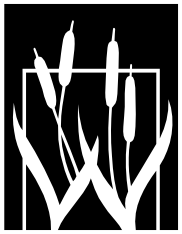


Whitemarsh
Information Systems Corporation

Database Management Systems: Understanding and Applying Database Technology
Chapters 4

Physical Database

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

1.0 Physical Database

- Physical Database Introduction
- Storage Structure
- Access Strategy
- Data Update
- Database Maintenance
- Physical Database Summary

1.1 Physical Database Introduction

- The DBMS Facilities, and Project Activities Required to Map, and Realize the Database.
- Components:
- Storage Structure
  - Access Strategy
  - Data Loading
  - Data Update
  - Database Maintenance

1.2 Objectives

- Describe How the DBMS Really Implements a Database
- Illustrate How the DBMS Makes it Work Together
- Identify the Basic Strategies for Database Loading
- Understand the Implications of Data Update
- Define the Alternatives for Database Backup

2.0 Storage Structure

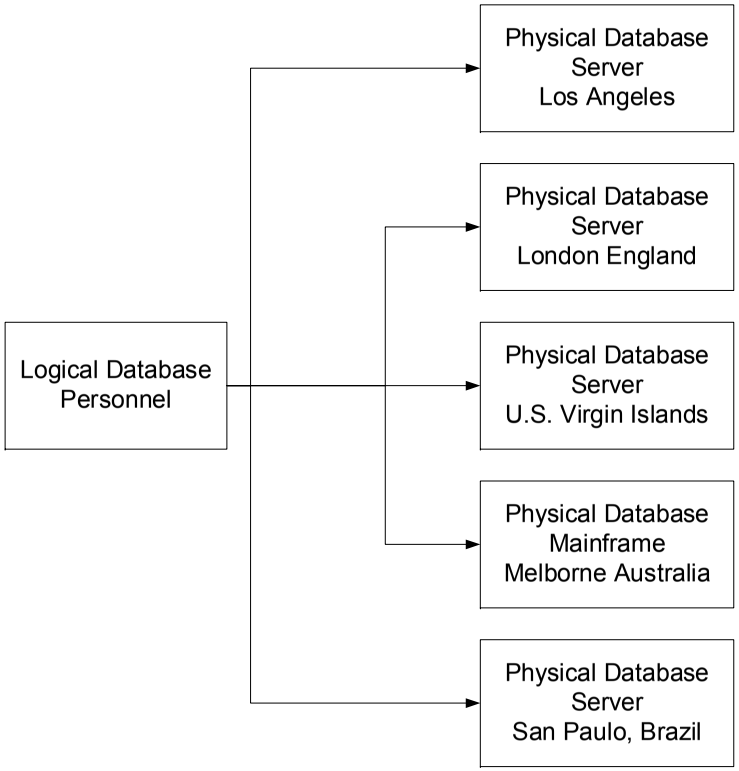
- Introduction
- Components
  - Dictionary
  - Indexes
  - Relationships
  - Data
- Summary

Storage Structure		
Storage Structure Component	**Static**	**Dynamic**
	Often multiple component physical files. Typically dictionary, indexes, relationships, and data	Often single component physical files Typically dictionary, indexes, and data
Access Strategy	Primary key and relationship searching via row processing	High number of indexes. Dynamic matching/ merging of row extracts via column values
Data Loading	Complete logical rows. Requires Careful Planning, Exact Placement, Large Volumes, All or Nothing	Table by table. Load what you have. Incremental building
Data Update	Very careful far reaching effects HLI only. Periodic database reorganization	Casual, add new rows, columns at will. Seldom needs reorganization
Database Management	Usually one or a few databases Global save/restore at high level	Many storage structures. Careful planning. Many commands at low levels

2.1 Storage Structure Introduction

- Definition: Physical Organization of All
- Files Which Comprise a Physical Database.

Dictionary	index	relationships	data



2.2 Dictionary

- Definition: "Intelligence" Part of a Database

- Key Components

  - Database Name
  - Security Passwords, Etc.
  - Field Names
  - Field Characteristics
  - Natural Language Stored Programs
  - Relationship Definitions
  - Available Space Lists
  - Names of All Other Files in Physical Database

- Subschemas

  - Common Part
  - Specific Language Part

Typical Dictionary Organizations

- Sequential -- Search Through Entries
  - ♦ Since size Is Small No Big Deal
  - ♦ Slowest
- Direct -- Organized Through Relative Location Of Field Within Table
  - ♦ Little Faster
- Hashed -- Field Name Used to Compute Direct Address of Field Description
  - ♦ Fastest

Dictionary Summary

- Next to Data, the Most Critical Component of a Database
- Trade-offs
  - Speed (Fastest = Hashed)
  - Space (Smallest = Sequential)
  - Dictionary Reorganization
    - Least = Sequential
    - Most = Hashed

2.3 Indexes

- Definition: Data Value Based Access Paths to Row Contexts
- Types of Indexes
  - Primary -- One in Which the Data Value Is Guaranteed to Be Present In One and Only One Row
  - Secondary -- One in Which the Data Value May Be Present in Multiple Rows

2.3.1 Primary Indexes

- Designated Field with Guaranteed Unique Value
- Often Used to Physically Locate/ Store Rows
- Often Used In:

  - ♦ Secondary Indexes

Social Security Number Within a Secondary Index		
Field	**Value**	**Row Containing "Male"**
SEX	Male	249-24-7649
		262-26-5949
		648-64-5913

  - ♦ And also...

- Relationships

Table: Department
Name: Administration Tot nbr: 85 Tot sal: 345,000
Pointer to First Employee <249-24-1142>
Pointer to Last Employee <649-56-1118>

2.3.2 Secondary Indexes

- Field Values That May Repeat
- Used to Locate Class of Rows
- Access Strategy Involvement
- Points Directly to Data
  - ♦ Example: Job Title

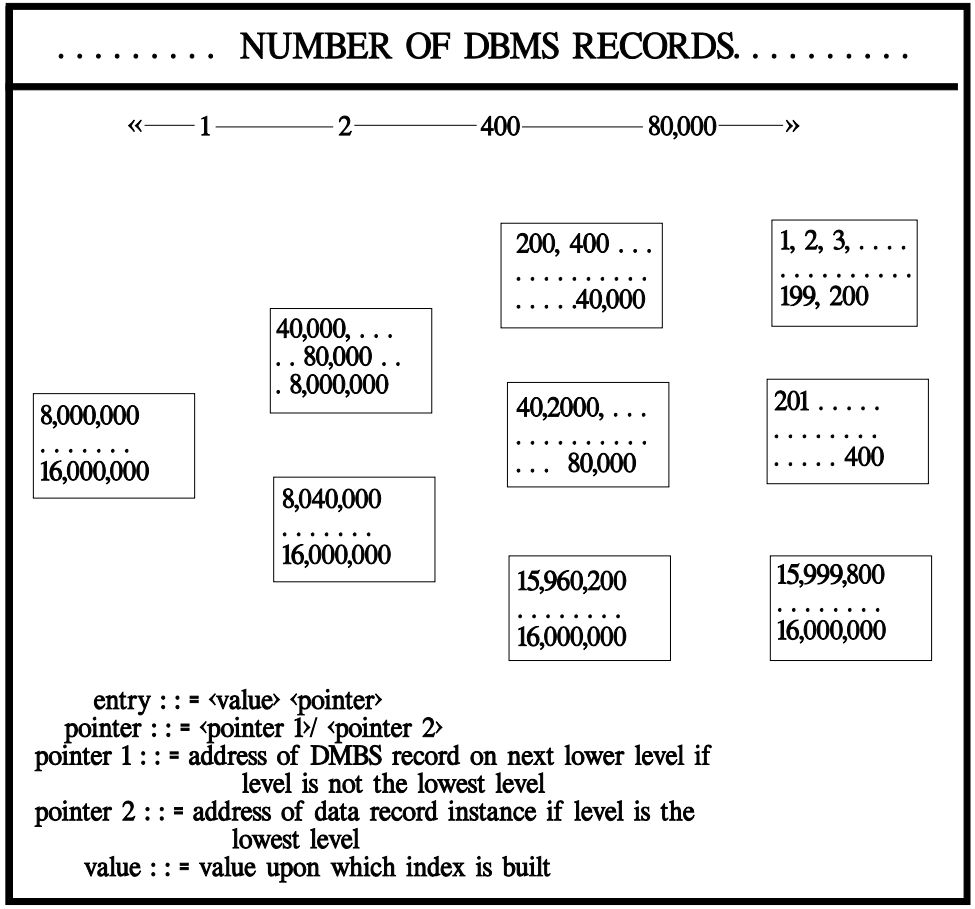
Field	Value	"Pointer" to Row
Job	Cook	249-24-7649 or RRA 17 262-26-5949 or RRA 18 648-64-5913 or RRA 39

"Pointer" Is Either Row's Primary Key, or the Relative Row Address (DBkey)

2.3.3 Multi-level Relative Row (A.k.a Inverted)

- Description
  - ♦ Logically Oriented
  - ♦ 3 to 4 Levels of Unique Values
  - ♦ Bottom Level Contains Row Identifier
  - ♦ Row Identifier Is Used by
    - ■ Alternate Row Access Technique
- If Row Pointed to Contains Data, Key = Primary Key
- If Row Pointed to Contains Lists of Row Identifiers Then it Serves As a Secondary Key.
- Advantages -- Organization of Index Is Separate from Organization of Rows.
  - ♦ Range Searches Practical.
- Disadvantages -- an Additional Level of Complexity, Thus Longer Access Times.

Multi-level, Logical Row



Hierarchical Index Organization
200 index component entries per DBMS record
16,000,000 Records Represented in Four Levels

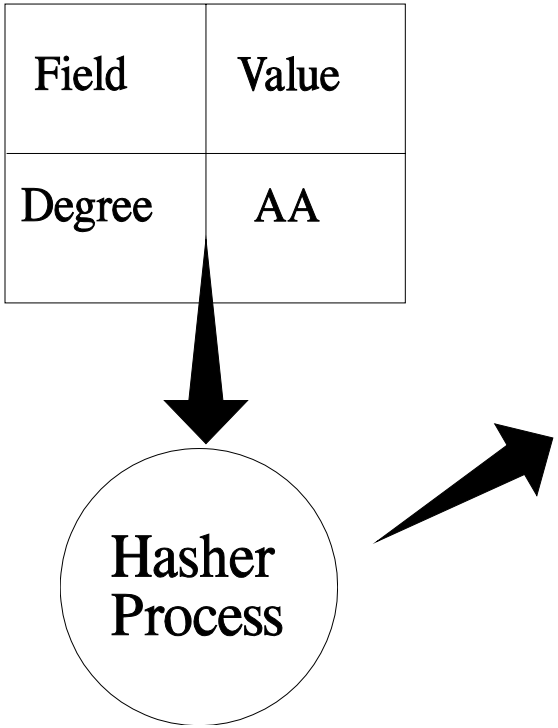
Note: See appendix A for notation definition

2.3.4 Hashed Index

- Description
  - ♦ Logically Oriented
  - ♦ No-levels, Value Is Used in Computation To Derive a Direct Access Location Of a Row Identifier.
  - ♦ If Row Identifier Points to A "Data" Row Then the Key Functions as a Primary Key.
  - ♦ If the Row Identifier Points to A Row of Primary Keys Then the Key Functions as a Secondary Key.
- Advantages -- Fastest Method of Indexing
- Disadvantages -- No Range Searching, Reorganizations Sometimes Required.

Some Examples of Hashing Algorithms

- Prime Divisor: Key Is Divided by Nearest Prime Number below the Number of Rows in the File and Remainder Is Used to Define the Position of the Row.
- Folding: Key Split into Several Parts Which Are Added Together to Give the Row Position
- Radix Transformation: Key Is Regarded as Being Rowed in Another Radix (Or Base) and Is Converted to Radix 10 to Give the Row Position
- Selected Digits
  - Position of a Row Is Derived from Chosen Digits of the Key Number
- Mid-square
  - Key Is Multiplied by Itself and the Central Digits of the Square Are Taken and Adjusted to Fit the Range of Address.

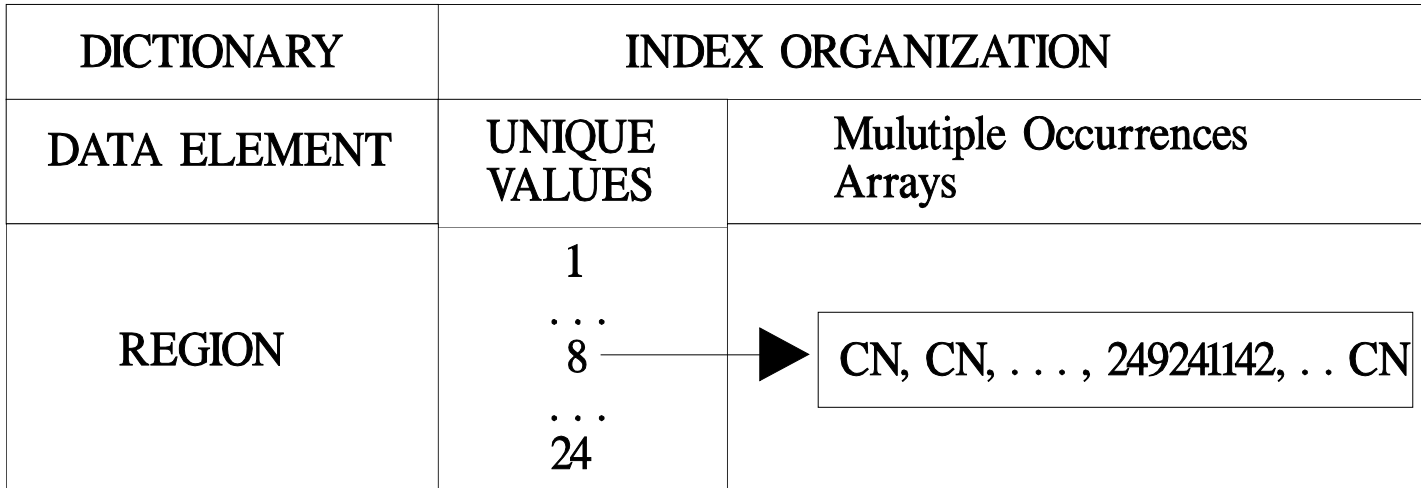


Value	Record Location				
BS	10	25	83	109	174
PhD	85	93	118	175	181
MA	15	24	67	115	125
AA	93	25	108	175	
BA	13	20	28	175	

Secondary Keys Consist of Two Parts

- Unique Value Directory
  - ♦ Organized Hierarchically
  - ♦ Organized Through Hash/calc
- Multiple Occurrence Lists
  - ♦ Organized as Pages of Dbkeys
  - ♦ Each Page Contains Ordered Lists of Dbkeys Or Bit Map of Relative Row Numbers.
  - ♦ 1=present, 0=not Present
  - ♦ Index Select Clauses Then, Either And/or Process the Dbkey Lists, or And/or the Bit Maps

Multiple Occurrence component of an index structure for Customer Numbers (CN)



Multiple Occurrence DBMS Row Reads				
Select Condition	**Number of Rows Selected**	**Traditional Dbkey Reads**	**Hi Level Bit Map Ands**	**Lo Level Bit Map Reads**
1	50,000	25	333	3
2	100,000	50	167	3
3	500,000	250	83	3
4	1,500,000	750	42	3
5	2,000,000	1000	21	3
6	500,000	250	11	3
7	100,000	50	6	3
8	2,000,000	1000	3	3
		3375		24

2.3.5 Index Summary

- Generally Added on to Some DBMS, So Not Too Well Integrated
- Integral Part of Other DBMS Designs, So Well Integrated and Utilized
- Unique Value Portions:
  - ♦ Multi-level Relative Row -- Most Flexible
  - ♦ Good All Around Design
  - ♦ Hashed -- Super for Some Specialized Uses
  - ♦ Not So Good for Most Others.
- Multiple Occurrence Portions:
  - ♦ Dbkey Lists -- Able to Be Used with Most DBMS
  - ♦ Bit Maps -- Only Used with Specialized DBMS

2.4 Relationships

- Definition: Method Employed by a DBMS To Relate Rows of the Same Or Different Types in Any of Three Ways.
  - Member to Owner
  - Owner to Member
  - Member to Member
- Fundamental Types
  - DBMS Generated Pointers
    - Embedded
    - Segregated
  - User Designated Column Values

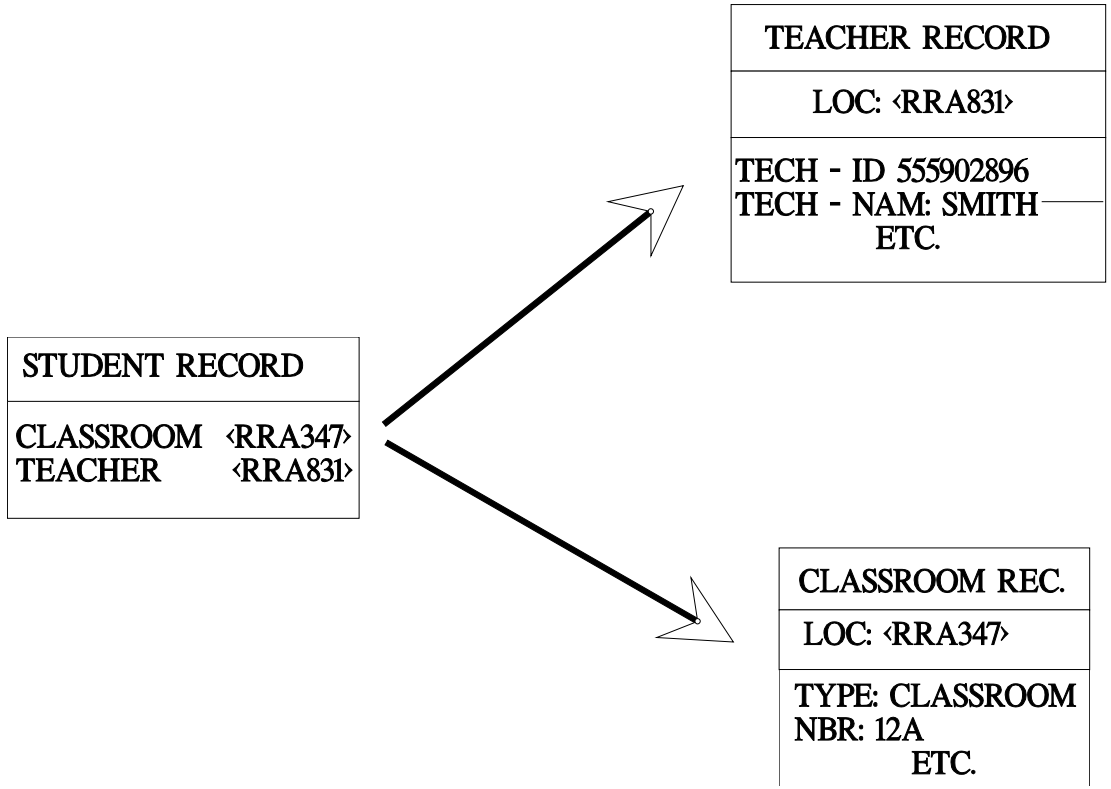
2.4.1 Static Relationships

- Embedded: Computer generated mechanisms that are contained within rows.
- Description: A DBMS technique that places the address of the target row Within the space occupied by the source row.
- Advantages: Processing related rows very fast.
- Disadvantages: Relationship modification requires row reorganization
- Relationship processing requires accessing and processing the rows.

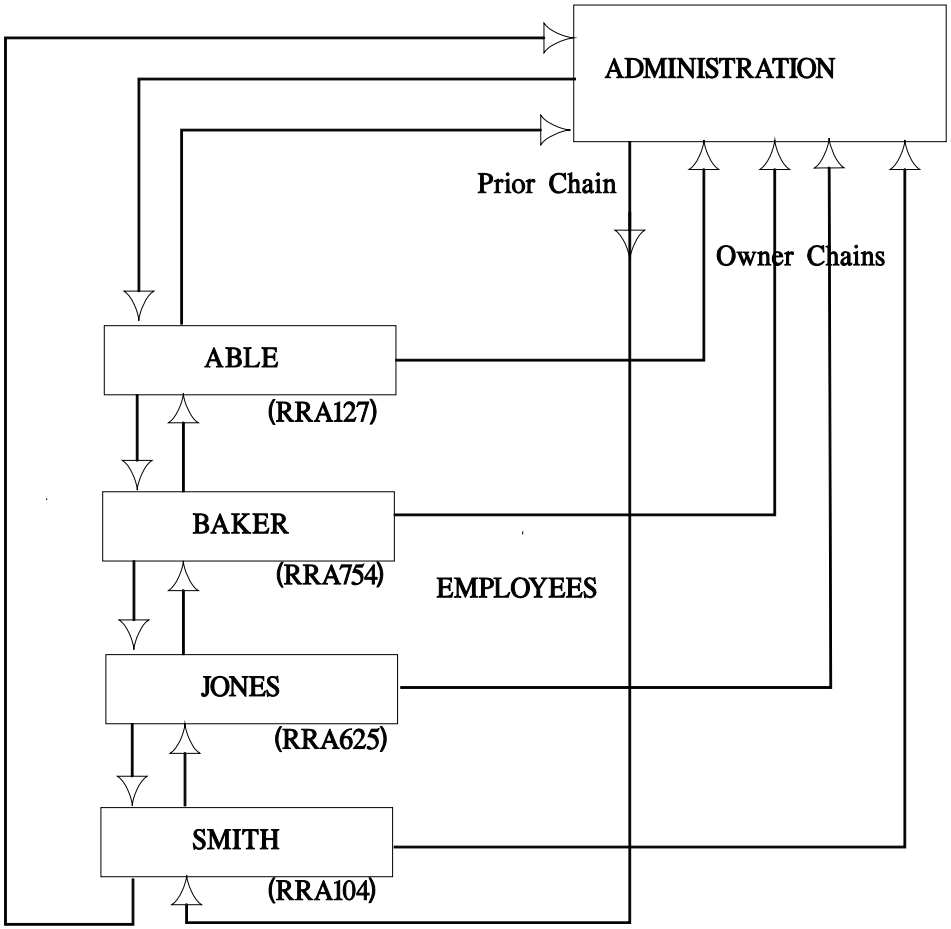
Primary Capability of Static Relationships

- Essential Sets
- Stored Arbitrarily
- Selected Arbitrarily
- Essential Ordering
- Before or after Placement
- Select: First-last-next-prior

2.4.1.1 Embedded Relationship Pointers

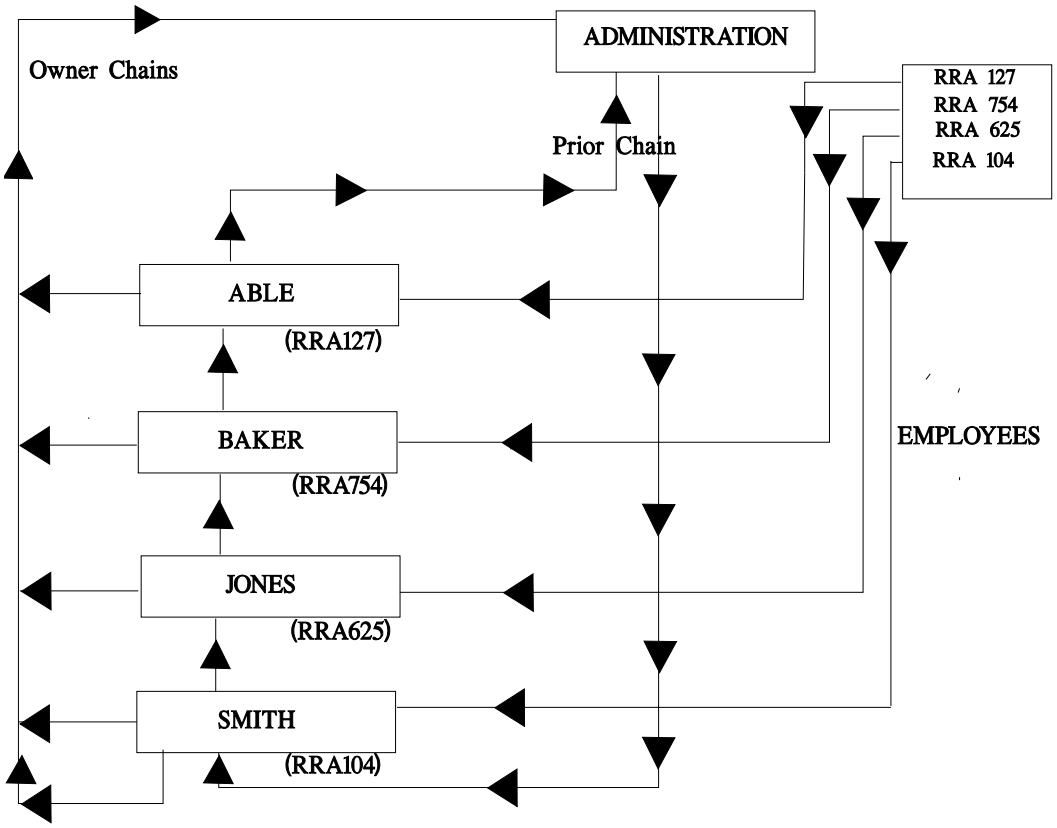


Embedded Pointers: Next, Prior, and Owner



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

Embedded Pointers: Pointer Array from Owner and then Prior and Owner.



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

2.4.1.2 Segregated Relationship Pointers

- Description: Embedded Pointers That Are Placed
  - Into a Segregated Area or File
  - Each Entry in the Pointer Area
  - Usually Consists of an Owner, Member, Sibling, and The Address of the Row
- Advantages
  - Relationships Can Be Processed Without Row Access
  - Relationships Are Pointed To by the Indexes
- Disadvantages
  - Extra Layer of Complexity
  - Slower Access to Related Rows

Segregated Relationship Pointers

RELATIONSHIP POINTERS

DATA RECORDS

10

RTI	NEXT	MEMBER	OWNER	ADDRESS
EMPLOYEE	? ? ? ?	20	? ? ? ?	5000

5000
EMPLOYEE

20

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	30	? ? ? ?	10	7500

7500
COURSE

30

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	40	? ? ? ?	10	8000

8000
COURSE

40

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	? ? ? ?	? ? ? ?	10	9700

9700
DEGREE

RTI means record type identifier

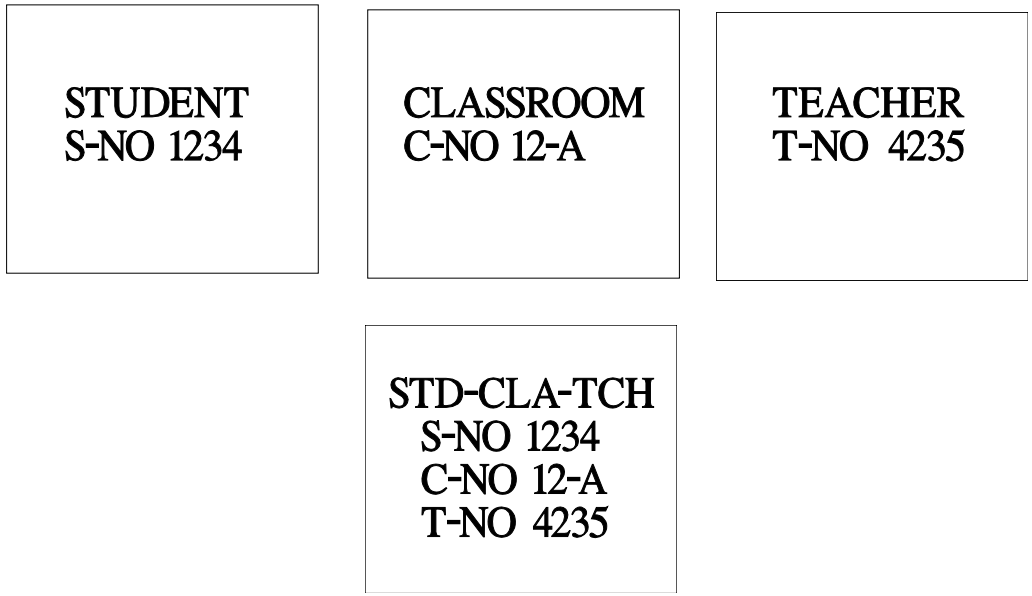
2.4.2 Dynamic Relationship

- Description
  - ♦ Rows of the Same or Different Types Are Related on the Basis of Common Values in Designated Fields.
- Advantages
  - ♦ Relationships Can Be Created, Modified, or Deleted Without Row Reorganization
  - ♦ A Row Is Related to Another Through a Field
  - ♦ Value Change.
  - ♦ Related Rows Are Determined at Retrieval Time
- Disadvantages
  - ♦ Relationship Determination Is Done for Every Request, Thus Slow Data Value Change Can Destroy Integrity

Fundamental Dynamic Relationship Capability

- Retrieval Time Set Definition
- Selected by Values
- Ordered by Values

Value Based Relationship



```
Create Table Movie_copy (
    Movie_copy_number    Integer Not Null,
    General_condition    Integer
);

Alter Table Movie_copy
    Add Primary Key (Movie_copy_number);

Create Table Movie (
    Movie_number         Integer Not Null,
    Movie_name           Character(25),
    Movie_director       Character(25),
    Description          Integer,
    Movie_star           Character(25),
    Rating               Integer,
    Movie_rental_rate    Decimal(7,2),
    Movie_date           Integer
);

Alter Table Movie
    Add Primary Key (Movie_number);
```

```
Create Table Customer (
    Customer_number      Integer Not Null,
    Customer_name        Character(25),
    Customer_street_address Character(45),
    Customer_city        Character(20),
    Customer_state       Character(2),
    Customer_zip_code    Integer,
    Customer_phone_number Integer
);

Alter Table Customer
    Add Primary Key (Customer_number);

Create Table Payment (
    Payment_transaction_number Integer Not Null,
    Customer_number      Integer,
    Employee_id          Integer Not Null,
    Payment_type         Character(20),
    Payment_amount       Decimal(7,2),
    Payment_date         Integer,
    Payment_status       Character(2),
    Check_bank_number    Integer,
    Check_number         Integer,
    Credit_card_number   Integer,
    Payment_credit_card_expiration Integer,
    Payment_credit_card_type Character(20)
);
```

```
Alter Table Payment
Add Primary Key (Payment_transaction_number);

Create Table Movie_rental_row (
    Employee_id         Integer Not Null,
    Rental_row_number Integer Not Null,
    Movie_copy_number    Integer Not Null,
    Movie_number         Integer Not Null,
    Customer_number      Integer Not Null,
    Payment_transaction_number Integer,
    Movie_rental_row_date Integer,
    Movie_rental_row_due_date Integer,
    Movie_rental_row_status Character(2),
    Movie_rental_row_rate Decimal(7,2),
    Movie_rental_row_overdue_charge Decimal(7,2)
);
Alter Table Movie_rental_row
    Add Primary Key (Employee_id, Rental_row_number,
        Movie_copy_number,
        Movie_number,
        Customer_number);

Alter Table Movie_rental_row
    Add Foreign Key (Payment_transaction_number)
    References Payment  (Payment_transaction_number)
    On Delete Restrict
    On Update Restrict;

Alter Table Movie_rental_row
    Add Foreign Key (Movie_number)
    References Movie  (Movie_number)
    On Delete Restrict
    On Update Restrict;

Alter Table Movie_rental_row
    Add Foreign Key (Movie_copy_number)
    References Movie_copy  (
    Movie_copy_number)
    On Delete Restrict
    On Update Restrict;

Alter Table Movie_rental_row
    Add Foreign Key (Customer_number)
    References Customer (Customer_number)
    On Delete Restrict
    On Update Restrict;
```

2.4.3 Relationship Summary

- Two Fundamental Types
- Static (DBMS Generated)
  - Relationships Created Through Row Add/deletes
  - Database Reorganization Required for Relationship Modification
  - Relationships Pre-wired, Determined Only Once, Thus Fast Access
  - Good for High-volume, Production Environments
- Dynamic (Value-based)
  - Relationships Created Through Field Value Changes
  - Row-type Only Reorganization When New Relationship (Field) Is Added
  - Relationships Are Determined at Every Query, Thus Slow Access
  - Good for Low-volume, Mis Environments

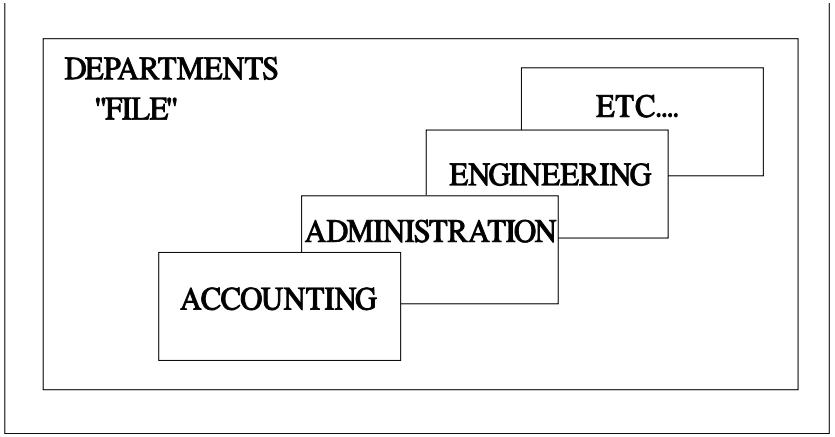
2.5 Data

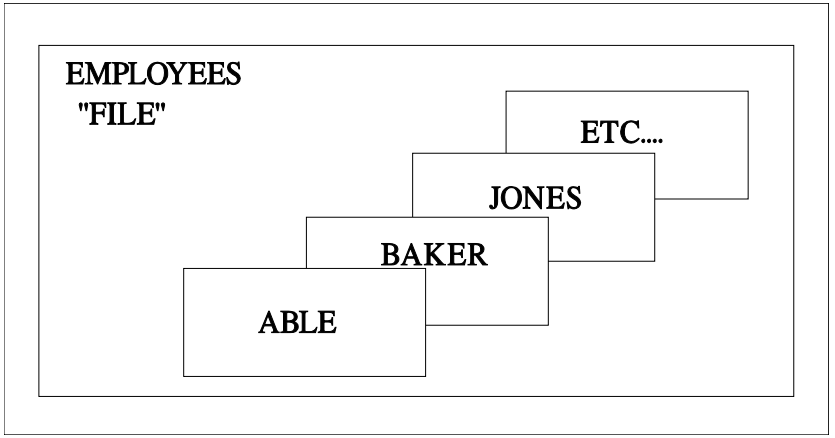
- Definition: That Part of the Physical Database in Which Actual Rows Are Stored
- Formats
  - ♦ Single Table Structure
  - ♦ Multiple Table Structure
  - ♦ Fixed Format Structure
  - ♦ Variable Format Structure
  - ♦ Complex Tables
  - ♦ Page and Key Storage
- Summary

2.5.1 Single Table

- Description: A Physical File in Which All Rows Are of the Same Type
- Advantages
  - ♦ Complete Table Independence
  - ♦ Row Order Can Be Modified Independently of Other Tables
  - ♦ Access of All Rows of One Type Is Very Fast
  - ♦ Database Damage Is Restricted to Single Table
- Disadvantages
  - ♦ Access of Collections of Rows of Different Types Is Slow
  - ♦ Multiple Table Updates, Backups & Recoveries Can Be Difficult

Example of Single Table Data





2.5.2 Multiple Table

- Description: A Single Physical File in Which Rows of Multiple Types Are Stored
- Advantages
  - ♦ Related Row Access Is Fast
  - ♦ Data Loading Can Cause Proximate Placement of Related Rows
  - ♦ Multiple Table Updates, Backups, & Recovery Is Easier to Control
- Disadvantages
  - ♦ Damage to Database Affects Multiple Tables Reorganization Affects Whole Database

Example of Multiple Table Data

- Personnel Database
- Departments Table
- Employee Table, Stored "Via"

BNF

Database ::= <file X>

<file X>::= <row S> [<row S>...]
[<row T> [<row T>...] ...]

DDL

DATABASE IS personnel
Row is department
Row is employee
RELATIONSHIP is deptempl
OWNER is department
MEMBER is employee
MODE is via

DSDL

FILE db CONTAINS department, employee

Example of Multiple Table Data (continued)

PHYSICAL Row 1		
ACCOUNTING	ABLE	BAKER
GRODMAN	KERR	PERKINSON
ADMINISTRATION	EDMONDS	CRITENDEN

PHYSICAL Row 2		
FRANCIS	GIMBLE	HARRISON
JONES	ENGINEERING	APPLETON
QUINCY	RICHARDSON	ZIEGFIELD

2.5.3 Fixed Format Tables

- Description: Each Occurrence of a Table Is Organized the Same Way
  - ♦ All Rows of a Given Type Are the Same Length
  - ♦ Each Column, Valued or Not Occupies the Same Space
- Advantages
  - ♦ No Row Decoding Is Required
  - ♦ Fastest Gets/puts with the Database
- Disadvantages
  - ♦ Sparsely Valued Databases Consume Much Space

Fixed Format--fixed Length

Row = { [{ Value | Space | Null }],...}

Last Name Type Is Character 11

First Name Type Is Character 12

Example:

SSN	Last Name	First Name	B-Date	Sex
505902896	Gorman	Michael	03221941	M

Fixed Format--variable Length

Row = { [{ Length, { Value | Space | Null } }],...}

Last Name Type Is Varchar

First Name Type Is Varchar

Example:

SSN	Last Name	First Name	B-Date	Sex
505902896	Gorman	Michael	03221941	M

2.5.4 Variable Format Rows

- Description Each Occurrence of the Row Is Created out of the Then Available Combinations of Field Codes and Field Values
- Advantages
- Maximum Flexibility in Row Construction
- New Field Types Require Only Dictionary Modification
- Value Storage into Rows, as Available
- Space Is Conserved for Sparsely Valued Rows
- Disadvantages
- Maximum Amount of Decoding Is Necessary

Variable Format--fixed Length

- Not Applicable
- Variable Format--variable Length

{ [{ Field Indicator, Length,{Value | Space | Null}}],...}

Field Code	DDL Field Name and Type
FA	SSN (SVI)
FB	Last Name (SVI)
FC	First Name (SVI)
FD	B-Date (SVI)
FE	Sex (SVI)
FF	Skill (MVI)
FG	Nickname (MVI)

Row Instances
FF~10programmer|FF~15systems Analyst
FA~525902896|FB~7gorman|FC~4mike|FD~03/22/1941|FE~m
FF~07manager|FG~5butch|FG~7tiger

2.5.5 Complex Table Structures

VECTOR:			
Telephone numbers , up to 10 occurrences			
3012229876	2019981284	.....	.....

GROUP:				
House number	**Street**	**City**	**State**	**Zip**
1234	Maple Street	Sadlebrook	NJ.	029879999

REPEATING GROUPS: DEPENDENTS	
First Name	Birthdate
Bobby	19780817

HOBBIES	
HOBBY NAME	**HOBBY-ANNUAL-COST**
Ice Hockey	2000.00
Model Boats	500.00
Soccer	250.00

2.5.6 Page and Key Storage

- Characteristics
- Single or Multiple Table Storage per Data Page
- Data Page Divided into Sequentially Stored Row Instances Key Map to Locate Start of Row

Rows and key formatted files

BNF

<file>::= { <simple DBMS row>...} / { <key formatted DBMS row>...}

<simple DBMS row >::= { <row A> [<row B>...] }

<key formatted DBMS row>::= { <row A> [<row B>...] }
{ <row address key A>
[<row address key B>...]}

<row address key>::= {<row start address>
<row key value>}

DDL

DATABASE is personnel
Row is departments
Row is employee
RELATIONSHIP is deptempl
OWNER is deparment
MEMBER is employee
MODE is via

DSDL

Any DSDLs from figures 4.28 through 4.31 (See book)

PHYSICAL RECORD 1		
RA=100 PK=ACCOUNTING ETC	RA=200 PK=ABLE ETC	RA=210 PK=BAKER ETC
RA=220 PK=GRODMAN ETC	RA=230 PK=KERR ETC	RA=240 PK=PERKINSON ETC
RA=250 PK=ADMINISTRATION ETC	RA=350 PK=EDMONDS ETC	RA=360 PK=CRITENDEN ETC
KEY-----RELATIVE ADDRESS MATCH		
100 ACCOUNTING 220 GRODMAN 250 ADMINISTRATION	200 ABLE 230 KERR 350 EDMONDS	210 BAKER 240 PERKINSON 360 CRITENDEN

Example of Data Record Instances and Key Formatted Files

2.5.7 Data Summary

Characteristic or Need	Tables Per File			
	Single		Multiple	
	Fixed	Variable	Fixed	Variable
Sequential Access	1	2	3	4
Multiple table access	3	4	1	2
Table Reorganization	1	2	3	4
Row Decoding	1	2	1	2
Production Application	3	4	1	2
MIS Type Application	2	1	4	3

Ranked Order: 1 = Best, through 4 = Worst

2.6 Storage Structure Summary

Physical Organization of the Database

Dictionary	Index	Relationships	Data

Dictionary	Intelligence
Indexes	Rapid Access Based on Values
Relationships	Rapid Access Based for Member to Owner Owner to Member Member to Member
Data	Storage of Rows

Storage Structure Orientations	Static highly engineered for production update and reporting Dynamic enables more sophisiticated "unknown" queries and reports

3.0 Access Strategy

- Introduction
- General Process
- Generic Example
- DBMS Examples
- Summary

3.1 Introduction

- Definition A Combination of DBMS Vendor Code And Operating Systems Access Methods That Access Data from The Various Parts of the Storage Structure

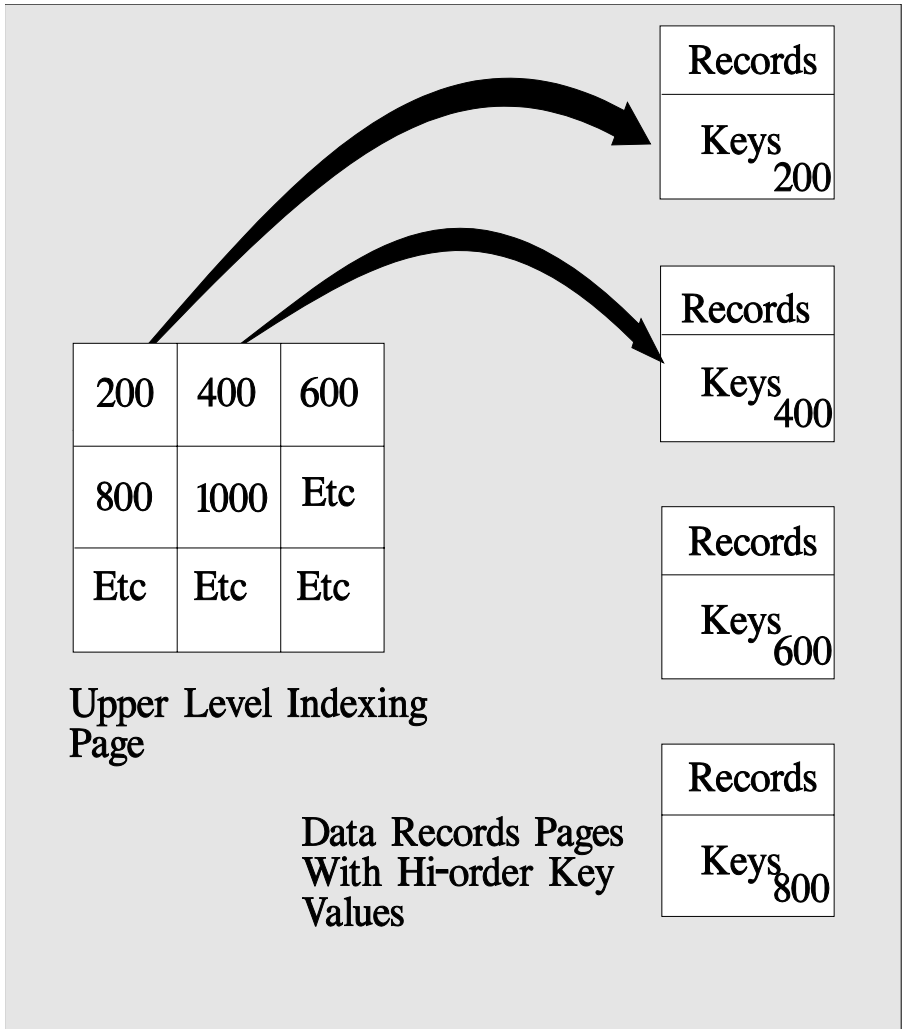
3.2 General Process

- Command is received and checked for grammatical correctness
- Dictionary is accessed to check for correct field names, types, value ranges, etc. if indexes are used then the <field name> <rel-op> <value>
- Expression is processed to narrow the selection of targets, that is, rows, or data-row-relationship-arrays
- If the command contains implicit relationship navigation, then owners, or members may be traversed.
- Target rows are retrieved into the user space
- If command was an update then values must be changed added, or deleted
- Row spaces are expanded or contracted as appropriate
- Supporting relationships are changed as appropriate. Supporting indexes are changed as appropriate.

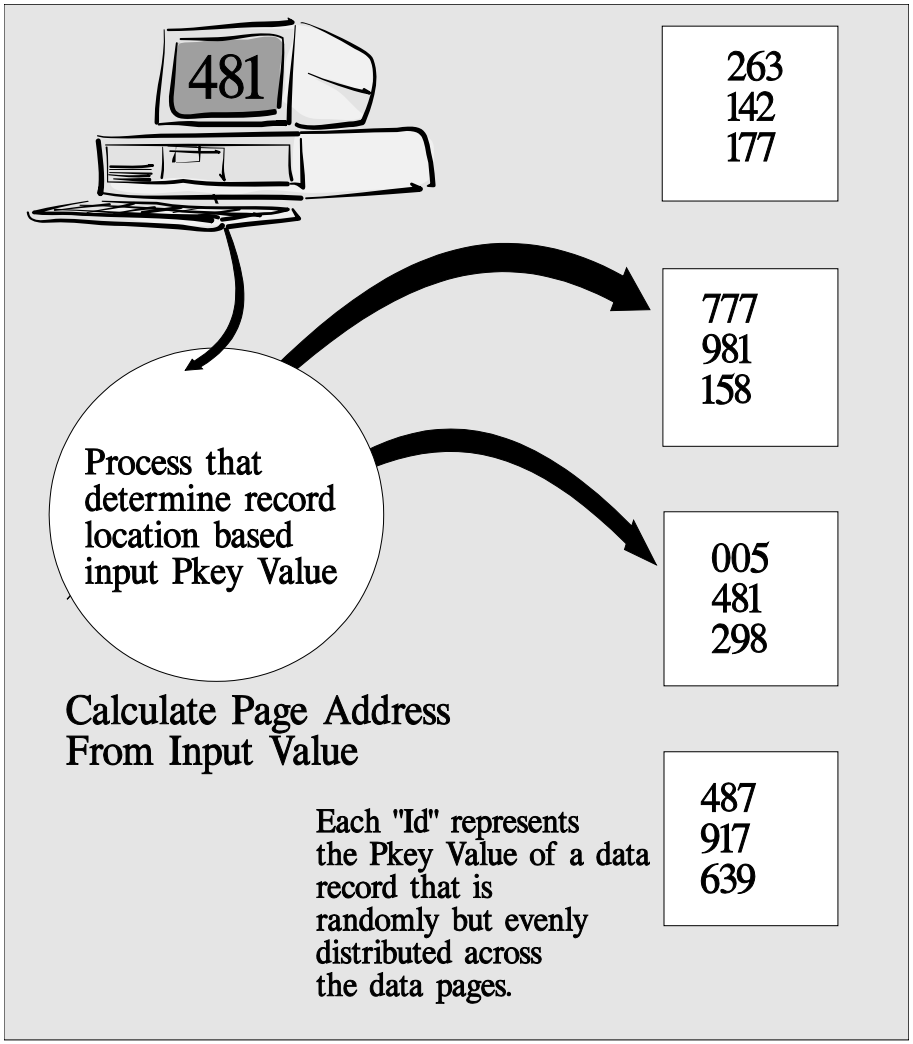
3.3 Generic Access Strategy Examples

- Primary Key Access
- Hashed Access
- Single Key Access
- Multi-key Access
- DBMS Based Pointers
- Value Based Pointers
- Dynamic Oriented Access Strategy

3.3.1 Primary Key Access (Static/dynamic)



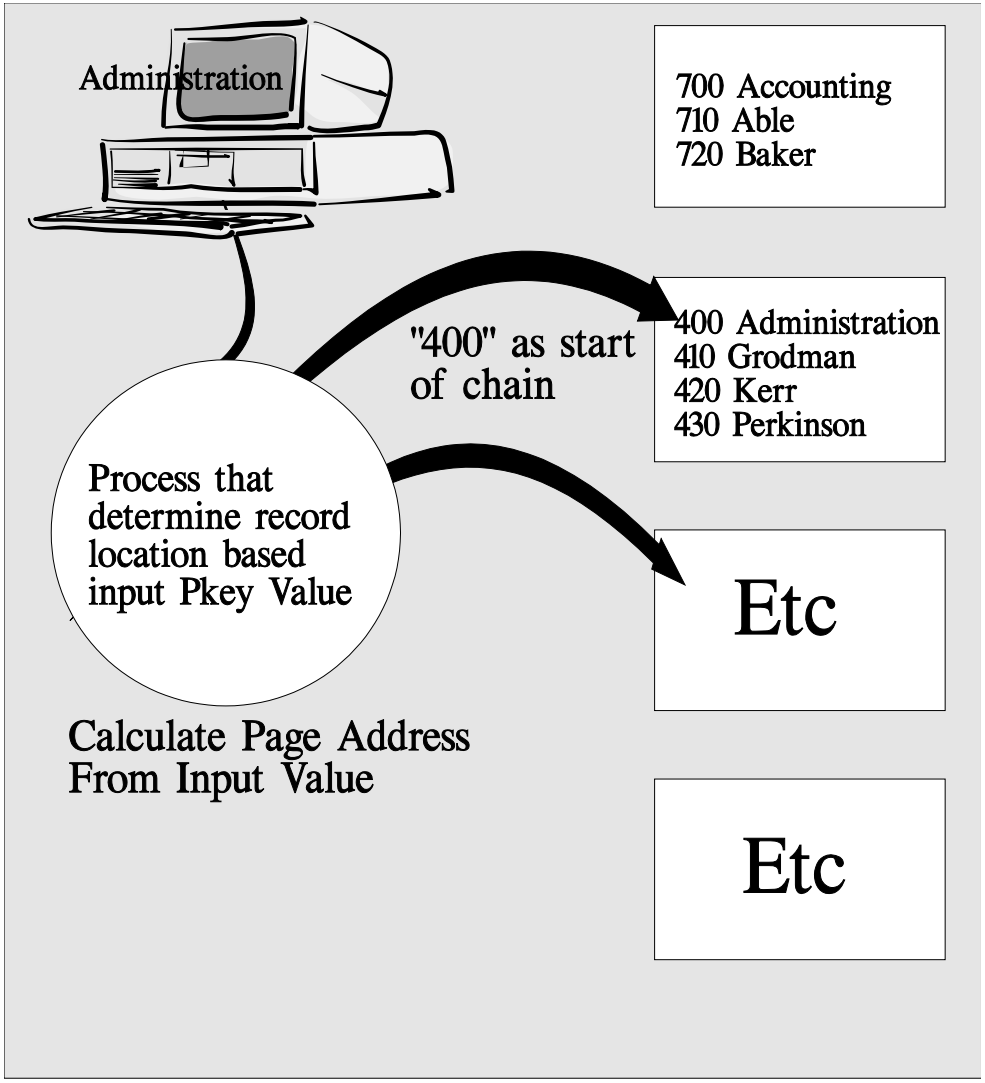
3.3.2 Hashed Addressing (Static/dynamic)



Hashed Addressing Combined with Stored "Via" (Static)

Personnel Database:

- Departments Table
- Employee Table, Stored "Via"



3.3.3 Single Key Access (Static/dynamic)

DDL Field	Index		Rows
	Unique Value	Multiple Occurrences	
Gender	Male	PK 1, PK 124, PK 185, PK 1119	12 Mary F 3.75
	Female	PK 12, PK 87, PK 187, PK 1940	124 George M 2.250
			185 Maurice M 3.04
			1940 Helen F 2.75

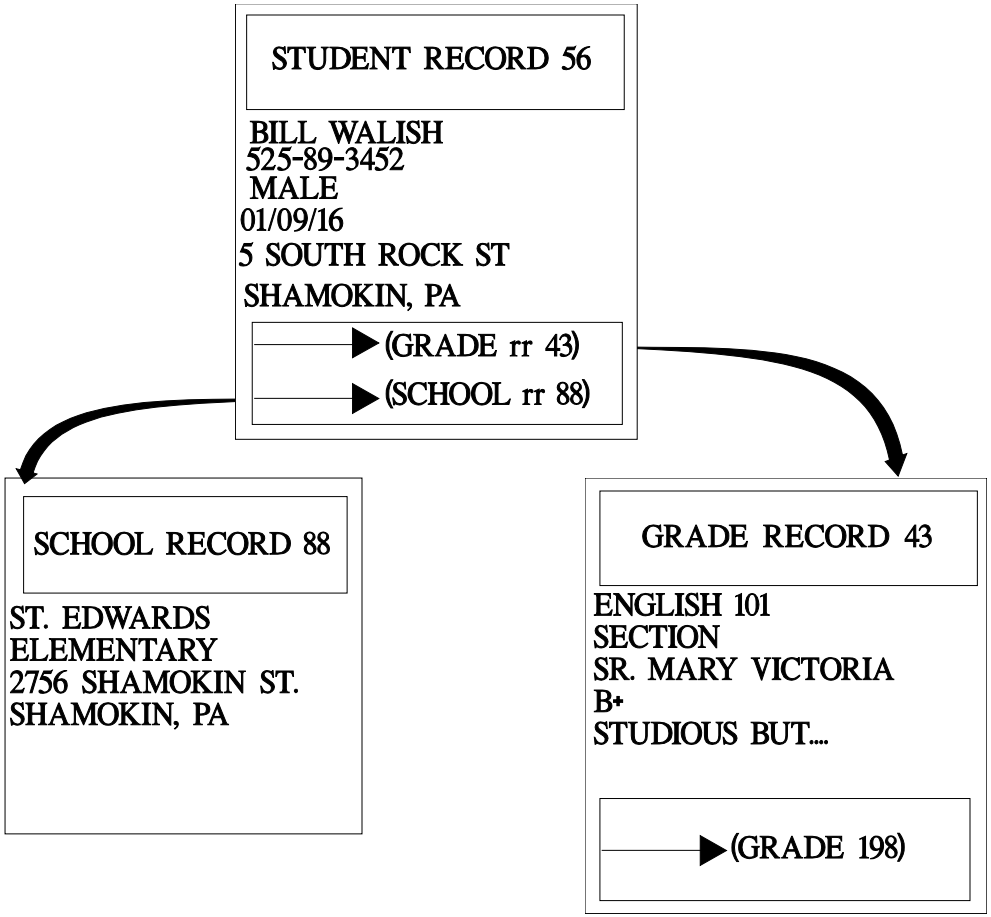
3.3.4 Multiple Key Access (Static/dynamic)

DDL Field	Index		Rows Access based on Primary Key Field Data Values
	Unique Value	Multiple Occurrences	
Gender	Male	PK 1, PK 124, PK 185, PK 1119	12 Mary F 3.75 ~~~~~~~~~~~~~~~~~~~~~~~~ 124 George M 2.250
	Female	PK 12, PK 87, PK 187, PK 1940	
GPA	1.1		185 Maurice M 3.04 ~~~~~~~~~~~~~~~~~~~~~~~~ 1940 Helen F 2.75
	2.25	PK 124	
	2.75	PK 1940	
	3.04	PK 185	
	3.75	PK 12	
	4.0		

3.3.5 DBMS Based Pointer (Static/dynamic)

DDL Field	Index		Rows Access based on Relative Row Address Values
	Unique Value	Multiple Occurrences	
Gender	Male	RRA 1, RRA 124, RRA 185, RRA 1119	12 Mary F 3.75 ~~~~~~~~~~~~~~~~~~~~~~~~~~ 124 George M 2.250
	Female	RRA 12, RRA 87, RRA 187, RRA 1940	
GPA	1.1		185 Maurice M 3.04 ~~~~~~~~~~~~~~~~~~~~~~~~~~ 1940 Helen F 2.75
	2.25	RRA 124	
	2.75	RRA 1940	
	3.04	RRA 185	
	3.75	RRA 12	
	4.0		

DBMS Based Pointers (Embedded)(Static)



NOTE : rr Means Realtive Record

DBMS Generated Pointers (Segregated)(Static)

RELATIONSHIP POINTERS

DATA RECORDS

10

RTI	NEXT	MEMBER	OWNER	ADDRESS
EMPLOYEE	? ? ? ?	20	? ? ? ?	5000

5000
EMPLOYEE

20

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	30	? ? ? ?	10	7500

7500
COURSE

30

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	40	? ? ? ?	10	8000

8000
COURSE

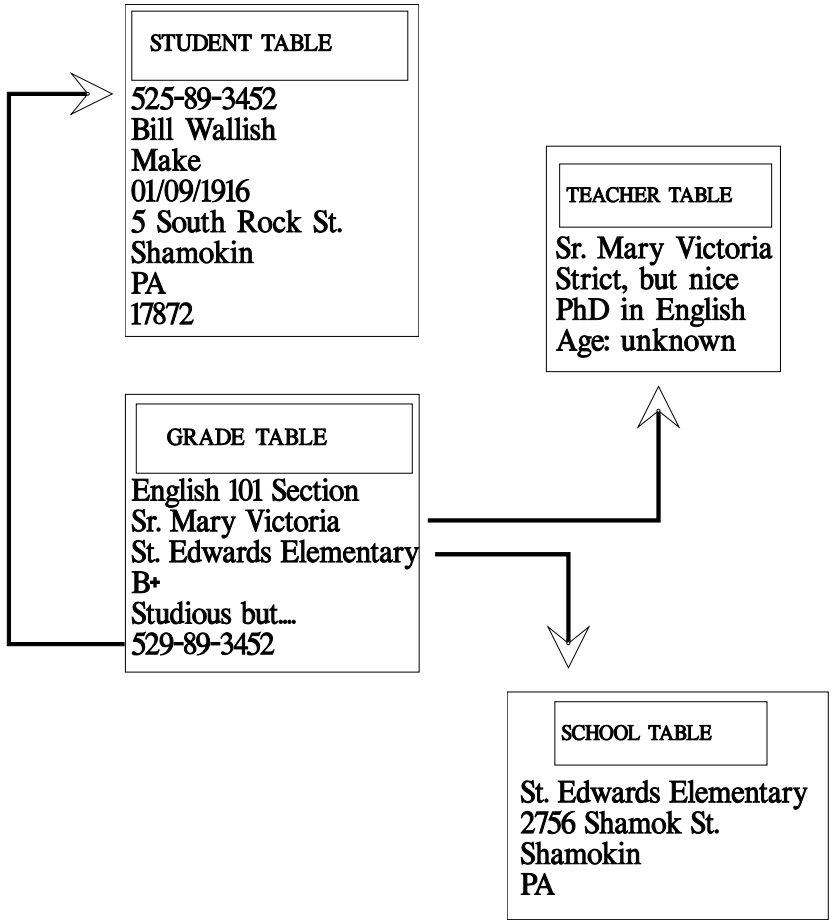
40

RTI	NEXT	MEMBER	OWNER	ADDRESS
Course	? ? ? ?	? ? ? ?	10	9700

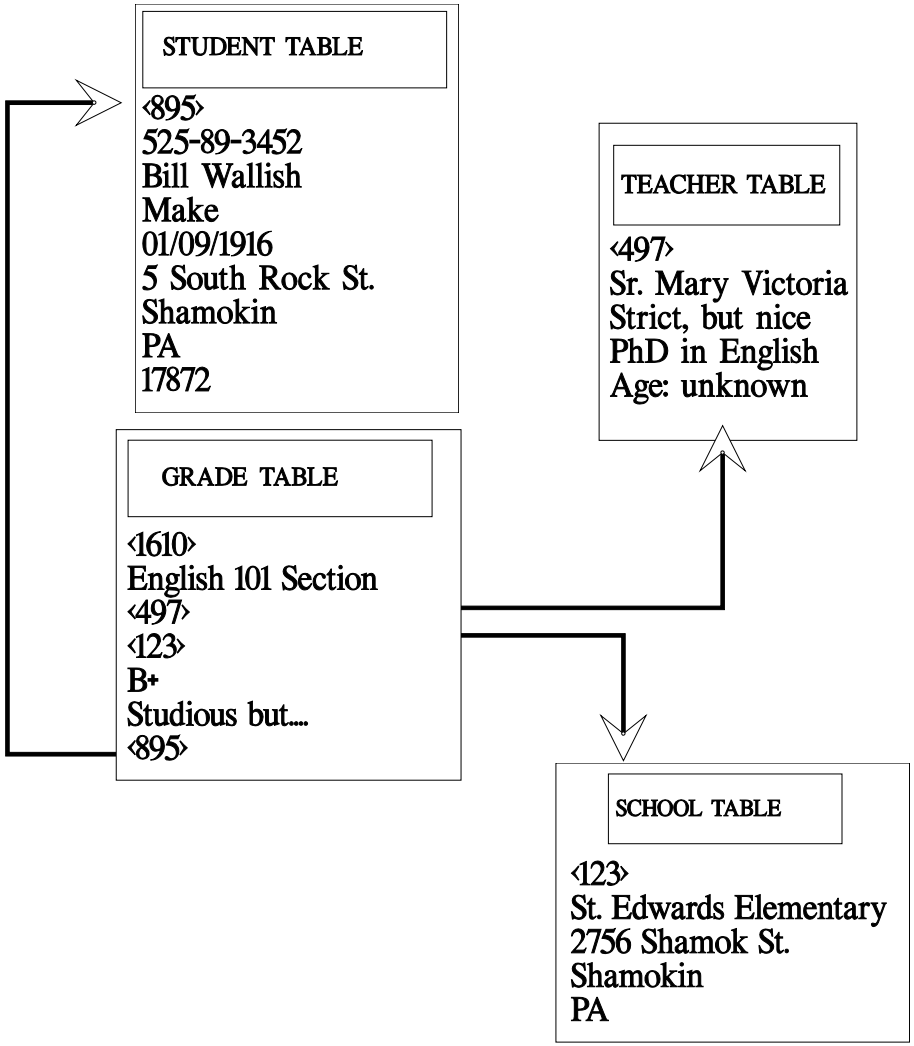
9700
DEGREE

RTI means record type identifier

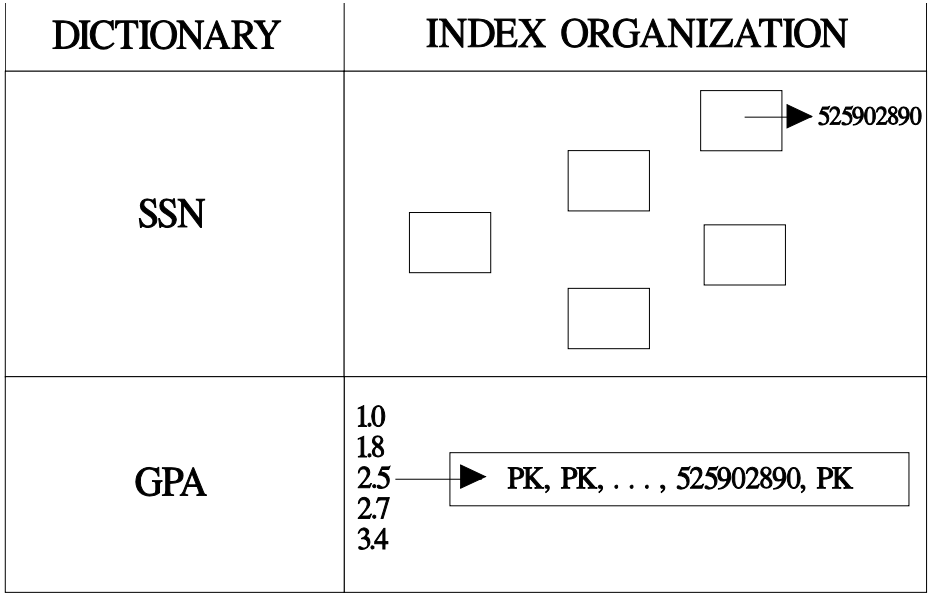
3.3.6 Business Value Based "Pointers"



Surrogate Key Value Based "Pointers"



3.3.7 Dynamic Oriented Access Strategy



Associator: Organized By Primary Key Value	
RECORD	
KEY	ADDRESS
149248819	123
219194187	512
325902896	250
423134781	183
525902890	1000

1000 DATA AREA			
			200 525902896
KEY AREA			
‹PK› ‹RA›			‹PK› ‹RA›
‹PK› ‹RA›			‹PK› ‹RA›
‹525902896› ‹200›			

3.3.8 Static Oriented Access Strategy (ANSI-NDL)

Data	Employ-Id: 1001 Name: Jones Skill: Lawyer Degree: LLB Job: Manager		
Set Type	**Owner**	**Next**	**Prior**
Person	xxx	1002	0895
Skill	xxx	0650	449
Degree	xxx	339	275
Job	189	125	etc

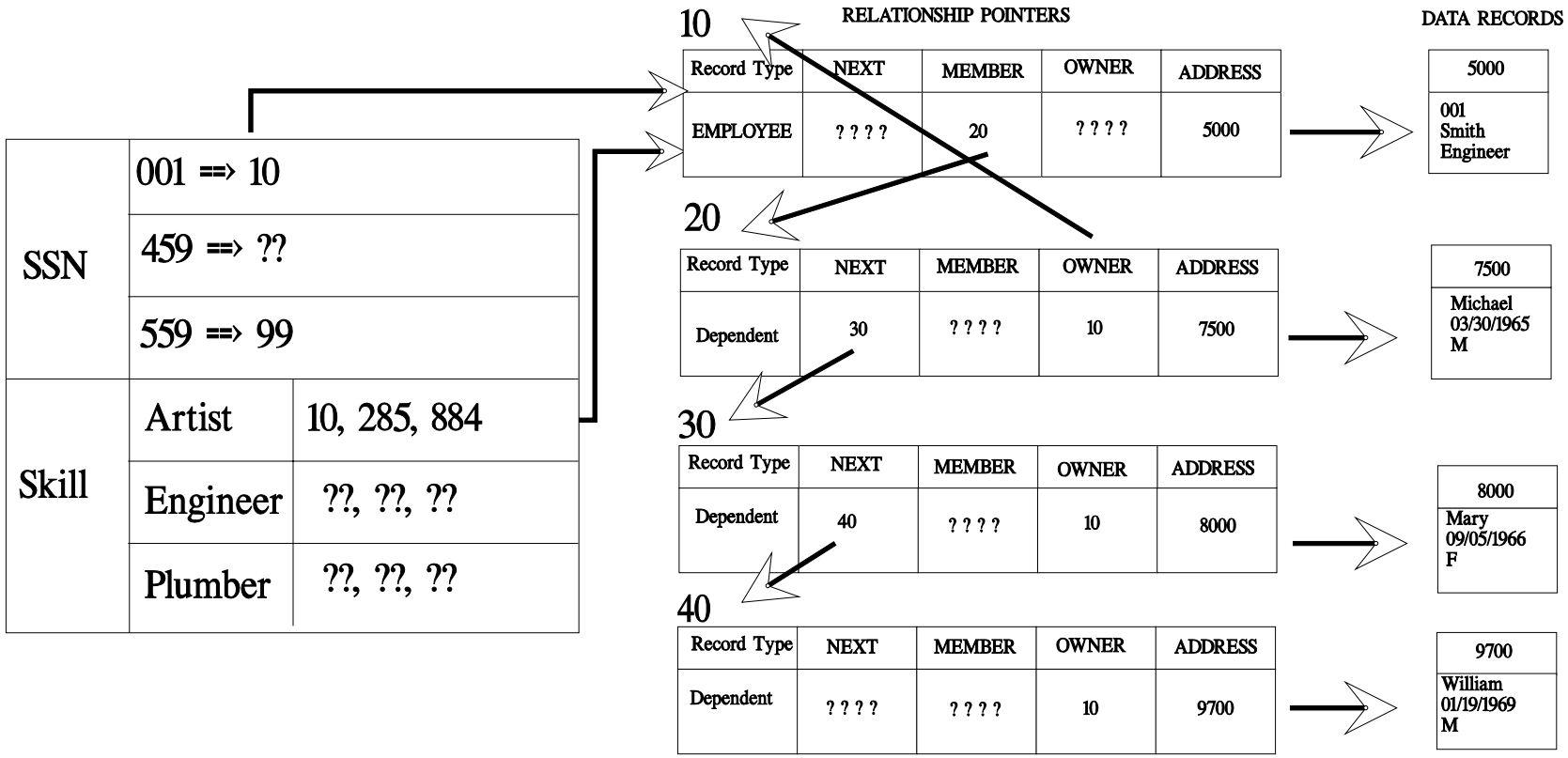
Disk Page 4789		
Data Area		
Row # 108	Row # 100	Row # 65
Row # 42	Row # 37	Row # 34
Row # 25	Row # 17	Row # 12
Key Area		
1001: 108	1002:100	0895:65
0650:42	449:37	339:34
275:25	189:17	125:12

Set Types in which this Person Participates:
Person – Singular, Ordered on Employee Id
Skill -- Singular, Ordered
Degree -- Singular, Ordered
Job -- Regular (Owner: Department)

Legend:
P-RRA: Pointer to Relative Row
Pk-RRA: Primary Key of Row
Row: Row
xxx: Not applicable

3.3.9 Static Oriented Hierarchical Access

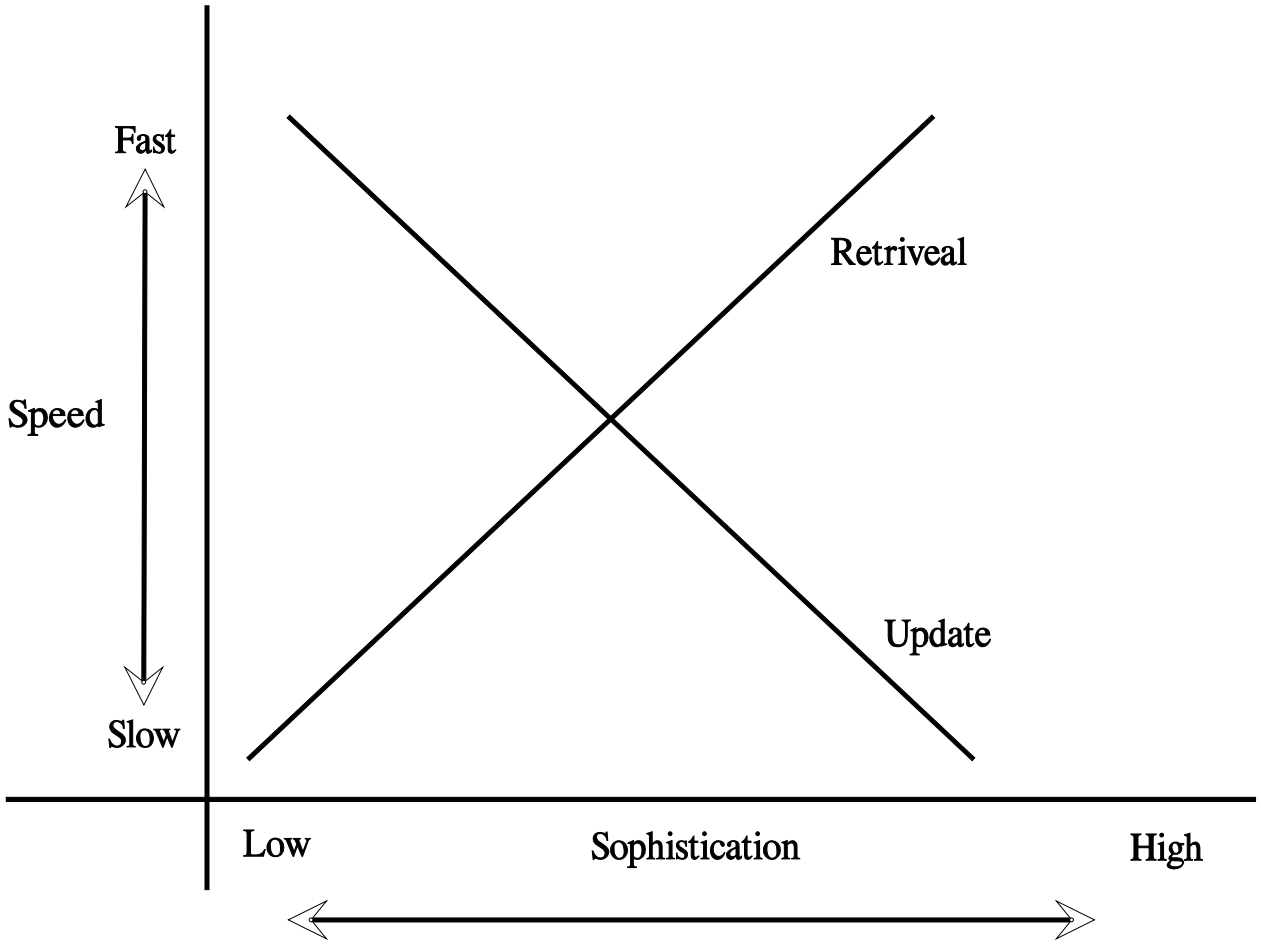
- (System 2000)
- Single Relationship Between Rows



3.4 Access Strategy Summary

- DBMS Provided Method of Getting & Putting Data
- Two Basic Orientations
  - Static
    - Production
    - High Volume
    - Low Change
    - HLI Orientation
  - Dynamic
    - Ad-hoc, MIS
    - Low Volume
    - High Change
    - Natural Language
    - Orientation

Access Strategy – Tradeoffs



4.0 Data Loading

- General Process
- Key Issues
- General Process
- Data Loading Strategies
- Load Engineering
- Summary

4.1 Key Issues for Data Loading Capabilities

- Editing Data Prior to Entry
- Validation of Data Once Loaded
- Data Conversion from Different Formats
- Creating Audit Data for Loading Integrity
- Database Transformation

Data Editing Capabilities Within the DBMS

- Simple -- Alphanumeric Check Clauses
- Example
  - ♦ Employee-name Type Is Char 15
  - ♦ SSN Type Is Numeric 9(9)
- Sophisticated -- Legal and Illegal Dates
- Example
  - ♦ Payment-due-date Type Is Date

Data Validation Automatically Provided

- Simple Validation
- Valid, Invalid, and Values Ranges
- Example
  - Sex Type Is Character 6 Valid Values Are "Male", "Female" Null Not Allowed
- Sophisticated Validation
- Arithmetic, Boolean, and Relational Operations on Single and Combinations of Fields
- Example
  - Salary Is Numeric
  - If Grade Lt 14 Then Range Is Lt 40000
  - If Grade Ge 14 Then Range Is Ge 40000
  - Null Is Not Allowed

Data Conversion

- Transformation from One Form to Another
- Examples:
  - ♦ EBCDIC to ASCII
  - ♦ ASCII Numeric to Binary
  - ♦ ASCII Decimal to Single Precision
  - ♦ Double Precision to ASCII Decimal

Rationale:

All Data in Single Internal Format

Problems

Possible Loss of Precision
Loss of Significance

Audit Data Creation

- Data Fields Especially for Audit Trails.

- Examples

  - ♦ Entry Date
  - ♦ Change Time
  - ♦ Validity Expiration Date
  - ♦ Removal Time/date
  - ♦ Transaction Id
  - ♦ Person Id
  - ♦ Source File Id

- Rationale:
  Audit Trails of Loading Transactions
  Time/date/edition Stamps

- Problems:
  Only If Not Done

Database Transformation

- Moving Data from One Database to Another.

Due to Database Redesign, or Creation of Summary Database.

- General Process
  - ♦ Graphically Illustrate Source and Target
  - ♦ Column by Column Within Each Table
  - ♦ Note Any Algorithmic Transformations
  - ♦ Between Source and Target.
- Implement Programs to Carry out Process.

Database Transformation

- Key Issue
  - ♦ One to One Conversion : Has Editing and Validation Been Completed
  - ♦ Many to One Conversion: Has Consensus Editing and Validation Been Performed
  - ♦ Requirement for Parallel Systems
- Small Db -- Transform Between Update Cycles
- Large Db -- Conversion of Whole May Have to Be Done in Pieces

Transformation Rationale

- Phased Implementation
  - Product Line by Product Line
  - Business Unit by Business Unit
  - Year by Year
- Design Evolution (Plan for Change)
  - "Weld" Well-Designed Components For Processing Performance
  - "Loosely" Tie down Immature Design Components For Flexibility in Design Change
  - Change Is a Sign of Progress, Not Always Defectiveness

4.2 General Process

1. User Program Reads Row from External Source

2. User Program "Inserts" Row into Database.

3. DBMS Edits for Correctness, Etc., and Places Row into Buffer.

4. When the Buffer Is Full, the DBMS Writes Row to a Loader Scratch File.

5. When All Rows Are Read and on the Scratch File, the DBMS:

    - Loads All Rows onto the Data, En Mass.
    - Updates at the Same Time, All Inter-row
    - Relationships Builds All Supporting Indexes, En Mass.

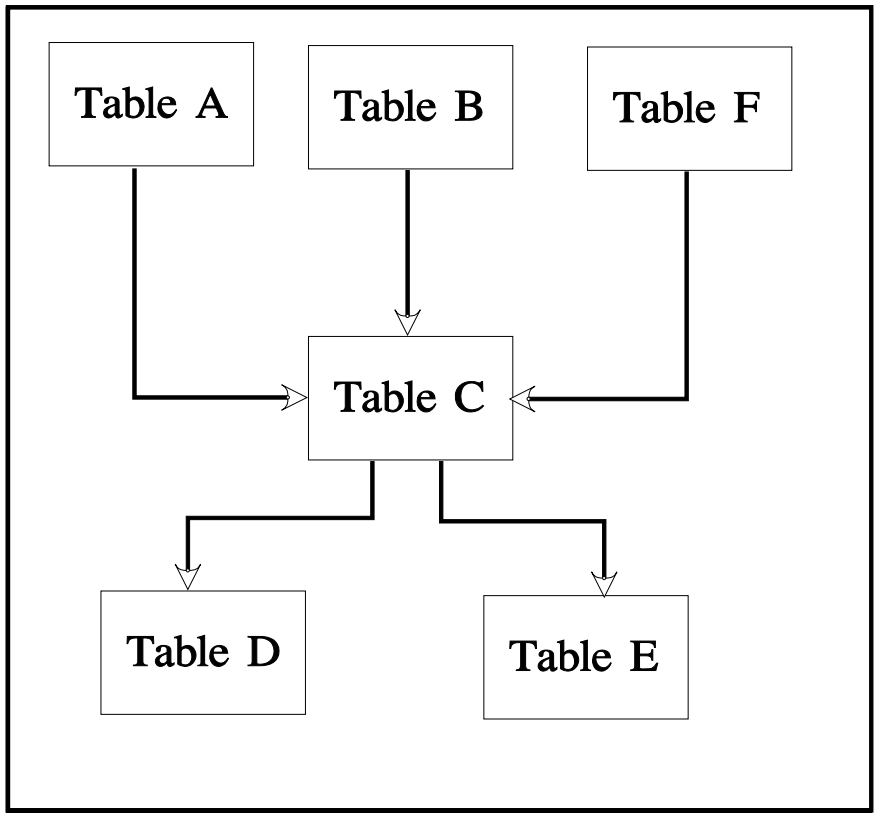
4.3 Data Loading Strategies

- Static Database
  - ♦ Logical Row Loads (Whole Employees, Projects, Etc)
  - ♦ Batch-up Loads for Cost-effective Backups and Recoveries
  - ♦ Hierarchical Data Model--Load Logical Rows as Above.
  - ♦ Network Data Model – Hierarchal Loads Plus, Connects to Other Parents.
- Dynamic Database
  - ♦ Tables Within Separate Jobs As Data Is Provided
  - ♦ Batch Jobs for Backups and Recovery

4.3.1 Static Data Loading

- Sort Data into Load Order
  - ♦ Single File
  - ♦ Separate Files
- Insert into Descending Modes of Principal Sets
  - ♦ Connect into Additional "Member" Sets
  - ♦ Proceed to next "Root" Mode

Network Data Loading Strategy



Strategy: Manual

Store <A-1> Store <C-1>, Insert C1 into Ac Set
Store <B-1>, Insert C1 into Bc Set
Store <F-1>, Insert C1 into Fc Set
Store <D-1>, Insert D1 into Cd Set
Store <E-1>, Insert E1 into Ce Set

Strategy: Automatic

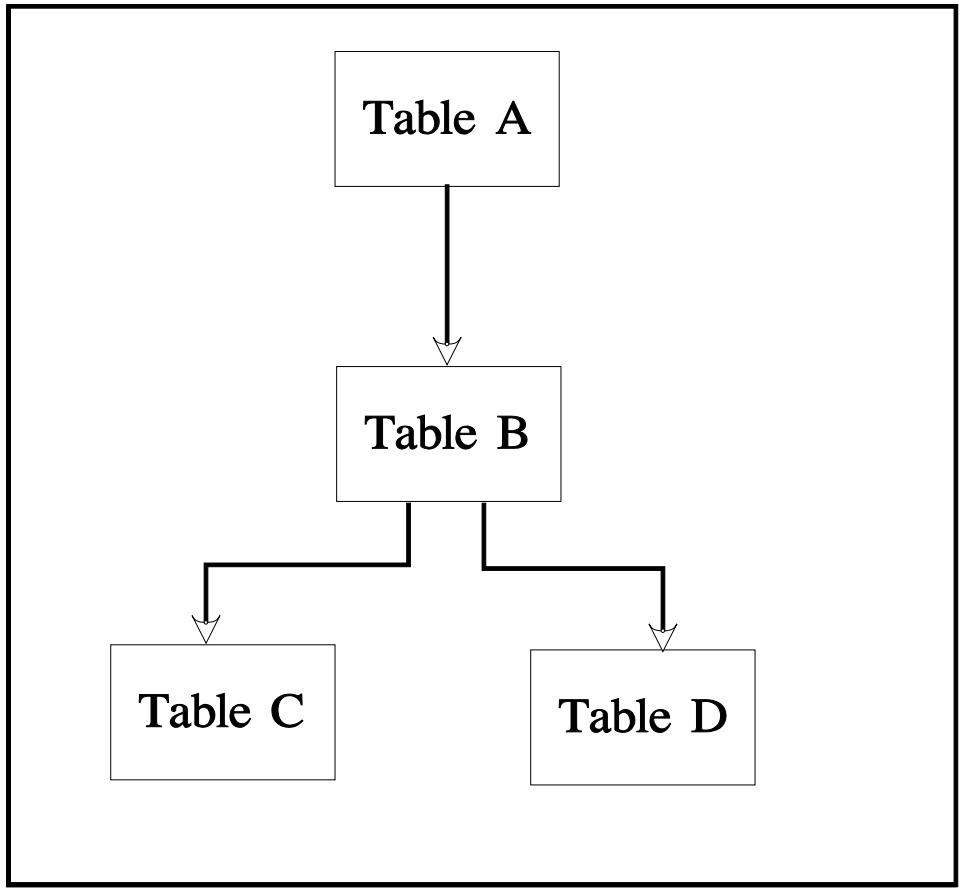
Store <A-1> Store <B-1> Store <F-1>
Store <C-1>, Automatic Insert into Ac,bc, & Fc Sets
Store <D-1>, Automatic Insert into Ce Set
Store <E-1>, Automatic Insert into Ce Set

Hierarchical Data Loading Strategy

Store A(1)
Store B(1)
Store C(1). . . .C(n)
Store D(1). . . .D(n)

Store B(2)
Store C(1). . . .C(n)
Store D(1). . . .D(n)

Store A(2). . . .



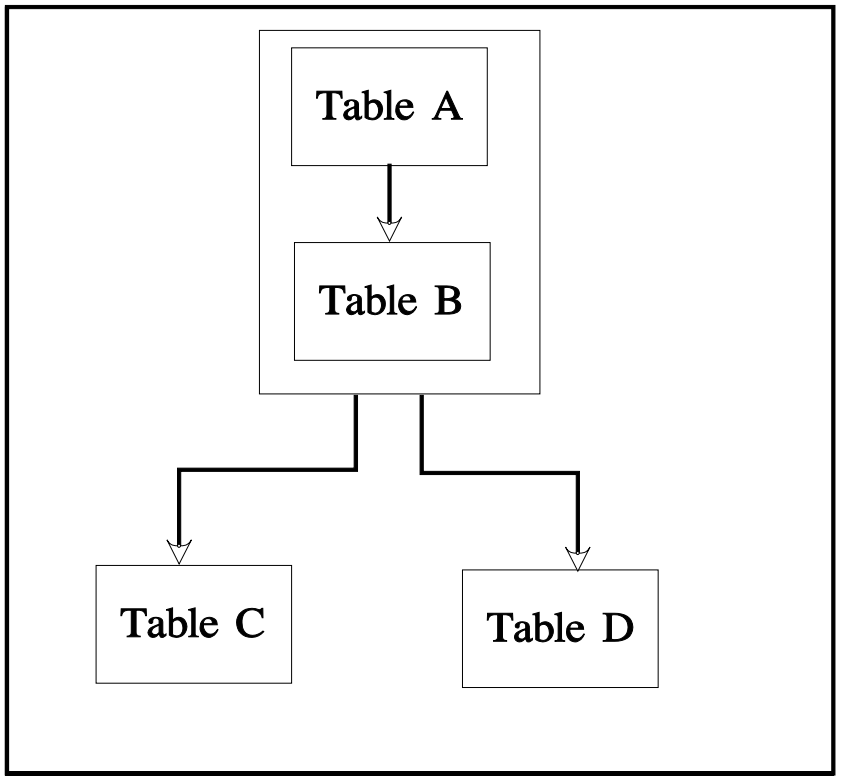
4.3.2 Dynamic Database Loading

- Sort rows of same type into specific order, for example, primary key, then load rows.
- Then proceed to the next table.
- BUT you must account for referential integrity....

Independent Logical File Example

Store A1,
Store B1 within the context of A1
Store B2, within the context of A1,...., Store B3
Store C1with reference to A1
Store C2 with reference to A1
Store D1 with reference to A1
Store D2 with reference to A1

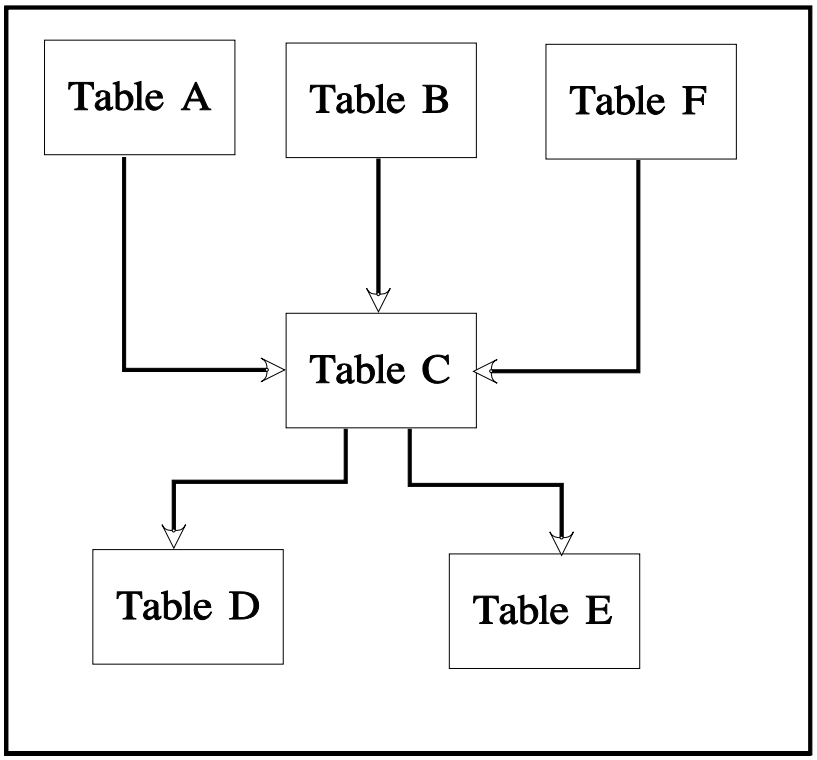
Store A2, store B1 within the context of A2
Store B2, within the context of A2,...., Store B3
Store C1with reference to A2
Store C2 with reference to A2
Store D1 with reference to A1
Store D2 with reference to A2



Store A1, ..., A <last>
Store B1, ..., B <last>
Store F1, ..., F <last>

Store C1, with appropriate reference to A1, B1, and C1.

Store D1 with reference appropriate C1
Store E1 with reference appropriate C

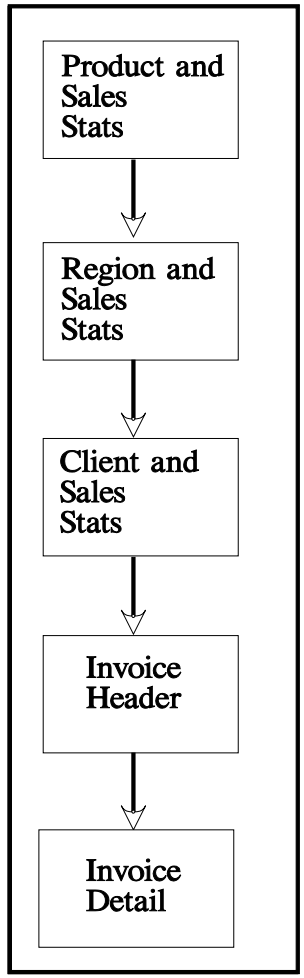


4.4 Load Engineering

- Know DBMS's storage structure and access strategy
- Know how precisely you can "place" rows of different types on DBMS files
- Carefully "layout" principal table access
- Organize rows to load into physical placement that mirrors principal retrieval order.

Load Engineering Example for a Product Sales Data Warehouse

- Load Products, regions within products, and clients within regions.
- Load Invoices one after another and build statistics and update "up the tree"



4.5 Data Loading Summary

- Large Volume Data Loading Strategy
- Data Editing -- must Be Done for Integrity
- Data Validation -- Critical to Have Database
- Data Conversion -- the Essence of Database: Common Semantics
- Data Transformation -- Database Evolution: Not Revolution.
- Phased Loads -- Moving over Gradually
  - Product Line by Product Line
  - Business Unit by Business Unit
  - Year by Year
- Data Loading Strategies -- All at Once, And Go Broke!!

5.0 Data Update

- Introduction
- Content Changes
- Storage Structure Effects
- Lockout
- Summary

5.1 Introduction

- Data update relates to content changes for:
  - Rows
  - Columns
  - Relationships
- Storage structure effects
- Data update often requires lockout of the
  - Database
  - Physical Row
  - Logical Row
- Emphasis: Not on updating, but physical database effects from updating

5.2 Content Changes

Row Add/delete:

- Static
  - Add
    - New Owner
    - New Member
    - New Relationship
    - New Index Pointers
  - Delete
    - Delete Whole Family (Owner)
    - Delete Only Member
    - Break and Mend Links
    - Delete Index Pointer
- Dynamic
  - Add Row
  - Add New Index Links
  - Delete Link
  - Delete Index Links

Column Changes

	Operation		
Effect	**Add**	**Delete**	**Update**
Fixed Length Rows	Change Null to Blank for Value	Change value to null or blank	Change one value to another
Variable Length Rows	Expand row size and add value	Shrink row size	Expand or shrink row size
Indexed Fields	Add new links	Delete links	Delete link or add link
Primary Keys	Add new row	Remove row	Delete row or add row

Relationship Changes

Change Type	DBMS	
	Static	**Dynamic**
Add	Add segment	change null or blank to value
Delete	Delete segment	Change value to blank or value
Change	Delete then find and add segment	Change one value to another

Content Change Summary

Change Type	DBMS Type	
	Static	**Dynamic**
Rows	Affects indexes and relationships	Affects only indexes
Fields	Affects indexes	Affects indexes
Relationships	Connect, Disconnect, Next Prior, and Owner effects	Field update

5.3 Storage Structure Effects

- Data Changes
- Index Effects
- Relationship Effects
- Dictionary Effects
- Summary

Data Change Effects

Row Structure	Operation	Change Effects	
		In place changes	Expansion or contraction
Simple row or column	Add or modify	Exchange one value for another	Remove or add padded blanks
Complex row, multi-valued item, or segment	Add or delete	Exchange one value for another	Reduce padded space, or logical row must fit across pages or logical row must move.

Index Updates -- Caused by Field Updates

Example: Adding Another Multiple Occurrence

Transaction: Add Employee Where SSN = 124578171, Sex= Male, Job = Programmer

DDL Field	Index		Rows Access based on Primary Key Field Data Values
	Unique Value	Multiple Occurrences	
Gender	Male	Pkey <124578171>	124578171 Maurice Male Programmer
Job	Analyst	Pkey <existing SSN value>	
	Programmer	Pkey <124578171>	
	Systems Analyst	Pkey <existing SSN value>	
	Systems Analyst - Programmer	Pkey <existing SSN value>	

Storage Structure Effects: May result in both unique value and multiple occurrence page splitting. May result in data page splitting.

Relationship Updates

- Static Imbedded
  - Modify Owner If First
  - Modify Prior If next
  - Connect/disconnect to Set,
  - Modifying Owner, next & Prior
- Static Segregated
  - Modify Owner Array If First
  - Modify Prior Is next
- Dynamic
  - Change Field Value

Dictionary Update

Data Definition Language Change	Scope of Change
Add Modify	Data Field
Add Delete Modify	Table
Add Delete	Relationship Type

- These are logical reorganizations that are treated in system control, below

Storage Structure Change Summary

- Few Changes Are Simple
- Careful Analysis of Storage Structure, Access Strategy Determines Effect
- Complex Storage Structures Hate Updates!

5.4 Lockout

- DBMS Imposed Single-threaded DBMS Activities

- Lockout Levels
  - Column (Least Course)
  - Logical Row
  - Physical Row (Page)
  - Dictionary
  - Indexes
  - Database

Column Lockouts

- Indexed columns imply index lockout
- This level of lockout is non-existent

Logical Row Lockouts

- Add/delete of Logical Row Can Lock:
  - Either Part of a Physical Row,
  - or Parts of Multiple Physical Rows
  - Depends upon the Logical Row Size.

- Physical Row Lockouts
  - Add/delete of Logical Row Can Invoke:
    - Single Physical Row Lock--entirely
    - Multiple Physical Row Lock--entirely
    - Multiple Index Physical Rows
    - Multiple Relationship Rows (Static Only)

Dictionary Lockouts

- Changes to the dictionary reflect:
  - ♦ new/changed column or contained clause
  - ♦ New/changed table or clause
  - ♦ Deleted table or clause
- Dictionary lockouts often lock the database
- Database lockout: only one user allowed database access during the lockout

Static or Dynamic Implication on Lockouts

- Dynamic DBMS (1 Table = 1 O/S File)
  - ♦ Lockouts Typically Affect Only 1 Table
  - ♦ Logical Row Lock (One Logical Row Instance)
  - ♦ Physical Row Lock (>1 Logical Row Instance)
- Static DBMS (1 O/S File = >1 Table), or (>1 O/S File = 1 Table)
  - ♦ Lockouts Typically Affect >1 Tables
  - ♦ Physical Row Lock (>1 Logical Row Instance)

Caution: As dynamic DBMS increases performance, storage structures tend toward static, and thus, locking granularity becomes more coarse

5.5 Data Update Summary

- Critical Issues
  - Extent of Control Exercised by User
    - HLI--high Control over Users
    - POL--moderate Control over Users
    - QUL--no Control over Users
  - Processing (On-line/batch)
- Lockout Granularity
  - Column (Not Common)
  - Logical or Physical Row
  - Static / Dynamic Implications
    - Static -- Typically More Coarse
    - Dynamic-- Typically less Coarse, but As Performance Increases, Granularity Is More Coarse
- Not All Changes Are Simple
- Lockout Has to Be Examined in Detail
- Dictionary Changes Often Lock Database

6.0 Database Maintenance

- Introduction
- Key Issues
- Summary

6.1 Introduction

- The DBMS vendor-supplied utility for backing-up a physical database for later restoration and use.

- General Process
  - ♦ DBMS Receives Backup Request from User
  - ♦ The DBMS Accesses the Database to Identify All the Files
  - ♦ Each File Is Copied to Another Media

6.2 Critical Issues

- Resource Requirements for Backup
  - ♦ Allocate Extra Space on Disk/tape
  - ♦ Allocate Time to Backup
  - ♦ Minutes, Hours,...??
- Is Database Locked During Backup
- Are All Backups Able to Be Restored?
  - ♦ New DBMS Version Might Not Recognize
  - ♦ Archived Database.
- Time Bombs
  - ♦ Old DBMS Might Not Operate
  - ♦ Because of a DBMS Version Expiration Date.
- Finally, Is Your Backup Good ???????

6.3 Database Maintenance Summary

- Understand the archiving process.
- Determine the time and disk resources required
- Know the extent of lockout
- Determine process to recover old archive.
- Can you keep old DBMS copy and still use it in the future for use with old database copies?
- Determine how to know if backups are good

7.0 Physical Database Summary

- Storage Structure
- Access Strategy
- Data Loading
- Data Updating
- Database Maintenance
- Static Vs. Dynamic
- DBMS Configurations

7.1 Storage Structure Summary

Physical Organization of the Database

Dictionary	Index	Relationships	Data

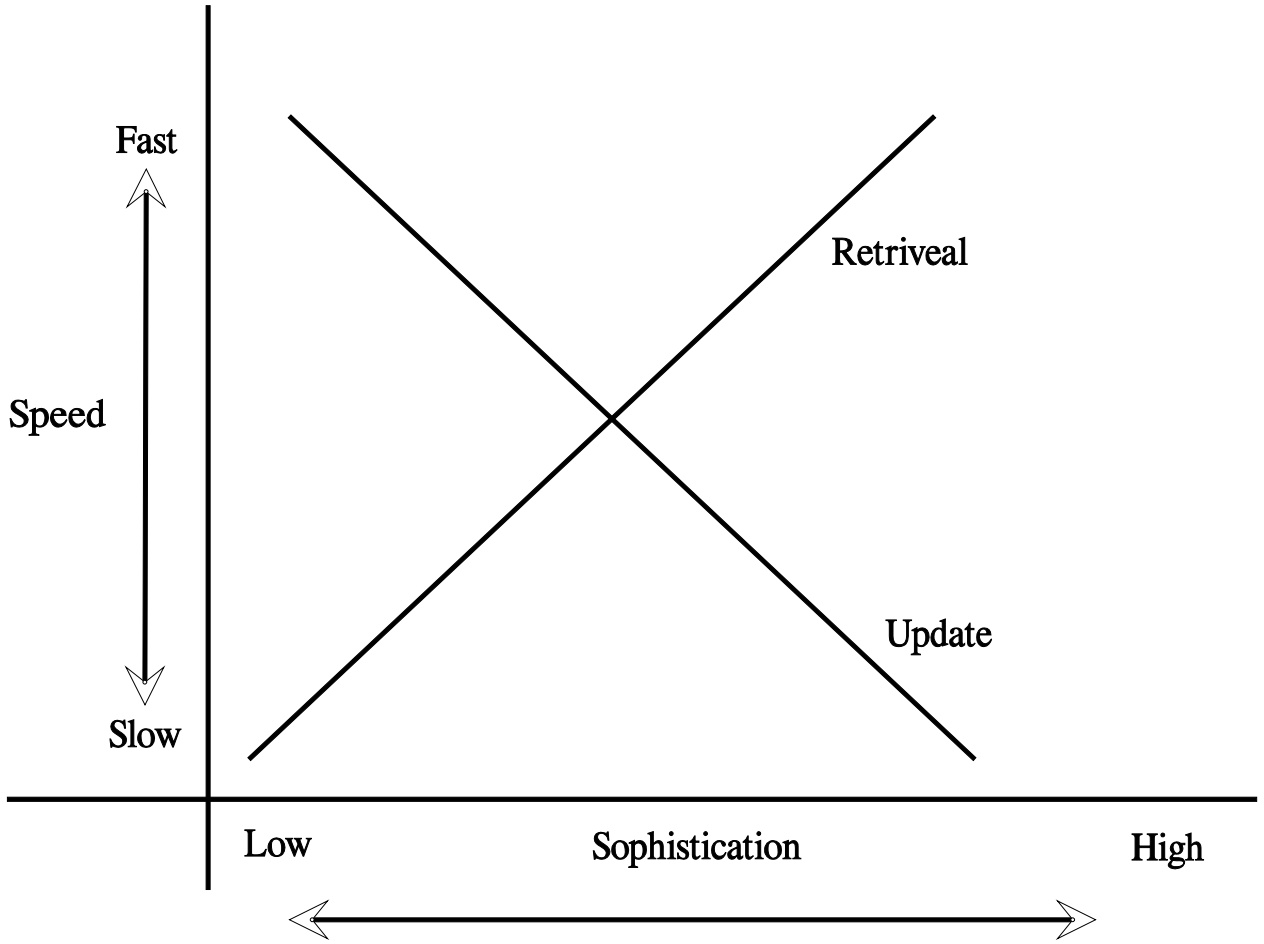
Dictionary	Intelligence
Indexes	Rapid Access Based on Values
Relationships	Rapid Access Based for Member to Owner Owner to Member Member to Member
Data	Storage of Rows

Storage Structure Orientations	Static Dynamic	highly engineered for production update and reporting enables more sophisticated "unknown" queries and reports

7.2 Access Strategy Summary

- DBMS Provided Method of Getting & Putting Data
- Two Basic Orientations
  - Static
    - Production
    - High Volume
    - Low Change
    - HLI Orientation
  - Dynamic
    - Ad-hoc, MIS
    - Low Volume
    - High Change
    - Natural Language
    - Orientation

Access Strategy – Tradeoffs



7.3 Data Loading Summary

- Large Volume Data Loading Strategy
- Data Editing -- must Be Done for Integrity
- Data Validation -- Critical to Have Database
- Data Conversion -- the Essence of Database: Common Semantics
- Data Transformation -- Database Evolution: Not Revolution.
- Phased Loads -- Moving over Gradually
  - Product Line by Product Line
  - Business Unit by Business Unit
  - Year by Year
- Data Loading Strategies -- All at Once, And Go Broke!!

7.4 Data Update Summary

- Critical Issues
  - Extent of Control Exercised by User
    - HLI--high Control over Users
    - POL--moderate Control over Users
    - QUL--no Control over Users
  - Processing (On-line/batch)
- Lockout Granularity
  - Column (Not Common)
  - Logical or Physical Row
  - Static / Dynamic Implications
    - Static -- Typically More Coarse
    - Dynamic-- Typically less Coarse, but As Performance Increases, Granularity Is More Coarse
- Not All Changes Are Simple
- Lockout Has to Be Examined in Detail
- Dictionary Changes Often Lock Database

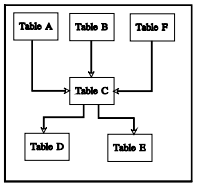
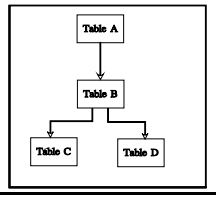
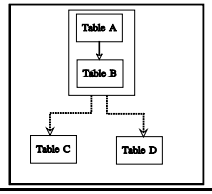
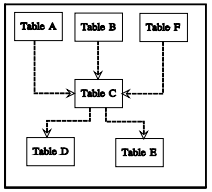
7.5 Database Maintenance Summary

- Understand the archiving process.
- Determine the time and disk resources required
- Know the extent of lockout
- Determine process to recover old archive.
- Can you keep old DBMS copy and still use it in the future for use with old database copies?
- Determine how to know if backups are good

7.6 Static vs Dynamic

Features	DBMS Type	
	Static	**Dynamic**
Integrity	Easier	Harder
Design changes	Harder	Easier
Application type and orientation	Production Centralized Corporate-wide Long-development Long-lifespan	MIS Decentralized Business unit/department Short development Short life span
Most significant features	Relationships via dbms generated pointers Relationships manifest at row load and update Relationship change via row delete and re-store	Relationships via user supplied field values Relationships manifest at row retrieval Relationship changes via field value changes

7.7 DBMS Configuration

Interrow Relationship Mechanism				
Data Model	**Static**		**Dynamic**	
	Network	**Hierarchy**	**Independent Logical File**	**Relational**
Key Data Structures				
Typical systems	Supra IDMS/R IDS DMS-2 DMS-1100 VAX/DBMS	System 2000 IMS	Inquire Adabas Nomad Focus Ramis CA/Datacom Model 204	DB/2 CA/ingress Oracle Sybase Informix SQL/server
Governing ANSI standard	ANSI-NDL			ANSI-SQL/86 ANSI-SQL/89 ANSI-SQL/92
	ANSI SQL:1999