Chapter 5: Database Objects

IBM DB2 Universal Database V8.1

Database Administration Certification Preparation Course

Maintained by Clara Liu

Objectives

- In this section, we will cover:
  - Buffer Pools
  - Table Spaces
  - Schemas and Catalogs
  - Data Types
  - Tables
  - Identity Columns
  - Temporary Tables
  - Views
  - Indexes
  - Constraints
  - Packages
  - Triggers, Functions, and Stored Procedures
  - Federated Database Support

DB2 Data Management Software

IBM

e business software

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

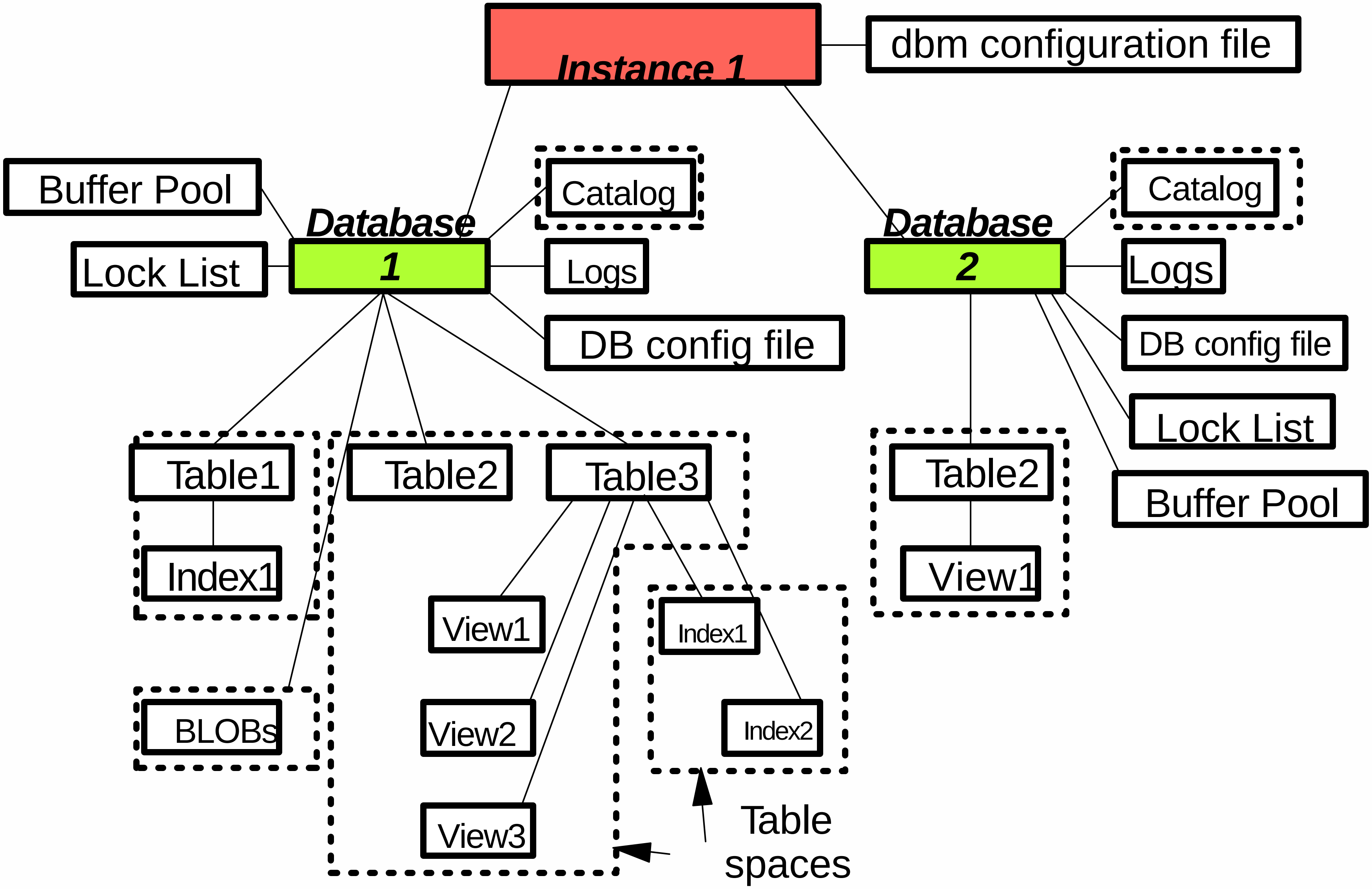
Packages

Triggers, Functions, Stored Procedures

Federated Database Support

IBM Software Group

Database Object Hierarchy



DB2 Data Management Software

IBM

Buffer Pools

- Used to buffer data in memory to reduce the number of I/O operations to the physical database
- Keep often requested data/index pages in memory
- Keep infrequently accessed tables (e.g. random access into very large table) out of main memory
- Ability to keep large number of pages in extended storage cache
- IBMDEFAULTBP is the default bufferpool created with every database

Table Spaces

- Table space is a logical grouping of tables created within a database
- Tables are created within table spaces
- Two types of table spaces:
  - System Managed Space (SMS)
  - Database Managed Space (DMS)

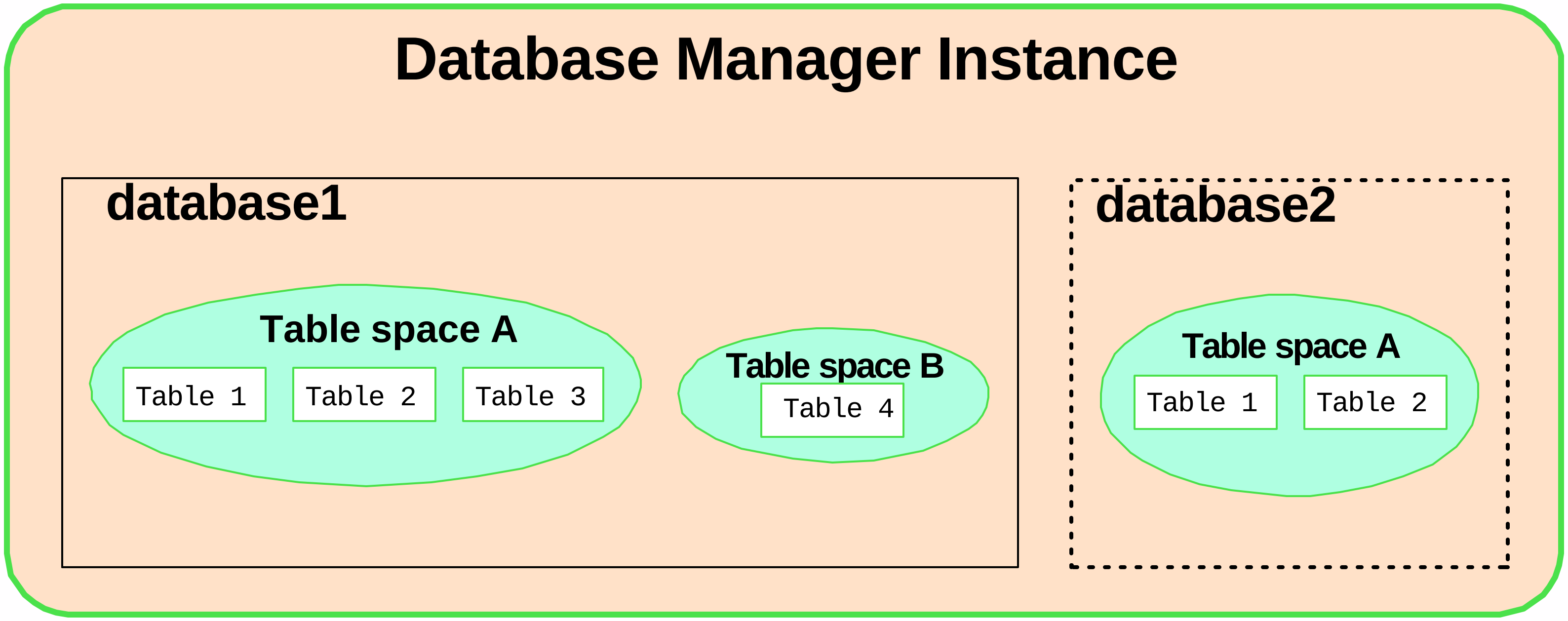


Table Spaces

- All database objects are stored within table spaces
- Two types of storage:
  - System Managed Space (SMS)
  - Database Managed Space (DMS)
- A table space is composed of one or more containers
- Data allocated by extents within containers
- Table spaces are either 4K, 8K, 16K or 32K pages
  - 4K is default size
  - Cannot mix page sizes within a table space
  - Must be associated with a buffer pool with same page size

Table Spaces

- With a simple CREATE DATABASE command:
  - CREATE DATABASE sample
- Three SMS table spaces are created automatically in default locations:
  - SYSCATSPACE - system catalog tables
  - USERSPACE1 - default user data
  - TEMPSPACE1 - temporary data
- Can change table space storage type and explicitly specify the locations of the containers, example:

  -
    ```
    CREATE DATABASE sample
     CATALOG TABLESPACE
       MANAGED BY SYSTEM
       USING ( 'c:\catdir1' );
     USER TABLESPACE
       MANAGED BY DATABASE
       USING ( FILE 'c:\db2files\usertbsp1 100, FILE 'c:\db2files\usertbsp2 100 )
     TEMP TABLESPACE
       MANAGED BY SYSTEM
       USING ( 'c:\tempspace' ) ;
    ```

DB2 Data Management Software

IBM

e business software

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

IBM Software Group

Schema

- A schema is a collection of database objects such as tables, views, indexes, or triggers. It provides a logical classification of database objects
- How is a Schema name used?
  - To fully-qualified table or other object name
  - "schemaname.tablename"
  - Can have multiple tables with the same name, but different schema names
    - eyerman.staff != jones.staff
- Following schema names reserved
  - SYSCAT, SYSIBM, SYSSTAT, SYSFUN
  - Avoid schema names beginning with SYS
    - Enforced with triggers, UDFs, and UDTs
- If database object does not specify a schema name, table qualified with current authorization ID
- Alternate schema names can be specified using
  - SET CURRENT SCHEMA or SET CURRENT SQLID command
  - CREATE ALIAS <aliasname> FOR <tab/view name>
  - CREATE VIEW

SYS Schemas

- Created with every database and placed into the SYSCATSPACE table space
- SYSIBM
  - Base catalogs
  - Access not recommended
- SYSCAT
  - SELECT authority GRANTed to PUBLIC
  - Catalog Read-only Views
  - Recommended way to obtain catalog information
- SYSSTAT
  - Updateable Catalog Views - Influence the Optimizer
- SYSFUN
  - User-Defined Functions

Roadmap to Catalog Tables

Schema: Table = SYSIBM View = SYSCAT

TABLE	VIEW	DESCRIPTION
SYSDBAUTH	DBAUTH	Authorities on database
SYSCHECKS	CHECKS	Check constraints
SYSCOLUMNS	COLUMNS	Column definitions
SYSCOLCHECKS	COLCHECKS	Columns referenced by check constraints
SYSCOLDIST	COLDIST	Detailed columns statistics
SYSKEYCOLUSE	KEYCOLUSE	Columns used in keys
SYSCONSTDEP	CONSTDEP	Constraint dependencies
SYSDATATYPES	DATATYPES	Datatype definitions (built-in & UDT)
SYSEVENTMONITORS	EVENTMONITORS	Event Monitor Definitions
SYSEVENTS	EVENTS	Events currently monitored
SYSFUNCPARMS	FUNCPARMS	Definitions of Parameters/Results of UDFs
SYSFUNCTIONS	FUNCTIONS	UDF definitions
SYSINDEXAUTH	INDEXAUTH	Index privileges
SYSINDEXES	INDEXES	Index definitions

Roadmap to Catalog Tables

Schema: Table = SYSIBM View = SYSCAT

TABLE	VIEW	DESCRIPTION
SYSPACKAGEAUTH	PACKAGEAUTH	Authorities on packages
SYSPACKAGEDEP	PACKAGEDEP	Package dependencies
SYSPACKAGES	PACKAGES	Package definitions
SYSREFERENCES	REFERENCES	Referential constraints definitions
SYSSTATEMENTS	STATEMENTS	Details of package SQL Statements
SYSTABAUTH	TABAUTH	Table Authorities
SYSTABCONST	TABCONST	Table constraint definitions
SYSTABLES	TABLES	Table definitions
SYSTABLESPACES	TABLESPACES	Table Space Definitions
SYSTRIGDEP	TRIGDEPEVENTS	Trigger dependencies
SYSTRIGGERS	TRIGGERS	Definitions of triggers
SYSVIEWDEP	VIEWDEP	View dependencies
SYSVIEWS	VIEWS	View definitions

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

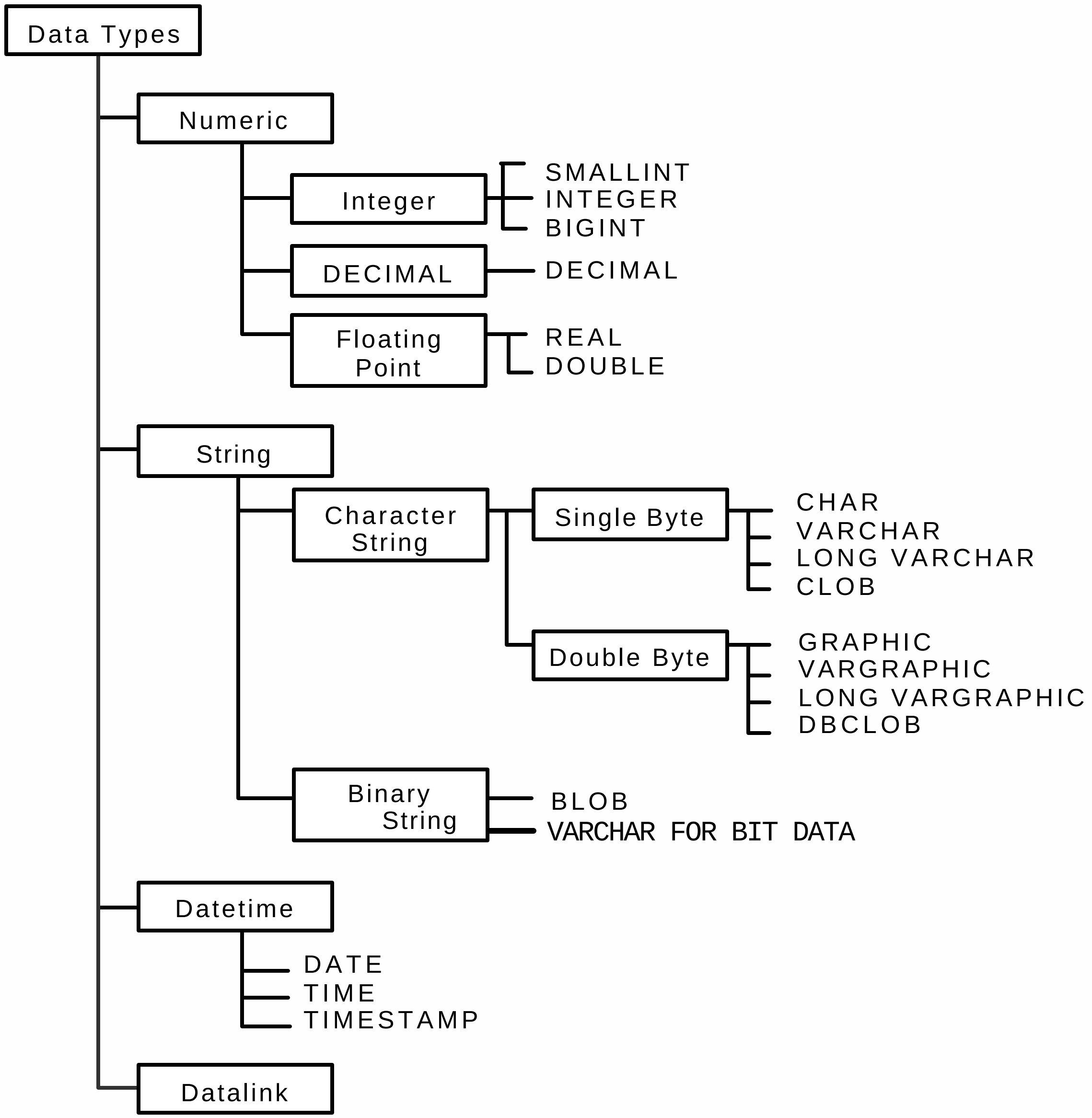
Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

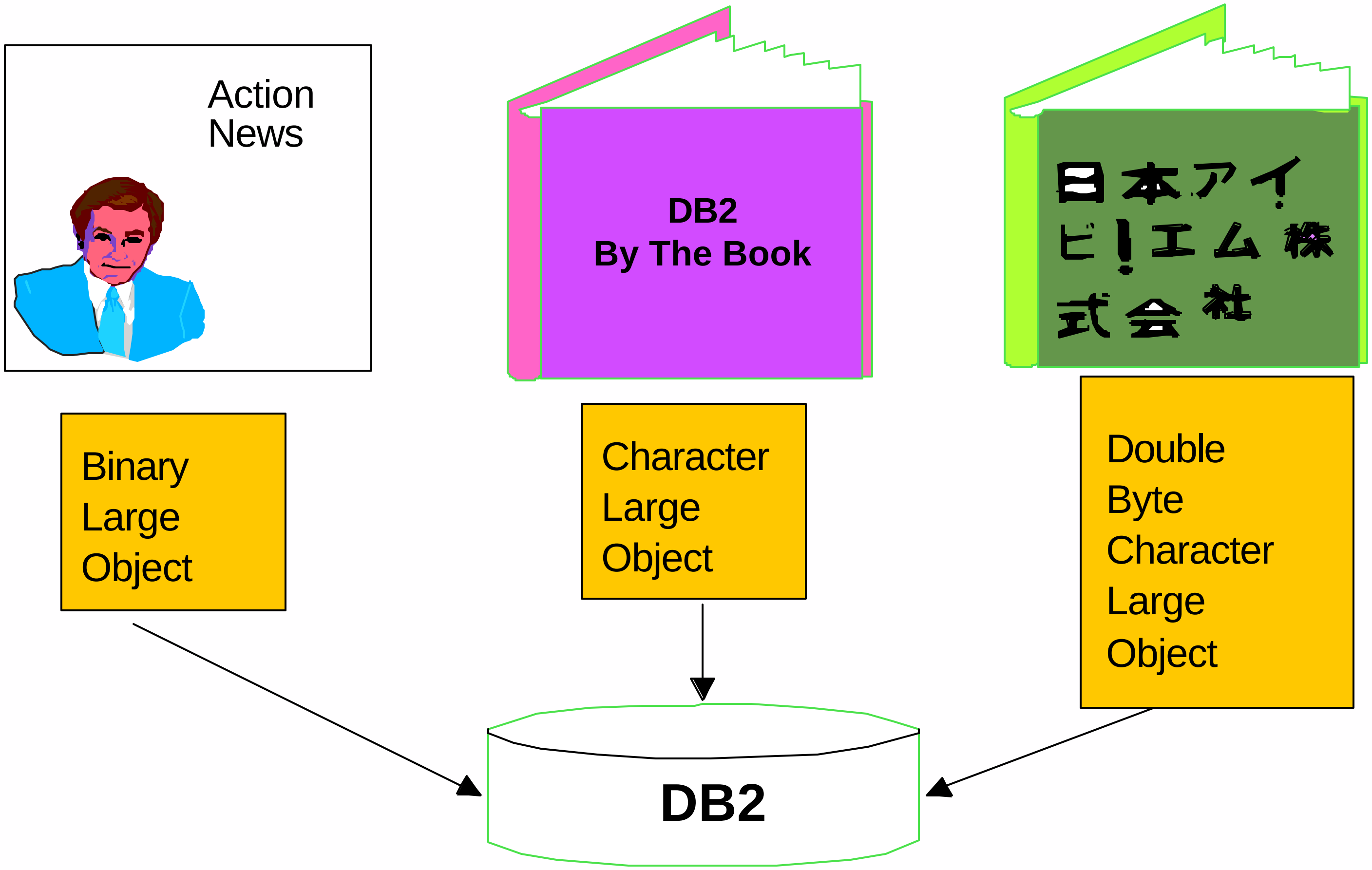
Data Types



- Data Types
  - Numeric
    - Integer
      - SMALLINT
      - INTEGER
      - BIGINT
    - DECIMAL
      - DECIMAL
    - Floating Point
      - REAL
      - DOUBLE
  - String
    - Character String
      - Single Byte
        - CHAR
        - VARCHAR
        - LONG VARCHAR
        - CLOB
      - Double Byte
        - GRAPHIC
        - VARGRAPHIC
        - LONG VARGRAPHIC
        - DBCLOB
    - Binary String
      - BLOB
      - VARCHAR FOR BIT DATA
  - Datetime
    - DATE
    - TIME
    - TIMESTAMP
  - Datalink

LARGE Objects

- To store large character strings or files
- To store large binary strings or files
- Maximum size is 2 GB (1 GB for DBCLOBs)



Action News

DB2
By The Book

日本アイ
ビーエム株
式会社

Binary Large Object

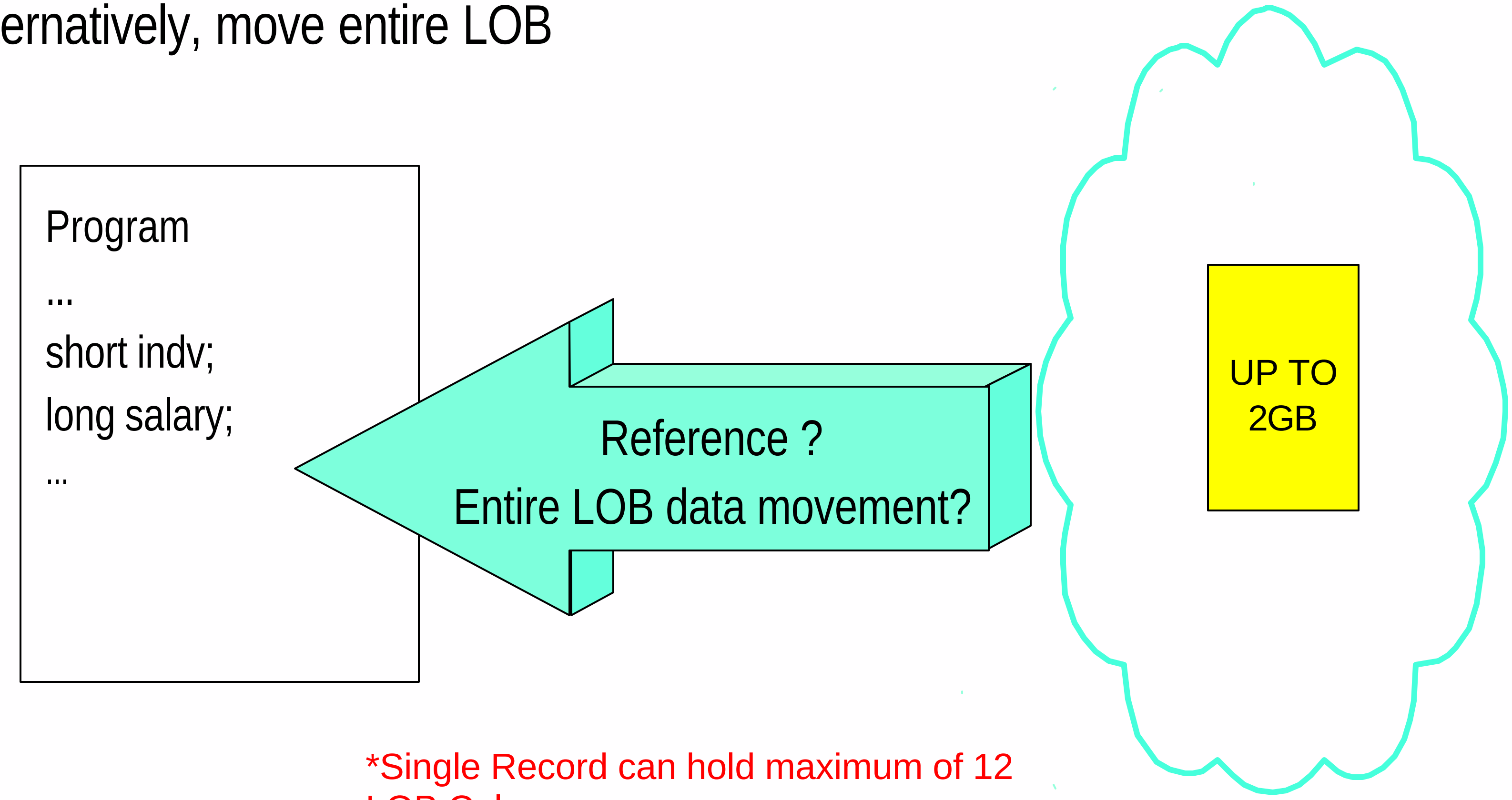
Character Large Object

Double Byte Character Large Object

DB2

LARGE Objects - Memory Considerations

- Use LOB Locators to move LOBs by reference
- Alternatively, move entire LOB



Program
...
short indv;
long salary;
...

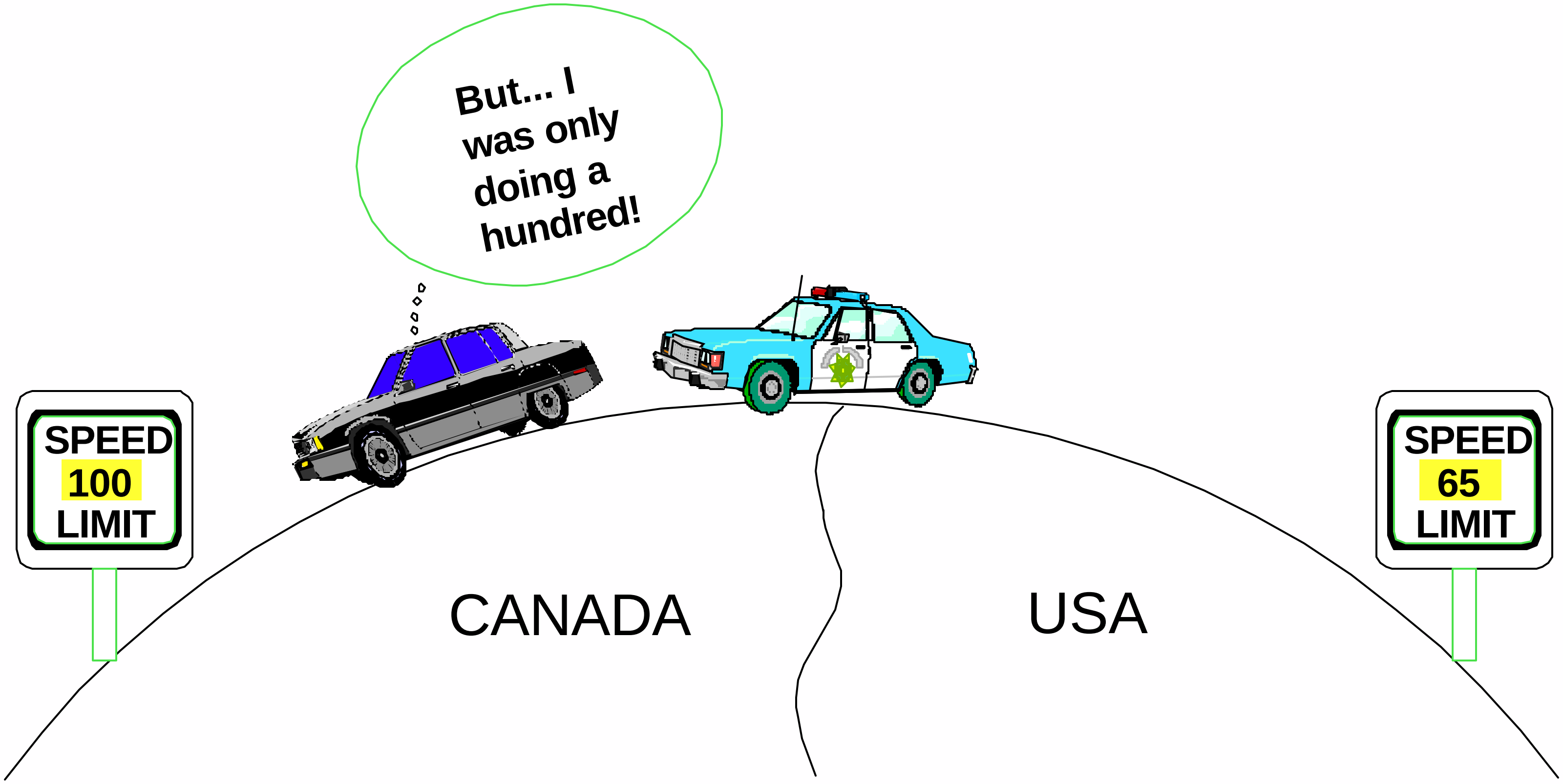
Reference ?
Entire LOB data movement?

UP TO
2GB

*Single Record can hold maximum of 12 LOB Columns

User-Defined Types

- Need to establish context for values
- DB2 enforced typing



User-Defined Types - Example

```
CREATE DISTINCT TYPE pound AS INTEGER WITH COMPARISONS ;

CREATE DISTINCT TYPE kilogram AS INTEGER WITH COMPARISONS ;

CREATE TABLE person
   ( f_name          VARCHAR (30)
   , weight_p        pound NOT NULL
   , weight_k        kilogram NOT NULL ) ;

SELECT f_name FROM person
     WHERE weight_p > pound(30);

SELECT f_name FROM person
     WHERE weight_p > weight_k;
```

Fails

Selecting the Correct Data Type

Question	DataType
Is the data fixed in length? Stored in binary format?	CHAR CHAR for bit data
Is the data variable in length? Stored in binary format?	VARCHAR VARCHAR for bit data
Do you need to sort(order) the data?	CHAR, VARCHAR NUMERIC
Is the data to be used in arithmetic operations?	DECIMAL,REAL DOUBLE,BIGINT INTEGER, SMALLINT
Does it contain decimal?	DECIMAL, REAL DOUBLE
Does the data have a specific meaning (beyond DB2 base data type)?	UDT

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

CREATE TABLE Command

- Connect to database first
- You must have SYSADM or DBADM authority or CREATETAB privilege on the database
- Example:

```
connect to eddb;
create table artists
 ( artno              SMALLINT NOT NULL
 , name               VARCHAR(50) WITH DEFAULT 'abc'
 , classification     CHAR(1) NOT NULL
 , bio                CLOB(100K) LOGGED
 , article             DATALINK LINKTYPE URL FILE
                         LINK CONTROL MODE DB2OPTIONS,
 , picture            BLOB(2M) NOT LOGGED COMPAT )
 INDEX IN indtbsp
 LONG IN longtbsp
 IN datatbsp;
```

Where is table placed by default

- If a table is created without the IN clause, the table data (and its indexes and LOB data) will be placed:
  - In the IBMDEFAULTGROUP table space (if it exists and if the page size is sufficient)
  - In a user created table space which is of the smallest pagesize that is sufficient for the table
  - Then it will go in USERSPACE1 (if it exists and has a sufficient page size)
- The IN, INDEX IN, and LONG IN clauses specify which table spaces regular data, index, and large objects are to be stored in

CREATE TABLE ... LIKE

- Table columns have exact same names and attributes
  - One for one copy of columns
    - no constraints, triggers, or indexes copied
    - data not copied
  - May specify table or view
- Example:
  - CREATE TABLE tab1new LIKE tab1;

Definition Only Table

- Query used to define table
- Can be subset of single table or combination of tables.
- Table not populated
- Column attributes of defined table based upon referenced table
- Example:

  -
    ```
    CREATE TABLE t1new
      AS (SELECT c1, c8, c10 FROM t1)
      DEFINITION ONLY;
    ```

NULL Values

- A null value represents an unknown state
- The CREATE TABLE statement can contain the phrase NOT NULL following the definition of each column.
- This will ensure that the column contains a known data value.
- Can specify a default value if NULL is entered
- Example:

```
CREATE TABLE staff
    ( id       SMALLINT NOT NULL WITH DEFAULT 10
    , name     VARCHAR(9)
    , dept     SMALLINT NOT NULL WITH DEFAULT 20
    , job      CHAR(5)
    , years    SMALLINT
    , salary   DECIMAL(7, 2)
    , comm     DECIMAL(7, 2) WITH DEFAULT );
```

user-defined default value

system default value

System Default Values

- If a specific default value is not specified following the DEFAULT keyword, the system default value of the column data type is used
- For example:
  - Numeric - 0
  - CHAR - Blanks
  - VARCHAR - A string of length 0
  - BLOB - A string of length 0
- Check the DB2 Command Reference under the 'ALTER TABLE' command for a complete list of data types' system default values

NULL and 0-Length Data Value Compression

- Reduce storage for typical data warehousing scenarios
  - Increase performance of large scans
- Available for all tables except global temporary tables
- Specifies the VALUE COMPRESSION clause in the CREATE TABLE command so that NULL and 0-length data values are to be stored more efficiently for most data types
- Eligible data types
  - NUMERIC
  - CHAR
  - VARCHAR
  - DBCS (fixed and variable)
  - BLOB
- Not supported data types
  - DATE
  - TIME
  - TIMESTAMP
  - These values are dynamic and are always changing
- Example:

```
► CREATE TABLE comp_t1
      ( c1 INTEGER        DEFAULT 0
      , c2 CHAR(10)       DEFAULT NULL
      , CONSTRAINT comp_t1mpk PRIMARY KEY (c1)
      ) VALUE COMPRESSION ;
```

System Default Value Compression

- If VALUE COMPRESSION is used, use the optional COMPRESS SYSTEM DEFAULT option to further reduce disk space usage
- Minimal disk space is used if the inserted or updated value is equal to the system default value for the data type of the column
- The default value will not be stored on disk. Data types that support COMPRESS SYSTEM DEFAULT:
  - All numeric, fixed-length character, and fixed-length graphic string data types
  - This means that zeros and blanks can be compressed.
- Must specify VALUE COMPRESSION if COMPRESS SYSTEM DEFAULT is used, otherwise warningraised and commpression is not enabled
- Example:
  - CREATE TABLE comp_t1

```
CREATE TABLE comp_t1
   ( c1 INTEGER NOT NULL       COMPRESS SYSTEM DEFAULT
   , c2 CHAR(10)               COMPRESS SYSTEM DEFAULT
   , CONSTRAINT comp_t1mpk PRIMARY KEY (c1)
   ) VALUE COMPRESSION ;
```

Some Useful Commands

- LIST TABLES
  - List tables for the current user
- LIST TABLES FOR ALL
  - List all tables defined in the database
- LIST TABLES FOR SCHEMA <schema>
  - List tables for the specified schema
- DESCRIBE TABLE <tablename>
  - Show the structure of the specified table
  - Example: **DESCRIBE TABLE department**

```
Column                Type      Type
name                  schema    name                  Length   Scale Nulls
--------------------- ------------ --------------------- ------------ --------- -----------
DEPTNO                SYSIBM    CHARACTER             3            0       No
DEPTNAME              SYSIBM    VARCHAR               29           0       No
MGRNO                 SYSIBM    CHARACTER             6            0       Yes
ADMRDEPT              SYSIBM    CHARACTER             3            0       No
LOCATION              SYSIBM    CHARACTER             16           0       Yes
```

Restrict Drop Table

- ALTER TABLE tab1 ADD RESTRICT ON DROP
- DROP TABLE tab1
  - SQL0672N Operation DROP not allowed on table USER.TAB1
- ALTER TABLE tab1 DROP RESTRICT ON DROP

Identity Columns

- A numeric column in a table which automatically generates a unique numeric value for each row that is inserted

- One Identity column per table maximum

- Values can be generated by DB2 always or by default
  - Generated always
    - values are always generated by DB2
    - applications are not allowed to provide an explicit value.
  - Generated by default
    - values can be explicitly provided by an application or if no value is given, then DB2 generates one
    - DB2 cannot guarantee uniqueness
    - intended for data propagation, unload/reload of a table

Identity Column - Generated Always Example

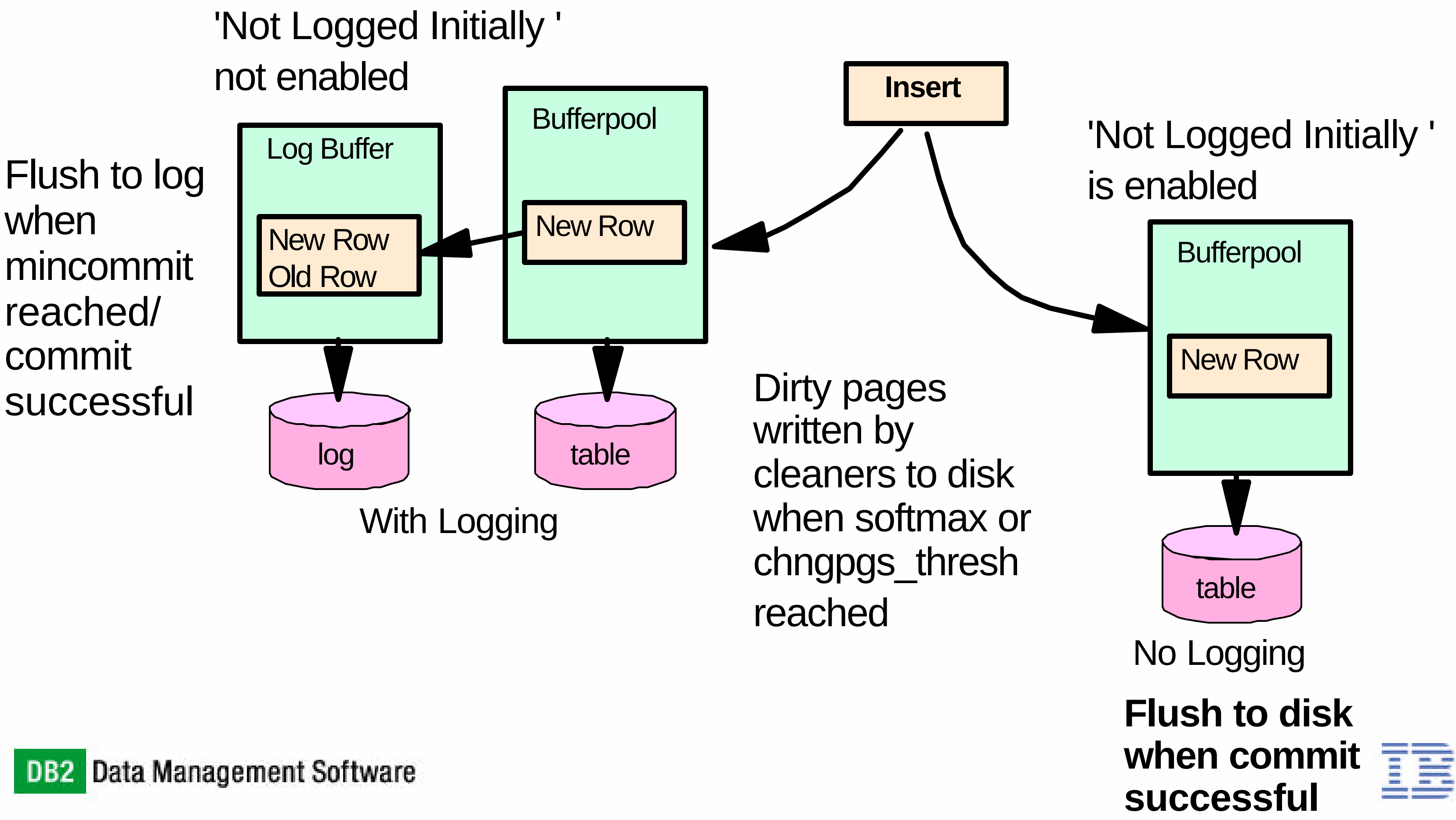
```
CREATE TABLE inventory
    (partno INTEGER
            GENERATED ALWAYS AS IDENTITY
            (START WITH 100 INCREMENTED BY 1),
      description CHAR(20) );
COMMIT;
INSERT INTO inventory VALUES (DEFAULT,'door');              --->inserts 100,door
INSERT INTO inventory (description) VALUES ('hinge');       --->inserts 101,hinge
INSERT INTO inventory VALUES (200,'windor');                --->error
COMMIT;

INSERT INTO inventory (description) VALUES ('lock');        --->inserts 102,lock
ROLLBACK;

INSERT INTO inventory (description) VALUES ('frame');       --->inserts 103,frame
COMMIT;

SELECT * FROM inventory;
                100   door
                101   hinge
                103   frame
```

Identity Columns - Generated By Default Example

```
CREATE TABLE inventory
   (partno INTEGER PRIMARY KEY
           GENERATED BY DEFAULT AS IDENTITY  (START WITH 100 INCREMENTED BY 1),
     description CHAR(20) );
COMMIT;
INSERT INTO inventory VALUES (DEFAULT,'door');           --->inserts 100,door
INSERT INTO inventory (description) VALUES ('hinge');    --->inserts 101,hinge
INSERT INTO inventory VALUES (200,'window');             --->inserts 200,window
INSERT INTO inventory VALUES (102,'handle');             --->inserts 102,handle
INSERT INTO inventory VALUES (101,'bolt');               --->error, duplicate
COMMIT;
INSERT INTO inventory (description) VALUES ('lock');     --->error, duplicate
INSERT INTO inventory (description) VALUES ('lock');     --->inserts 103,lock
ROLLBACK;
INSERT INTO inventory (description) VALUES ('frame');    --->inserts 104,frame
COMMIT;
SELECT * FROM inventory order by partno;
            100    door
            101    hinge
      102  handle
            104    frame
            200    window
```

'Not Logged Initially' Tables

- Useful for situations needing to insert large amounts of data from alternate source (another table or file)
- Data inserted without logging
- Use when recovery of table not required



'Not Logged Initially ' not enabled

Flush to log when mincommit reached/ commit successful

With Logging

Dirty pages written by cleaners to disk when softmax or chngpgs_thresh reached

'Not Logged Initially ' is enabled

No Logging

Flush to disk when commit successful

'Not Logged Initially' Tables ...

- To use this option, the table must first be created using ..
  - ▸ CREATE TABLE table-name ...   NOT LOGGED INITIALLY
- To improve concurrency after the CREATE
  - ▸ COMMIT should be issued ('Not Logged Initially' state turned off)
- To reactivate the 'not logged initially' state:
  - ALTER TABLE table-name ACTIVATE NOT LOGGED INITIALLY;
  - INSERT INTO ... SELECT FROM ... ;
  - COMMIT; ('Not Logged Initially' state turned off)

    OR
  - ALTER TABLE table-name ACTIVATE
NOT LOGGED INITIALLY
WITH EMPTY TABLE ;
  - INSERT INTO ... SELECT FROM ... ;
  - COMMIT ('Not Logged Initially' state turned off)

Declare Global Temporary Tables

- Created and used by an application and dropped (automatically) when the application terminates
- Can only be accessed by the application that created the table
- No entry exists in any catalog table to avoid catalog contention
- If multiple applications create a table of the same name, each application has a unique instance of that declared temporary table
  - No authority checking
- No table locking or row locking
- Minimal undo logging
  - Support the rollback of data changes made to global tempory table
  - NOT LOGGED clause manditory in V7, and now option in V8
- Hold intermediate results during complex processing
- Automatic cleanup
- Index support
  - Any standard index can be created on a temporary table
- Statistics support
  - RUNSTATS supported against the table

Temporary Tables

- Declared temporary tables reside in a user temporary tablespace
- Must be defined prior to creating any declared temporary tables

```
CREATE USER TEMPORARY TABLESPACE apptemps
     MANAGED BY SYSTEM USING ('apptemps');

DECLARE GLOBAL TEMPORARY TABLE temployess
     LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempdept
     ( deptid CHAR(6), deptname CHAR(20) )
       ON COMMIT DELETE ROWS NOT LOGGED ;

DECLARE GLOBAL TEMPORARY TABLE tempprojects
     AS  ( fullselect ) DEFINITION ONLY
     ON COMMIT PRESERVE ROWS NOT LOGGED
     WITH REPLACE IN TABLESPACE apptemps;
```

Chapter 5: Database Objects

- Buffer Pools and Table Spaces
- Schemas and Catalogs
- Data Types
- Tables, Identity Columns, Temporary Tables
- **Views**
- Indexes
- Constraints
- Packages
- Triggers, Functions, Stored Procedures
- Federated Database Support

Creating Views

- Data for view not stored separately
- Nested view supported
- Needs to have at least SELECT privilege on the base tables of the view
- View information kept in:
  - SYSCAT.VIEWS, SYSCAT.VIEWDEP, SYSCAT.TABLES

```
CONNECT TO TESTDB
CREATE VIEW DEPTSALARY
  AS SELECT DEPTNO, DEPTNAME, SUM(SALARY) AS TOTSAL
  FROM PAYROLL GROUP BY DEPTNO,DEPTNAME
```

```
CREATE VIEW EMPSALARY
    AS SELECT EMPNO, EMPNAME, SALARY
  FROM PAYROLL, PERSONNEL
  WHERE EMPNO=EMPNUMB AND SALARY > 30000.00
```

```
SELECT * FROM DEPTSALARY


   DEPTNO       DEPTNAME                  TOTSAL
   ------       ----------------      ----------
     10         MANUFACTURING         1000000.00
     20         ADMINISTRATION         300000.00
     30         MARKETING              250000.00
     ...
```

Views With Check Option

- Specifies the constraint that every row that is inserted or updated through the view must conform to the definition of the view
- A row that does not conform to the definition of the view is a row that does not satisfy the search conditions of the view
- Example:

  -
    ```
    CREATE VIEW emp_view2
       ( empno, empname, deptno ) AS
       ( SELECT id, name FROM employee WHERE dept = 10 )
       WITH CHECK OPTION;
    ```
- When this view is used to insert or update with new values, the WITH CHECK OPTION will restrict the input values for the dept column

CASCADED and LOCAL Check Option

- If a view is defined based on another view or a table with check constraints, it is possible to inherit or not to inherit the search condition, two options available:
  - WITH CASCADED CHECK OPTION (default)
  - WITH LOCAL CHECK OPTION
- Example:
  - CREATE VIEW emp_view3 AS

    ```
    ( SELECT empno, empname, deptno FROM emp_view2 WHERE empno > 20 )
    WITH CASCADED CHECK OPTION ;
    ```
  - Conditions deptno = 10 AND empno > 20 will be checked for insert and update operations against this view
- Example:
  - CREATE VIEW emp_view4 AS

    ```
    ( SELECT empno, empname, deptno FROM emp_view3
      WHERE name = 'Smith'  )
    WITH LOCAL CHECK OPTION ;
    ```
  - Only condition name='Smith' (defined in emp_view4) is checked for inserts and updates

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

Creating Indexes

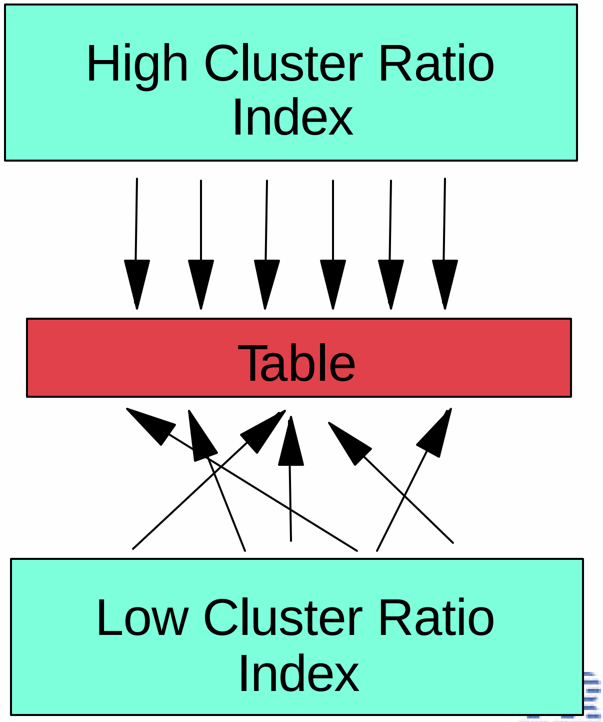
- Index Characteristics:
  - ascending or descending
  - unique or non-unique
  - compound
  - cluster
  - bi-directional - may specify ALLOW or DISALLOW REVERSE SCANS
  - include columns - may only be used if UNIQUE is specified
- Examples:

```
create unique index itemno on albums (itemno) desc
create index clx1 on stock (shipdate) cluster allow reverse scans
create unique index incidx on stock (itemno) include (itemname)
create index item on stock (itemno) disallow reverse scans collect detailed statistics
```

- RENAME INDEX xyz TO pdq
  - Allows to create new index, remove old, rename new name to old name for consistency

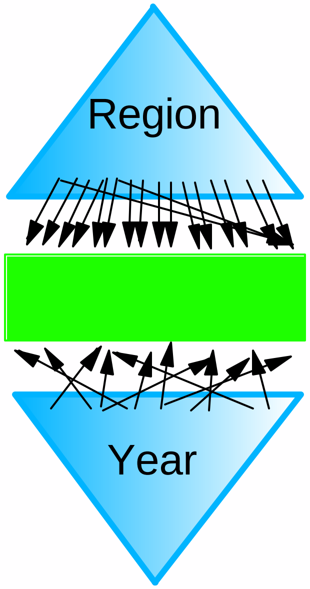
Clustering Index

- DB2 will attempt to store rows with equal or near key values physically close together
  - Improve range searches (e.g. BETWEEN clauses)
  - Only one clustering index may exist for a table
  - Cannot be created on table with APPEND ON
  - Cluster ratio - the degree of data clustering of the index in percentage
  - Higher clustering means rows are ordered on the data pages in index key sequence
  - Cluster factor - a more detailed measurement than the cluster ratio



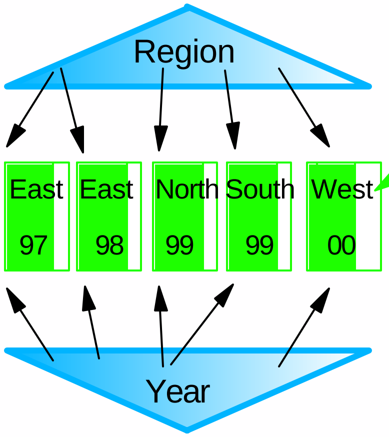
Multi-Dimensional Clustering

- Multi-dimensional Clustering
  - Provides range partitioning on multiple dimensions
  - Reduces need for indexing
  - Roll-in / roll-out improvements



Prior to MDC

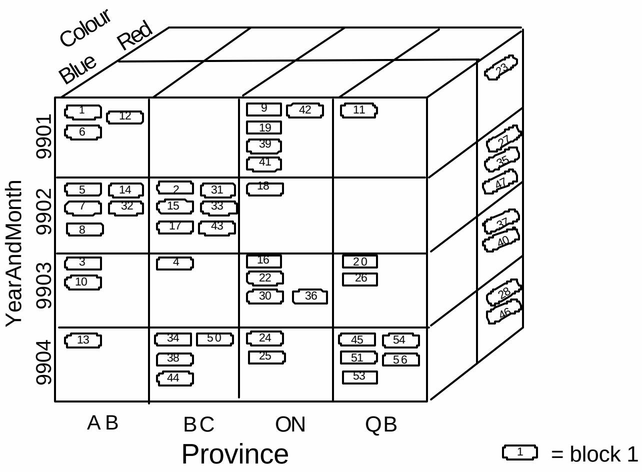
- Clustering in one dimension only
- clustering NOT guaranteed (degrades once page free space is exhausted)



All records in this block are from the **West** region and from the year **2000**

With MDC

- Clustering guaranteed !
- Smaller indexes
- Faster query response
- Simple definition syntax
- Fast roll-in & roll-out



```
CREATE TABLE MDC1 (
   Date DATE,
   Province CHAR(2),
   Color VARCHAR(10),
   YearAndMonth generated as INTEGER(Date)/100, ... )
DIMENSIONS ( YearAndMonth, Province, Colour )
```

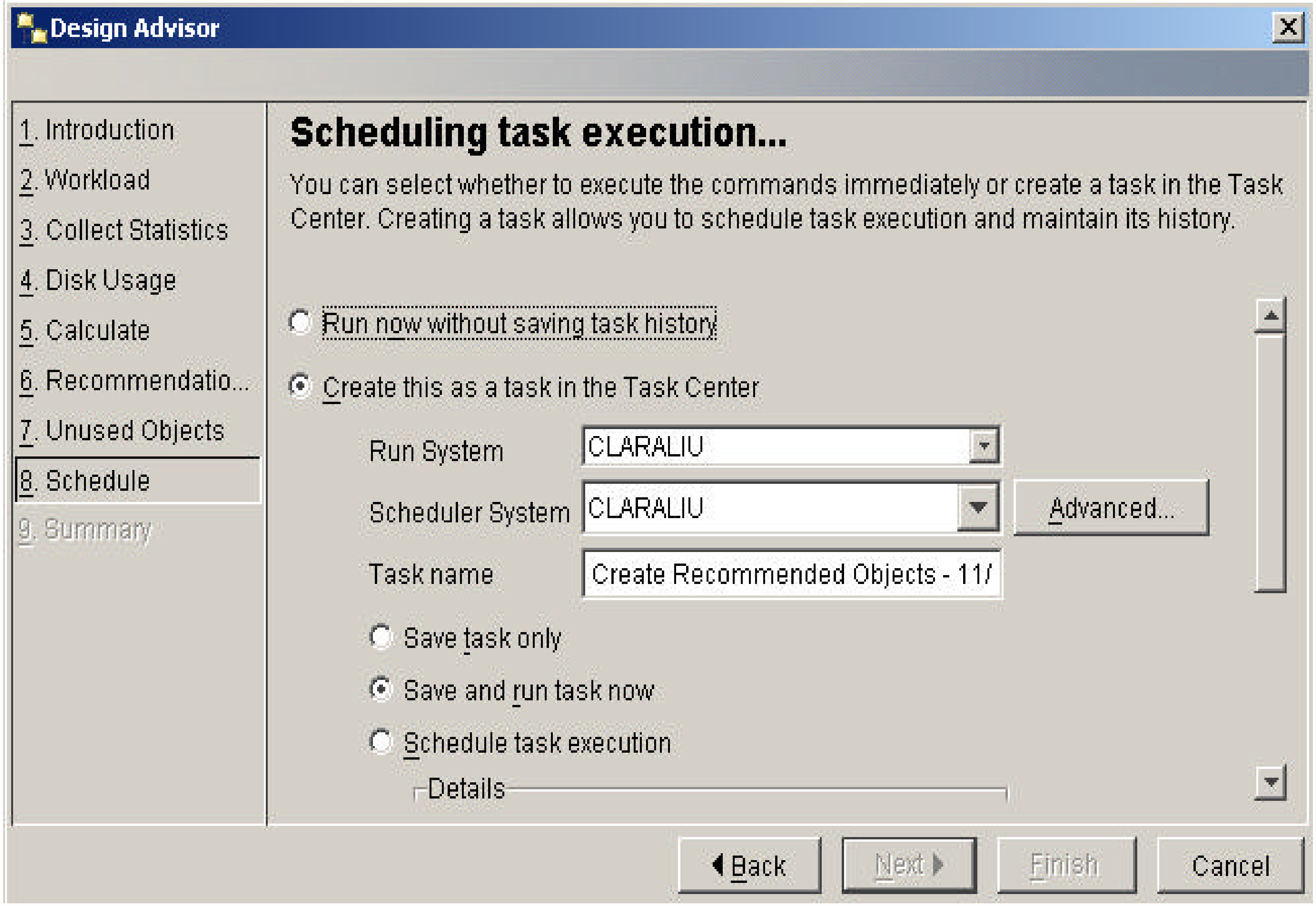
Type-2 Indexes

- Version 8 adds support for type-2 indexes, advantages are:
  - Improve concurrency because the use of minimal next-key locking
  - An index can be created on columns that have a length greater than 255 bytes
  - Allowed online table reorg and online table load to be used on the table
  - Allowed usage of the new multidimensional clustering facility
- All new indexes are created as type-2 indexes
- If type-1 indexes already exist in a table, new index will also be a type-1 index because type-1 and type 2 indexes cannot coexist on a table
- All indexes created before Version 8 were type-1 indexes
- To convert type-1 indexes to type-2 indexes, use the REORG INDEXES command
  - REORG INDEXES ALL FOR TABLE <tablename> CONVERT
- To find out what type of index exists for a table, use the INSPECT command
  - INSPECT CHECK TABLE NAME <tablename> INDEX NORMAL RESULTS <filename>
  - See Chapter 10 for more information

Design Advisor

- Design Advisor can help you design and define suitable indexes:
  - Find the best indexes for a problem query
  - Find the best indexes for a set of queries (a workload), subject to resource limits which are optionally applied
  - Test an index on a workload without having to create the index
- Can be invoked using either:
  - Control Center
  - db2advis command
- Design Advisor Graphical Interface allows you to:
  - Specify the workload for which indexes are to be advised
  - Specify the SQL statement whose indexes are to be advised
  - Specify the input file containing one or more SQL statements
  - Specify the maximum space to be used for all recommended indexes in the existing schema
  - Specify the maximum allowable time (in minutes) to complete the operation
  - Save the script to create the recommended objects in outfile
  - Update the catalog statistics
  - Create and schedule a task to create the recommeded indexes (if any)

Design Advisor GUI



Scheduling task execution...

You can select whether to execute the commands immediately or create a task in the Task Center. Creating a task allows you to schedule task execution and maintain its history.

○ Run now without saving task history

● Create this as a task in the Task Center

Run System: CLARALIU

Scheduler System: CLARALIU

Task name: Create Recommended Objects - 11/

○ Save task only

● Save and run task now

○ Schedule task execution

Design Advisor - db2advis Command

```
>>-db2advis---d--database-name--+-------------------+----------->
                                +--w--workload-name-+
                                +--s--"statement"---+
                                +--i--filename------+
                                '--g----------------'

>--+-------------------------+--+----------------+-------------->
   '--a--userid--+---------+-'  '--l--disk-limit-'
                 '-/passwd-'

>--+---------------------+--+----+--+----+--+-------------+----><
   '--t--max-advise-time-'  '--h-'  '--p-'  '--o--outfile-'
```

- -d database-name
  - Specifies the database name
- -w workload-name
  - Specifies the name of the workload for which indexes are to be advised
- -s "statement"
  - Specifies the text of a single SQL statement whose indexes are to be advised
- -i filename
  - Specifies the name of an input file containing one or more SQL statements
- -l disk-limit
  - Specifies the maximum space to be used for all recommended indexes in the existing schema
- -t max-advise-time
  - Specifies the maximum allowable time (in minutes) to complete the operation
- -o outfile
  - Saves the script to create the recommended objects in outfile

Design Advisor - Examples

- Example #1:
  - db2advis -d prototype -s "SELECT * FROM addresses a
                                              WHERE a.zip IN ('93213', '98567', '93412')
                                              AND (company LIKE 'IBM%' OR company LIKE '%otus')"
    - The utility connects to the PROTOTYPE database, and recommends indexes for the ADDRESSES table
- Example #2:
  - db2advis -d prototype -w production -l 53 -t 20
    - The utility connects to the PROTOTYPE database, and recommends indexes that will not exceed 53MB for queries and workload name is "production", the maximum allowable time for finding a solution is 20 minutes

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

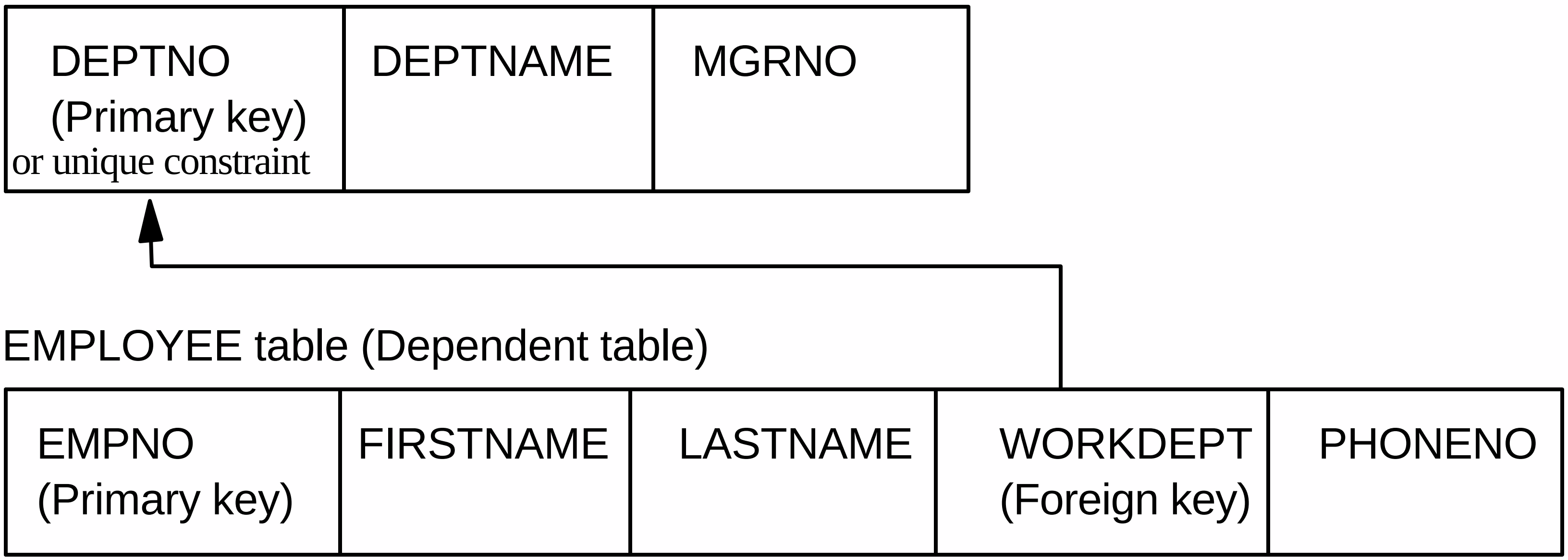
Referential Integrity

- Referential Integrity or Referential Constraints are established with the
  - Primary Key clause
  - Unique constraint clause
  - Foreign Key clause
  - References clause

- In the CREATE/ALTER TABLE statements

```
create table artists (artno INT, .............
    primary key (artno)
    foreign key dept (workdept)
    references department on delete no action)
in DMS01 ;
```

Referential Integrity Example



DEPARTMENT table (Parent table)

DEPTNO (Primary key) or unique constraint	DEPTNAME	MGRNO

EMPLOYEE table (Dependent table)

EMPNO (Primary key)	FIRSTNAME	LASTNAME	WORKDEPT (Foreign key)	PHONENO

```
CREATE TABLE artists ( artno INT, .............
    PRIMARY KEY (artno)
    FOREIGN KEY dept (workdept)
    REFERENCES department ON DELETE NO ACTION )
IN DMS01
```

Referential Integrity Rules

- Insert Rules
  - Rule is implicit when a foreign key is specified.
  - backout insert if not found
- Delete Rules
  - Restrict
    - Parent row not deleted if dependent rows are found.
  - Cascade
    - Deleting row in parent table automatically deletes any related rows in dependent tables.
  - No Action (default)
    - Enforces presence of parent row for every child after all other referential constraints applied
  - Set Null
    - Foreign key fields set to null; other columns left unchanged.
- Update Rules
  - Restrict
    - Update for parent key will be rejected if row in dependent table matches original values of key.
  - No Action (default)
    - Update will be rejected for parent key if there is no matching row in dependent table.

Unique Constraints

Unique Key	Unique Index	Primary Key	Unique Constraints
all values of the key are unique	can have multiple unique index in a table	can only have one primary key in a table	created when primary key or unique clause is used
cannot contain NULL	allow only one NULL value	cannot contain NULL	cannot contain NULL
		it is a type of unique index	if an index already exists, unique index is created
			if an index does not already exist, primary key is created
			can have multiple unique constraint in a table but only one can be primary key
			cannot have more than one unique constraint on the same set of columns

Check Constraints

- Enforce data integrity at a table level
- Once defined every update/insert must conform, otherwise it will fail

```
CREATE TABLE artists
(artno               SMALLINT NOT NULL,
 name                VARCHAR(50) WITH DEFAULT 'abc',
 classification      CHAR(1) NOT NULL,
 bio                 CLOB(100K) LOGGED,
 picture             BLOB(2M) NOT LOGGED COMPACT )
   CONSTRAINT classify
   CHECK (classification IN ('C','E','P','R'))
 IN dms01
```

If some rows do not meet the constraint then it will fail.

You can turn off checking, add the data and then add the constraint, but the table will be placed in CHECK PENDING.

To modify a constraint you must drop it and create a new constraint.

Informational Constraints

- Rules that can be used in query rewrite but are not enforced
  - Standard constraints may result in the overhead for Insert/Update/Delete operations
  - A better alternative if application already verifies data
  - Informational constraint can be used by the SQL compiler but is not enforced by the database manager
  - The SQL compiler includes a rewrite query stage which transforms SQL statements into forms that can be optimized and improve data access path

- Constraint Options
  - ENFORCED
    - The constraint is enforced by the database manager during normal operations such as insert, update, or delete
  - NOT ENFORCED
    - When used, DB2 may return wrong results when any data in the table violates the constraint
  - ENABLE QUERY OPTIMIZATION
    - The constraint can be used for query optimization under appropriate circumstances
  - DISABLE QUERY OPTIMIZATION
    - The constraint can not be used for query optimization

Informational Constraints - Example

```
CREATE TABLE artists
   ( artno              SMALLINT NOT NULL,
     name               VARCHAR(50) WITH DEFAULT 'abc',
     classification     CHAR(1) not null,
    CONSTRAINT classify
   CHECK (classification IN ('C','E','P','R') )
   NOT ENFORCED ENABLE QUERY OPTIMIZATION )
```

- INSERT INTO artists
  VALUES (1, 'SMITH', 'C'), (2, 'DONALD', 'P'), (3, 'MAX', 'E') ;
- INSERT INTO artists VALUES (4, 'ELLIOT', 'X') ;
- SELECT * FROM artists ;

```
ARTNO  NAME        CLASSIFICATION
------ ----------- --------------
     1 SMITH       C
     2 DONALD      P
     3 MAX         E
     4 ELLIOT      X

4 record(s) selected.
```

- SELECT * FROM artists WHERE classification = 'X' ;

```
ARTNO  NAME        CLASSIFICATION
------ ----------- --------------
     4 ELLIOT      X
1 record(s) selected.
```

- DELETE FROM artists WHERE classification = 'X' ;

```
DB20000I  The SQL command completed successfully.
```

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

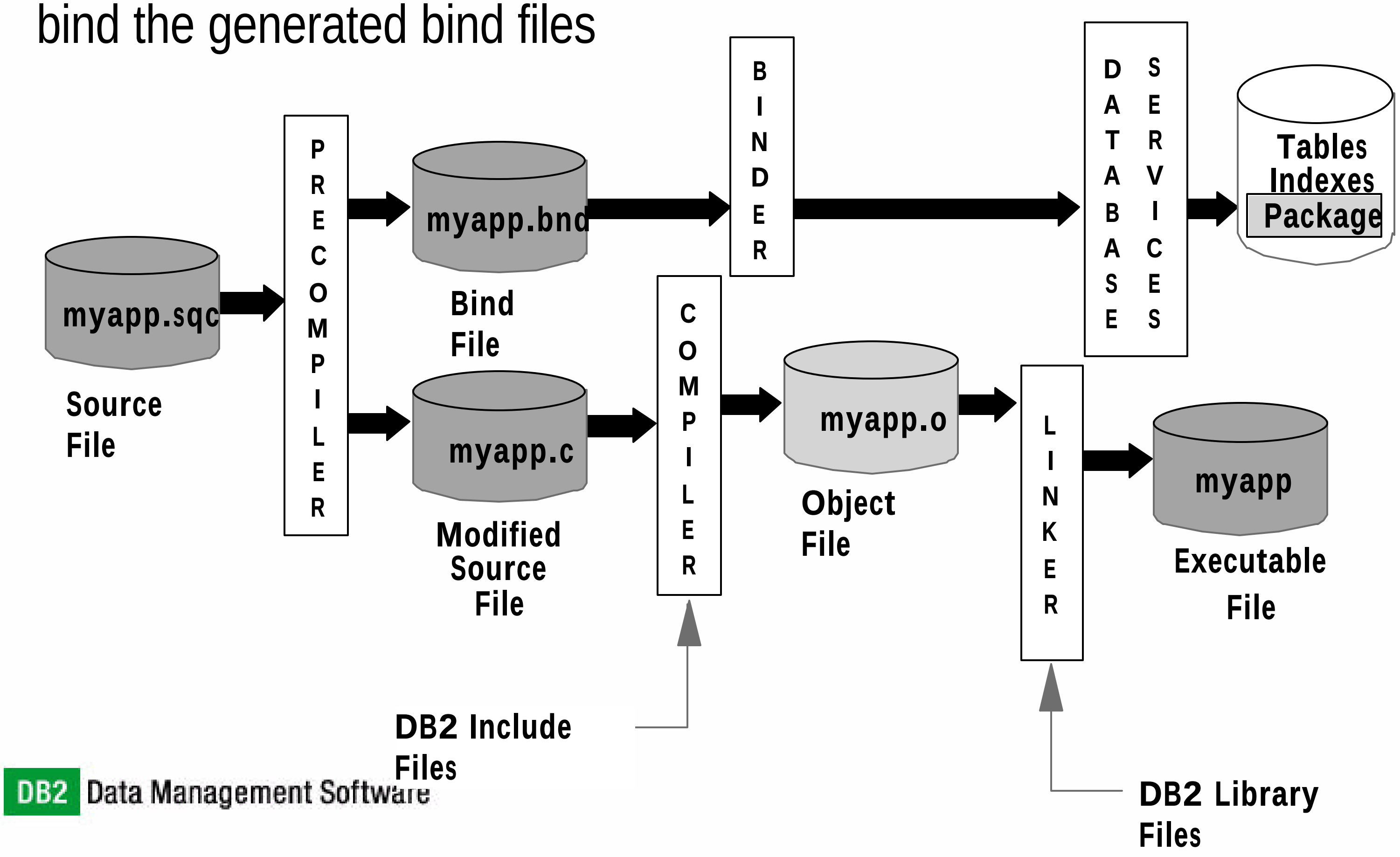
Packages

Triggers, Functions, Stored Procedures

Federated Database Support

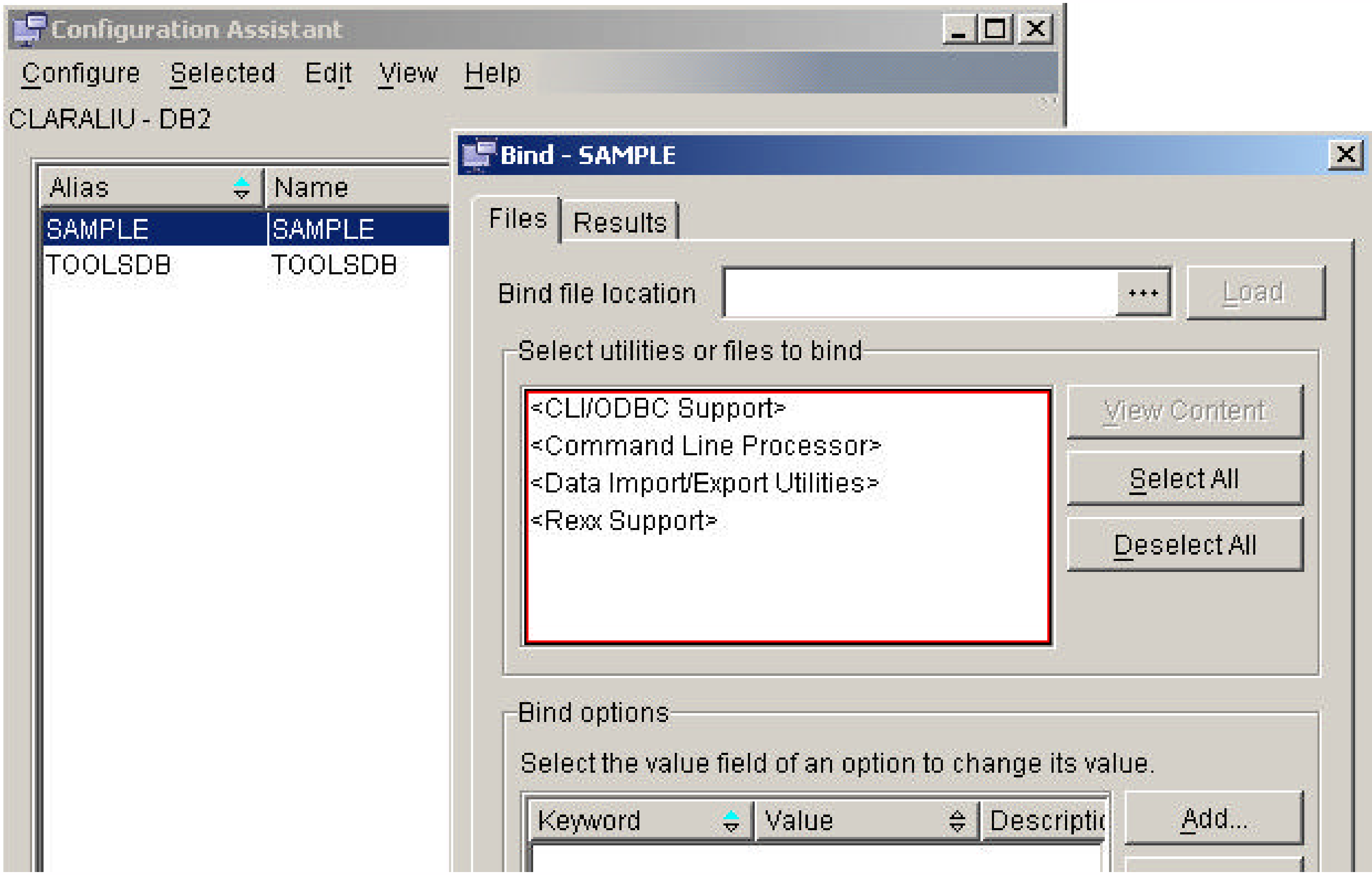
Packages

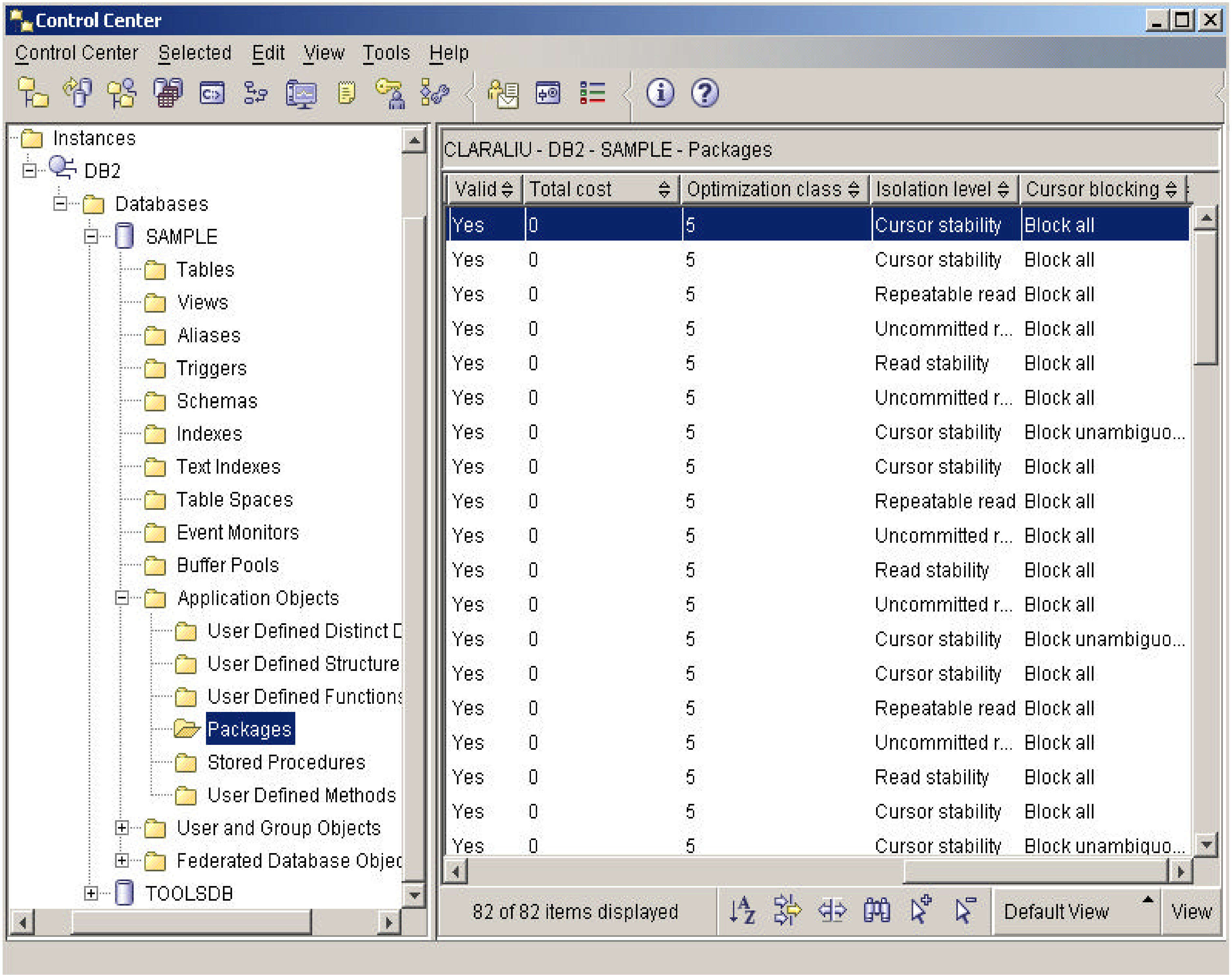
- A package is a database object that contains information needed to execute specific SQL statements in a single source file
- A database application uses one package for every precompiled source file that contains static or dynamic SQL statements
- Packages are created by running the precompiler against a source file and bind the generated bind files



Binding a Bind File

- The BIND Command:
  - BIND <bind filename>
- Use the Configuration Assistant:





CLARALIU - DB2 - SAMPLE - Packages

Valid	Total cost	Optimization class	Isolation level	Cursor blocking
Yes	0	5	Cursor stability	Block all
Yes	0	5	Cursor stability	Block all
Yes	0	5	Repeatable read	Block all
Yes	0	5	Uncommitted r...	Block all
Yes	0	5	Read stability	Block all
Yes	0	5	Uncommitted r...	Block all
Yes	0	5	Cursor stability	Block unambiguo...
Yes	0	5	Cursor stability	Block all
Yes	0	5	Repeatable read	Block all
Yes	0	5	Uncommitted r...	Block all
Yes	0	5	Read stability	Block all
Yes	0	5	Uncommitted r...	Block all
Yes	0	5	Cursor stability	Block unambiguo...
Yes	0	5	Cursor stability	Block all
Yes	0	5	Repeatable read	Block all
Yes	0	5	Uncommitted r...	Block all
Yes	0	5	Read stability	Block all
Yes	0	5	Cursor stability	Block all
Yes	0	5	Cursor stability	Block unambiguo...

82 of 82 items displayed

DB2 Data Management Software

IBM

e business software

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

IBM Software Group

Triggers

- A trigger defines a set of actions that are activated or triggered by an update operation on a specified base table.
- Actions may
  - cause other changes to the database
  - raise an exception
- Use to
  - VALIDATION
    - Similar to constraints but more flexible
  - CONDITIONING
    - Allows new data to be modified/conditioned to a predefined value.
  - INTEGRITY
    - Similar to RI but more flexible
- Three types of triggers:
  - INSERT
  - UPDATEs
  - DELETEs
- A trigger can be fired BEFORE or AFTER an event

AFTER Trigger

- A trigger is defined to set the value of the column *passfail* dependent on the score attained

```
CREATE TRIGGER passfail AFTER INSERT ON test_taken
    REFERENCING NEW AS N
    FOR EACH ROW MODE DB2SQL

    UPDATE test_taken
        SET PASS_FAIL = CASE
        WHEN N.SCORE >=
            ( SELECT CUT_SCORE FROM TEST
                WHERE NUMBER = N.NUMBER ) THEN 'P'
        WHEN N.SCORE <
            ( SELECT CUT_SCORE FROM TEST
                WHERE NUMBER = N.NUMBER) THEN 'F'
        END
    WHERE N.CID = CID
      AND N.TCID = TCID
      AND N.NUMBER = NUMBER
      AND N.DATA_TAKEN = DATA_TAKEN
```

BEFORE Trigger

- The triggers are defined to prevent a booking either before 09:00 or after 17:00

```
CREATE TRIGGER pre9 NO CASCADE BEFORE INSERT ON test_taken
    REFERENCING NEW AS N
    FOR EACH ROW MODE DB2SQL
    WHEN (N.START_TIME < '09:00:00')
        SIGNAL SQLSTATE '70003'
          ('Cannot assign seat before 09:00!')

CREATE TRIGGER aft5 NO CASCADE BEFORE INSERT ON test_taken
    REFERENCING NEW AS N
    FOR EACH ROW MODE DB2SQL
    WHEN (N.START_TIME +
         (SELECT SMALLINT(LENGTH) FROM test
          WHERE NUMBER = N.NUMBER) MINUTES > '17:00:00')
        SIGNAL SQLSTATE '70004'
          ('Cannot assign seat after 17:00!')
```

INSTEAD OF Triggers

- Use view as single interface for ALL SQL operations
- Specifies that the associated triggered action replaces the action against a view
- Only one INSTEAD OF trigger is allowed for each kind of operation (i.e. INSERT, UPDATE, DELETE) on a given view
- Restrictions:
  - The WHEN clause may not be specified for INSTEAD OF triggers (SQLSTATE 42613).
  - FOR EACH STATEMENT cannot be specified

INSTEAD OF Triggers - Examples

```
CREATE TRIGGER student_v_insert
    INSTEAD OF INSERT ON student_v
    REFERENCING NEW AS N DEFAULTS NULL
    FOR EACH ROW MODE DB2SQL
    BEGIN ATOMIC
        INSERT INTO students VALUES ( n.name, n.studentid ) ;
        INSERT INTO person VALUES ( n.name, n.studentid, n.age, n.enrolldate ) ;
    END

CREATE TRIGGER student_v_delete
    INSTEAD OF DELETE ON student_v
    REFERENCING OLD AS O
    FOR EACH ROW MODE DB2SQL
    BEGIN ATOMIC
        DELETE FROM students WHERE id = o.studentid ;
        DELETE FROM PERSON WHERE name = o.name ;
    END
```

Functions

- DB2 UDB provides three types of functions:
  - Scalar or Row Functions
    - Provide a value for each row in the result set
    - Date/Time, Mathematical, Character, etc
  - Column or Vector Functions
    - Provide a value based on a group of rows
    - Count, Min, Max, Avg, etc
  - Table Functions
    - Returns columns of a table, resembling a table created by a simple CREATE TABLE statement
- User Defined Functions can be in any of those types
  - Mechanism for creating extensions to SQL
  - Functions can be written in C, Java, OLE, and SQL

Functions

- Example:

```
CREATE FUNCTION TAN (X DOUBLE)
      RETURNS DOUBLE
      LANGUAGE SQL
      CONTAINS SQL
      NO EXTERNAL ACTION
      DETERMINISTIC
      RETURN SIN(X)/COS(X)
```

- Usage:

```
SELECT tan(rect_length) FROM shapes;
```

Stored Procedures

- Perform intermediate processing avoiding transmitting data across network
- Centralized administration and maintenance
- Execute on the database server
- Stored procedures can be written in C, Java, COBOL, OLE, and SQL
- Example:

```
CREATE PROCEDURE update_salary
    ( IN employee_number CHAR(6)
    , IN rate INTEGER
    , OUT newsalary INTEGER )
    LANGUAGE SQL
    BEGIN
        UPDATE emp
          SET salary = salary  * ( 1.0 + rate / 100.0 )
          WHERE empno = employee_number ;
        SELECT salary INTO newsalary
          FROM emp WHERE empno = employee_number ;
    END
```

- Usage:

```
CALL update_salary ( 3422, 50, ? );
```

Chapter 5: Database Objects

Buffer Pools and Table Spaces

Schemas and Catalogs

Data Types

Tables, Identity Columns, Temporary Tables

Views

Indexes

Constraints

Packages

Triggers, Functions, Stored Procedures

Federated Database Support

Federated Database

- DB2 federated database support is part of the IBM Information Integrator
- Allows access to DB2 family databases and non-DB2 databases

- DB2 has federated built-in support for:
  - DB2 for iSeries
  - DB2 for zSeries
  - Informix

- Relational Connect adds transparent access to other databases:
  - Oracle
  - Sybase
  - Microsoft SQL Server

Setting Up a Federated Database System

- Create WRAPPER
  - Routines stored in a library that allows the federated server to perform operations such as connecting to a data source and retrieving data from it iteratively
  - **CREATE WRAPPER DRDA LIBRARY 'libdb2drda.a' ;**

- Create SERVER
  - The SERVER defines the data source to the federated database with information that pertains to the data source
  - **CREATE SERVER crandall**

    ```
    CREATE SERVER crandall
      TYPE DB2/MVS  VERSION 4.1
      WRAPPER DRDA
      AUTHORIZATION userid PASSWORD passwd
      OPTIONS ( ... ) ;
    ```

Setting Up a Federated Database System *(continued)*

- Create User Mapping
  - An association between the federated server and the data source user ID and password
  - Needs to define user mapping so that the federated server can pushdown requests to the data source if required

  -
    ```
    CREATE USER MAPPING FOR user3
       SERVER s1
       OPTIONS
       ( REMOTE_AUTHID 'SYSTEM'
       , REMOTE_PASSWORD 'MANAGER' )
    ```

- Create NICKNAME
  - An identifier used to reference the object located at the data source that will be accessed

  -
    ```
    CREATE NICKNAME dept
       FOR os390a.hedges.department
    ```

Object Definition Review

- You can CREATE or DROP the following objects:

Table	View	Alias
Bufferpool	Schema	Event Monitor
UDF	Trigger	Table Space
Index	UDT	Stored Procedure
Nickname	Wrapper	Server

- But you can only ALTER:

Table	Table Space	Nickname
Type	Buffer Pool	Wrapper
View		Server

Only for structured types

Only for views built on typed tables