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

- **Customer**
  - Customer No
  - Customer Address

- **Customer Order**
  - Order No
  - Quantity
  - Delivery Address

- **Part**
  - Part No
  - Name
  - Unit Price

- **Relationships**
  - Shipment to Customer
  - Customer Orders Parts
  - Order for Part
  - Purchase of Part
  - Purchase Order

- **Attributes**
  - Shipment No
  - Dispatch
  - Date
  - Customer No
  - Customer Address
  - Order No
  - Quantity
  - Delivery Address
  - Part No
  - Name
  - Unit Price
  - 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
- Determines rules for accessing data and monitors its security
- Approves the use of any programs that access production databases
- Guides, reviews and approves database designs
- Controls database integrity & availability; monitors activities for backup and recover
- DBA
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

These are twins

All segments are dependents of PART

DETAIL is:
- Dependent of ORDER
- Dependent of PART
- Child of ORDER
- Grandchild of PART
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 Terminology

Level 1 (or root)

PART

Parent of STOCK and PURCHASE ORDER

Level 2

STOCK

Child of Part and Parent of DETAIL

PURCHASE ORDER

Level 3

DETAIL

Child of PURCHASE ORDER

DETAIL
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

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

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 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**

<table>
<thead>
<tr>
<th>Member List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 John Smith, Access, DB2, FoxPro</td>
</tr>
<tr>
<td>2 Dave Jones, dBASE, Clipper</td>
</tr>
<tr>
<td>3 Mike Beach</td>
</tr>
<tr>
<td>4 Jerry Miller, DB2, Oracle</td>
</tr>
<tr>
<td>5 Ben Stuart, Oracle, Sybase</td>
</tr>
<tr>
<td>6 Fred Flint, Informix</td>
</tr>
<tr>
<td>7 Joe Blow</td>
</tr>
<tr>
<td>8 Greg Brown, Access, MSSql Server</td>
</tr>
<tr>
<td>9 Doug Hope</td>
</tr>
</tbody>
</table>

**Member Table**

<table>
<thead>
<tr>
<th>MID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Smith</td>
</tr>
<tr>
<td>2</td>
<td>Dave Jones</td>
</tr>
<tr>
<td>3</td>
<td>Mike Beach</td>
</tr>
<tr>
<td>4</td>
<td>Jerry Miller</td>
</tr>
<tr>
<td>5</td>
<td>Ben Stuart</td>
</tr>
<tr>
<td>6</td>
<td>Fred Flint</td>
</tr>
<tr>
<td>7</td>
<td>Joe Blow</td>
</tr>
<tr>
<td>8</td>
<td>Greg Brown</td>
</tr>
<tr>
<td>9</td>
<td>Doug Hope</td>
</tr>
</tbody>
</table>

**Database Table**

<table>
<thead>
<tr>
<th>DID</th>
<th>MID</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Access</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>DB2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>FoxPro</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>dBASE</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>Clipper</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>DB2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Oracle</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>Oracle</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>Sybase</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>Informix</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>Access</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>MSSql Server</td>
</tr>
</tbody>
</table>
Example: 2NF

Database Table

<table>
<thead>
<tr>
<th>DID</th>
<th>MID</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Access</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>DB2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>FoxPro</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>dBASE</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>Clipper</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>DB2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Oracle</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>Oracle</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>Sybase</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>Informix</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>Access</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>MSSqlServer</td>
</tr>
</tbody>
</table>

Member Table

<table>
<thead>
<tr>
<th>MID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Smith</td>
</tr>
<tr>
<td>2</td>
<td>Dave Jones</td>
</tr>
<tr>
<td>3</td>
<td>Mike Beach</td>
</tr>
<tr>
<td>4</td>
<td>Jerry Miller</td>
</tr>
<tr>
<td>5</td>
<td>Ben Stuart</td>
</tr>
<tr>
<td>6</td>
<td>Fred Flint</td>
</tr>
<tr>
<td>7</td>
<td>Joe Blow</td>
</tr>
<tr>
<td>8</td>
<td>Greg Brown</td>
</tr>
<tr>
<td>9</td>
<td>Doug Hope</td>
</tr>
</tbody>
</table>

MbrDB Table

<table>
<thead>
<tr>
<th>MID</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Database Table

<table>
<thead>
<tr>
<th>DID</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access</td>
</tr>
<tr>
<td>2</td>
<td>DB2</td>
</tr>
<tr>
<td>3</td>
<td>FoxPro</td>
</tr>
<tr>
<td>4</td>
<td>dBASE</td>
</tr>
<tr>
<td>5</td>
<td>Clipper</td>
</tr>
<tr>
<td>6</td>
<td>Oracle</td>
</tr>
<tr>
<td>7</td>
<td>Sybase</td>
</tr>
<tr>
<td>8</td>
<td>Informix</td>
</tr>
<tr>
<td>9</td>
<td>MSSqlServer</td>
</tr>
</tbody>
</table>
**Example: 3NF**

<table>
<thead>
<tr>
<th>Member Table</th>
<th>Member Table</th>
<th>Company Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MID</strong></td>
<td><strong>Name</strong></td>
<td><strong>Company</strong></td>
</tr>
<tr>
<td>1</td>
<td>John Smith</td>
<td>ABC</td>
</tr>
<tr>
<td>2</td>
<td>Dave Jones</td>
<td>MCI</td>
</tr>
<tr>
<td>3</td>
<td>Mike Beach</td>
<td>IBM</td>
</tr>
<tr>
<td>4</td>
<td>Jerry Miller</td>
<td>MCI</td>
</tr>
<tr>
<td>5</td>
<td>Ben Stuart</td>
<td>AIC</td>
</tr>
<tr>
<td>6</td>
<td>Fred Flint</td>
<td>ABC</td>
</tr>
<tr>
<td>7</td>
<td>Joe Blow</td>
<td>RU Nuts</td>
</tr>
<tr>
<td>8</td>
<td>Greg Brown</td>
<td>XYZ</td>
</tr>
<tr>
<td>9</td>
<td>Doug Hope</td>
<td>IBM</td>
</tr>
</tbody>
</table>
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

Diagram showing the hierarchy of a database in DB2, with different types of table spaces and index spaces.
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
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 than 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

- **Authority: Installation SYSADM**
  - No additional named privileges

- **Authority: SYSADM**
  - EXECUTE privilege on all plans
  - All privileges on all packages

- **Authority: SYSCTRL**
  - System Privileges:
    - EINDADD
    - EINDAGENT
    - EINDERROR
    - EINDACTIONS
    - EINDDBA
  - Privileges on all tables:
    - ALTER
    - REFERENCES
  - Privileges on catalog tables:
    - SELECT
    - INSERT
    - DELETE
  - Privileges on all plans:
    - EIND
  - Privileges on all packages:
    - EIND
  - Privileges on all collections:
    - CREATE IN
  - Use privileges on:
    - DBCOPY
    - TABLESPACE
  - STOPGROUP

- **Authority: PACKADM**
  - Privileges on a collection:
    - CREATE IN
  - Privileges on all packages in the collection:
    - BIND
    - COPY
    - EXECUTE

- **Authority: DRACOM**
  - Privileges on tables and views in one database:
    - ALTTAB
    - INSERT
    - DELETE
    - SELECT
    - INDEX
    - UPDATE
    - REFERENCES

- **Authority: DBCTRL**
  - Privileges on one database:
    - DROP
    - LINK
    - RECOVER
    - REPAIR

- **Authority: DMOADM**
  - Privileges on one database:
    - CREATE
    - STARTDB
    - CREATESYS
    - DISPLAYDB
    - STOPDB
    - DBCOPY

- **Authority: SYSOPR**
  - Privileges:
    - DISPLAY
    - RECOVER
    - STOPIX
    - TRACE
Responsibilities: SYSADM

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

⇒ has all privileges for the entire DB2 subsystem!
Responsibilities: DBADM

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

⇒ 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)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter the input data set name: (Can be sequential or partitioned)</td>
</tr>
<tr>
<td>2</td>
<td>Enter the output data set name: (Must be a sequential data set)</td>
</tr>
<tr>
<td>3</td>
<td>Specify processing options:</td>
</tr>
<tr>
<td>4</td>
<td>For remote SQL processing:</td>
</tr>
</tbody>
</table>

### Processing Options:

- **CHANGE DEFAULTS**: Yes/No (Y/N - Display SPUFI defaults panel?)
- **EDIT INPUT**: Yes/No (Y/N - Enter SQL statements?)
- **EXECUTE**: Yes (Y/N - Execute SQL statements?)
- **AUTOCOMMIT**: Yes (Y/N - Commit after successful run?)
- **BROWSE OUTPUT**: Yes (Y/N - Browse output data set?)

### Connect Location:

- **F1=HELP**
- **F2=SPIT**
- **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, FIRSTNAME, PHONENO
000200 FROM DSN8S10.EMP
000300 WHERE WORKDEPT= 'DI1'
000400 ORDER BY LASTNAME;
****************************************************************************
BOTTOM OF DATA ****************************
Using SPUFI (Result Dataset)

```sql
BROWSE -- userid.RESULT

SELECT LASTNAME, FIRSTNAME, PHONENO
FROM SQA0516.EMP
WHERE WORKDEPT = 'D11'
ORDER BY LASTNAME;
```

<table>
<thead>
<tr>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>PHONENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAMSON</td>
<td>BRUCE</td>
<td>4610</td>
</tr>
<tr>
<td>BROWN</td>
<td>DAVID</td>
<td>4501</td>
</tr>
<tr>
<td>JOHN</td>
<td>REGI</td>
<td>0672</td>
</tr>
<tr>
<td>JONES</td>
<td>WILLIAM</td>
<td>0642</td>
</tr>
<tr>
<td>LUTZ</td>
<td>JENNIFER</td>
<td>0672</td>
</tr>
<tr>
<td>PIANKA</td>
<td>ELIZABETH</td>
<td>3782</td>
</tr>
<tr>
<td>SCOUTEN</td>
<td>MARILYN</td>
<td>1682</td>
</tr>
<tr>
<td>STEIN</td>
<td>IRVING</td>
<td>6423</td>
</tr>
<tr>
<td>WALKER</td>
<td>JAMES</td>
<td>2986</td>
</tr>
<tr>
<td>YAMAMOTO</td>
<td>KIYOSHI</td>
<td>2890</td>
</tr>
<tr>
<td>YOSHIMURA</td>
<td>MASATOSHI</td>
<td>2890</td>
</tr>
</tbody>
</table>

DSNE6101: NUMBER OF ROWS DISPLAYED IS 11
DSNE6161: STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

-----------

DSNE6171: COMMIT PERFORMED, SQLCODE IS 0
DSNE6161: STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

-----------

DSNE6011: SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE6201: NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE6211: NUMBER OF INPUT RECORDS READ IS 4
DSNE6221: NUMBER OF OUTPUT RECORDS WRITTEN IS 30
The Query Management Facility (QMF)
QMF Usage

QMF provides results in 4 easy steps:

1. Build and save your question
2. Database obtains the answer set
3. Customize the format of the data
4. Browse/Print formatted data

Diagram:
- Query
- Data
- Form
- Report
- Chart
- Web
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 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
deppname 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

1. Source Program
2. Precompile
3. Include Member
4. Modified Source
5. Compile
6. Object Module
7. Linkedit
8. Load Module
9. DBRM
10. Bind
11. Package
12. Bind
13. Plan
14. DCLGEN
15. RUN
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.