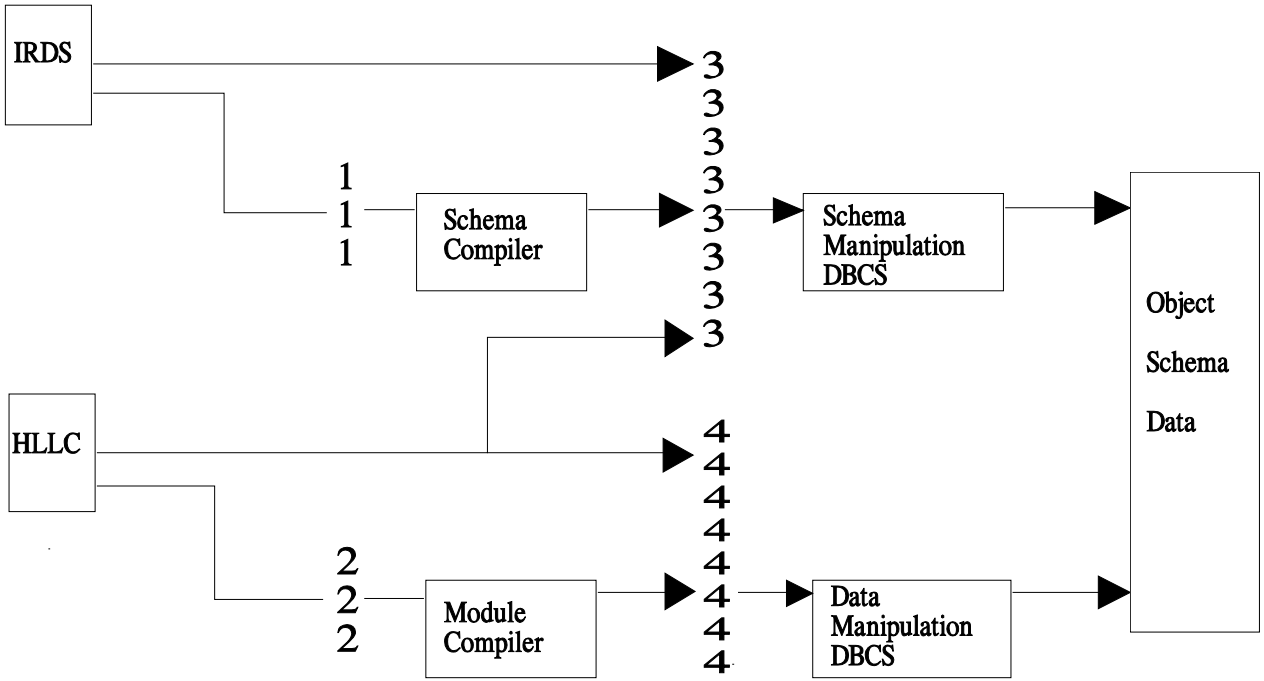
- SQL
  - ♦ SQL, founded on IBM's Sequel
  - ♦ Implemented on almost every brand of hardware
  - ♦ Implemented on all three hardware tiers: micro-computers, mini-computers, and mainframes.
  - ♦ SQL relationships must be value-based
  - ♦ SQL relationships occur in programming logic.
  - ♦ Row orderings are requested by application program and performed by the DBMS
  - ♦ SQL has referential integrity to control insertion and retention.
  - ♦ SQL and NDL are complementary.

- Any DBMS vendor could implement NDL, SQL, or both
- Both ANSI standards were created to be independent of the method of physical implementation.

7.0 ANSI/NDL

Data Model		
Characteristics		**Network Data Model**
Record Organization		Single valued, multi-valued, groups and repeating groups.
Relationships		Owner & 1 Member Owner & >1 Member No Owner & 1 Member No Owner & >1 Member Recursive One to One
Operations	Record	Add, Delete, Find, Modify
	Relationships	Connect, Disconnect, Get Owner, Get Member, Get Next
Data Definition Language (DDL)		Record Organizations and Relationships
Data Manipulation Language (DML)		Operations

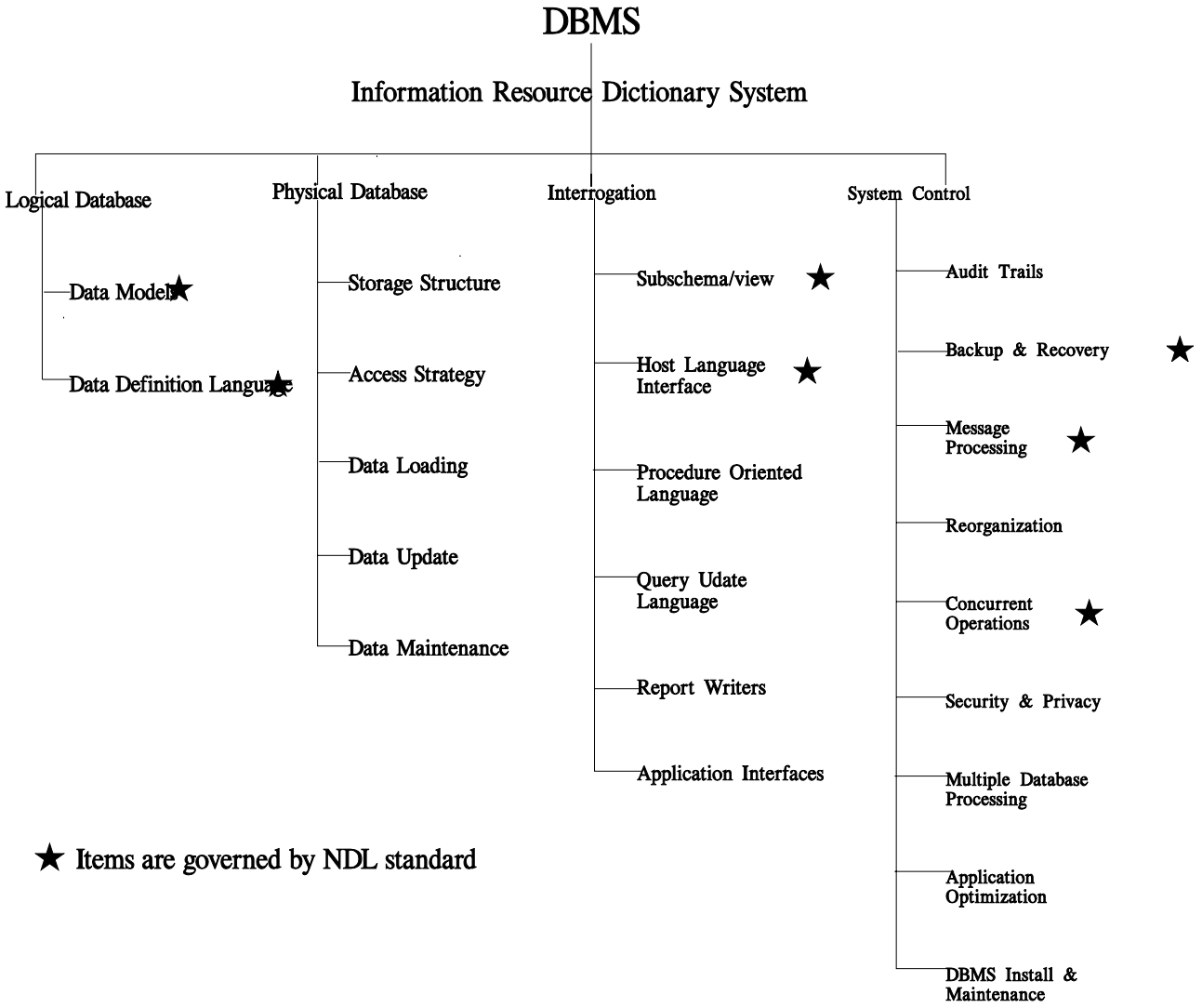
NDL Architecture



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

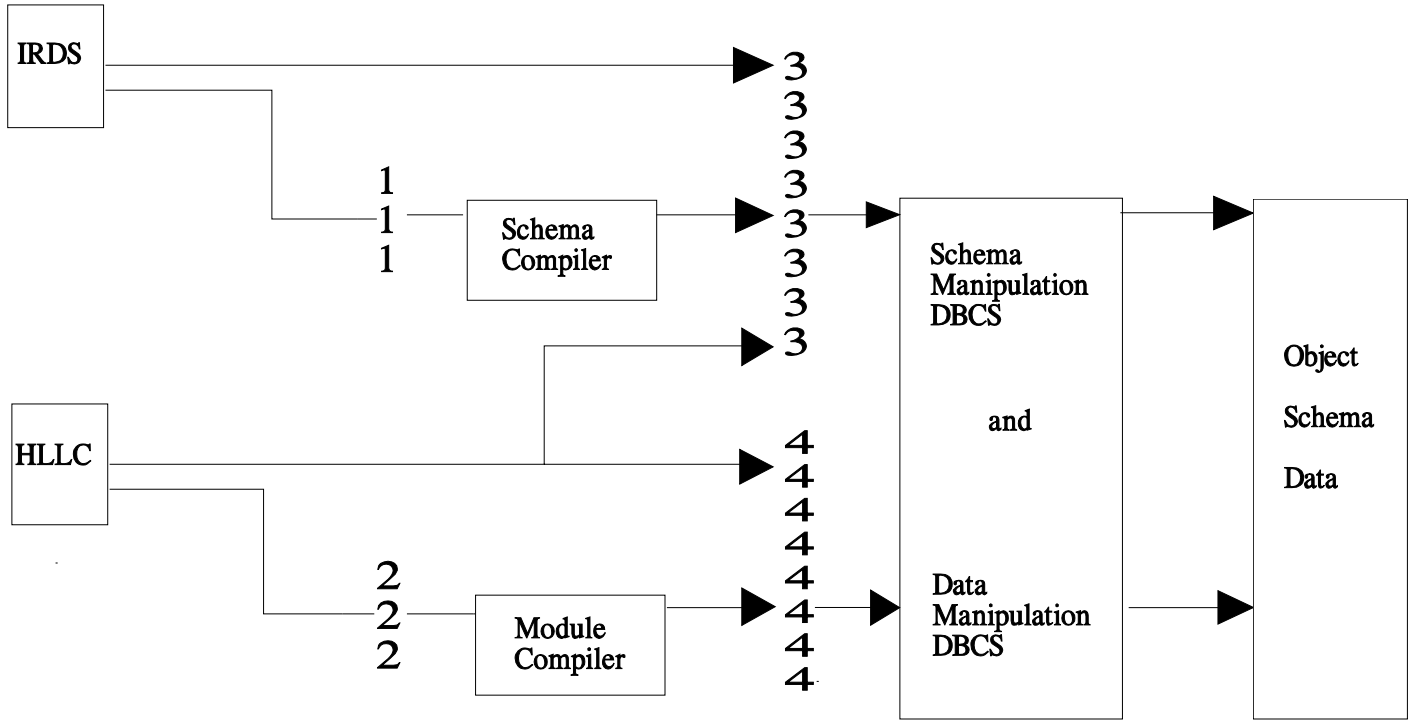
Standardized NDL DBMS Components



8.0 ANSI/SQL (1986, 1989, & 1992)

Data Model		
Characteristics		**SQL Data Model (1986, 1989, 1992)**
Record Organization		Single valued
Relationships		Owner & multiple member (via referential integrity) One to One (via referential integrity)
Operations	Record	Add, Delete, Find, Modify, Project
	Relationships	Join, Intersection, Division, Union, Project, Divide, Product
Data Definition Language (DDL)		Record Organizations
Data Manipulation Language (DML)		Relationships and Operations

SQL Architecture

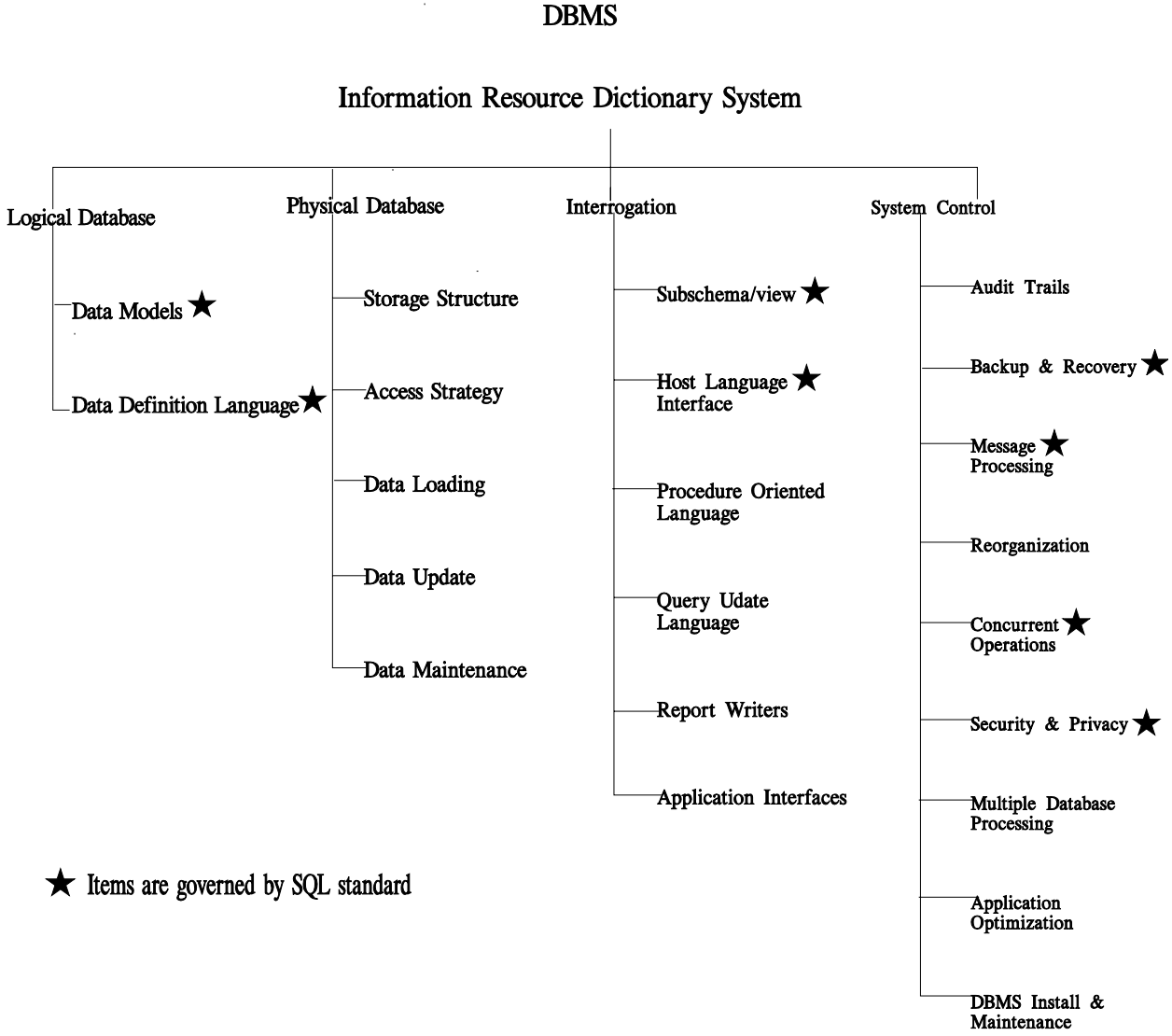


Legend:

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

Standardized SQL DBMS Components



9.0 ANSI/SQL (1999)

Data Model		
Characteristics		**SQL Data Model (1999, 2003)**
Record Organization		Single valued, multi-valued, groups and repeating groups.
Relationships		Owner & one member Owner & multiple member Singular & one member Singular & multiple member Recursive Many-to-many One to One Inferential
Operations	Record	Add, Delete, Find, Modify
	Relationships	Connect, Disconnect, Get Owner, Get Member, Get Next
Data Definition Language (DDL)		Record Organizations and Relationships
Data Manipulation Language (DML)		Operations

9.1 Evolution of the SQL Language

SQL/1986	SQL/1989	SQL/1992	
Basic Tables	SQL/1986 plus	SQL/1989 plus	
Some integrity constraints Language bindings to COBOL, FORTRAN, C, etc.	Partial Referential Integrity	Assertions Bit data type CASE Character and National Character Sets Connection Management DATETIME Domains Dynamic SQL Enhanced constraints Full Referential Integrity Get Diagnostics Grouped operations	Information Schema Multiple module support Natural joins (inner & outer) Row & Table constraints Schema manipulation Subqueries in check clauses Table constraints Temporary tables Transaction Management Union and intersect

9.2 SQL:1999 Data Model

- Data Structures
- Relationships
- Operations

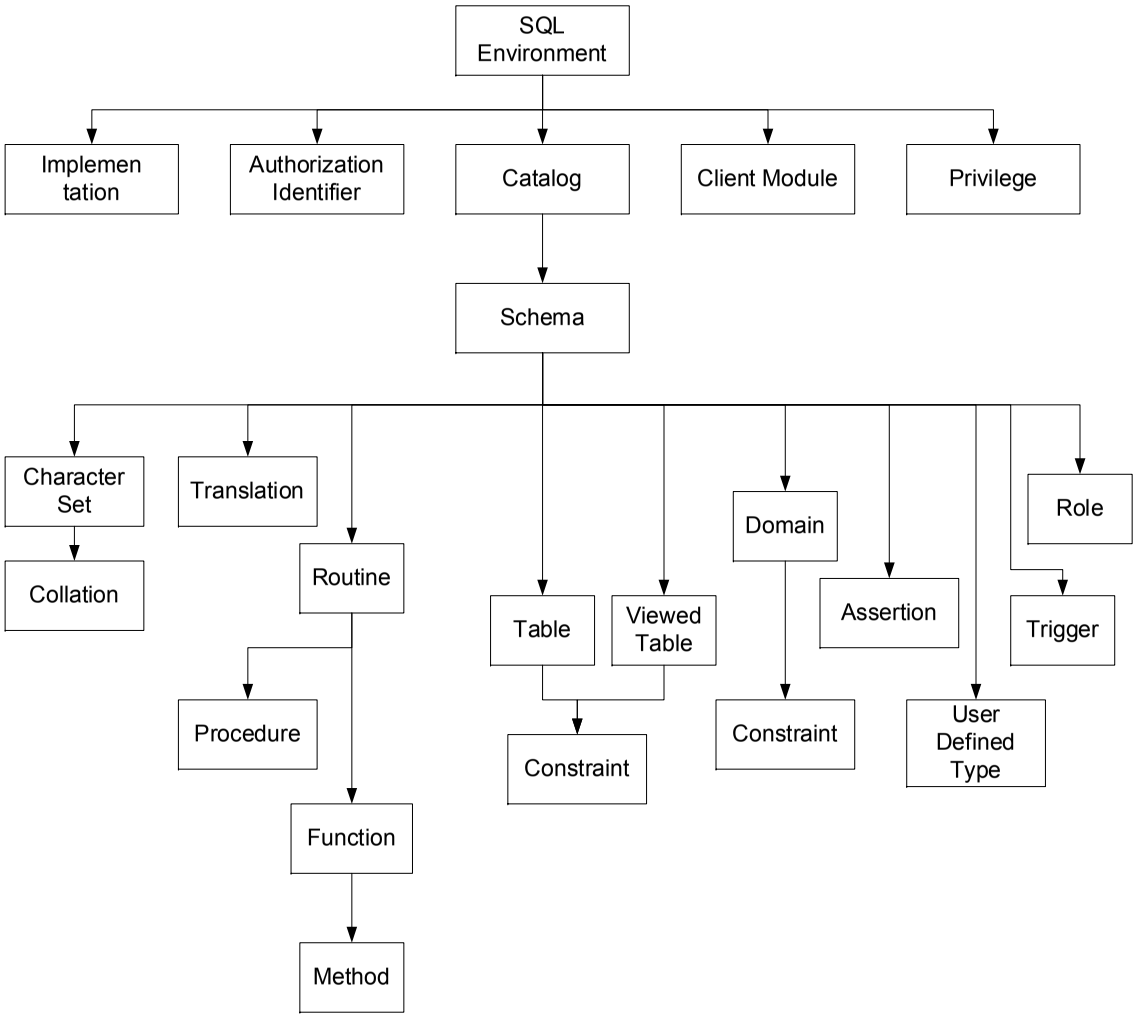
9.2.1 Data Structures

Column Data Type	Definition	SQL:1999
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"	✔-Note 1
Groups	Each component has subcomponents to represent single-set of values such as Address with Street-1, Street-2, City, State, Zip	✔-Note 2
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.	✔-Note 3
Nested Repeating Groups	Employee (Dependents (Hobbies))	✔-Note 4

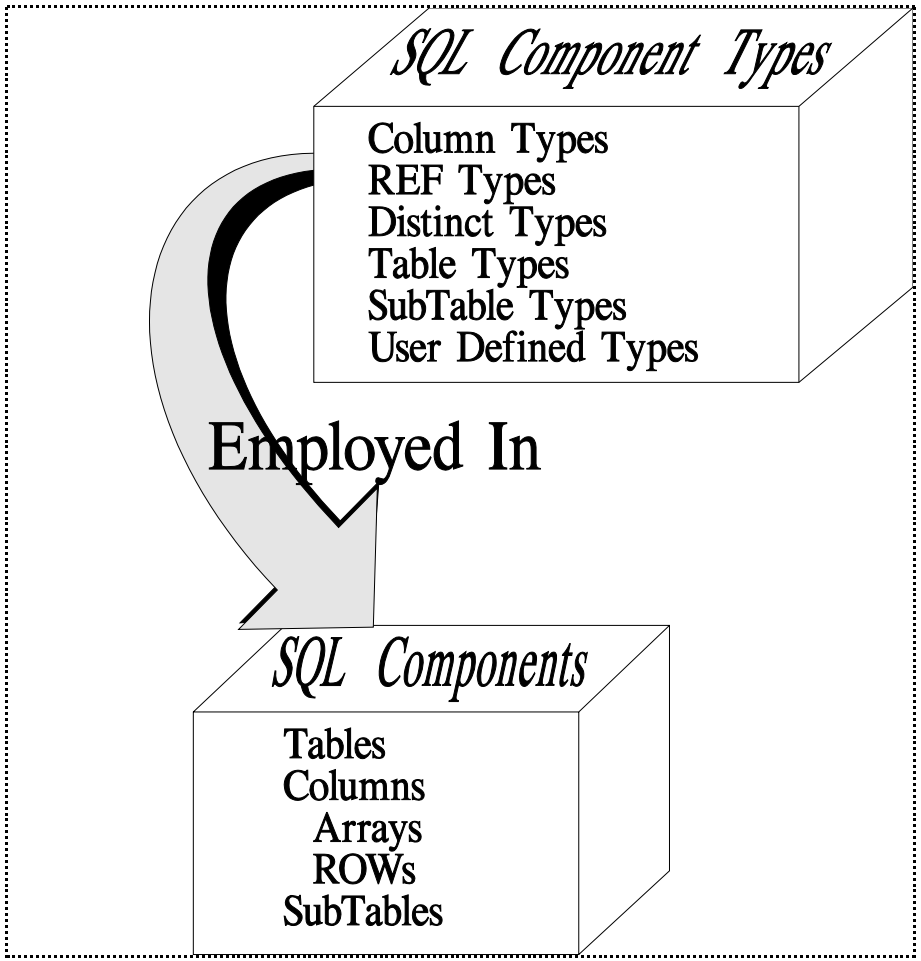
Notes:

1 Arrays as a data type for a column
2 ROW data structure of a column
3 ROW data structure for a column wherein each Row structure field has the data type, ARRAY
4 ROW data structure for a column with contained ARRAYs where each ARRAY column is a ROW data structure with contained ARRAYs where each ARRAY column , etc....

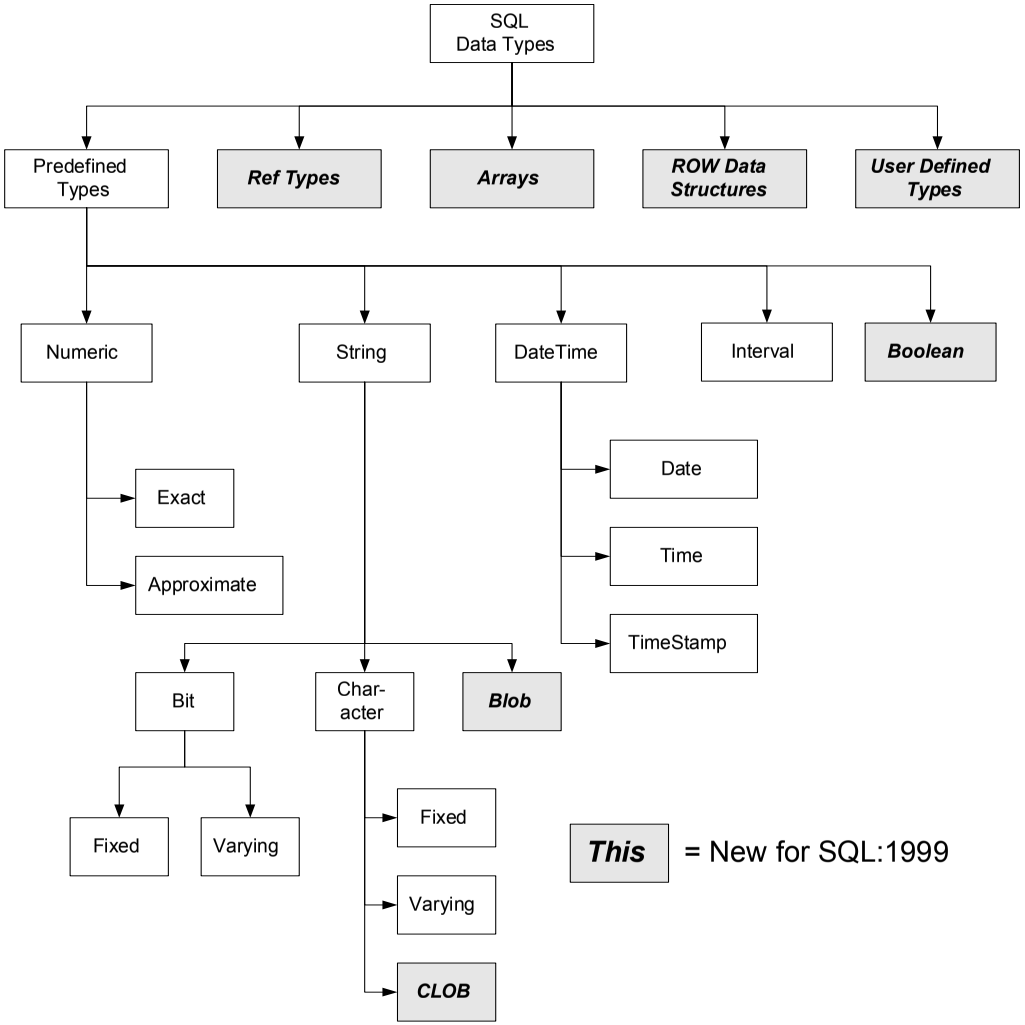
SQL:1999 Objects Under Management



Fundamental Change in SQL Architecture



SQL:1999 Data Types (really data structures)



SQL Data Type: String (BLOB, or Character (CLOB))

- WAR_AND_PEACE CLOB(25M) CHARACTER SET CYRILLIC
- EMPLOYEE_PHOTO BLOB(50K)

Notes:

- Normal character string literals and hex string literals apply
- SUBSTRING, TRIM, ||, etc., all apply
- Comparison only for = and <>
- No GROUP BY or ORDER BY
- No DISTINCT
- No PRIMARY KEY or FOREIGN KEY

SQL Data Type: Boolean

- POLITICIANS_LIE BOOLEAN
- Boolean value expressions are, in effect, predicates (and vice versa)
- Boolean literals: TRUE, FALSE, UNKNOWN
- COL1 AND (COL2 OR NOT COL3) IS NOT FALSE

SQL Data Type: Ref Types

References to instances of structured types that can be used wherever other types are used

Representation

- User generated (REF USING <predefined type>)
- System generated (REF IS SYSTEM GENERATED)
- Derived from a list of attributes (REF (<attribute definition>s)

<attribute definition> ::= <attribute name>
<data type>
[<reference scope check>]
[<attribute default>]
[<collate clause>]
<attribute default> ::= <default clause>

Example

```
CREATE SCHEMA s1

CREATE TYPE employee AS
        (lastname           CHARACTER VARYING (30),
        firstname           CHARACTER VARYING (30),
        manager             REF (employee) SCOPE EVERY EXISTING TABLE ) ;

CREATE TABLE employees OF employee
( empref REF (employee) VALUES ARE SYSTEM GENERATED ) ;

CREATE TABLE project_teams
        ( team_name         CHARACTER VARYING (30),
        team_leader REF (employee) SCOPE (employees),
        team                REF (employee) SCOPE (employees) ARRAY [10],
        reports             ROW ( frequency CHARACTER VARYING (30),
        report_to           REF (employee) SCOPE (employees)
```

SQL Data Type: Arrays

```
CREATE TABLE reports
        (id INTEGER,
        authors VARCHAR(15) ARRAY[20],
        title VARCHAR(100),
        abstract FullText)
```

Appropriate DML operations

```
INSERT INTO reports (id, authors, title)

VALUES (10, ARRAY ['Date', 'Darwen'], 'A Guide to the SQL Standard') Standard')
```

Access to array columns

- By ordinal position
- Declarative (i.e. query) facility
- Implicitly transforms array into table
- Selection by column content and/or position
- Unnesting

Examples:

SELECT id, authors[1] AS name FROM reports

SELECT r.id, a.name FROM reports AS r, UNNEST (r.authors) AS a (name)

SQL Data Type: ROW Data Structures

```
CREATE TABLE employees
        (name CHAR (40),
                address ROW ( street CHAR (30),
                                city CHAR (20),
                                zip ROW( original CHAR (5),
                                        plus4 CHAR (4)));


INSERT INTO employees
        VALUES ('John Doe', ('2225 Coral Drive', 'San Jose', ('95124', '2347')));
```

SQL Data Type: User Defined Types

- User-defined data types
  - User-defined, named types representing entities.
  - For example, employee, project, money, polygon, image, text, language, format, ...

- User-defined methods and functions (operators)
  - User-defined operation representing the behavior of entities in the application domain.
  - For example, hire, appraisal, convert, area, length, contains, ranking, ...

- Definition:
  - User-defined data type
    - Name
    - Representation
    - Relationship to other types
  - User-defined method (and function)
    - Name
    - Signature (i.e., parameter list)
    - Result
    - Implementation

User Defined Data Types

SQL Data Type: Classes of User Defined Types

- Distinct Types used within
  - Columns, variables, parameters types
- Structured Types
  - Columns, variables, parameters types
  - Table Types

SQL Data Type: UDT--Distinct Types

```
CREATE TABLE RoomProperties
        ( RoomID CHAR(10),
        RoomLength              INTEGER,
        RoomWidth               INTEGER,
        RoomArea                INTEGER,
        RoomPerimeter           INTEGER );


UPDATE RoomProperties
SET RoomArea = RoomLength;
```

No error results because the column names are all traditional data types that contain no enforced semantics other than fundamental nature of the value....

But with Distinct Types,

First Create the Distinct Types for the Columns

```
CREATE TYPE plan.roomtype AS CHAR(10) FINAL;

CREATE TYPE plan.meters AS INTEGER FINAL;

CREATE TYPE plan.squaremeters AS INTEGER FINAL;
```

Then Create the Table

```
CREATE TABLE RoomTable (
    RoomID plan.roomtype,          plan.roomtype
    RoomLength                     plan.meters,
    RoomWidth                      plan.meters,
    RoomPerimeter                  plan.meters,
    RoomArea                       plan.squaremeters)
```

```
UPDATE RoomTable
        SET RoomArea = RoomLength;
```

You get an ERROR because the data type of RoomArea <> data type of Room

```
UPDATE RoomTable
SET RoomLength =RoomWidth;
```

NO ERROR RESULTS because the data types are the same, plan.meters,

Using User Defined Types for Columns and Tables

First, define the user defined types

```
CREATE TYPE address AS
        (street CHAR (30),
        city CHAR (20),
        state CHAR (2),
        zip INTEGER) NOT FINAL

CREATE TYPE bitmap AS BLOB FINAL

CREATE TYPE real_estate AS
        (owner              REF (person),
        price               money,      !a TYPE definition
        rooms               INTEGER,
        size                DECIMAL(8,2),
        location            address,    !a TYPE definition
        text_description    text,       !a TYPE definition
        view_image bitmap,
        document            doc         !a TYPE definition
        ) NOT FINAL
```

Second, deploy the user defined types....

- Attributes of type become columns of table
- Plus one column to define REF value for the row (object id)
- An entire type can become a table

```
CREATE TABLE properties OF real_estate
    (REF IS oid USER GENERATED)
```

- real-estate is a table type
- address is a column type
- bitmap is a column type

User Defined Methods

- **What are methods?**--SQL-invoked functions "attached" to user-defined types

- How are they different from functions?

  - ♦ Implicit SELF parameter (called subject parameter)
  - ♦ Two-step creation process: signature and body specified separately.
  - ♦ Must be created in the type's schema
  - ♦ Different style of invocation (UDT value.method(...))

Example:

```
CREATE TYPE employee AS
        (name CHAR(40),
        base_salary DECIMAL(9,2),
        bonus DECIMAL(9,2))
        INSTANTIABLE NOT FINAL
        METHOD salary() RETURNS DECIMAL(9,2);

CREATE METHOD salary() FOR employee
        BEGIN
        ....
        END;
```

SQL Data Type: UDT Structured Types, Sub Tables

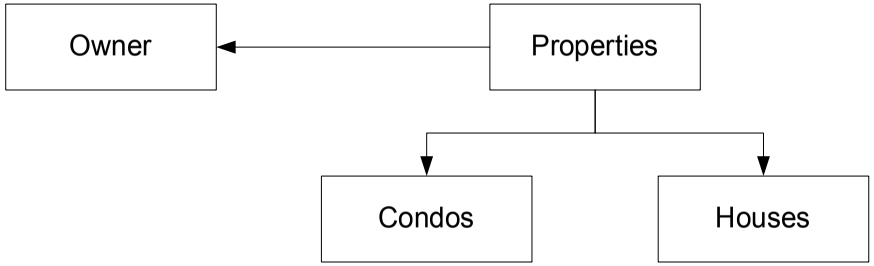
CREATE TYPE person ... NOT FINAL
CREATE TYPE real_estate ... NOT FINAL
CREATE TYPE condo UNDER real_estate ... NOT FINAL
CREATE TYPE house UNDER real_estate ... NOT FINAL

CREATE TABLE people OF person (...)
CREATE TABLE properties OF real_estate
CREATE TABLE condos OF condo **UNDER** properties
CREATE TABLE houses OF house **UNDER** properties

NB: Subtables are MERELY a method of partitioning a BIG table into subtyped subordinate tables. For each SuperTable row there is AT MOST one row from each subtable, but,

Count(Properties) >= Sum (Count(Condos) + Count(Houses))

Example: If there were 100 properties then if there were 75 houses there could only be a maximum of 25 condos...



9.2.2 Relationships

SQL: 1999 Relationship Types

Name	Example	SQL:1999
One-to-many	Employee to dependents	✔-Note 1
Owner-multiple-member	Territory contains salesmen and customers	✔-Note 2
Singular-one-member	Top performing employees	✔-Note 1
Singular-multiple-member	Top performing current, former, part-time, and retired employees	✔-Note 3
Recursive	Organization contains organization	✔-Note 4
Many-to-many	Automobiles and owners	✔-Note 5
One-to-one	Table and its primary key	✔-Note 6
Inferential	Many houses each with a location, and then buyer with desired location	✔-Note 7

Notes:

1 Traditional relational data model (SQL:1986, 1989, & 1992)

2 Implemented as a ROW(TerritoryId, Salesman REF (SalesmanId),

3 Developed using Subtables where Employees are partitioned off into their common columns (employee) and their unique columns (current, former, part-time, and retired)

4 Recursion operations built into the language (WITH RECURSION...)

5 Cross joins from within columns of ARRAYS contained as data types of column in different tables

6 Effectively as tables and subtables. Most directly with UNIQUE Fkey

7 A single valued non-primary key Location within House and same for Buyer

9.2.3 Operations

SQL: 1999 Operations

- **Record Operations –** traditional insert, delete, and modify of rows and columns within rows
- **Relationship Operations** – traditional operations that affect the relationships between rows of the same or different tables

Record Operations			
Operation	**Static**	**Dynamic**	**SQL:1999**
Find	SELECT According to STORED Order	SELECT and PUT into DML Specified Order	✔
Get	Obtain Record From Find	Ditto	✔
Add	Install a New Row Into Database	Ditto	✔
Delete	Remove an Existing Row From Database	Ditto	✔
Modify	Change Some Data Column Values in Existing Row	Ditto	✔

Relationship Operations			
Operation	**Static**	**Dynamic**	**SQL:1999**
Connect	Add to a Named RELATIONSHIP in Specific Order	N/A	✔-Note 1
Disconnect	Delete From RELATIONSHIP	N/A	✔-Note 2
Get Owner	Obtains The Parent of the Row That is Current	N/A	No
Get Member	Obtains the First Child of the Owner For the Named Relationship	N/A	✔-Note 3
Get next	Obtains the Next Row Within The Named Relationship	N/A	✔-Note 4
Intersect	N/A	Find and Keep Only the Common	✔
Difference	N/A	Find and Keep Only the Not Common	✔
Join	N/A	"Append" Relations to Each Other	✔
Divide	N/A	Subset	✔
Product	N/A	Cross-Product	✔
Union	N/A	Merge and Drop Duplicates	✔

SQL:1999 Relationship Operations(cont.)

Notes:

1	Value an Column in an ARRAY. Data type of Array is REF type pointing elsewhere.
2	Set value of column in array to null
3	Get the first column of an array. Data type of Array is REF type pointing elsewhere
4	Get next column in array. Data type of Array is REF type pointing elsewhere

9.3 SQL:1999 Foundation Components

- Tables that have been enhanced to support new built-in data types (boolean, extensions to character sets, translations, and collations)
- BLOB and CLOB data types
- User Defined Types (behavior, an encapsulated internal structure, and access characteristics of public, protected, or private)
- Array
- ROW (table person (SSN, name ROW(first, middle, last), address ROW(street, city, state, zip ROW(four, five)))))
- User Defined Functions
- Predicate extensions (for all, for some, similar to, cursor extensions, null values, assertions, view updatability, joins)

- Triggers
  - Different triggering events, update, delete, and insert
  - Optional condition
  - Activation time: before and after
  - Multiple statement action
  - Several triggers per table
  - User-defined ordering
  - Condition and multiple statement action per each row or per statement

- Roles (enhancements to security), & Save-points

- Recursion

- Information Schema

9.4 Triggers

- Triggers provide automatic execution of a set of SQL statements when a specific data change operation (UPDATE, INSERT, DELETE) occurs

- Bring application logic into the database

- Transforms a passive to active DBMS

  - Benefits of triggers include:
  - Code reuse
  - Faster application development
  - Easier Maintenance
  - Guaranteed enforcement of business rules

- Common Use of Triggers

  - Enforce "transitional" business rules
  - Validate input data
  - Generate new values for inserted / updated rows
  - Cross-reference other tables
  - Maintain audit, summary or mirror data in other tables
  - Support "alerts"
  - E-mail notification
  - Initiate external actions

Trigger Components

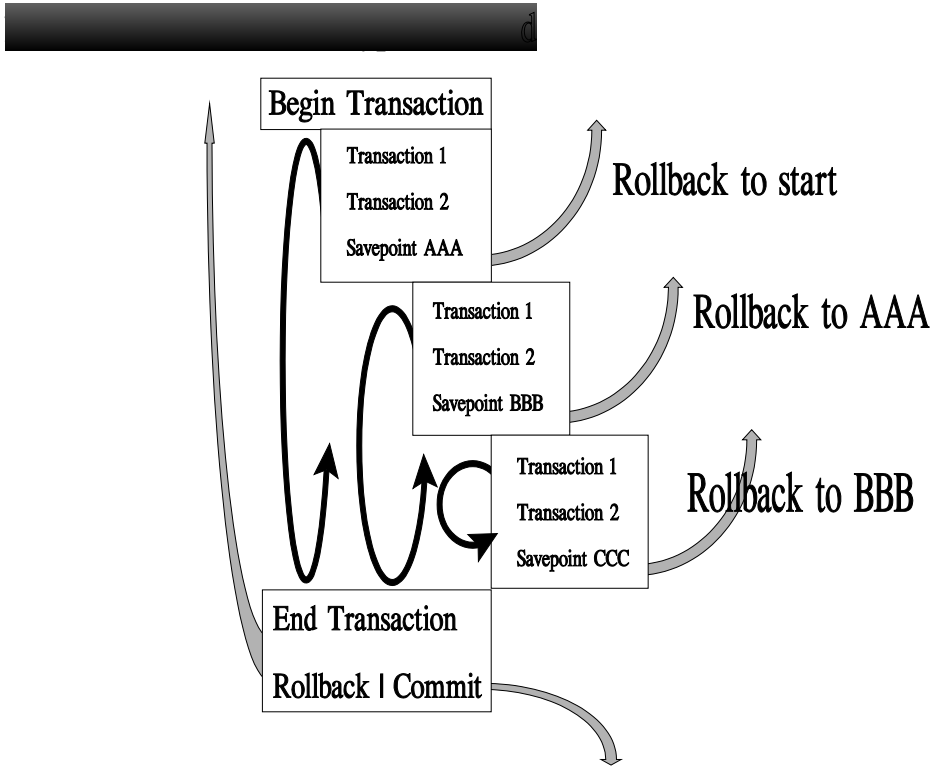
Triggering Table	Table on which the trigger is defined
Triggering Event	An SQL Data Change Operation (INSERT, DELETE, UPDATE)
UPDATE	instigating “target” can be a column
ON	the triggering table
Trigger Activation Time	BEFORE or AFTER
Trigger Granularity	FOR EACH ROW or FOR EACH STATEMENT

Example:

```
CREATE TRIGGER Payroll
AFTER UPDATE OF salary ON Paytable
FOR EACH STATEMENT
INSERT INTO PAYROLL_LOG ...;
```

9.5 Savepoints

- Behaves like single-nested subtransactions
- ROLLBACK TO SAVEPOINT allows "partial rollback" of transaction
- RELEASE SAVEPOINT acts like tentative commit of part of transaction



9.6 Roles Security Enhancement

New in SQL99; benefits:

- Simplifies definition of complex sets of privileges
- Roles are created

Note: definition of users implementation-defined

Example:

- CREATE ROLE Auditor

- CREATE ROLE AuditorGeneral

Roles may be assigned to users & roles

- GRANT Auditor TO AuditorGeneral WITH ADMIN OPTION
- GRANTED BY CURRENT ROLE
- GRANT Auditor TO Smith

Controllable whether to grant as user or role

9.7 Recursion

What is recursive SQL?

- self-referencing table expressions
- self-referencing views

Why use recursion?

- Bill of material processing
- Network traversals (e.g. airline routing)

Functionality and performance benefits

- Challenge: integration into SQL
- Syntax in analogy to Datalog
- Advanced recursion (e.g. mutual recursion)
- Integration with different forms of joins
- Allowing for duplicates
- Graph traversal in "depth first" or "breadth first"
- Cycle control

9.8 Information Schema

A set of views describing the metadata contained in a catalog

- Exist in the INFORMATION_SCHEMA schema
- Are fully defined (column names, data types, and semantics)
- May be queried by users
- Are read-only
- Reflect database objects that the user owns or for which the user has some privilege
  - TABLES
  - COLUMNS
  - VIEWS
  - DOMAINS
  - etc

9.9 Call Level Interface

The CLI specification contains more than 50 different call specifications that address:

- Connection control to SQL servers
- Allocate and de-allocate resource
- Execute SQL statements
- Obtain diagnostic information
- Control transaction termination
- Obtain information about implementation

It also contains Resource Management Handle routines for:

- Environment
- Connection
- Statement
- Context

The CLI also contains a Descriptor Area that accommodates:

- Application parameter
- Application row
- Implementation parameter
- Implementation row

9.10 SQL Multi-Media Components

- Framework (Part 1)
- Full Text (Part 2)
- Spatial (Part 3)
- Still Image (Part 5)

Rationale for SQL/MM

Enabling functionality of SQL3:

- Definition of user-defined, application specific data types
- Implementation of user-defined functions to support application specific operations on the data types
- Storage of large objects (BLOBs and CLOBs)
- Powerful trigger and constraint mechanisms to maintain the integrity and semantics of the new data types
- Storage and execution of user-defined stored procedures in the server

This enables ...

- Development of application specific collections of user-defined types, user-defined functions, triggers, constraints, and stored procedures (i.e. libraries) "tight" to the DBMS engine

9.10.1 SQL Multi-Media: Full Text

Why Full-Text standard library?

- Built-in search facilities (LIKE, SIMILAR) not powerful enough (text viewed as string of characters).
- Need higher level notion of text

Structural units in Full-Text:

- Words
- Sentences
- Paragraphs

Operations in Full-Text:

- Boolean Search
- Ranking
- Conceptual Search

SQL/MM Full Text Examples

Searched String: *Every text value is associated with a specific language.*

Single word search:

SELECT * FROM myDocs
WHERE 1 = CONTAINS(TextBody, '"specific"')
Result: Every text value is associated with a specific language.

Phrase search:

SELECT * FROM myDocs
WHERE 1 = CONTAINS(TextBody, '"specific language"')
Result: Every text value is associated with a specific language.

Context search:

SELECT * FROM myDocs WHERE 1 = CONTAINS(TextBody, '"text" IN SAME SENTENCE AS "language"')
Result: Every text value is associated with a specific language.

Stopwords:

SELECT * FROM myDocs WHERE 1 = CONTAINS(TextBody, '"value was associated"')
Result: Every text value is associated with a specific language.

SQL/MM Full Text Examples (cont.)

Searched String: *Every text value is associated with a specific language.*

Linguistic search:

SELECT * FROM myDocs WHERE 1 = CONTAINS(TextBody, 'STEMMED FORM OF "values are associated"')

Result: Every text value is associated with a specific language.

9.10.2 SQL Multi-Media: Spatial

Goals:

- Support for "flat world" (2-d) geometric objects and operations
- Coverage of important application areas
- Simple features

Motivation

- Break ground for standard type library
- Promote efficient access methods on relational platforms

9.10.3 SQL Multi-Media: Still Image

- Enable screening of large imagebases
- Support for proven set of image features
- Type structure adaptable to evolving image processing technology

Example:

Find all possibly infringed logos by scoring them against a new logo.

```
SELECT * FROM RegLogos

WHERE 1.2 < SI_findTexture(newLogo).SI_Score(Logo)
```

9.11 SQL Programming Language

Procedural Extensions

- Improve performance in centralized and client/server environments
- Multiple SQL statements in a single EXEC SQL
- Multi-statement procedures, functions, and methods

Gives great power to DBMS

- Several, new control statements (procedural language extension)
- (begin/end block, assignment, call, case, if, loop, for, singal/resignal,
- variables, exception handling)

SQL-only implementation of complex functions

- Without worrying about security ("firewall")
- Without worrying about performance ("local call")

SQL-only implementation of class libraries

SQL Programming Language (cont.)

Consider a C program with embedded SQL statements:

```
void main
EXEC SQL INSERT INTO employee
VALUES ( ...);
EXEC SQL INSERT INTO department
VALUES ( ...);
}
```

Using PSM-96 procedural extensions, the same program can be written as:

```
void main
{
EXEC SQL
BEGIN
INSERT INTO employee VALUES ( ...);
INSERT INTO department VALUES ( ...);
END;
}
```

SQL Programming Language (cont.)

If we create a SQL procedure first:

```
CREATE PROCEDURE proc1 ()
        {
        BEGIN
        INSERT INTO employee VALUES ( ...);
        INSERT INTO department VALUES ( ...);
        END;
        }
```

Then the embedded program can be written as

```
        void main
        {
        EXEC SQL CALL proc1();
        }
```

SQL Programming Language (cont.)

Statement	Effect
Compound statement	BEGIN ... END;
SQL variable declaration	DECLARE var CHAR (6);
If statement	IF subject (var <> 'urgent') THEN ... ELSE ...;
Case statement	CASE subject (var) WHEN 'SQL' THEN ... WHEN ...;
Loop statement	LOOP < SQL statement list> END LOOP;
While statement	WHILE i<100 DO .... END WHILE;
Repeat statement	REPEAT ... UNTIL i <100 ... END REPEAT;
For statement	FOR result AS ... DO ... END FOR;
Leave statement	LEAVE ...;
Return statement	RETURN 'urgent';

Statement	Effect
Call statement	CALL procedure_x (1,3,5);
Assignment statement	SET x = 'abc';
Signal/resignal statement	SIGNAL division_by_zero

SQL Programming Language (cont.)

A compound statement is a group of SQL statements to be executed sequentially.

<compound statement> :: =
[<beginning label> <colon>]
BEGIN [[NOT] ATOMIC]
[<local declaration list>]
[<local cursor declaration list>]
[<local handler declaration list>]
[<SQL statement list>]
END <ending label>

Example:

```
BEGIN
UPDATE accounts SET balance = balance-100 WHERE ...;
INSERT INTO account_history
(SELECT account#, CURRENT_DATE, 'debit', 100 FROM accounts WHERE ...);
END
```

9.12 Transaction, Connection, Session, and Diagnostics Management

Transaction Management

- Start transaction
- Set transaction
- Set constraints mode
- Savepoint
- Release savepoint
- Commit
- Rollback

Connection Management

- Connect
- Set connection
- Disconnect statement

Session Management

- Set session characteristics
- Set session user identifier
- Set role
- Set local time zone

Diagnostic Management

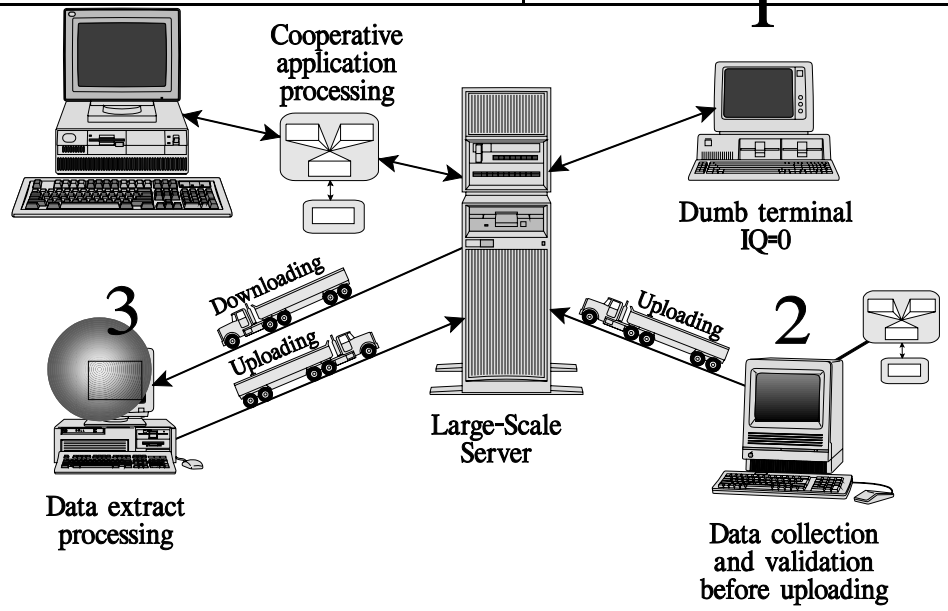
- Get diagnostics statement

9.13 SQL/MED

- Management of External Data
- Seen as way to give SQL access to non-SQL data (e.g., flat files, even sensors)
- Foreign tables, abstract LOBs(?): SQL API
- DataLink: SQL control, native API
- Federated database?
- (Non-final) Committee Draft late 1998

9.14 SQL:1999 Impact on Client Server Paradigm

SQL Standard	Client/Server Alternative Supported
SQL:1986	1
SQL:1989	1
SQL:1992	1,2,3
SQL:1999	1,2,3,4



Four Client/Server Alternatives

10.0 The ANSI/IRDS

The information resource dictionary system (IRDS) is designed to be an organization's repository of metadata. Uses of the IRDS include the following:

- A documentation tool
- A software life cycle and project management tool
- A data column standardization and management system
- An organizational planning tool
- A tool to support database administration, document administration, information resource management and data administration
- A tool for supporting a distributed processing and database environment
- A source and object library management system
- A configuration management facility
- The storage location for NDL & SQL schema and subschemas (views)

The minimum set of objects in the IRDS is specified in the IRDS part 2, Basic Functional Schema. It contains the following eight objects:

- data column
- row
- document
- file
- module
- program
- system
- user

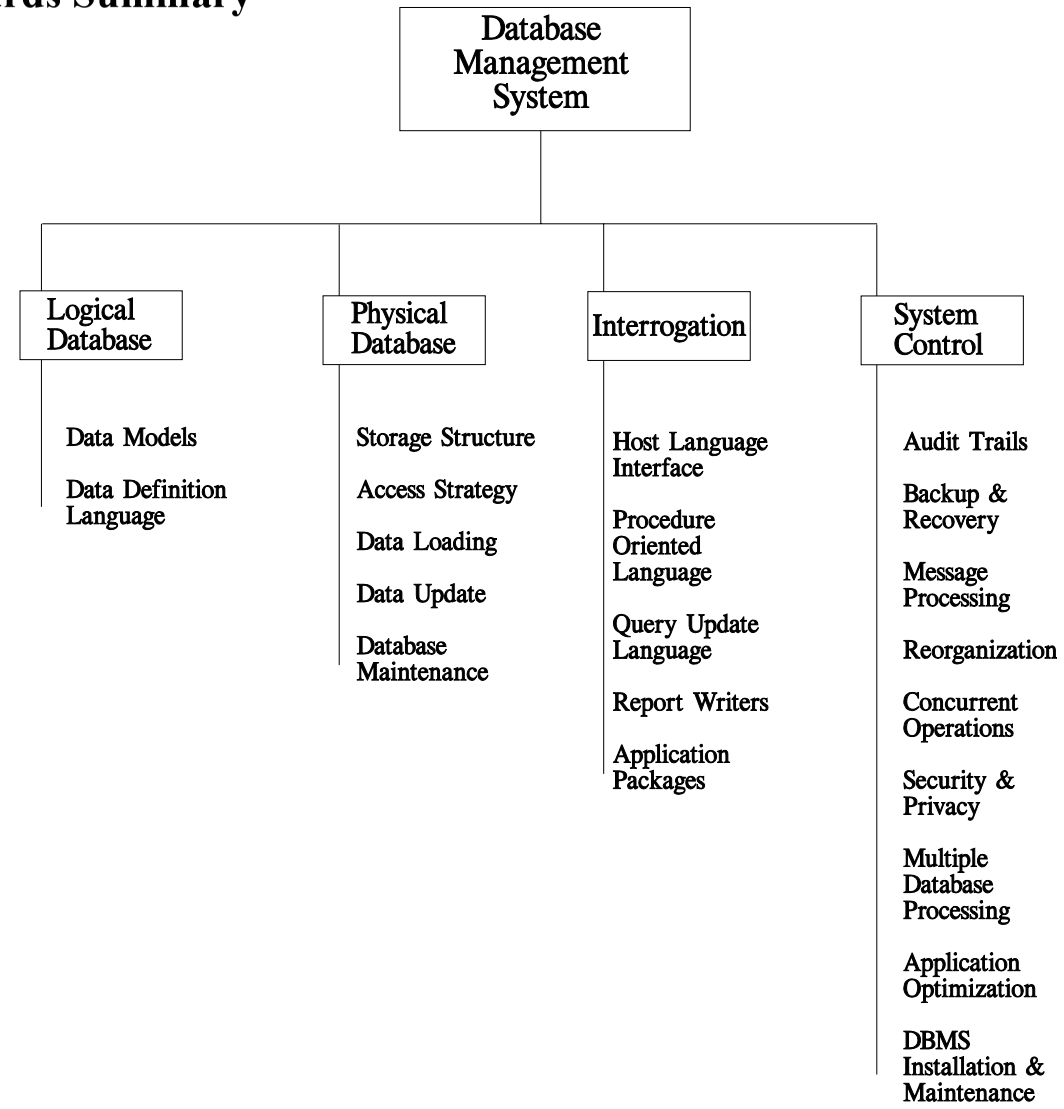
Also included in the IRDS are four types of relationships to interconnect the IRD objects:

- One to many
- One to one
- Many to many
- Recursion

The following are illustrative combinations of objects and allowed relationships:

- Document contains document
- Document contains column
- Document contains row
- Document derived from document
- Document derived from file
- Document derived from row

11.0 Database Standards Summary



The ANSI database standards process started in the mid-1970s and continues today. Its milestones and achievements include:

1975 Creation of the Draft Reference Model (three schema)
1978 Establishment of X3H2 (NDL charter)
1979 Establishment of X3H4 (IRDS)
1981 Extension of X3H2's Charter extended for SQL
1983 Replacement of the ANSI/SPARC 1975 Reference Model
1986 Approval of the ANSI/NDL database standard
1986 Approval of the ANSI/SQL database standard (i.e., SQL/86)
1986 Development of the ISO Reference model (replaces 1983 model)
1988 Approval of the ANSI/IRDS standard
1989 Approval of ANSI/SQL Addendum-1 (i.e., SQL/89)
1988 Development of the IEC/ISO/TC1 Reference model (replaces 1986 model)
1989 Approval of ANSI/SQL (embedded SQL language support)
1992 Completion of ANSI/SQL2 (probably SQL/92)
1999 Completion of ANSI/SQL 1999
2003 Completion of ANSI/SQL 2003

ANSI/X3H2 Event	Year									
	78	**80**	**82**	**84**	**86**	**88**	**90**	**92**	**94**	**96**
Start of X3H2	X									
NDL-DDL Development	X	X	X	X						
NDL-DML Development		X	X	X						
NDL Processing				X	X					
RDL Development			X	X						
Change RDL To SQL				X						
SQL Development				X	X					
SQL Processing				X	X					
SQL Referential Integrity Revision (SQL 1989)				X	X	X				
SQL Embedded Language Standard				X	X	X				
SQL2 Development						X	X			
SQL2 Processing (SQL 1992)								X		
	78	**80**	**82**	**84**	**86**	**88**	**90**	**92**	**94**	**96**

ANSI/X3H2 Event	Year									
	96	**98**	**00**	**02**	**04**	**06**	**08**	**10**	**12**	**14**
SQL1999 Development	X	X	X							
SQL1999 Processing				X						
SQL/MM Development	X	X	X	X	X					
SQL/XML Development				X	X					
SQL/XML Processing					X					
SQL/4 Development (SQL 2007?)				X	X	X				
SQL/4 Processing (SQL 2007?)						X	X			
	96	**98**	**00**	**02**	**04**	**06**	**08**	**10**	**12**	**14**