DB2 Data Management Software

IBM

e business software

DB2 Version 8 Database Objects

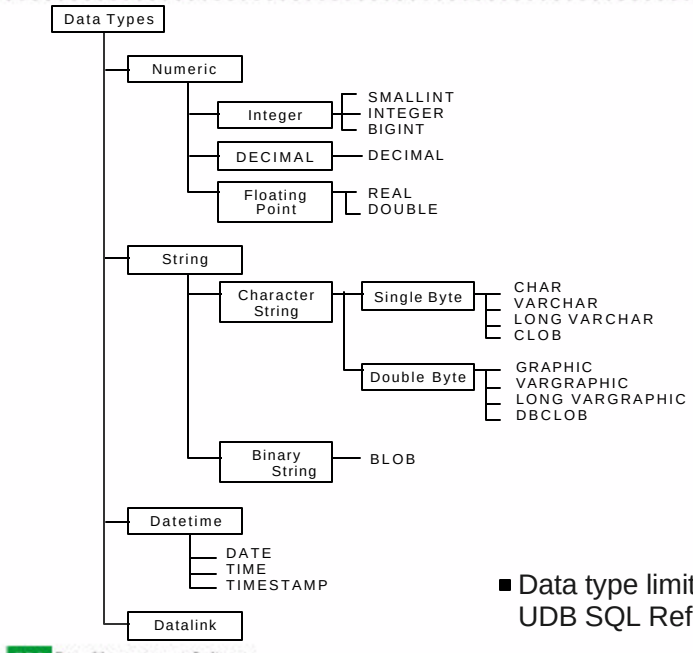
DB2 Quickstart Education

Maintained by Paul Yip (ypaul@ca.ibm.com)

February 2003

IBM Software Group

Data Types



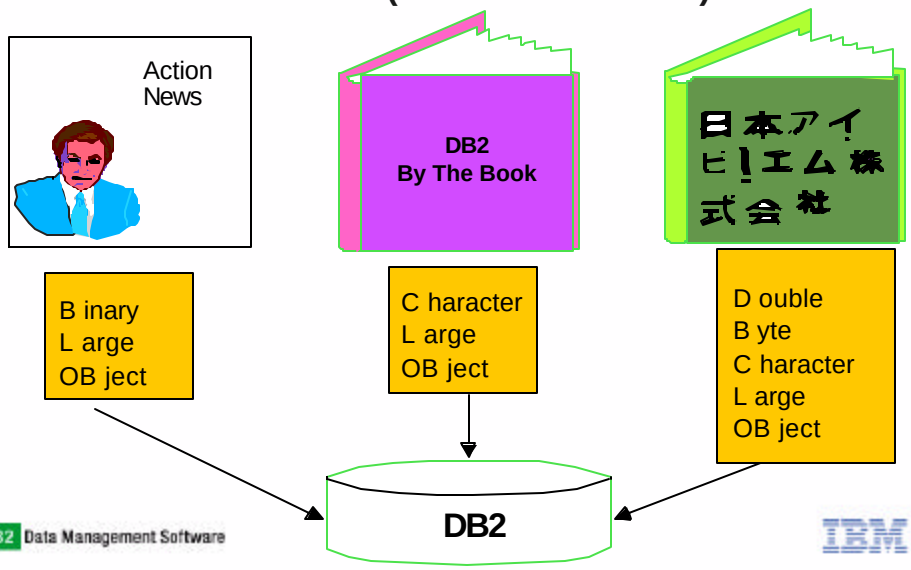
- Data type limits defined in DB2 UDB SQL Reference

DB2 Data Management Software

IBM

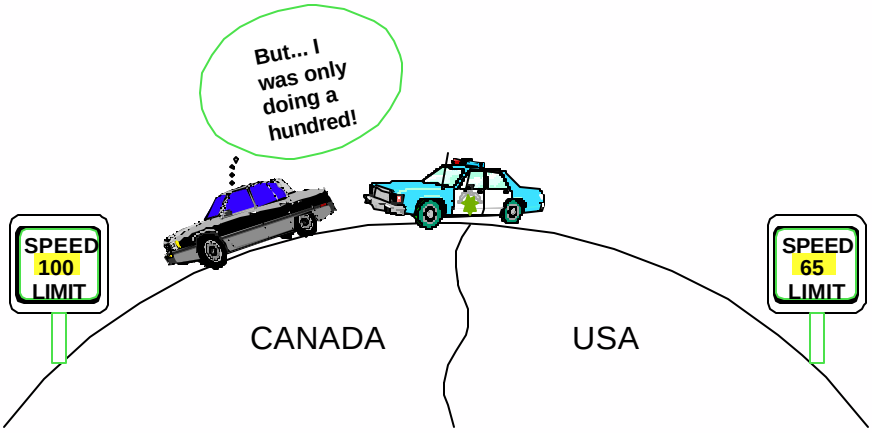
Large Objects - The Need

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



User-Defined Types - The Need

- Need to establish context for values
- DB2 enforced typing



User-Defined Types - Definition

```
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?	CHAR
Is the data variable in length?	VARCHAR
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

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

```
CREATE TABLE Staff (
    ID SMALLINT NOT NULL, NAME VARCHAR(9) ,
    DEPT SMALLINT not null with default 10,
    JOB CHAR(5) , YEARS SMALLINT ,
    SALARY DECIMAL(7,2) ,
    COMM DECIMAL(7,2) with default 15 )
```


Null and Default Value Compression

- Reduce storage for typical data warehousing scenarios
  - Increase performance of large scans
- Available for all tables except global temporary tables
- Must use system default value, not user-defined values
- Eligible datatypes
  - Numeric
  - Char
  - Varchar
  - DBCS (fixed and variable)
  - BLOB
- Not supported
  - Date
  - Time
  - Timestamp
  - These values are dynamic and are always changing

System Default Value Compression

- Example:

```
CREATE TABLE COMP_T1 (
    C1 INTEGER NOT NULL COMPRESS SYSTEM DEFAULT,
    C2 CHARACTER (10) COMPRESS SYSTEM DEFAULT,
    CONSTRAINT COMP_T1MPK PRIMARY KEY (C1)
    )
    VALUE COMPRESSION
```

- Enable existing table for compression:
  alter table t1 activate value compression
  alter table t1 alter column <column> compress system default
  reorg table t1 (if already populated)

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
```

DB2 Data Management Software

IBM

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
```

DB2 Data Management Software

IBM

SEQUENCE objects

- Unlike identity columns, sequences are independent of tables
- example:

```
CREATE SEQUENCE myseq
START WITH 1
INCREMENT BY 1
NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq, ...)

SELECT prevval for myseq FROM sysibm.sysdummy1
```


Create Table Command

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

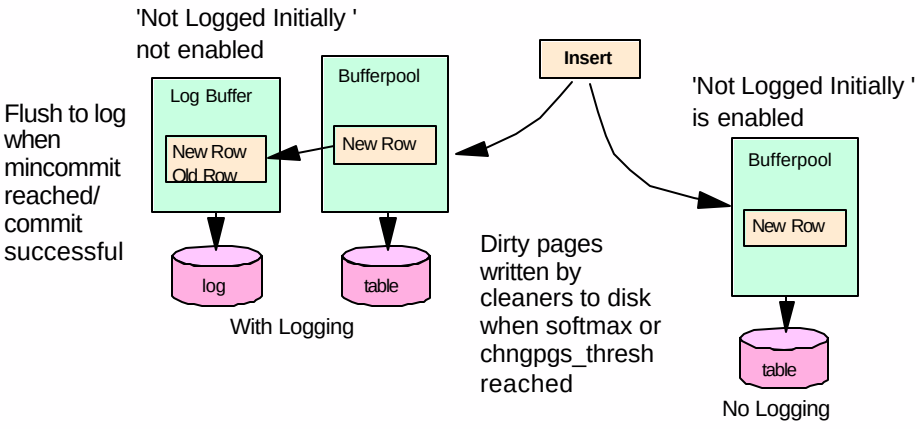
```
connect to eddb;
create table artists
(artno           smallint not null,
 name            varchar(50) with default 'abc',
 classification  char(1) not null,
 bio             clob(100K) logged,
 picture         blob(  2M) not logged compact)

 in dms01
 index in dms02
 long  in dms03

 not logged initially;
```


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:
  1. In the IBMDEFAULTGROUP table space (if it exists and if the page size is sufficient)
  2. In a user created table space which is of the smallest pagesize that is sufficient for the table.
  3. Then it will go in USERSPACE1 (if it exists and has a sufficient page size)
- The moral of the story: if there are multiple table spaces, then specify into which table spaces a table's data, indexes, and LOB field data is to be placed.

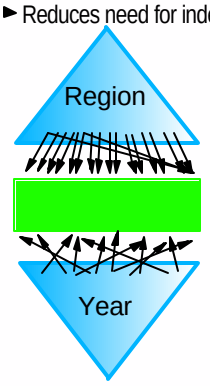
'Not Logged Initially' Tables

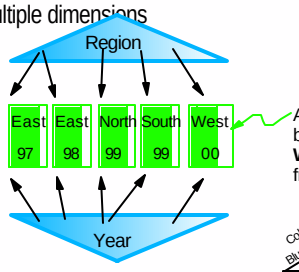
- **Useful for situations needing to insert large amounts of data from alternate source (another table or file)**
- **Data inserted without logging**
- **Use when recovery of table not required**



Multi-Dimensional Clustering

- Multi-dimensional Clustering
  - Provides range partitioning on multiple dimensions
  - Reduces need for indexing

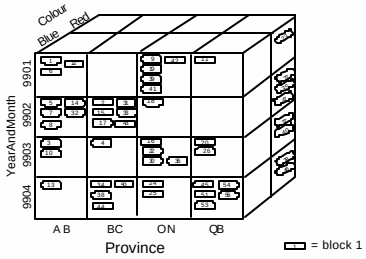




All records in this block are from the **West** region and from the year **2000**

With MDC

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



Prior to MDC

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

MDC Example

- example:

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


Declared 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
- Logging
  - ►NOT LOGGED clause mandatory in V7, and now option in V8
- Automatic cleanup
- Performace
  - ►avoid catalog contention
  - ►no locking of rows
  - ►no logging (but logging is optional)
  - ►no authority checking
- 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;
```


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
- 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
  - **CREATE TABLE** t1new
    AS
    (SELECT C1, C8, C10 FROM t1)
    DEFINITION ONLY;

Creating Views

- Data for view not stored separately
- Nested view supported
- 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
...		

Creating Indexes

- Index Characteristics:
  - ascending or descending
  - Unique or non-unique
  - compound
  - cluster
  - bi-directional
  - include columns
- Examples:

```
create unique index itemno on albums (itemno) desc

create index item on stock (itemno) allow reverse scans

create index clx1 on stock (shipdate) cluster allow reverse scans

create unique index incidx on stock (itemno)  include (shipdate)
```


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          .....
  primary key (artno)
   foreign key dept (workdept)
   references department on delete no action)
in DMS01
```

DB2

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.

Referential Integrity Rules (cont...)

- 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.

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
- 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,
 bio             clob(100K) logged,
 picture         blob(2M) not logged compact)
CONSTRAINT classify
CHECK (classification in ('C','E','P','R')
NOT ENFORCED ENABLE QUERY OPTIMIZATION)
 in dms01
```