

Objectives

- Explain the three primary DB models and their differences
- Explore the major features of database management systems
- Show the difference between logical and physical files
- Explain the relationship between logical and physical files
- Use DDS to create simple and join logical files

What Is a Database?

- A database is made up of related “groups of data”
- A DBMS is a collection of programs that allows the user to create, maintain and relate these groups of data
- There are three types of databases
 - ◆ Hierarchical
 - ◆ Network
 - ◆ Relational

Data Relationships

- Number of relationships a record has with records from another group
 - ◆ One-to-one (1:1)
 - ◆ One-to-many (1:M)
 - ◆ Many-to-many (M:M)
- Direction of access required
 - ◆ Unidirectional
 - ◆ Bidirectional

Hierarchical Database

- Can only support a parent-child relationship (1:M, unidirectional)
- Relationships between records are stored (“pointers”)
- One “root” data group is the only entry point into the DB (I.e. access to any data group must be through the root)
- To access a record, the data record key, as well as, the keys of all parent records must be provided

Hierarchical Database Pros & Cons

- Advantages

- ◆ Stored relationships mean fast retrieval

- Disadvantages

- ◆ Stored relationships take up space
- ◆ Stored relationships must be maintained, slowing updates
- ◆ Does not support many-to-many or bi-directional access

Network Database

- Like hierarchical DB's, relationships between data records are stored
- Allows multiple entry points into a database
- Supports M:M relationships and bi-directional access

Network Database Pros & Cons

- Advantages

- ◆ Supports complex relationships

- Disadvantages

- ◆ Complex relationships require more space, take longer to update and retrieval is slower
- ◆ Because of the complexity, specialized personnel are needed

Relational Database

- Groups of data are viewed as **tables** (also called relations) with rows and columns not records and fields
- No stored relationships
- Access to database can be through any table and doesn't have to be defined
- Tables can be accessed by any field
- To relate information between different tables, the relational model uses duplicate data fields

Relational Database Pros & Cons

- Advantages

- ◆ Easy to understand, use, and learn
- ◆ No stored relationships therefore faster updating and less space

- Disadvantages

- ◆ Requires data redundancy - extra space, longer to update
- ◆ Slow retrieval

DBMS Features

- Data access controls
- Backup and recovery
- Data integrity
- Data manipulation
- Query language
- Alternative views

IBM i Features

- Data access controls
 - ◆ **Object security**
- Backup and recovery
 - ◆ **GO Backup**
 - ◆ **SAV commands**
- Data integrity
 - ◆ **Referential constraints**
 - ◆ **Trigger programs**
 - ◆ **DDS keywords**

IBM i Features

- Data manipulation
 - ◆ Navigator
 - ◆ DFU
 - ◆ SQL
- Query language
 - ◆ QUERY
 - ◆ SQL
- Alternative views
 - ◆ Logical files
 - ◆ SQL views

A Logical File

- Refers to data that exists in a physical file
- Can reference data from many physical files
- Contains:

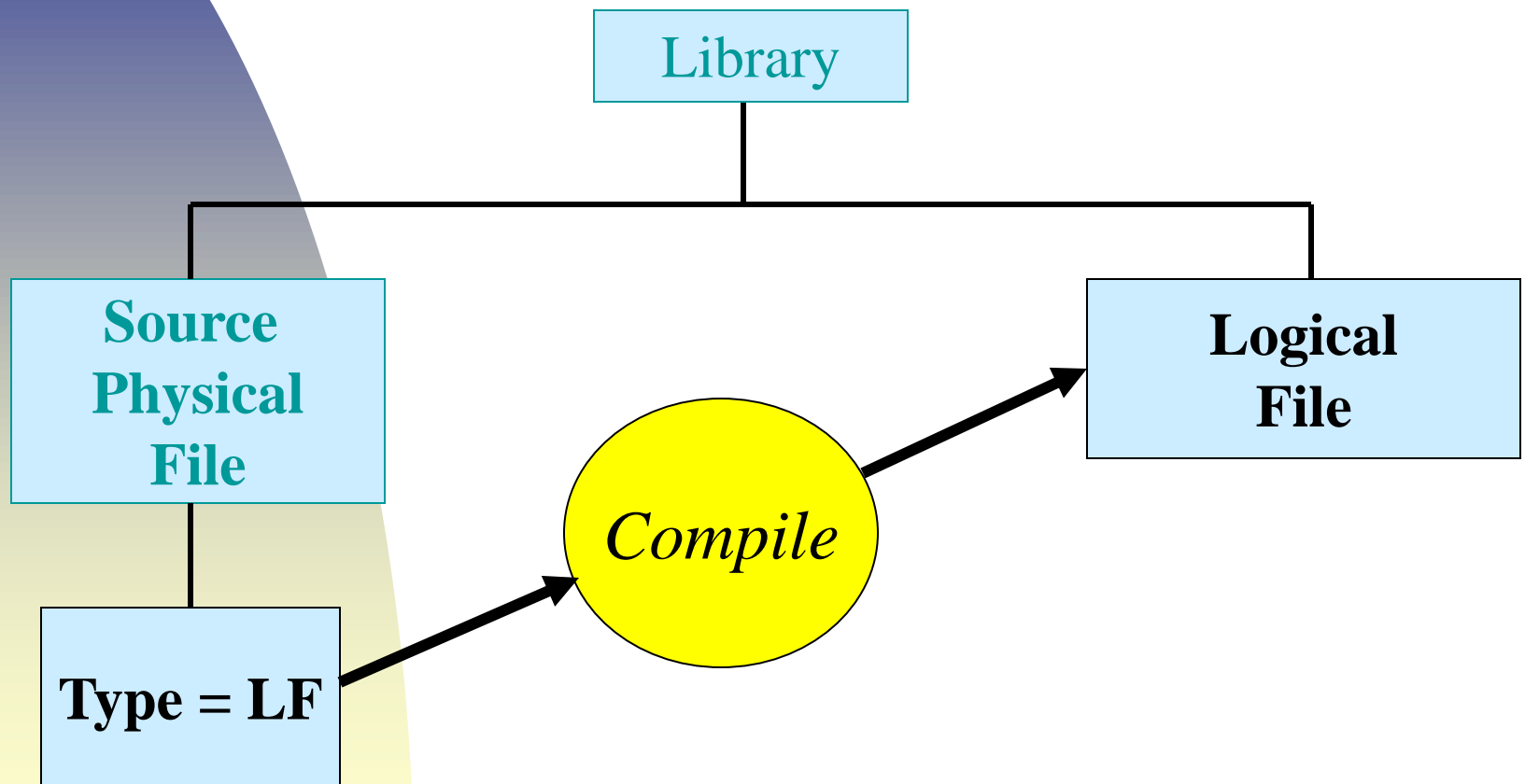
Field definitions

An access path

NO DATA

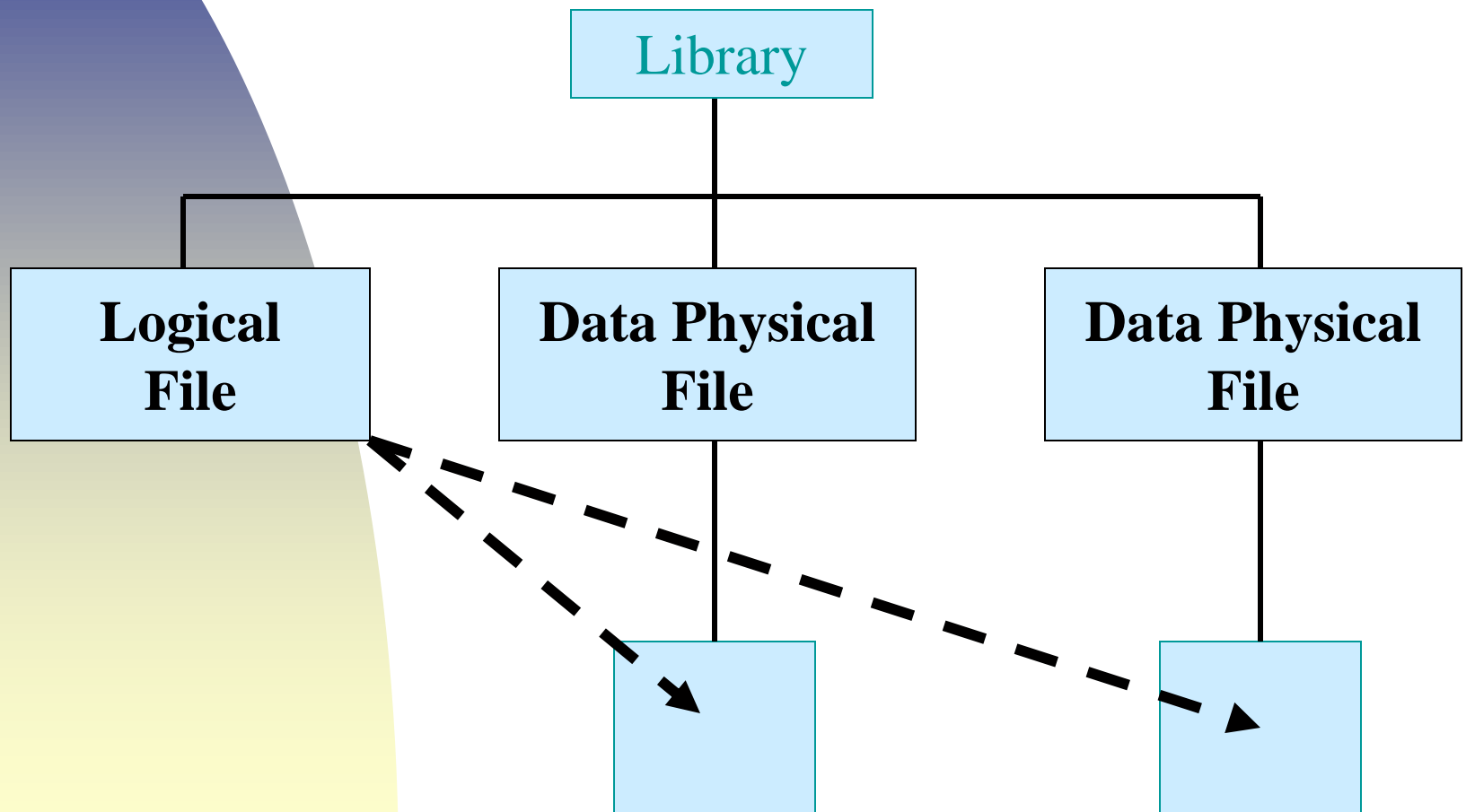
Logical File

- Created by compiling a source physical file member with type = LF



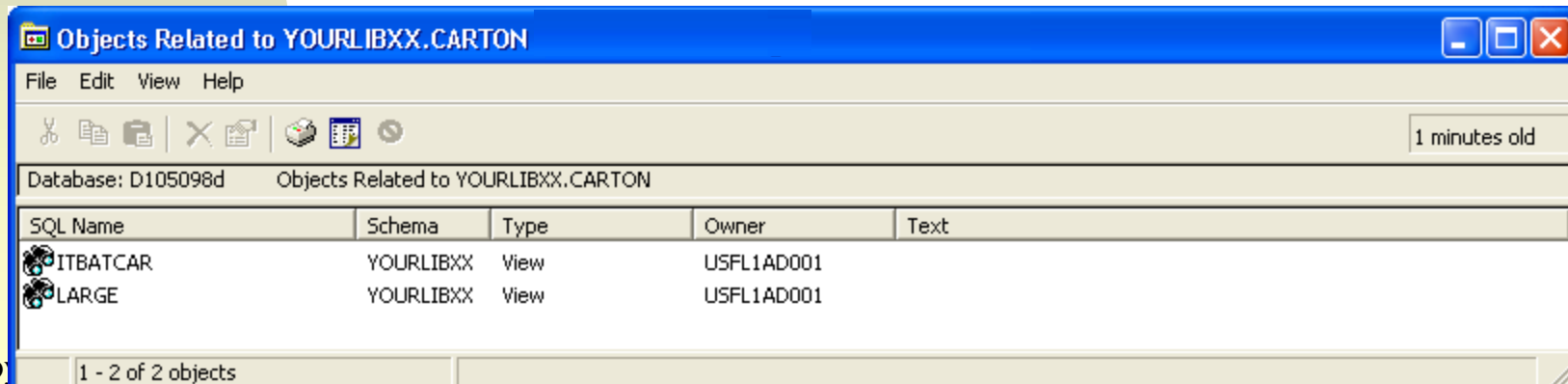
Logical File

- Acts as an alternate index to physical file data



Logical/Physical Relationship

- Changes to the physical file data will be reflected in the logical file
- A physical file cannot be deleted if there is a dependant logical file
- Use **DSPDBR** to find a physical files' dependant logical files
- Or Navigator: right click file, select **Show Related**



The screenshot shows a window titled "Objects Related to YOURLIBXX.CARTON". The window has a menu bar (File, Edit, View, Help) and a toolbar with icons for copy, paste, delete, and refresh. The status bar indicates "1 minutes old". The main content area displays a table with the following data:

SQL Name	Schema	Type	Owner	Text
ITBATCAR	YOURLIBXX	View	USFL1AD001	
LARGE	YOURLIBXX	View	USFL1AD001	

The status bar at the bottom of the window shows "1 - 2 of 2 objects".

Relational Operands

- Select
- Project
- Union
- Join

Select

- SELECT retrieves certain rows from a relational table, based on a condition
- SELECT FROM table name
WHERE condition
- Conditions take the form of:

Field Comparison Value/Field

e.g. Salary > 20,000

Gender = "F"

Date < CURDATE

Select

Lname	Fname	Gender	Salary	Dept
Jones	Adrian	M	14000	716
Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924

SELECT FOR Gender = "F"

Smith	Pat	F	23000	630
Adams	Chris	F	65000	630

Project

- Returns only specified columns (but all rows)

Jones	Adrian	M	14000	716
Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924

PROJECT ON Lname Dept

Jones	716
Smith	630
Adams	630
Conner	924

Project

Projects & Selects can also be performed together

Lname	Fname	Gender	Salary	Dept
Jones	Adrian	M	14000	716
Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924

**PROJECT (SELECT FOR Gender = "F")
ON Lname Dept**

Smith	630
Adams	630

Union

- A union of two tables combines and sorts the rows of multiple tables into a defined sequence
- The total number of rows from the result of a union is equal to the combined number of rows in each table

Union

Jones	Adrian	14000
Smith	Pat	23000
Adams	Chris	65000
Conner	Dale	34000

Evans	Joan	42000
Rudd	Scott	81000
Kahn	Bill	18000
Dole	Rene	36000

UNION ORDER BY Salary

Jones	Adrian	14000
Kahn	Bill	18000
Smith	Pat	23000
Conner	Dale	34000
Dole	Rene	36000
Evans	Joan	42000
Adams	Chris	65000
Rudd	Scott	81000

Joins

- Joins appear to combine data from different tables
- A general join matches every row in one table with every row in another table
- An equijoin matches rows between tables that have the same values in a column(s)
- A natural join matches rows like an equijoin but eliminates duplicate columns in the result table

Simple Logical Files

- Allow you to perform the **SELECT** and **PROJECT** functions
- Reference a physical file with the **PFILE** record level keyword

R DEPTREC
LNAME
DEPT

PFILE (PF1)

- Include all or some of the fields from the physical file with field level specifications (I.e. perform a project)

Simple Logical Files

- Select or omit rows with Select or Omit specifications
- Select or Omit specifications identified with a S or O in column 17 and follow field and key specifications

R DEPTREC
LNAME
DEPT
S GENDER

PF1 (PF1)

COMP (EQ 'F')

- Condition: Field Comparison Value

Compound Conditions

- Select or omit specifications can have multiple conditions
- Multiple conditions can be specified with ANDs and ORs
 - ◆ ANDs require both conditions be true for a record to be selected
 - ◆ ORs require either conditions be true

Compound Selects

Lname	Fname	Gender	Salary	Dept
Jones	Adrian	M	14000	716
Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924

SELECT FOR Gender = "F" AND Salary > 30000

Adams	Chris	F	65000	630
-------	-------	---	-------	-----

SELECT FOR Gender = "F" OR Salary > 30000

Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924

Compound Selects with DDS

AND

R DEPTREC

PFILE

LNAME

DEPT

S GENDER

COMP (EQ 'F')

SALARY

COMP (GT 30000)

R DEPTREC

PFILE

LNAME

DEPT

S GENDER

COMP (EQ 'F')

S SALARY

COMP (GT 30000)

OR

Logical/Physical File Relationship

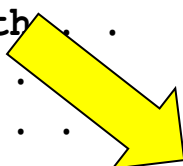
- When data is added or deleted from a physical file, the logical file will be immediately changed to reflect these modifications
- This updating is controlled by the logical files “Access path maintenance” parameter. Valid values are:
 - ◆ *IMMED
 - ◆ *REBLD
 - ◆ *DLY

Logical/Physical File Relationship

Change Logical File (CHGLF)

Type choices, press Enter.

Logical file	> DEANSLIST	Name
Library	*LIBL	Name, *LIBL, *CURLIB
System	*LCL	*LCL, *RMT, *FILETYPE
Force rebuild of access path	*NO	*NO, *YES
Maximum members	1	Number, *SAME, *NOMAX
Access path size	*MAX4GB	*SAME, *MAX4GB, *MAX1TB
Access path maintenance	*IMMED	*SAME, *IMMED, *REBLD, *DLY
Access path recovery	*NO	*SAME, *NO, *AFTIPL, *IPL
Force keyed access path	*NO	*SAME, *NO, *YES
Preferred storage unit	*ANY	1-255, *SAME, *ANY
Rcd format selector program	*NONE	Name, *SAME, *NONE
Library	_____	Name, *LIBL, *CURLIB
Records to force a write	*NONE	Number, *SAME, *NONE
Maximum file wait time	*IMMED	Number, *SAME, *IMMED, *CLS
Maximum record wait time	60	Number, *SAME, *IMMED, *NOMAX
Share open data path	*SAME	*SAME, *NO, *YES



More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

Synchronizing LF's and PF's

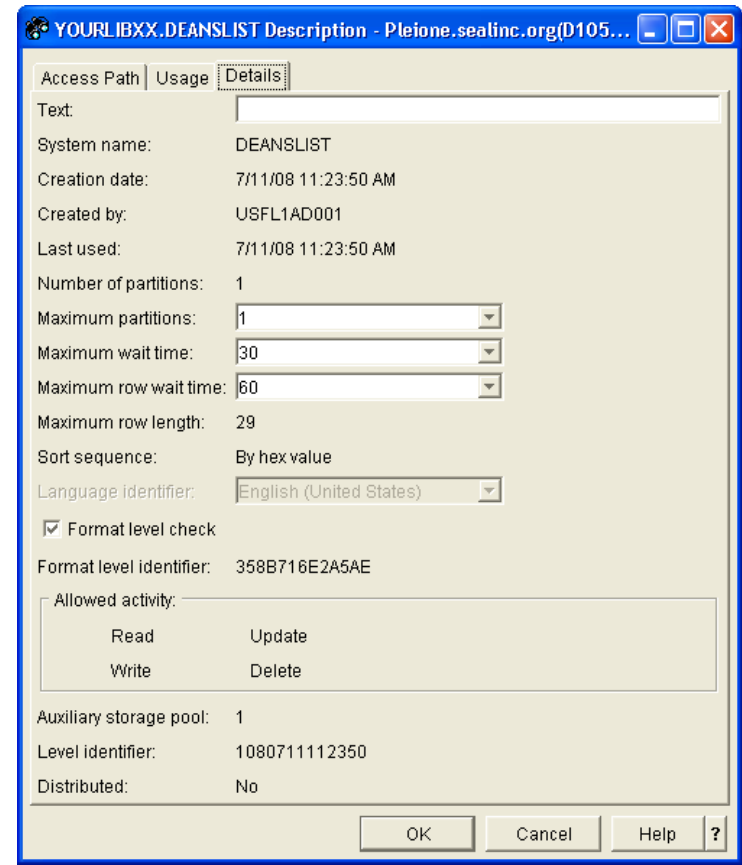
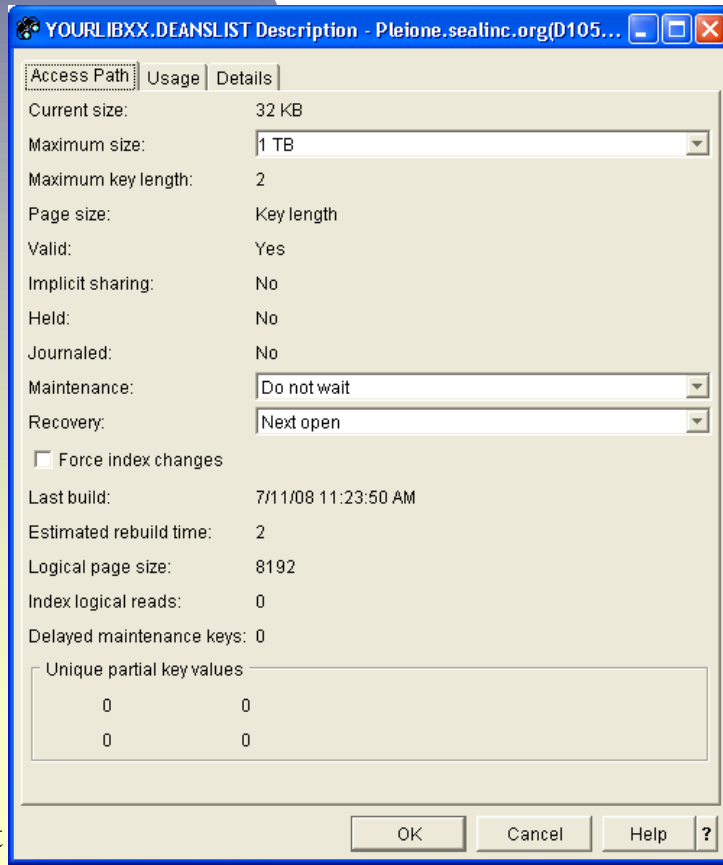
- *IMMED - is the default and only allowable value for uniquely keyed LF's
- *REBLD - the logical file access path is not saved. When the logical file is opened, the access path is rebuilt and maintained. When the logical file is closed the access path is not maintained
- *DLY - when the logical file is opened changes to the physical file are applied to the logical file access path

When to Use

- *REBLD - infrequently accessed LFs that are dependent on frequently accessed PFs. Saves time on constantly updating the LF
- *DLY - makes the user/job using the LF “pay” the performance cost
 - ◆ Jobs using the LF must wait for the PF changes to be made
 - ◆ Jobs modifying PF do not have to wait for LF updates

Navigator

- Logical files can be created in Navigator but control of the LF is more limited
 - ◆ For instance, cannot control when the logical file access path will be updated



Updating Data with Logical Files

- Changing logical file "data" actually changes the physical file
- Data can only be inserted/changed in fields specified in the logical file

Jones	Adrian	M	14000	716
Smith	Pat	F	23000	630
Adams	Chris	F	65000	630
Conner	Dale	M	34000	924
Cole				458
Palmer				188

- Changes can only be performed on records defined in the logical file

LF/PF File Relationship II

- Logical files are dependent on physical files
- Physical files can not be deleted if there are dependent files
- Physical files cannot be recompiled (since this deletes them), if there are dependent files
- The dependent logical files must be removed before recompiling

Removing Dependent Logical Files

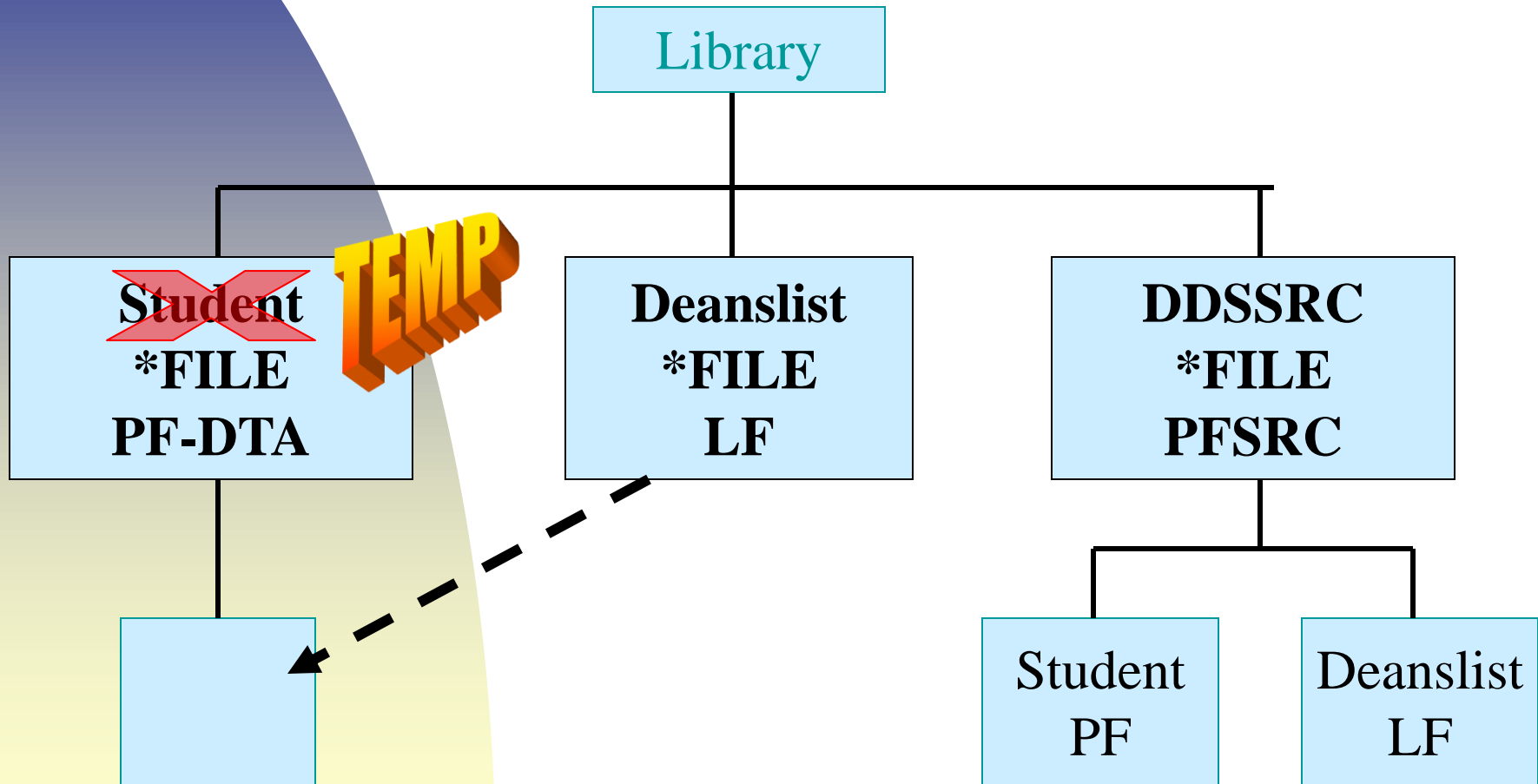
- Deleting the logical file will remove the dependency
- However, when a physical file is renamed all dependent logical files are modified to “point to” the new named object
- When the PF definition is recompiled there is no existing PF to be deleted.
- LF definitions can be recompiled and the new LFs will point to the newly created PF

Changing Physical & Logical Files

- To structurally change a physical file you must:
 - ◆ Find the dependent logical files (DSPDBR or Show Related in Nav)
 - ◆ Delete the dependent logical files or rename the physical file
 - ◆ After recompiling the physical file, recreate the dependent logical files
- Since they contain no data, logical files are easier to change than physical files
 - ◆ Change the DDS source code
 - ◆ Recompile

Changing Physical Files

- Renaming Student to Temp changes LF Deanslist



Renaming Physical Files

- LF Deanslist now dependent on PF TEMP

Display Spooled File

```
File . . . . . : QPDSPFD                               Page/Line   2/2
Control . . . . .                               Columns     1 - 78
Find . . . . .
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
Record format . . . . . :                               DLFMT
Key field . . . . . :                               GRADEPT
Sequence . . . . . :                               Descending
Sign specified . . . . . :                               SIGNED
Zone/digit specified . . . . . :                               *NONE
Alternative collating sequence . . . . . :                               No

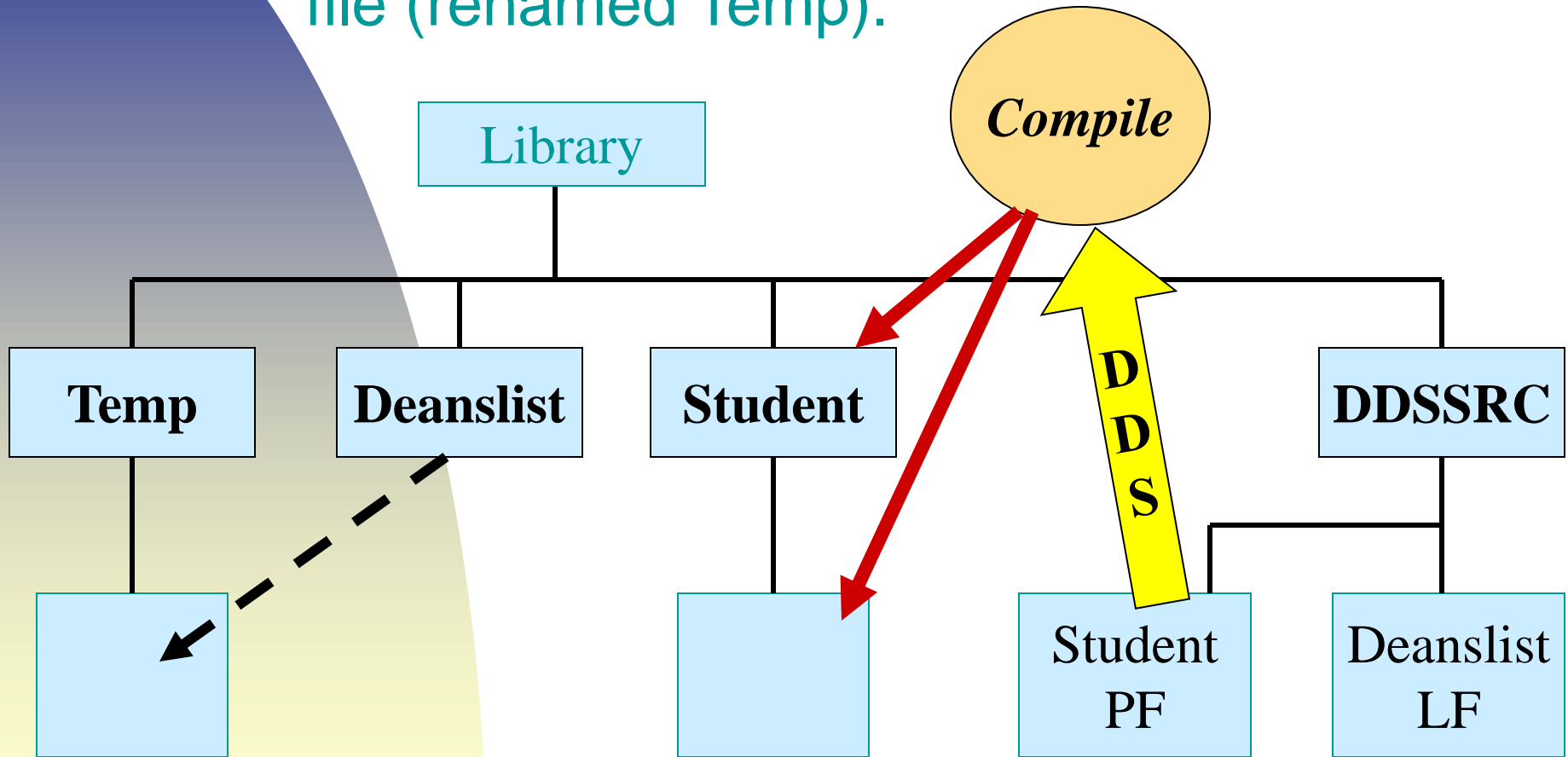
Files accessed by logical file                               PFILE
File                               Library                LF Format
TEMP                               YOURLIBXX          DLFMT

Select/Omit Description
Dynamic selection . . . . . : DYNSLT                    No
Number of rules . . . . . :                               2
Format . . . . . :                               DLFMT
Field . . . . . :                               GRADEPT
Rule . . . . . :                               SELECT
Comparison . . . . . :                               COMP GE
```



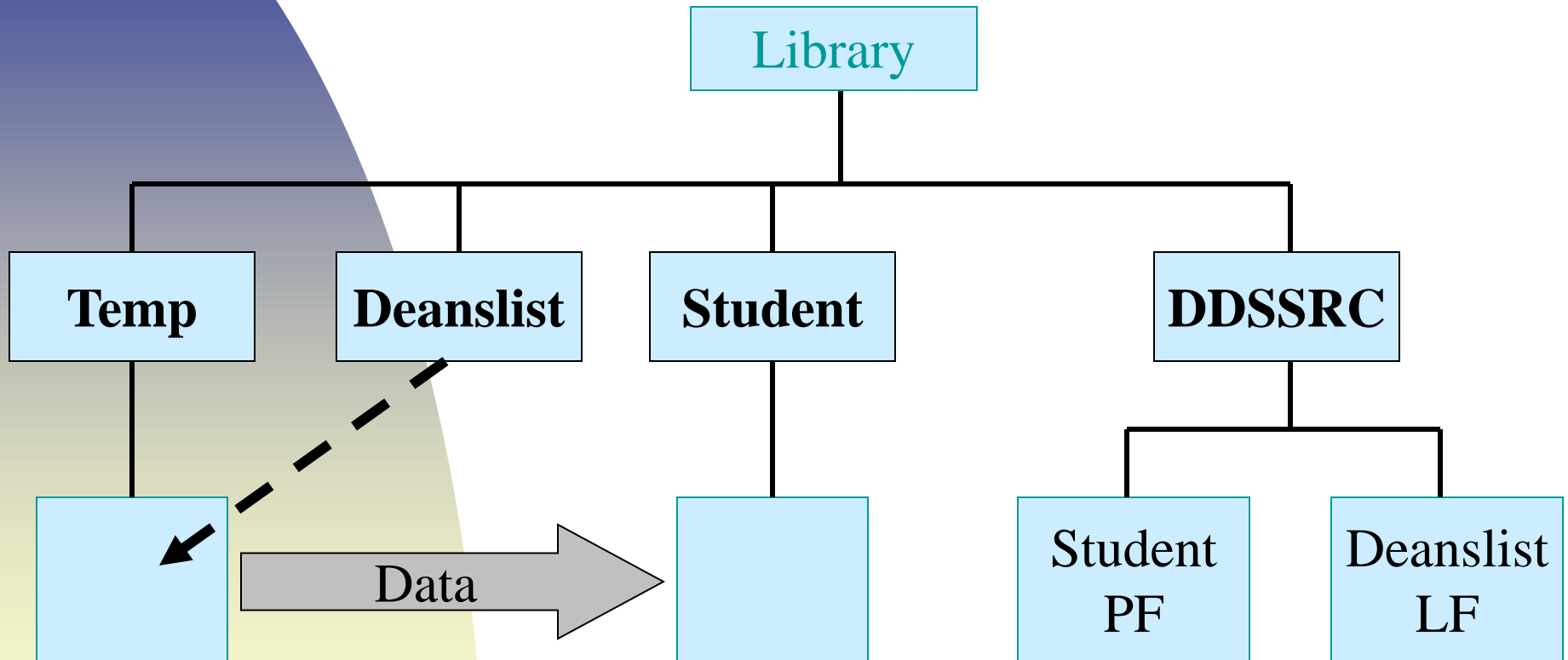
Changing Physical Files

- Changing Student's DDS and recompiling no problem because there is no Student file (renamed Temp).



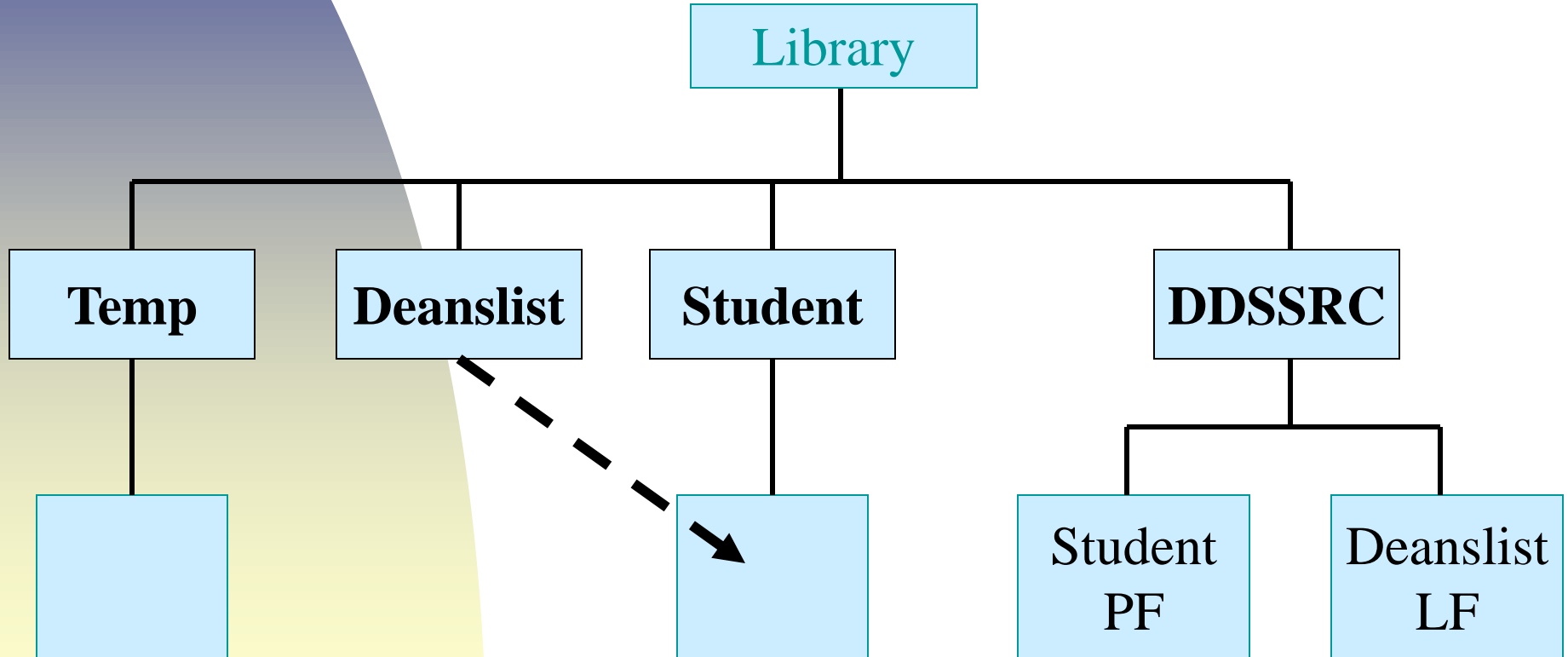
Changing Physical Files

- Copy the data from Temp to Student



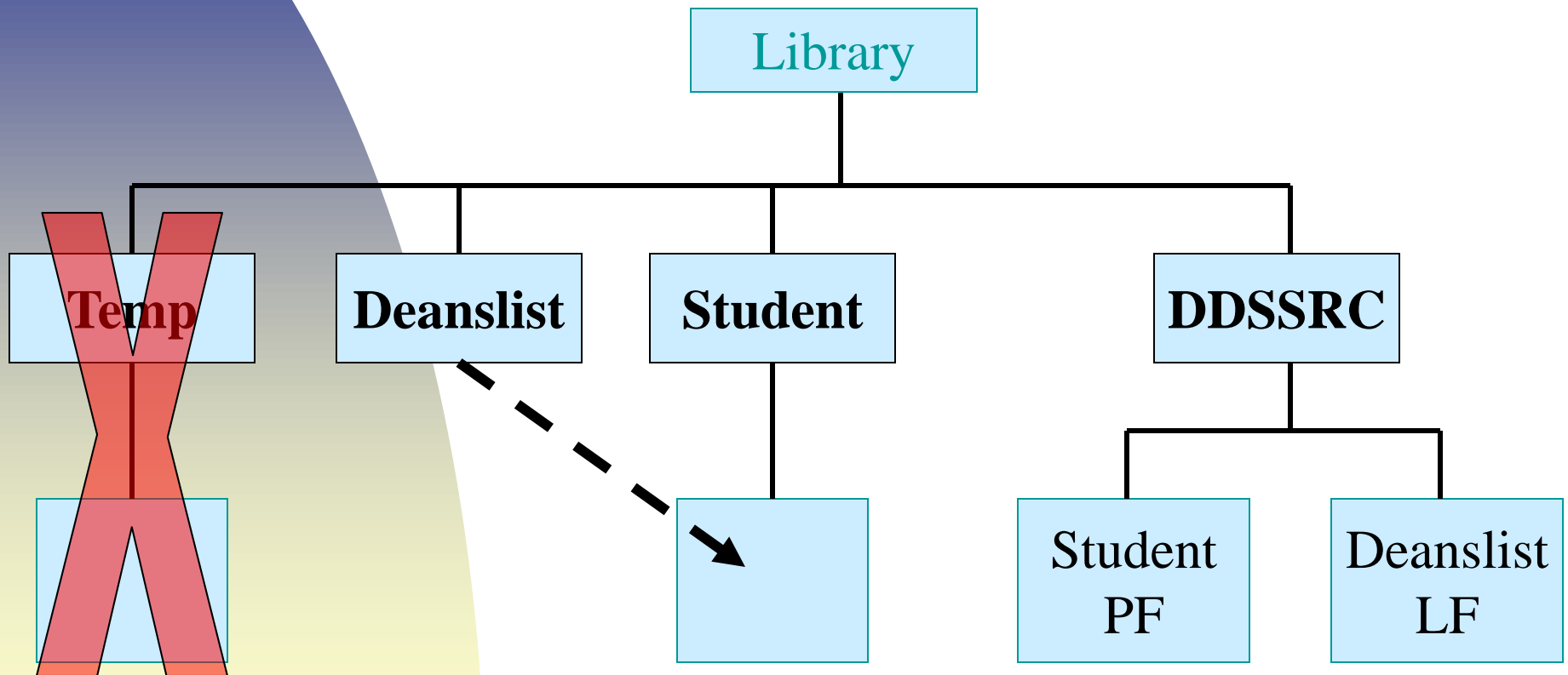
Changing Physical Files

- Recompiling Deanslist's DDS deletes old LF Deanslist and creates new Deanslist pointing to Student



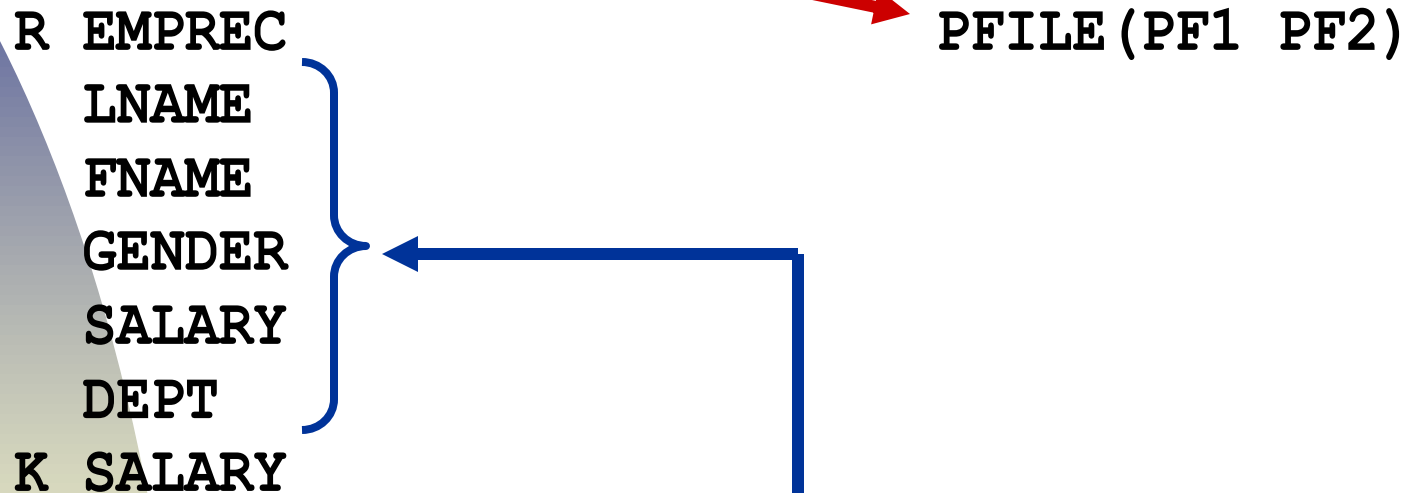
Changing Physical Files

- Clean up by deleting Temp



Union

- Use PFILE keyword to identify the files



- Only fields found in all files can be specified

Results in:

LNAME	FNAME	GENDER	SALARY	DEPT
	PF2 RECORD		17800	
	PF1 RECORD		18300	
	PF1 RECORD		18600	
	PF2 RECORD		19700	
	PF1 RECORD		20500	
	PF2 RECORD		24200	
	PF1 RECORD		26700	

: : :
: : :

Joining Files

- Logical files allow you to perform a JOIN
- JFILE - record level keyword that identifies the files to be joined

R R1FMT

J

J

FLDA

FLDC

Relative file numbers used
instead of file names

JFILE (PF1 PF2 PF3)

JOIN (1 2)

JFLD (FLDA FLDA)

JOIN (2 3)

JFLD (FLDB FLDC)

JREF (1)

Joining Files

- Join specification - defines the basis for joining the files (J in column 17)
- Join specs follow record specs and there is one for each two files joined

R R1FMT

J

J

FLDA

FLDC

JFILE (PF1 PF2 PF3)

JOIN (1 2)

JFLD (FLDA FLDA)

JOIN (2 3)

JFLD (FLDB FLDC)

JREF (1)

- JOIN keyword identifies the two files to be joined
- JFLD keyword(s) identifies the field(s) that will be used to join the records

Results in:

Data from PF1

Data from PF3

R1FMT

Field A	Field C
Field A	Field C
Field A	Field C
Field A	Field C
Field A	Field C
Field A	Field C

⋮
⋮

⋮
⋮

Multiple Format Logical Files

- Let you perform a union on files with different record formats

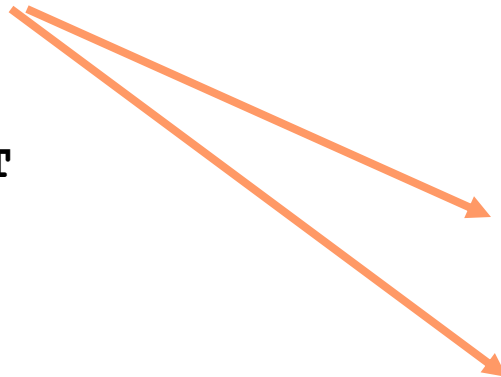
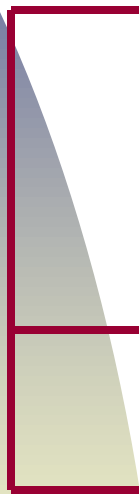
- Multiple record specifications
- Key fields must match between files

R DEPTFMT
K

R EMPFMP
K
K

PFILE (LIB1/DEPT)
DEPTNUM

PFILE (LIB2/EMP)
DEPT#
EMPNUM



Results in:

Department Record	
Employee Record	
Employee Record	
Department Record	
Employee Record	
Employee Record	
Employee Record	
Employee Record	
Department Record	
Employee Record	
Employee Record	

⋮ ⋮
⋮ ⋮

Why Use Logical Files

- Simplify program logic
- Cut down on program file reads (I.e. instead of 3 PF reads, 1 JLF read)
- These differences will:
 - ◆ Enhance program performance
 - ◆ Make the program easier to code and maintain

LFs Simplify Program Logic

- To read all DEPT & EMP records from two PFs

Read DEPT

Do While read successful

Read EMP where DEPT# = DEPNUM

Do While read successful

Read Next EMP where DEPT# = DEPNUM

Enddo

Read Next DEPT

Enddo

- To read from one multiple format LF

Read MFLF

Do While read successful

Read next MFLF

Enddo

LFs Cut Down on File Reads

- DBMS reads and writes are more efficient because DB2 is integrated into IBM i
- When a program reads a logical file, all the physical files are actually read by the DBMS
- Reading the PFs through the LF is faster because the DBMS is doing the reading not the program

Points to Remember

- There are three types of databases: hierarchical, network, and relational
- Relational DBMS's provide the capability to perform: select, project, union, and join
- With these relational operands you can create various user views of data
- Logical files provide all the relational DBMS functions