COMP 3400 Mainframe Administration[1]

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

[1]These slides are based in part on materials provided by IBM's Academic Initiative.

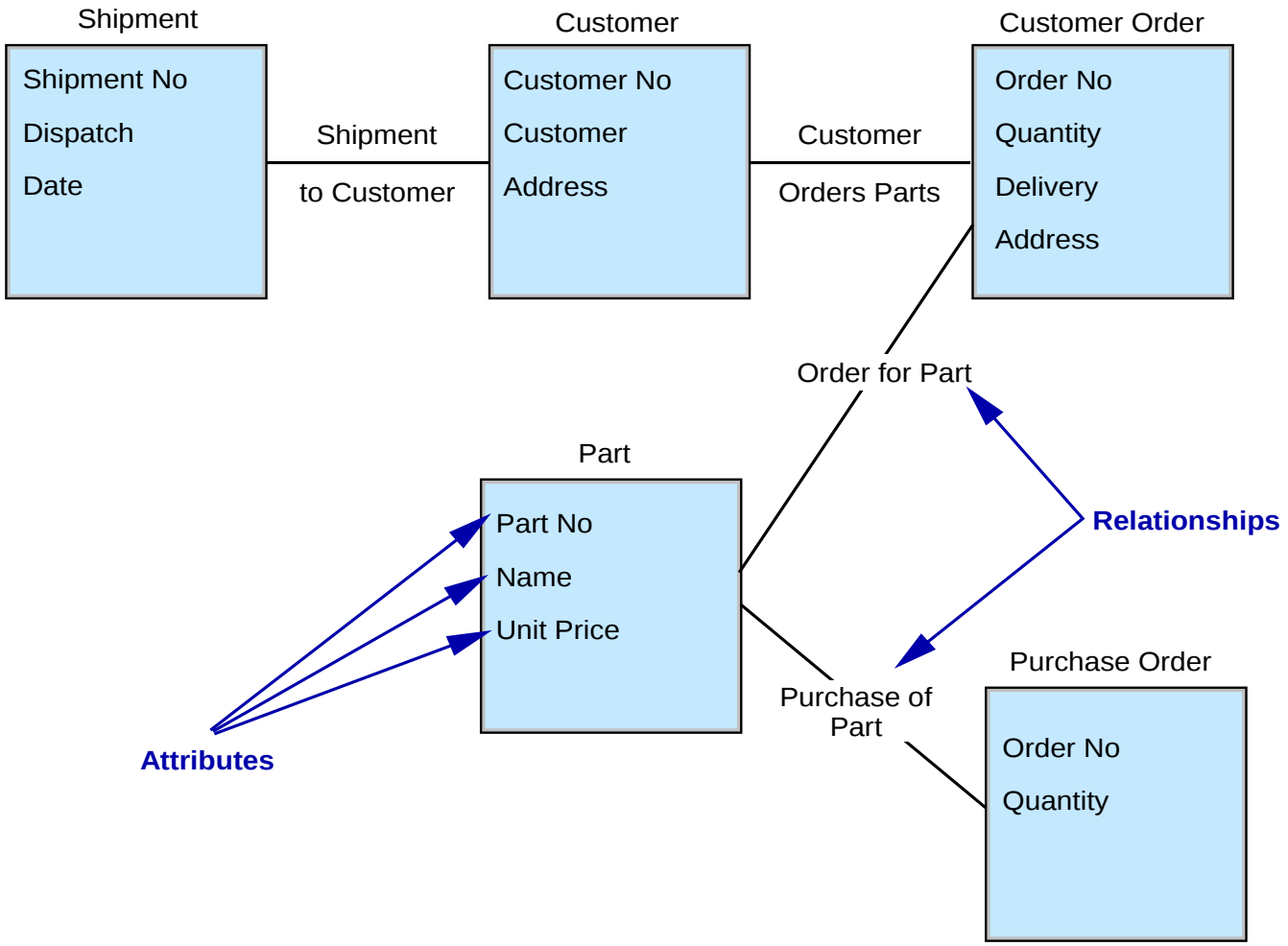
Databases

- Stores (business) information
- Controls access
- Storage is independent of one or more applications
- Supports processing requirements of the applications

Terminology

- Entities – what does the database store information about?
- Attributes – properties of an entity
- Relationships – links between entities

Example


Shipment
Shipment No
Dispatch
Date
Shipment
to Customer
Customer
Customer No
Customer
Address
Customer
Orders Parts
Customer Order
Order No
Quantity
Delivery
Address
Order for Part
Part
Part No
Name
Unit Price
Relationships
Attributes
Purchase of
Part
Purchase Order
Order No
Quantity

Types of Relationships

- One-to-one (rarely used, why?)
- One-to-many (most common)
- Many-to-many

Note: relationships can be recursive.

Reasons for Using Databases

- Reduce programming effort (writing IO, performance analysis and performance tuning)
- Improve security
- Manage concurrency
- Ensure consistency
- Simplify backup and recovery

The Role of the Database Administrator

Provides standards for databases;administers databases

Guides, reviews and approves database designs

Determines rules for accessing data and monitors its security

Controls database integrity & availability; monitors activities for backup and recover

Approves the use of any programs that access production databases

Application functions

An *application function* is the smallest application unit representing a user's interaction with the database:

- Process a single order
- Query inventory status

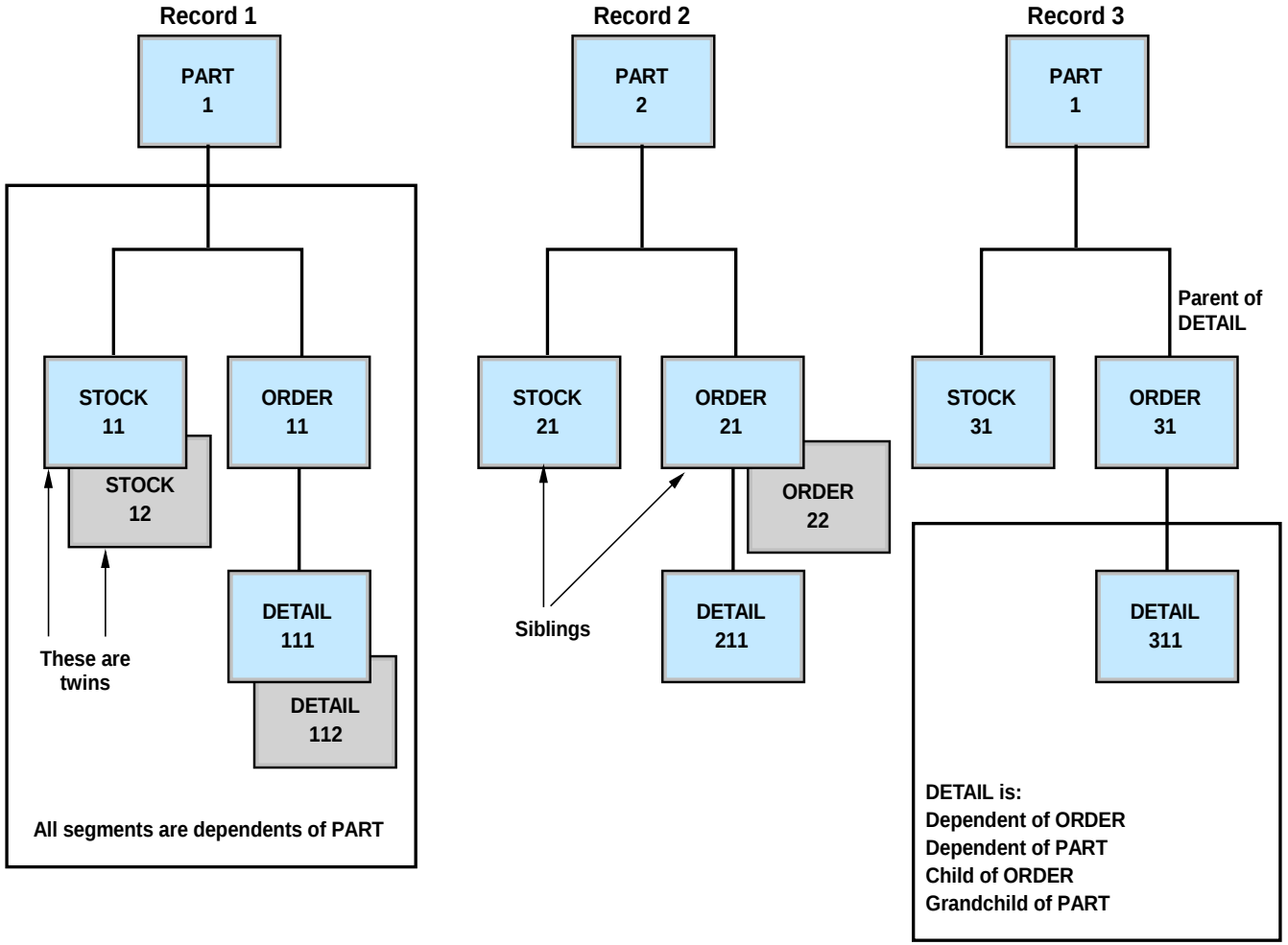
Databases on z/OS

- Hierarchical databases (IMS)
- Relational databases (DB2)

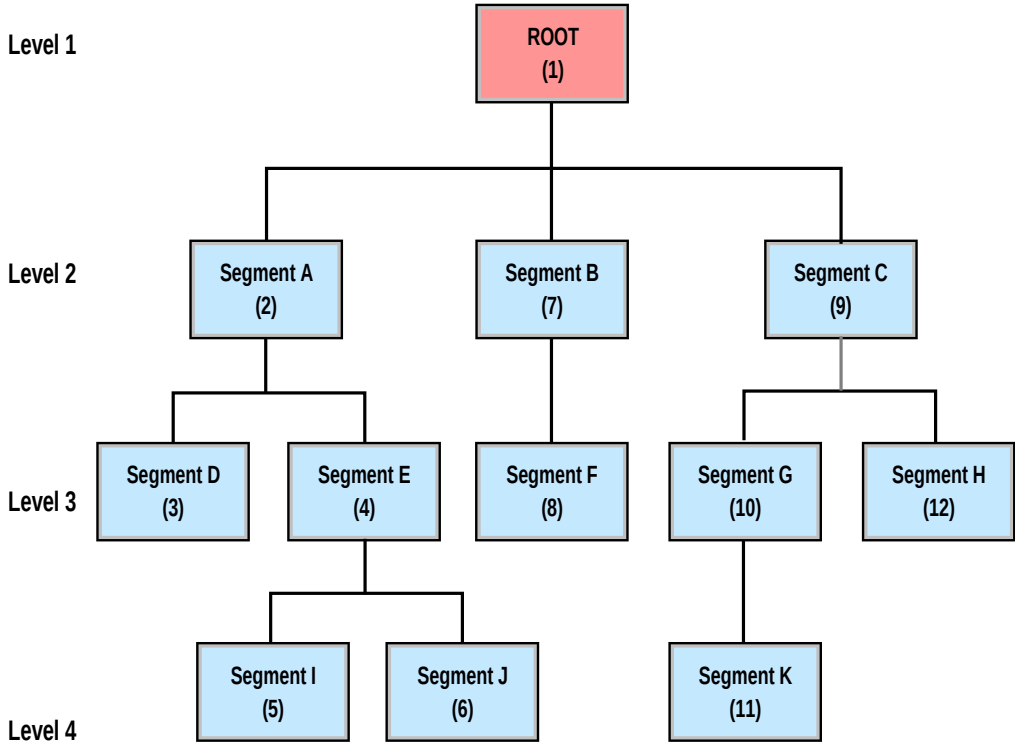
Hierarchical Databases

- Data organized in tree-like structure
- Each parent can have many children
- Good for modeling one-to-many relationships
- Hierarchical data model could be represented in RDBMS using an adjacency list model (but maybe less efficient)
- Nodes in tree may contain different attributes

Hierarchical DB Terminology

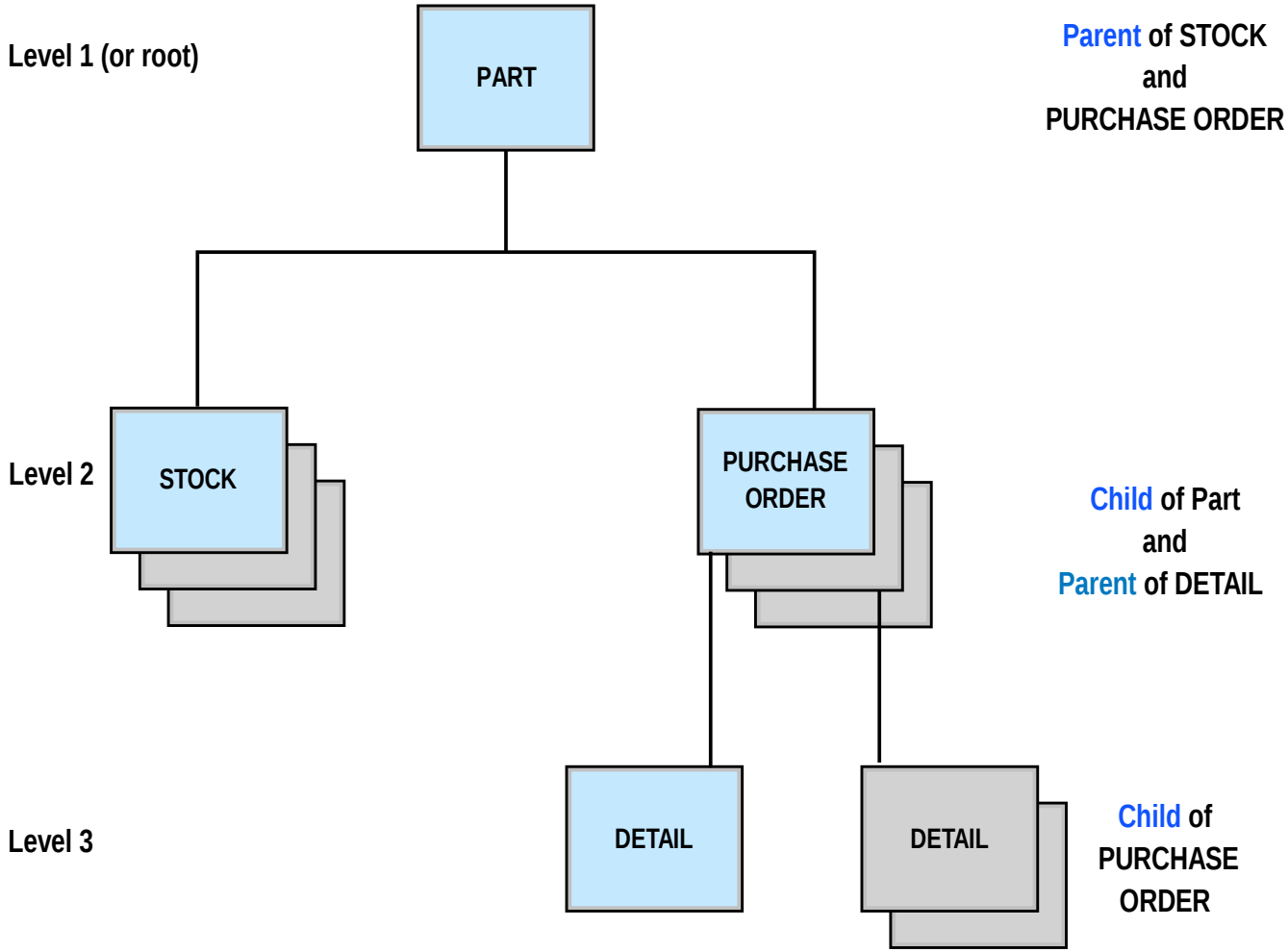


Hierarchical DB Traversal Order



The sequence of traversing the hierarchy is top to bottom, left to right, front to back (for twins).

More Hierarchical DB Terminlogy



Information Management System (IMS)

- IMS/DB is IBM's Hierarchical Database[2]
- Navigational: applications need to know the structure to get to the right data
- Data is relatively static
- IMS/DB "database" is equivalent to a RDBMS table

[2]Note that IMS/TM can also be used as a TM for DB2...

Relational Databases Terminology

Database: logical grouping of data for one or more applications (with tables, accounts and access rules)

Table: Logical structure composed of rows and columns

Index: Ordered set of pointers to rows of a table

Key: Columns identified as keys are used for index creation and/or referential integrity checking

Example of a Table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00
B01	PLANNING	000020	A00
C01	INFORMATION CENTER	000030	A00
D01	DEVELOPMENT CENTER		A00
E01	SUPPORT SERVICES	000050	A00
D11	MANUFACTURING SYSTEMS	000060	D01
D21	ADMINISTRATION SYSTEMS	000070	D01
E11	OPERATIONS	000090	E01
E21	SOFTWARE SUPPORT	000100	E01

At the intersection of every column and row is an data item called an *atomic value*.

Keys

- Primary Key – defines the entity, only one per table (example: student ID)
- Unique Key – alternative (unique) key also used for access (example: SSN)
- Foreign Key – key for a different table, used for referential integrity (example: student ID in book check-out table of library)

Access paths

- The *access paths* of an application function specify identify what entities are used in the operation
- Sequential access can be used if all or most entities are accessed
- Random access is used if only a few particular entities are accessed
- For efficient random access, the access path should utilize the entity's key (for which hopefully the appropriate index exists)

Rules of Data Normalization

1NF Eliminate Repeating Groups: Make a separate table for each set of related attributes, and give each table a primary key.

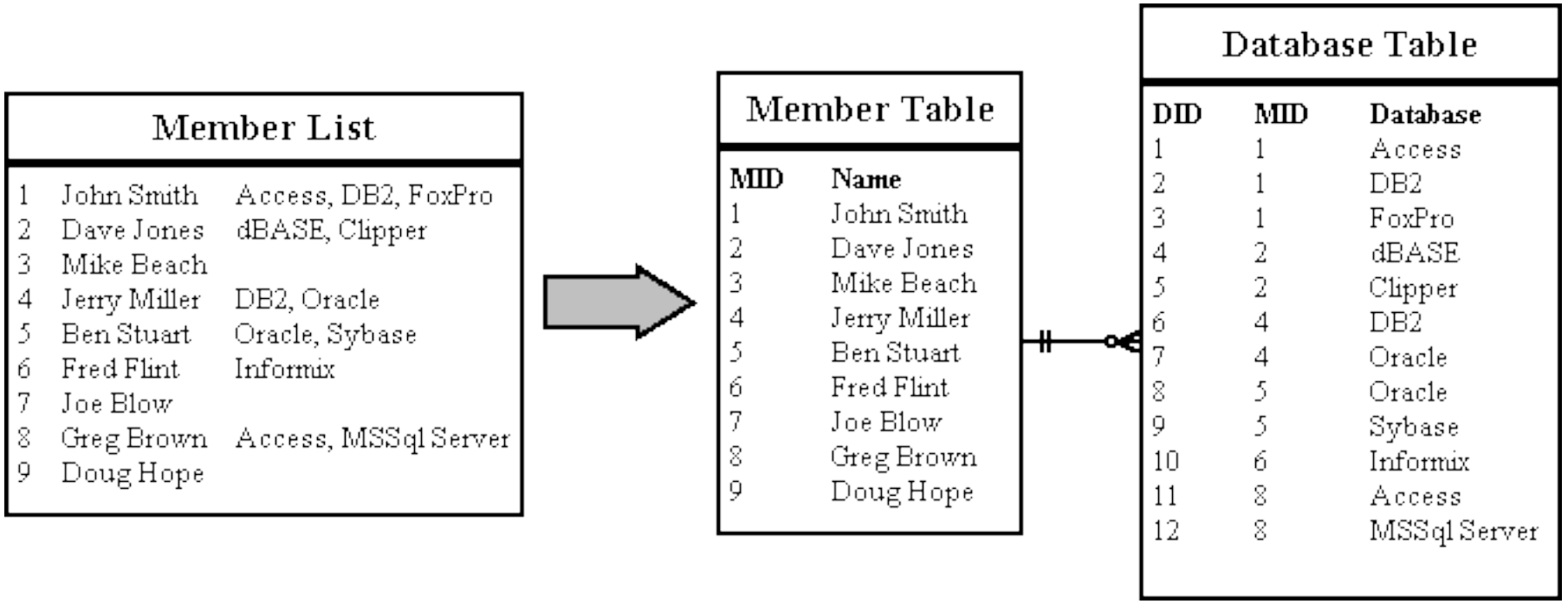
2NF Eliminate Redundant Data: If an attribute depends on only part of a multi-valued key, move it to a separate table.

3NF Eliminate Columns Not Dependent On a Key: If attributes do not contribute to a description of the key, move them to a separate table.

4NF Isolate Independent Multiple Relationships: No table may contain two or more $1 : n$ or $n : m$ relationships that are not directly related.

5NF Isolate Semantically Related Multiple Relationships

Example: 1NF



Member List

1	John Smith	Access, DB2, FoxPro
2	Dave Jones	dBASE, Clipper
3	Mike Beach	
4	Jerry Miller	DB2, Oracle
5	Ben Stuart	Oracle, Sybase
6	Fred Flint	Informix
7	Joe Blow	
8	Greg Brown	Access, MSSql Server
9	Doug Hope	

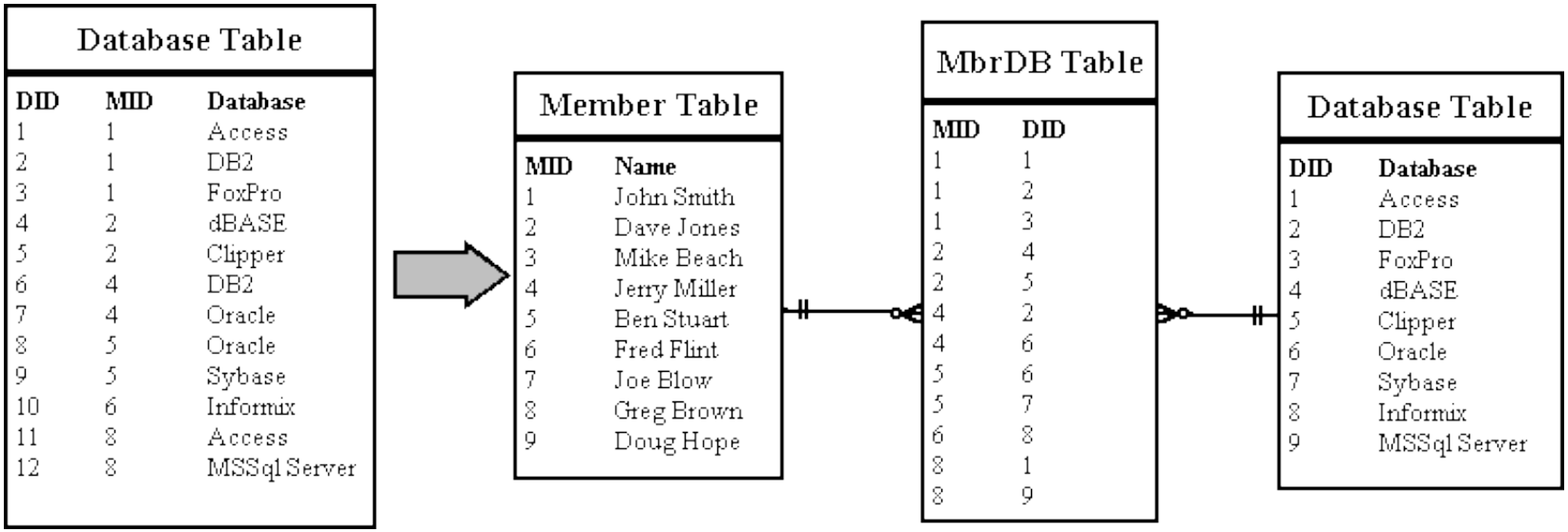
Member Table

MID	Name
1	John Smith
2	Dave Jones
3	Mike Beach
4	Jerry Miller
5	Ben Stuart
6	Fred Flint
7	Joe Blow
8	Greg Brown
9	Doug Hope

Database Table

DID	MID	Database
1	1	Access
2	1	DB2
3	1	FoxPro
4	2	dBASE
5	2	Clipper
6	4	DB2
7	4	Oracle
8	5	Oracle
9	5	Sybase
10	6	Informix
11	8	Access
12	8	MSSql Server

Example: 2NF



Database Table

DID	MID	Database
1	1	Access
2	1	DB2
3	1	FoxPro
4	2	dBASE
5	2	Clipper
6	4	DB2
7	4	Oracle
8	5	Oracle
9	5	Sybase
10	6	Informix
11	8	Access
12	8	MSSql Server

Member Table

MID	Name
1	John Smith
2	Dave Jones
3	Mike Beach
4	Jerry Miller
5	Ben Stuart
6	Fred Flint
7	Joe Blow
8	Greg Brown
9	Doug Hope

MbrDB Table

MID	DID
1	1
1	2
1	3
2	4
2	5
4	2
4	6
5	6
5	7
6	8
8	1
8	9

Database Table

DID	Database
1	Access
2	DB2
3	FoxPro
4	dBASE
5	Clipper
6	Oracle
7	Sybase
8	Informix
9	MSSql Server

Example: 3NF

Member Table

MID	Name	Company	CompLoc
1	John Smith	ABC	Alabama
2	Dave Jones	MCI	Florida
3	Mike Beach	IBM	Delaware
4	Jerry Miller	MCI	Florida
5	Ben Stuart	AIC	Nebraska
6	Fred Flint	ABC	Alabama
7	Joe Blow	RU Nuts	Iowa
8	Greg Brown	XYZ	New York
9	Doug Hope	IBM	Delaware

Member Table

MID	Name	CID
1	John Smith	1
2	Dave Jones	2
3	Mike Beach	3
4	Jerry Miller	2
5	Ben Stuart	4
6	Fred Flint	1
7	Joe Blow	5
8	Greg Brown	6
9	Doug Hope	3

Company Table

CID	Name	Location
1	ABC	Alabama
2	MCI	Florida
3	IBM	Delaware
4	AIC	Nebraska
5	RU Nuts	Iowa
6	XYZ	New York

Data Structures in DB2

- Table spaces — z/OS data sets holding tables with data
- Index spaces — z/OS data sets holding index information
- Storage Groups
- Views

DB2 Hierarchy Structure

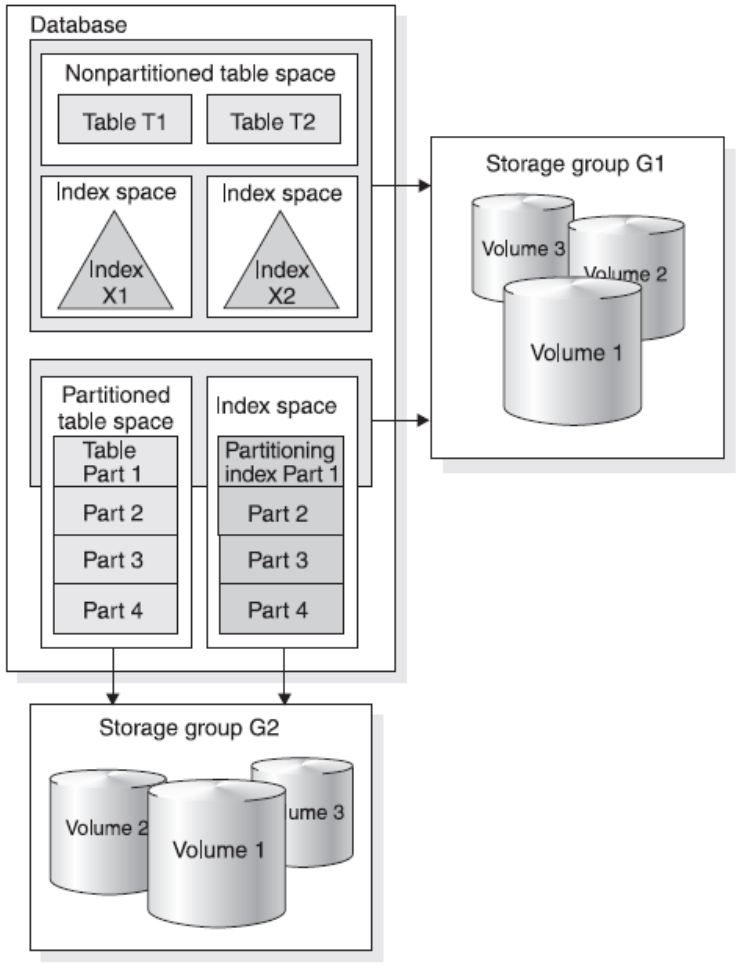


Table Space Choices

Partitioned Divide storage into partitions, one for each table

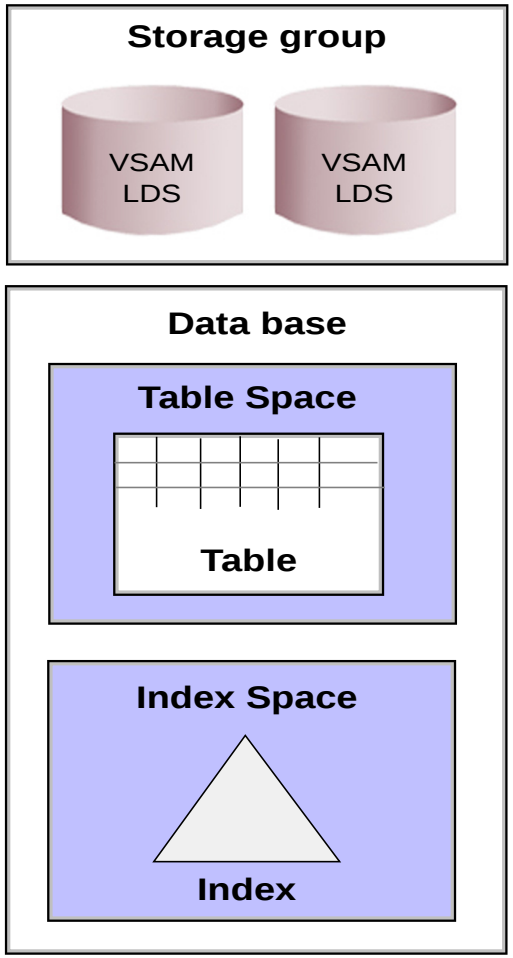
Segmented Divide storage into segments of equal size; each segment contains rows from the same table

Simple Data from multiple tables is not separated

Large Object Separate large objects (graphics, video)

These choices can be used to improve performance and/or administration flexibility.

Views


Storage group
VSAM LDS
VSAM LDS
Data base
Table Space
Table
Index Space
Index

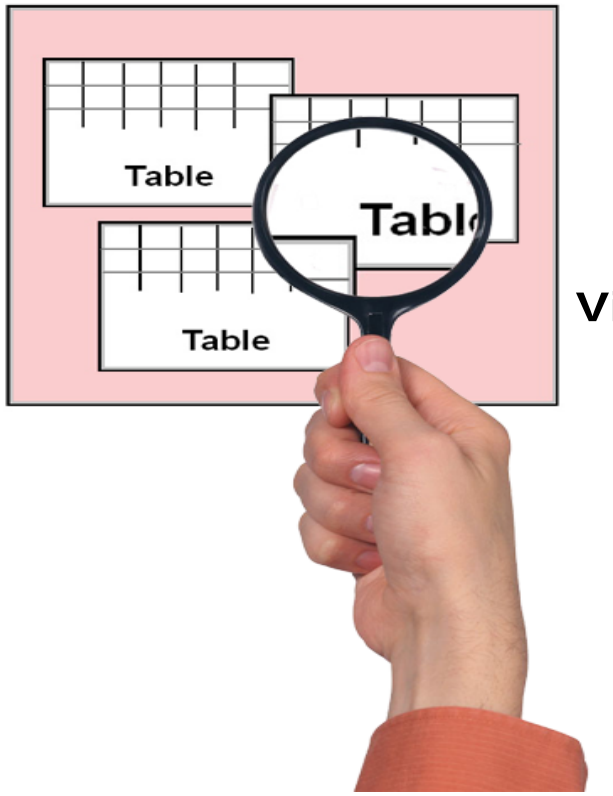


Table
Table
Table

Views

Schema Structures

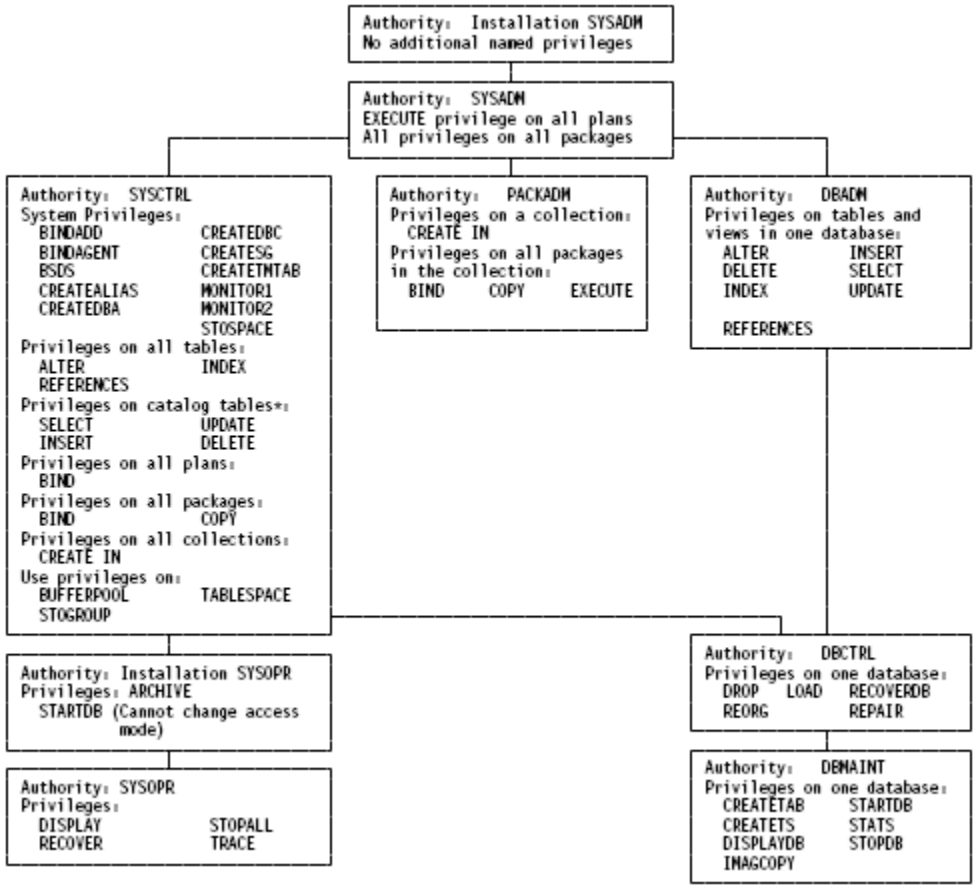
DB2 supports:

- User-defined Data Types (UDTs) – like EUROs or YEN; UDTs must be based on the existing DB2 data types
- User-defined Functions (UDFs) – more complex arithmetic then what is provided by DB2
- Triggers – actions executed when a specific operation occurs
- Large Objects (LOBs) – large data stored in special auxiliary tables
- Stored procedures – user-written application program stored and run on the DB2 server

DB2 System Structures

- The DB2 *Catalog* keeps information about tables, views, indexes, table spaces, etc.; you can query the catalog using SQL
- The DB2 *Directory* keeps information about application programs (information about plans and packages, open/closed table spaces and indices, meta-information about databases, etc.); you can **not** query the directory using SQL
- Buffer pools – virtual storage for DB2 caching
- Active and archive logs – to support recovery or rollback in case of failures

Administrative Authorities within DB2



Responsibilities: SYSADM

- Installation
- System Object Management
- System and Disaster Recovery
- Monitoring System Performance

$\Rightarrow$ has all priviledges for the entire DB2 subsystem!

Responsibilities: DBADM

- Creation & Management of DB2 Objects for a particular DB2 database
- Data (re-)organization
- Backup & recovery
- Data consistency

$\Rightarrow$ can execute most utility commands

DB2 Address Spaces

DB2 is a multi-address space subsystem requiring at least three address spaces:

- System Services
- Database Services
- Lock Manager Services (IRLM)

The Distributed Data Facility (DDF) is an AS used for distributing DB2 across systems.

DB2 Attachment Facilities

An *Attachment Facility* is a way to connect to / communicate with DB2. DB2 supports the following attachment facilities:

- CICS Attachment facility (CA)
- Call Attachment facility (CAF)
- IMS Attachment facility (IA)
- TSO Attachment facility (TA)

Attachment interfaces only attach to a DB2 subsystem running on the same MVS system as the application!

Important DB2 Utilities

- LOAD – populate tables
- UNLOAD – move or copy data
- REORG – change order of data (based on performance data from RUNSTATS or EXPLAIN)
- COPY – backup data
- RECOVER – recover data from backup
- MERGECOPY – merge incremental copies with full copy
- REBUILD INDEX – recover indexes
- CHECK – validate data consistency

DB2 Administrative Interfaces

- DB2I – interactive panel to enter DB2 commands, which includes SPUFI
- SQL Processor Using File Input (SPUFI[3]) – SQL interface through TSO/ISPF providing transactional facility of DBAs
- Query Management Facility (QMF) – integrated tool for performing queries and gathering reports (no TSO/ISPF/PDS knowledge required)

[3] Pronounced "spoo fee"

Locating SPUFI

```
DSNEPRI                    DB2I PRIMARY OPTION MENU          SSID: DSN
COMMAND ===> 1

Select one of the following DB2 functions and press ENTER.

 1  SPUFI                 (Process SQL statements)
 2  DCLGEN                (Generate SQL and source language declarations)
 3  PROGRAM PREPARATION   (Prepare a DB2 application program to run)
 4  PRECOMPILE            (Invoke DB2 precompiler)
 5  BIND/REBIND/FREE      (BIND, REBIND, or FREE plans or packages)
 6  RUN                   (RUN an SQL program)
 7  DB2 COMMANDS          (Issue DB2 commands)
 8  UTILITIES             (Invoke DB2 utilities)
 D  DB2I DEFAULTS         (Set global parameters)
 X  EXIT                  (Leave DB2I)



PRESS:  END to exit        HELP for more information
```

Using SPUFI (Allocate Output Dataset)

```
                          SPUFI                                SSID: DB8H
===>

Enter the input data set name:        (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> 'ZPROF.SPUFI.CNTL(dept)'
 2  VOLUME SERIAL ... ===>             (Enter if not cataloged)
 3  DATA SET PASSWORD ===>             (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
 4  DATA SET NAME ... ===> 'ZPROF.SPUFI.OUTPUT'

Specify processing options:
 5  CHANGE DEFAULTS   ===> NO          (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>

 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

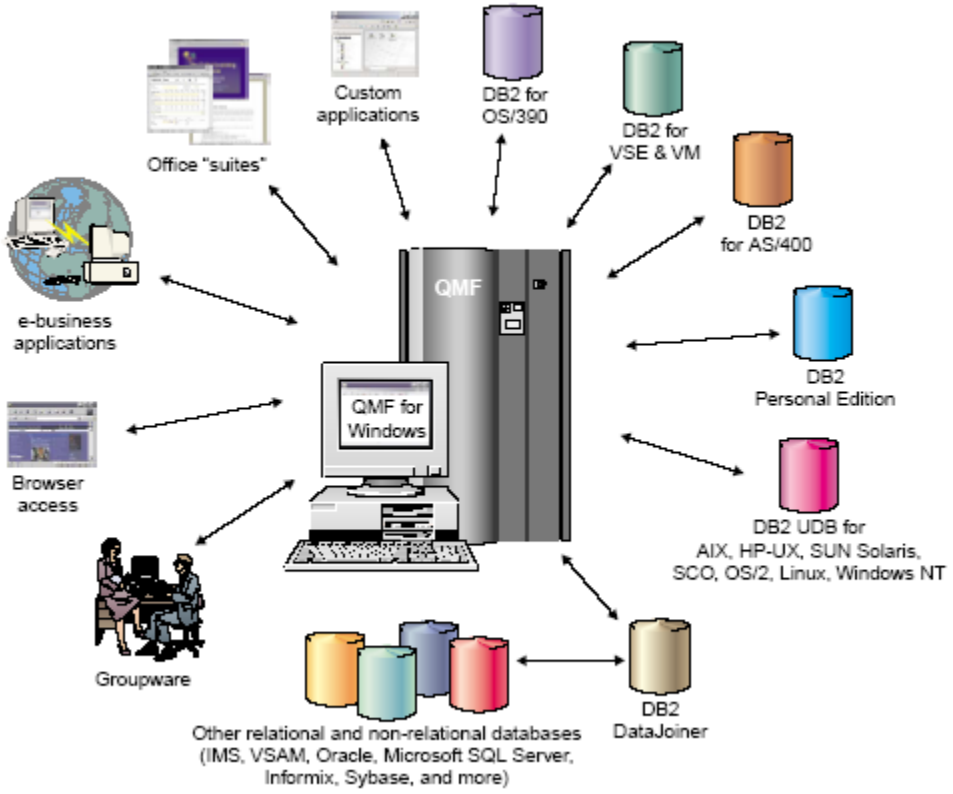
Using SPUFI (EDIT panel)

```
EDIT --------userid.EXAMPLES(XMP1) --------------------- COLUMNS 001 072
COMMAND INPUT ---> SAVE                          SCROLL ---> PAGE
******************************** TOP OF DATA ***********************
000100 SELECT LASTNAME, FIRSTNME, PHONENO
000200   FROM DSN8510.EMP
000300   WHERE WORKDEPT= 'D11'
000400   ORDER BY LASTNAME;
******************************** BOTTOM OF DATA ********************
```

Using SPUFI (Result Dataset)

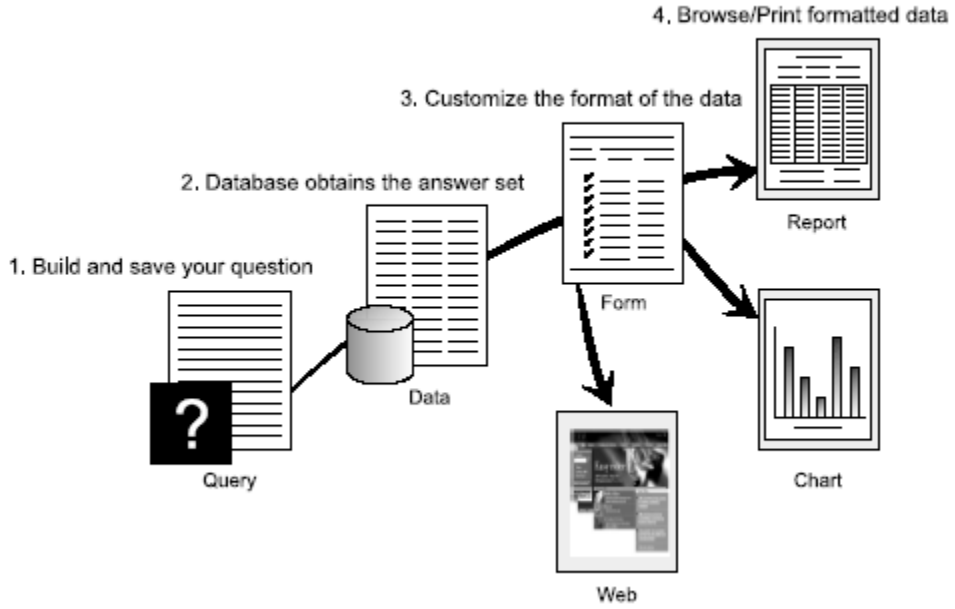
```
BROWSE-- userid.RESULT                        COLUMNS 001 072
COMMAND INPUT ===>                                 SCROLL ===> PAGE
--------+---------+---------+---------+---------+---------+---------+---------+
SELECT LASTNAME, FIRSTNME, PHONENO                                    00010000
 FROM DSN8510.EMP                                                     00020000
 WHERE WORKDEPT = 'D11'                                               00030000
 ORDER BY LASTNAME;                                                   00040000
---------+---------+---------+---------+---------+---------+---------+---------+
LASTNAME         FIRSTNME     PHONENO
ADAMSON          BRUCE        4510
BROWN            DAVID        4501
JOHN             REBA         0672
JONES            WILLIAM      0942
LUTZ             JENNIFER     0672
PIANKA           ELIZABETH    3782
SCOUTTEN         MARILYN      1682
STERN            IRVING       6423
WALKER           JAMES        2986
YAMAMOTO         KIYOSHI      2890
YOSHIMURA        MASATOSHI    2890
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+----
---------+---------+---------+---------+---------+---------+----
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 4
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 30
```

The Query Management Facility (QMF)



QMF Usage

QMF provides results in 4 easy steps:



Structured Query Language (SQL)

SQL is a high level language for processing relational structures and is the only way to access data in DB2 databases.

SQL functions fall into three sublanguages / categories:

- Data manipulation language (DML): SELECT, UPDATE, INSERT, DELETE
- Data definition language (DDL): CREATE, ALTER, DROP
- Data control language (DCL): GRANT, REVOKE

CREATE

```
CREATE TABLE smith.tempdept
  (deptno    CHAR(3)      NOT NULL,
   deptname VARCHAR(36) NOT NULL,
   mgrno     CHAR(6)      NOT NULL,
   admrdept CHAR(3)      NOT NULL,
   PRIMARY KEY (deptno)           )
  IN DSN8D81A.DSN8S81D;
```

INSERT

```
INSERT INTO smith.tempdept
  VALUES ('X05', 'EDUCATION', '000631', 'A01');
```

CREATE INDEX

```
CREATE INDEX IX1 ON ACCOUNTS(ACCTID);
CREATE INDEX IX2 ON ACCOUNTS(NAME);
```

SELECT

```
SELECT deptname FROM DSN8610.VDEPT
  WHERE deptno = 'X42';
```

CREATE VIEW

```
CREATE VIEW DSN8610.VDEPT
  AS SELECT ALL deptno, deptname, mgrno
  FROM DSN8610.DEPT;
```

EXPLAIN

- Command used to determine the access path used for a SQL statement
- Used to analyze the performance of SELECT, UPDATE, INSERT and DELETE statements
- The query is not executed!
- The access path is placed in `userid.PLAN_TABLE` (if it exists)

Example: EXPLAIN

```
EXPLAIN ALL SET QUERYNO = 1
  SELECT empno, lastname
  FROM emp
  WHERE lastname = 'Miller';
```

Statement Types

- Static (SQL) statements: access path known at compile time; values supplied dynamically

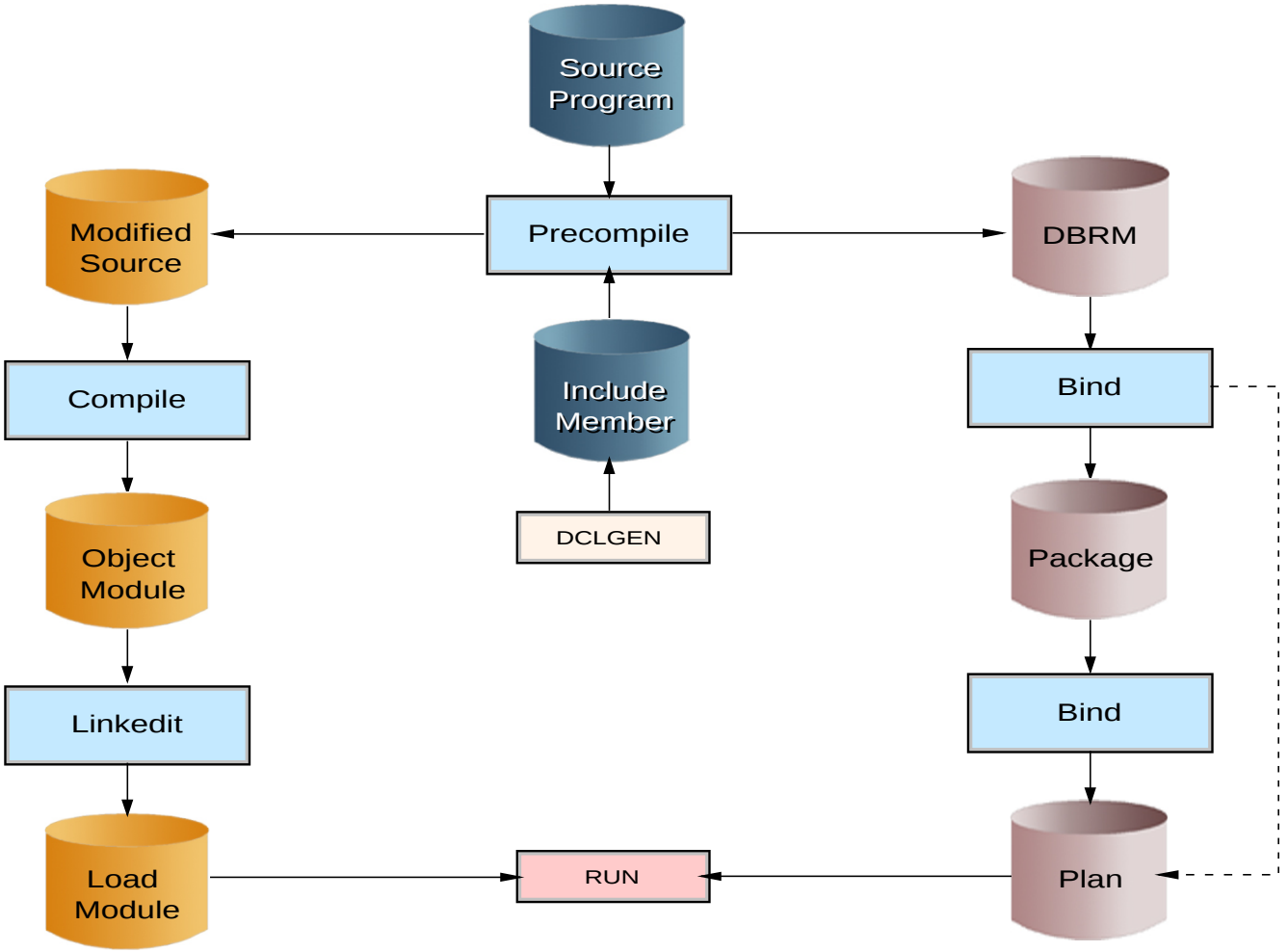
⇒ Bind process determines access path and stores executable SQL code in a package

- Dynamic (SQL) statements: entire logic only known at run-time

⇒ SQL statement interpreted at run-time, can be significantly slower

⇒ Used by management GUIs for administrative operations that run only once (CREATE, ALTER, DROP, GRANT, REVOKE)

DB2 Application Programming



DCLGEN

- Generates source definitions for DB2 objects
- Requires running DB2 database with existing tables, views, etc.
- Optional (you could write the source definitions manually as well)
- Usually run by DBA

PRECOMPILE

- Identifies SQL statements and replaces them with a CALL to DB2, passing host variable addresses, statement numbers and a *consistency token*
- SQL statements identified syntactically:

```
EXEC SQL
  SELECT empno, lastname
  INTO :number, :name
  FROM emp
END-EXEC.
```

- Precompile also generates a database request module (DBRM) for BIND

BIND

- BIND takes DBRMs to create $packages$
- BIND checks for syntax errors, authorization and determines access paths using a cost-based optimizer
- The resulting package is stored in the DB2 directory

Plans

- Contains all packages of one project – every run uses the same plan
- Plans can also store subroutines
- Plans are stored in the DB2 directory

Cursors

- Many SQL statements have more than one row in the result set
- A *cursor* allows you to fetch, update or delete one row at a time

Questions

?

Exercise!

See textbook Section 12.14 on page 415.