

**SYBASE TO UDB**

**By**

**Database Migration CoE**

**Securities Vertical**

**Version 1.0**

**Dated Oct – 2005**

**Confidentiality**

**Wipro Technologies** explicit understanding is that the contents of the document would not be divulged to any third party without prior written consent from Wipro Technologies.

**Revision History**

| S.No. | Ver. No. | Impacted Section | Description of Change | Remarks |
| --- | --- | --- | --- | --- |
| 1. | 1.0 | Baseline Version |  |  |
|  |  |  |  |  |

**Table of Contents**

**1** **Executive Summary**

**2** **Introduction**

2.2.2.16 Solution Overview

2.2.2.16 What is covered in this document

2.2.2.16 What is not covered in this document

2.2.2.16 Document Conventions

2.2.2.16 References

**3** **TOPICS**

3.1 Queries

3.1.1 Select Statements

3.1.2 Subqueries

Subqueries can also be divided as

3.1.3 Correlated Subqueries

3.1.4 Group By

3.2 Built In Functions

3.2.1 System Functions

3.2.2 String Functions

3.2.2.3 SEARCHING FOR A SUBSTRING WITHIN ANOTHER

3.2.2.5 COMPARING THE SOUNDEX VALUES OF TWO STRINGS

3.2.2.6 CONVERTING A STRING TO LOWER CASE

3.2.2.9 REPLICATING A STRING

3.2.2.10 REVERSING A STRING

3.2.2.11 EXTRACTING A SUBSTRING FROM A STRING

3.2.2.12 TRIMMING A STRING

3.2.2.13 SOUNDEX VALUES

3.2.2.14 CREATING A STRING OF SPACES

3.2.2.15 CONVERTING A FLOATING POINT NUMBER TO A STRING

3.2.2.16 STUFFING ONE STRING WITHIN ANOTHER

3.2.3 Text Functions

3.2.4 Aggregate Functions

3.2.5 Date Functions

3.3 Joins

3.3.1 Datatypes in Joins

3.3.2 Equi/Natural Joins

3.3.3 Joins based on Equality

3.3.4 Self Joins

3.3.5 Not equal Joins

3.3.6 Outer Joins

3.4 Stored Procedures

3.4.1 Using Default Parameters

3.4.2 Null as Default Parameters

3.4.3 Wildcard as Default Parameters

3.4.4 Procedure Groups

3.4.5 Using Recompile

3.4.6 Using Recompile in Execute

3.4.7 Nesting Procedures

3.4.8 Temp Tables

3.4.9 Setting options in stored procedures

3.4.10 Arguments for stored procedures

3.4.11 Executing procedures

3.4.12 Cross database references

3.4.13 Executing procedures after a time delay

3.4.14 Executing procedures remotely

3.4.15 Return status

3.4.16 Reserved return status values

3.4.17 User-generated return values

3.4.18 Return parameters

3.4.19 Passing values in parameters

3.4.20 Output variable

3.4.21 Qualifying names inside procedures

3.4.22 Renaming objects referenced by procedures

3.5 TRIGGERS

3.5.1 Creating Triggers

3.5.2 Insert, update triggers

3.5.3 Delete Triggers

3.5.4 if update (column name)

3.5.5 Difference between ROLLBACK TRANSACTION and ROLLBACK TRIGGER

3.6 Datatypes

3.6.1 User Defined Datatypes

3.6.2 System Defined Datatypes

3.7 Tables

3.7.1 Creating Table

3.7.2 Adding, Dropping, Modifying Columns

3.7.3 Using Select Into

3.7.4 User Defined Datatypes

3.8 Cursors

3.9 Views

3.9.1 Creating View

3.9.2 Modifying Underlying Objects

3.9.3 Modifying Data thru Views

3.9.4 Updating Multitable Views

3.10 Rules

3.10.1 Creating

3.10.2 Binding to Columns & User Defined Datatypes

3.10.3 Rules & Nulls

3.11 Defaults

3.11.1 Creating

3.11.2 Binding and Unbinding

3.12 Control Flow

3.12.1 If else case

3.12.2 Begin end

3.12.3 While break continue

3.12.4 Declare goto return

3.12.5 Waitfor print raise error

3.12.6 Local Variables

3.12.7 Global Variables

3.13 Transactions

3.13.1 Isolation Levels

3.13.2 Modes

3.13.3 Begin Commit

3.13.4 Transaction States

3.13.5 Deadlocks

3.14 Temp Tables

3.15 System Procedures

3.16 System Tables

3.17 Utility Commands

3.17.1 BCP

3.17.2 ISQL

3.17.3 DEFNCOPY

3.18 NULL VALUES

Wildcard as Default Parameters

Procedure Groups

Setting options in stored procedures

BCP

DEFNCOPY

Using Recompile

Using Recompile in Execute

Binding to Columns & User Defined Datatypes

Binding and Unbinding

Transaction States

**4.** **glossary of terms**

Appendix A

1. **Executive Summary**

1. **Introduction**

## **Solution Overview**

## **What is covered in this document**

## **What is not covered in this document**

## **Document Conventions**

## **References**

1. **TOPICS**

## **Queries**

### **Select Statements**

DESCRIPTION: The select command retrieves data stored in the rows and columns of database tables using a procedure called a query.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select au\_fname, au\_lname  from authors  where city = "Oakland" | select au\_fname, au\_lname  from authors  where city = "Oakland" |
| RESULT  au\_fname au\_lname  -------------------- ----------------------------------------  Marjorie Green  Dick Straight  Dirk Stringer  Stearns MacFeather  Livia Karsen | AU\_FNAME AU\_LNAME  -------------------- ------------  Marjorie Green  Dick Straight  Dirk Stringer  Stearns MacFeather  Livia Karsen |

CONCLUSION: Results are same in both SYBASE and DB2.

### **Subqueries**

DESCRIPTION

A query sometimes needs information from one table in order to query another table. This type of query which forms a part of another query or subquery are called subquery.They are also known as inner queries.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = “New Age Books” ) | select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = 'New Age Books' ) |
| RESULT:  title\_id title price -------- ----------------------------- -------  BU2075 You Can Combat Computer Stress! 23.92  PS2091 Is Anger the Enemy? 87.60  PS2106 Life Without Fear 56.00  PS3333 Prolonged Data Deprivation: Four Case Studies 159.92  PS7777 Emotional Security: A New Algorithm 63.92 | TITLE\_ID TITLE PRICE  -------- ------------------------- ---------------------  BU2075 You Can Combat Computer Stress! 23.9200  PS2091 Is Anger the Enemy? 87.6000  PS2106 Life Without Fear 56.0000  PS3333 Prolonged Data Deprivation: Four Case Studies 159.9200  PS7777 Emotional Security: A New Algorithm 63.9200 |

CONCLUSION: Subqueries are used the same way in both the SYBASE and DB2.

#### **Subquery restrictions in Sybase and DB2:**

* **Select list in a Subquery**

EXAMPLE:

| SYBASE | DB2 |
| --- | --- |
| QUERY:  select titles  from titles  where pub\_id =  ( select \*  from publishers  where pub\_name = “New Age Books” ) | select titles  from titles  where pub\_id =  ( select \*  from publishers  where pub\_name = ‘New Age Books’ ) |
| RESULT:  The symbol '\*' can only be used for a subquery select list when the subquery is introduced with EXISTS or NOT EXISTS. | Multiple columns are returned from a subquery that is allowed only one column. |

Conclusion:The select list in a subquerycan consist of only one column name, except in the exists subquery, where an (\*) is usually used in place of the single column name both in SYBASE and DB2.

* **Subqueries with order by, group by, or compute by list.**

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select title\_id, title  from titles  order by ( select pub\_id  from publishers  where pub\_name = "New Age Books" ) | select title\_id, title  from titles  order by ( select pub\_id  from publishers  where pub\_name = ‘New Age Books’ ) |
| RESULT  Statement fails to execute. | TITLE\_ID TITLE  -------- ---------------------------------------------------------  BU1032 The Busy Executive's Database Guide  BU1111 Cooking with Computers: Surreptitious Balance  BU2075 You Can Combat Computer Stress!  BU7832 Straight Talk About Computers  MC2222 Silicon Valley Gastronomic Treats  MC3021 The Gourmet Microwave  MC3026 The Psychology of Computer Cooking  PC1035 But Is It User Friendly?  PC8888 Secrets of Silicon Valley  PC9999 Net Etiquette |

CONCLUSION: SYBASE doesn’t support subqueries in an ORDER BY clause whereas DB2 allows it.

* **Limit in nesting levels.**

**Conclusion:** There is a limit of 16 nesting levels in SYBASE whereas limit of nesting levels in

DB2 is 14.

[Examples are avoided since the nesting level is high]

* **Subqueries having the order by clause.**

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select title  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = "New Age Books"  order by pub\_id ) | select title  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = "New Age Books"  order by pub\_id ) |
| RESULT  SYBASE does not allow order by inside subqueries. | TITLE  ---------------------------------------------------------------------  You Can Combat Computer Stress!  Is Anger the Enemy?  Life Without Fear  Prolonged Data Deprivation: Four Case Studies  Emotional Security: A New Algorithm |

CONCLUSION: SYBASE does not allow order by inside the subqueries whereas DB2 allows it.

**Types of Subquery:**

* **Expression subqueries**

DESCRIPTION

Expression subqueries are introduced with an unmodified comparison operator, must return a single value, and can be used almost anywhere an expression is allowed in SQL.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = “New Age Books” ) | select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = 'New Age Books' ) |
| RESULT  title\_id title price  -------- ----------------------------- ---------  BU2075 You Can Combat Computer Stress! 23.92  PS2091 Is Anger the Enemy? 87.60  PS2106 Life Without Fear 56.00  PS3333 Prolonged Data Deprivation: Four Case Studies 159.92  PS7777 Emotional Security: A New Algorithm 63.92 | TITLE\_ID TITLE PRICE  -------- -------------------------------------- ----------  BU2075 You Can Combat Computer Stress! 23.9200  PS2091 Is Anger the Enemy? 87.6000  PS2106 Life Without Fear 56.0000  PS3333 Prolonged Data Deprivation: Four Case Studies 159.9200  PS7777 Emotional Security: A New Algorithm 63.9200 |

CONCLUSION: Expression subqueries are same in both SYBASE and DB2

* **Quantified predicate subqueries**

DESCRIPTION

These subqueriesoperate on lists introduced with in or with a comparison operator modified by any or all. Quantified predicate subqueries return 0 or more values. This type is also used as an existence test, introduced with exists.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select pub\_name  from publishers  where exists  (select \*  from titles  where pub\_id = publishers.pub\_id  and type = "business") | select pub\_name  from publishers  where exists  (select \*  from titles  where pub\_id = publishers.pub\_id  and type = ‘business’) |
| RESULT  pub\_id pub\_name  ------ --------  0736 New Age Books  1389 Algodata Infosystems | PUB\_ID PUB\_NAME  ------ ----------------------------------------  0736 New Age Books  1389 Algodata Infosystems |

CONCLUSION: Quantified predicate subqueries exist both in SYBASE and DB2.

### **Subqueries can also be divided as**

* **Noncorrelated subquery**

DESCRIPTION

These are the subqueries which can be evaluated independently. The results of the subquery are substituted in the main statement, or outer query.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = “New Age Books” ) | select title\_id, title, price  from titles  where pub\_id =  ( select pub\_id  from publishers  where pub\_name = 'New Age Books' ) |
| RESULT  title\_id title price  -------- ----------------------------------------- -------  BU2075 You Can Combat Computer Stress! 23.92  PS2091 Is Anger the Enemy? 87.60  PS2106 Life Without Fear 56.00  PS3333 Prolonged Data Deprivation: Four Case Studies 159.92  PS7777 Emotional Security: A New Algorithm 63.92 | TITLE\_ID TITLE PRICE  -------- ------------------------------------------ ---------  BU2075 You Can Combat Computer Stress! 23.9200  PS2091 Is Anger the Enemy? 87.6000  PS2106 Life Without Fear 56.0000  PS3333 Prolonged Data Deprivation: Four Case Studies 159.9200  PS7777 Emotional Security: A New Algorithm 63.9200 |

CONCLUSION: Noncorrelated subqueries are used the same way in both the SYBASE and DB2.

### **Correlated Subqueries**

DESCRIPTION

These are the queries which cannot be evaluated independently but can reference columns in a table listed in the from list of the outer query.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select au\_lname, au\_fname  from authors  where 100 in  (select royaltyper  from titleauthor  where au\_id = authors.au\_id) | select \*  from project  where exists  (select \*  from department  where department.deptno = project.deptno) |
| RESULT  au\_lname au\_fname  -------- --------  White Johnson  Green Marjorie  Carson Cheryl  Straight Dick  Locksley Chastity | AU\_LNAME AU\_FNAME  ---------------------------------------- --------------------  White Johnson  Green Marjorie  Carson Cheryl  Straight Dick  Locksley Chastity |

CONCLUSION: Correlated subqueries exist in both the SYBASE and DB2.

**NOTE**: A subquery can be either Expression or Quantified predicate noncorrelated subquery

or Expression or Quantified predicate correlated subquery.

### **Group By**

**Simple Group By:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, advance from titles group by type | SYNTAX:  select type, advance from titles group by type |
| RESULT:  type advance  ------------ ------------------------  business 5,000.00  business 5,000.00  business 10,125.00  business 5,000.00  mod\_cook 0.00  mod\_cook 15,000.00  UNDECIDED NULL  popular\_comp 7,000.00  popular\_comp 8,000.00  popular\_comp NULL  psychology 7,000.00  psychology 2,275.00  psychology 6,000.00  psychology 2,000.00  psychology 4,000.00  trad\_cook 7,000.00  trad\_cook 4,000.00  trad\_cook 8,000.00  (18 rows affected | RESULT:  TYPE ADVANCE  ------------ ---------------------  business 5000.0000  business 10125.0000  Four Case 2000.0000  mod\_cook 0.0000  mod\_cook 15000.0000  popular\_comp 7000.0000  popular\_comp 8000.0000  popular\_comp -  psychology 2275.0000  psychology 4000.0000  psychology 6000.0000  psychology 7000.0000  trad\_cook 4000.0000  trad\_cook 7000.0000  trad\_cook 8000.0000  UNDECIDED -  16 record(s) selected. |

**Note:** There is difference in the output of Sybase & Db2 because DB2 doesn’t allow, repeating values. Db2 allows only distinct values whereas Sybase allows the repeating values in the output of the group by query.

**Nesting groups with group by:**

You can list more than one column in the group by clause to nest groups.

Once the sets are established with group by, the aggregates are applied.

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select pub\_id, type, avg(price), sum(total\_sales)  from titles  group by pub\_id, type | SYNTAX:  SELECT pub\_id,type,AVG(price),SUM(total\_sales) FROM titles GROUP BY pub\_id,type |
| RESULT:  pub\_id type  ------ ------------ ------------------------ -----------  0736 business 23.92 18722  0736 psychology 91.86 9564  0877 UNDECIDED NULL NULL  0877 mod\_cook 91.92 24278  0877 psychology 172.72 375  0877 trad\_cook 127.71 19566  1389 business 138.48 12066  1389 popular\_comp 171.80 12875  (8 rows affected) | RESULT:  PUB\_ID TYPE 3 4  ------ ------------ --------------------------------- -----------  0736 business 2.9900000000000000 18722  0736 psychology 11.4825000000000000 9564  0877 mod\_cook 11.4900000000000000 24278  0877 psychology 21.5900000000000000 375  0877 trad\_cook 15.9633333333333333 19566  0877 UNDECIDED - -  1389 business 17.3100000000000000 12066  1389 popular\_comp 91.4750000000000000 12875  8 record(s) selected. |

Note: You can nest many groups within groups, up to the maximum of 16 columns or expressions specified with group by.

**Referencing other columns in queries using group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, title\_id, avg(price), avg(advance)  from titles  group by type | SYNTAX:  select type, title\_id, avg(price), avg(advance) from titles group by type, title\_id |
| RESULT:  type title\_id  ------------ -------- ------------------------ ------------------------  business BU1032 109.84 6,281.25  business BU1111 109.84 6,281.25  business BU2075 109.84 6,281.25  business BU7832 109.84 6,281.25  mod\_cook MC2222 91.92 7,500.00  mod\_cook MC3021 91.92 7,500.00  UNDECIDED MC3026 NULL NULL  popular\_comp PC1035 171.80 7,500.00  popular\_comp PC8888 171.80 7,500.00  popular\_comp PC9999 171.80 7,500.00  psychology PS1372 108.03 4,255.00  psychology PS2091 108.03 4,255.00  psychology PS2106 108.03 4,255.00  psychology PS3333 108.03 4,255.00  psychology PS7777 108.03 4,255.00  trad\_cook TC3218 127.71 6,333.33  trad\_cook TC4203 127.71 6,333.33  trad\_cook TC7777 127.71 6,333.33  (18 rows affected) | RESULT:  TYPE TITLE\_ID 3 4  ------------ -------------------- --------------------------------- -  ---------------------  business BU1032 19.9900000000000000  5000.0000000000000000  business BU1111 11.9500000000000000  5000.0000000000000000  business BU2075 2.9900000000000000  0125.0000000000000000  business BU7832 19.9900000000000000  5000.0000000000000000  mod\_cook MC2222 19.9900000000000000  0.0000000000000000  mod\_cook MC3021 2.9900000000000000  5000.0000000000000000  popular\_comp PC1035 22.9500000000000000  7000.0000000000000000  popular\_comp PC8888 160.0000000000000000  8000.0000000000000000  popular\_comp PC9999 -  -  psychology PS1372 21.5900000000000000  7000.0000000000000000  psychology PS2091 10.9500000000000000  2275.0000000000000000  psychology PS2106 7.0000000000000000  6000.0000000000000000  psychology PS3333 19.9900000000000000  2000.0000000000000000  psychology PS7777 7.9900000000000000  4000.0000000000000000  trad\_cook TC3218 20.9500000000000000  7000.0000000000000000  trad\_cook TC4203 11.9500000000000000  4000.0000000000000000  trad\_cook TC7777 14.9900000000000000  8000.0000000000000000  UNDECIDED MC3026 -  -  18 record(s) selected. |

**Expressions and group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select avg(total\_sales), total\_sales \* price  from titles  group by total\_sales \* price | SYNTAX:  select avg(total\_sales),  total\_sales \* price  from titles  group by total\_sales \* price |
| RESULT:    ----------- ------------------------  NULL NULL  111 6,216.00  375 62,850.00  375 64,770.00  2045 179,142.00  3336 213,237.12  2032 324,957.44  3876 370,545.60  18722 447,830.24  4095 491,072.40  22246 532,124.32  4072 651,194.24  4095 654,872.40  4095 655,200.00  15096 1,443,177.60  8780 1,612,008.00  (16 rows affected) | RESULT:  1 2  ----------- --------------------------------  111 777.0000  375 7856.2500  375 8096.2500  2045 22392.7500  3336 26654.6400  2032 40619.6800  3876 46318.2000  18722 55978.7800  4095 61384.0500  22246 66515.5400  4072 81399.2800  4095 81859.0500  15096 180397.2000  8780 201501.0000  4095 655200.0000  - -  16 record(s) selected. |

**Nesting aggregates with group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select max(avg(price))  from titles  group by type | SYNTAX:  select max(avg(price)) from titles group by type |
| RESULT:    ------------------------  171.80  (1 row affected) | RESULT:  SQL0112N The operand of the column function "MAX" includes a column function,  a scalar fullselect, or a subquery. SQLSTATE=42607  db2 => |

**Null values and group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select advance, avg(price \* 2)  from titles  group by advance | SYNTAX:  select advance,avg(price \* 2) from titles group by advance |
| RESULT:  advance  ------------------------ ------------------------  NULL NULL  0.00 319.84  2,000.00 319.84  2,275.00 175.20  4,000.00 159.52  5,000.00 276.96  6,000.00 112.00  7,000.00 349.28  8,000.00 279.92  10,125.00 47.84  15,000.00 47.84  (11 rows affected) | RESULT:  ADVANCE 2  --------------------- ---------------------------------  0.0000 39.98000  2000.0000 319.84000  2275.0000 21.90000  4000.0000 75.87000  5000.0000 127.90666  6000.0000 14.00000  7000.0000 43.66000  8000.0000 174.99000  10125.0000 5.98000  15000.0000 5.98000  - -  11 record(s) selected. |

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select price, count(price), count(\*)  from titles  group by price | SYNTAX:  SELECT price,COUNT(price),COUNT(\*) FROM titles GROUP BY price |
| RESULT:  price  ------------------------ ----------- -----------  NULL 0 2  23.92 2 2  56.00 1 1  63.92 1 1  87.60 1 1  95.60 2 2  119.92 1 1  159.92 4 4  160.00 1 1  167.60 1 1  172.72 1 1  183.60 1 1  (12 rows affected) | RESULT:  PRICE 2 3  --------------------- ----------- -----------  2.9900 2 2  7.0000 1 1  7.9900 1 1  10.9500 1 1  11.9500 2 2  14.9900 1 1  19.9900 4 4  20.9500 1 1  21.5900 1 1  22.9500 1 1  160.0000 1 1  - 0 2  12 record(s) selected. |

**where clause and group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, avg(price)  from titles  where advance > 5000  group by type | SYNTAX:  SELECT type, AVG(price) FROM titles  WHERE advance > 5000 GROUP BY type |
| RESULT:  type  ------------ ------------------------  business 23.92  mod\_cook 23.92  popular\_comp 171.80  psychology 114.36  trad\_cook 143.76  (5 rows affected) | RESULT:  TYPE 2  ------------ ---------------------------------  business 2.9900000000000000  mod\_cook 2.9900000000000000  popular\_comp 91.4750000000000000  psychology 14.2950000000000000  trad\_cook 17.9700000000000000  5 record(s) selected. |

**group by and all:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, avg(advance)  from titles  where advance > 1000 and advance < 10000  group by type | SYNTAX:  SELECT type,AVG(advance) FROM titles WHERE advance > 1000 AND advance < 10000 GROUP BY type |
| RESULT:  type  ------------ ------------------------  business 5,000.00  popular\_comp 7,500.00  psychology 4,255.00  trad\_cook 6,333.33  (4 rows affected) | RESULT:  TYPE 2  ------------ ---------------------------------  business 5000.0000000000000000  popular\_comp 7500.0000000000000000  psychology 4255.0000000000000000  trad\_cook 6333.3333333333333333  4 record(s) selected. |

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, avg(advance)  from titles  where advance > 1000 and advance < 10000  group by all type | SYNTAX:  SELECT type, AVG(advance) FROM titles WHERE advance > 1000 AND advance < 10000 GROUP BY type |
| RESULT:  type  ------------ ------------------------  UNDECIDED NULL  business 5,000.00  mod\_cook NULL  popular\_comp 7,500.00  psychology 4,255.00  trad\_cook 6,333.33  (6 rows affected) | RESULT:  TYPE 2  ------------ ---------------------------------  business 5000.0000000000000000  popular\_comp 7500.0000000000000000  psychology 4255.0000000000000000  trad\_cook 6333.3333333333333333  4 record(s) selected. |

**Note:** Group by all types is not used in the Db2.

**Using aggregates without group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select pub\_id, count(pub\_id)  from publishers | SYNTAX:  SELECT pub\_id, COUNT(pub\_id) FROM publishers |
| RESULT:  pub\_id  ------ -----------  0736 4  0877 4  1389 4  9999 4  (4 rows affected) | RESULT:  SELECT pub\_id,COUNT(pub\_id) FROM publishers  SQL0119N An expression starting with "PUB\_ID" specified in a SELECT clause,  HAVING clause, or ORDER BY clause is not specified in the GROUP BY clause or  it is in a SELECT clause, HAVING clause, or ORDER BY clause with a column  function and no GROUP BY clause is specified. SQLSTATE=42803 |

**Selecting groups of data: the having clause**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select title\_id  from titles  where title\_id like "PS%"  having avg(price) > $2.0 | SYNTAX:  SELECT title\_id FROM title\_test WHERE title\_id LIKE 'PS%' having avg(price) >2.0 |
| RESULT:  title\_id  --------  BU1032  BU1111  BU2075  BU7832  MC2222  MC3021  MC3026  PC1035  PC8888  PC9999  PS1372  PS2091  PS2106  PS3333  PS7777  TC3218  TC4203  TC7777  (18 rows affected) | RESULT:  SQL0119N An expression starting with "TITLE\_ID" specified in a SELECT clause,  HAVING clause, or ORDER BY clause is not specified in the GROUP BY clause or  it is in a SELECT clause, HAVING clause, or ORDER BY clause with a column  function and no GROUP BY clause is specified. SQLSTATE=42803 |

Note: Equivalent not found.

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type from titles group by type having count(\*) > 1 | SYNTAX:  SELECT type FROM titles GROUP BY type HAVING COUNT(\*) > 1 |
| RESULT:  type  ------------  business  mod\_cook  popular\_comp  psychology  trad\_cook  (5 rows affected) | RESULT:  TYPE  ------------  business  mod\_cook  popular\_comp  psychology  trad\_cook  5 record(s) selected. |

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type  from titles  group by type  having type like "p%" | SYNTAX:  SELECT type FROM titles GROUP BY type HAVING type LIKE 'p%' |
| RESULT:    type  ------------  popular\_comp  psychology  (2 rows affected) | RESULT:  TYPE  ------------  popular\_comp  psychology  2 record(s) selected. |

**Having, Group by, and Where clauses interact:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT title\_id,SUM(advance) FROM titles WHERE title\_id LIKE 'PS%'GROUP BY title\_id HAVING SUM(advance) > 200 | SYNTAX:  SELECT title\_id,SUM(advance) FROM titles WHERE title\_id LIKE 'PS%' GROUP BY title\_id HAVING SUM(advance) > 200 |
| RESULT:  title\_id  -------- ------------------------  PS1372 7,000.00  PS2091 2,275.00  PS2106 6,000.00  PS3333 2,000.00  PS7777 4,000.00  (5 rows affected) | RESULT:  TITLE\_ID 2  -------------------- ---------------------------------  PS1372 7000.0000  PS2091 2275.0000  PS2106 6000.0000  PS3333 2000.0000  PS7777 4000.0000  5 record(s) selected. |

**Using having without group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select title\_id, advance, price  from titles  where advance < 4000  having price > avg(price) | SYNTAX:  SELECT title\_id,advance,price FROM titles WHERE advance < 4000 |
| RESULT:  title\_id advance price  -------- ------------------------ ------------------------  BU1032 5,000.00 159.92  BU7832 5,000.00 159.92  MC2222 0.00 159.92  PC1035 7,000.00 183.60  PC8888 8,000.00 160.00  PS1372 7,000.00 172.72  PS3333 2,000.00 159.92  TC3218 7,000.00 167.60  (8 rows affected) | RESULT:  TITLE\_ID ADVANCE PRICE  ---------- --------------------- ---------------------  MC2222 0.0000 19.9900  PS2091 2275.0000 10.9500  PS3333 2000.0000 159.9200  3 record(s) selected. |

Note: Having clause is not getting converted.

**Sorting query results: the order by clause**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select pub\_id, type, title\_id  from titles  order by pub\_id | SYNTAX:  select pub\_id, type, title\_id from titles order by pub\_id |
| RESULT:  pub\_id type title\_id  ------ ------------ --------  0736 business BU2075  0736 psychology PS2091  0736 psychology PS2106  0736 psychology PS3333  0736 psychology PS7777  0877 UNDECIDED MC3026  0877 mod\_cook MC2222  0877 mod\_cook MC3021  0877 psychology PS1372  0877 trad\_cook TC3218  0877 trad\_cook TC4203  0877 trad\_cook TC7777  1389 business BU1032  1389 business BU1111  1389 business BU7832  1389 popular\_comp PC1035  1389 popular\_comp PC8888  1389 popular\_comp PC9999  (18 rows affected) | RESULT:  PUB\_ID TYPE TITLE\_ID  ------ ------------ ---------------  0736 psychology PS7777  0736 psychology PS3333  0736 business BU2075  0736 psychology PS2091  0736 psychology PS2106  0877 mod\_cook MC2222  0877 trad\_cook TC7777  0877 trad\_cook TC4203  0877 mod\_cook MC3021  0877 trad\_cook TC3218  0877 UNDECIDED MC3026  0877 psychology PS1372  1389 popular\_comp PC8888  1389 business BU1032  1389 business BU1111  1389 popular\_comp PC1035  1389 business BU7832  1389 popular\_comp PC9999  18 record(s) selected. |

**order by and group by:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, avg(price)  from titles  group by type  order by avg(price) | SYNTAX:  SELECT type,AVG(price) FROM titles GROUP BY type ORDER BY AVG(price) |
| RESULT:  type  ------------ ------------------------  UNDECIDED NULL  mod\_cook 91.92  psychology 108.03  business 109.84  trad\_cook 127.71  popular\_comp 171.80  (6 rows affected) | RESULT:  TYPE 2  ------------ ---------------------------------  mod\_cook 11.4900000000000000  psychology 13.5040000000000000  business 13.7300000000000000  trad\_cook 15.9633333333333333  popular\_comp 91.4750000000000000  UNDECIDED -  6 record(s) selected. |

**order by and group by used with select distinct:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select distinct pub\_id  from titles  order by type | SYNTAX:  SELECT DISTINCT pub\_id FROM title\_test ORDER BY type,pub\_id |
| RESULT:  pub\_id  ------  0877  0736  1389  0877  1389  0736  0877  0877  (8 rows affected) | RESULT:  SQL0214N An expression in the ORDER BY clause in the following position, or  starting with "TYPE" in the "ORDER BY" clause is not valid. Reason code =  "2". SQLSTATE=42822 |

Note: No equivalent found. If a query has an order by or group by clause that includes columns not in the select list, adds those columns as hidden columns in the columns being processed. The columns listed in the order by or group by clause are included in the test for distinct rows. To comply with ANSI standards, include the order by or group by column in the select list.

**Computeclause: Summarizing groups of data**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select type, price, advance  from titles  where type like "%cook"  order by type  compute sum(price), sum(advance) by type  compute sum(price), sum(advance) | SYNTAX:  select coalesce(price,10), sum(price) \  from titles \  group by rollup(type, price) \  order by type, price |
| RESULT:  type price advance  ----------- ----------------- ------------  mod\_ cook 2.99 15,000.00  mod\_cook 19.99 0.00  Compute Result:  --------------- ---------  22.98 15,000.00  type price advance  ----------- ----------------- ------------  trad\_cook 11.95 4,000.00  trad\_cook 14.99 8,000.00  trad\_cook 20.95 7,000.00  Compute Result:  --------------- ---------  47.89 19,000.00  Compute Result:  --------------- ---------  70.87 34,000.00  (8 rows affected) | RESULT:  PRICE SUM  ----------- -----------  10 2468  1234 2468  **10**  **12342**  12342 12342  **10 8888**  8888 8888  **10 23698**  7 record(s) selected..  The numbers in bold are the total and subtotal. |

Note: There is no Equivalent for Compute clause in Db2 but workaround is possible as shown above.

## **Built In Functions**

### **System Functions**

#### **Retrieve the Current User Name**

DESCRIPTION: Functions to retrieve the Current User Name

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> SELECT user\_name()  2> go | SELECT current user  FROM sysibm.sysdummy1; |
| RESULT  ------------------------------  guest  (1 row affected) | KOUSHID |

CONCLUSION The result is the same.

#### **GETTING THE COLUMN NAMES**

#### DESCRIPTION: **Retrieving the Column Name**

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select col\_name(object\_id("titles"),2)  2> go | select COLNAME from syscat.columns where TABNAME='TITLES' and COLNO=1 and TABSHEMA= ‘THUWAL’ |
| RESULT  ------------------------------  title |  |

CONCLUSION: ONLY Sybase supports this function. DB2 needs SQL queries in order to get the same information.

NOTE: DB2 numbers the column starting from 0 where as SYBASE numbers it from 1.

#### **Retrieving the Column length**

DESCRIPTION: Only Sybase allows the column lengths to be retrieved.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select col\_length("titles","title")  2> go | select LENGTH from syscat.columns where TAB  NAME='TITLES' and COLNAME='TITLE' and TABSHEMA= ‘THUWAL’ |
| RESULT  -----------  80  (1 row affected) | 80 |

CONCLUSION: Only Sybase allows the column length to be retrieved through a function.DB2 needs SQL queries to achieve the objective.

#### **Retrieving the length of expression in bytes.**

DESCRIPTION: This feature is available in both Sybase and DB2.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datalength("user")  2> go  NOTE: (*expression* is usually a column name BUT can also be a literal) | select length(deptname) from department; |
| RESULT  -----------  4  (1 row affected) | | 28 | | --- | | 8 | | 18 | | 18 | | 21 | | 22 | | 16 | | 10 | | 16 | |

CONCLUSION: Supported by both Sybase and DB2.

#### **RETRIEVING THE CURRENT DATABASE NAME**

DESCRIPTION: To obtain the database name one is currently logged into.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select db\_name()  2> go | QUERY  Values current server |
| RESULT  ------------------------------  pubs2  (1 row affected) | RESULT  1  ------------------  SAMPLE  1 record(s) selected. |

CONCLUSION: Supported by Sybase only.

#### **RETRIEVING THE CURRENT DATABASE ID**

DESCRIPTION: To obtain the database id one is currently logged into.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select db\_id()  2> go | In DB2, there is no concept of Database ID. The name is the identifier |
| RESULT  ------  4  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING THE HOST ID**

DESCRIPTION: To retrieve the host id on which the database server is running

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select host\_id()  2> go | DB2 does not have the concept of host-ids as the database server is identified by its name or IP address(in conjunction with the port number) |
| RESULT  ---------------------  8746  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING THE SERVER NAME**

DESCRIPTION: Retrieves the name of the database server

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select host\_name()  2> go | QUERY  select host\_name from table(SYSPROC.ENV\_GET\_SYS\_INFO()) as systeminfo |
| RESULT  -----------------------  ps-server1  (1 row affected) | Can be checked only with FP (Fix Pack) 9 and higher. |

CONCLUSION Supported by Sybase only.

#### **SUBSTITUTING FOR A NULL VALUE**

DESCRIPTION: To specify a default value for a column when it is NULL.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select avg(price)  2> from titles  3> go | select avg(price)  from titles |
| RESULT  ------------------------  118.13  (1 row affected) | 118.1300000000000000 |

**WHEN DEFAULT IS SPECIFIED FOR THE NULL**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select avg(ISNULL(price,0))  2> from titles  3> go | select avg(COALESCE(price,0))  from titles |
| RESULT  ------------------------  105.00  (1 row affected) | 105.0044444444444444 |

CONCLUSION Supported by both Sybase and DB2.Both uses functions with different names.

#### **RETRIEVING AN OBJECT ID**

DESCRIPTION: Retrieves the ID of any object like tables, indexes, views etc.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select object\_id("titles")  2> go | In DB2, objects are identified by their type and fully qualified name. For routines (like functions and procedures) we have the concept of routineid and for sequences we have seqid. |
| RESULT  -----------  224000798  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING AN OBJECT NAME**

DESCRIPTION: Retrieves the name of any object like tables, indexes, views etc.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select object\_name( 224000798)  2> go | In DB2, objects are identified by their type and fully qualified name. For routines (like functions and procedures) we have the concept of routineid and for sequences we have seqid. |
| RESULT  ------------------------------  titles  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **SETTING THE SERVER USER NAME**

DESCRIPTION Retrieves the Server User name of the current user or any other user.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select suser\_name()  2> go | select current user from sysibm.sysdummy1; |
| RESULT  ----------------------  mohm  (1 row affected) | KOUSHID |

CONCLUSION Supported by both Sybase and DB2.In Sybase if no server user user name is supplied, the name of the Current user is returned.

#### **FETCHING THE SERVER USER ID OF THE USER**

DESCRIPTION: Retrieves the server user id of the User.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select suser\_id("mohm")  2> go | In DB2, identifies users with their names. There isn’t any concept of server user id. |
| RESULT  -----------  23  (1 row affected) |  |

CONCLUSION Supported in Sybase Only

#### **COMPARING TIME STAMP VALUES**

DESCRIPTION: Compares two timestamp values.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Use the tsequal system function to compare timestamps when you are using browse mode in a front-end application. For example, the following  statement updates a row in publishers that has been browsed. It compares the timestamp column in the browsed version of the row with the hexadecimal timestamp in the stored version. If the two timestamps are not equal, you receive an error message, and the row is not updated.  EXAMPLE:  update publishers  set city = "Springfield"  where pub\_id = "0736"  and tsequal(timestamp,0x0001000000002ea8) | There is no such timestamp in hex format in DB2. In Sybase also this format is generated by the  db engine only for its own internal purpose. The user does not have to bother about that. after exporting also, this data can be loaded into  DB2 tables using varchar or char(20) data types. |

NOTE: Do not use the tsequal function in the where clause as a search argument.When you use tsequal, the rest of the where clause should match a single row uniquely. Use the tsequal function only in insert and update statements.

#### **FETCHING THE USER ID**

DESCRIPTION: Retrieves the database user id of the User

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select user\_id()  2> go | In DB2, the users are identified by their names. The Concept of user id does not exist. |
| RESULT  -----------  2 |  |

CONCLUSION Supported in Sybase Only.

#### **VALIDATING USER NAME**

DESCRIPTION: Checks to see if the user is authorised to the database.Returns 0 if the username is invalid. Else returns a non-zero value.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select valid\_name("mohm")  2> go | QUERY:  db2 “select grantee from syscat.dbauth where grantee=’MOHM’ “ | grep ‘record(s)’ | cut –f3 –d” “ |
| RESULT  -----------  1  (1 row affected) | RESULT  0 |

CONCLUSION: Supported in Sybase **Only** but workaround is possible as shown above.

#### **VALIDATING USER THROUGH USER ID**

DESCRIPTION Checks to see if the user with the given user id is authorised to the database.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select valid\_user(23)  2> go | In DB2, there isn’t any concept of userid. Thus, one cannot check for the validity based on user id’s. |
| RESULT  -----------  1  (1 row affected) |  |

CONCLUSION Supported in Sybase **Only** but work around is possible as shown above.

### **String Functions**

#### **FETCHING THE ASCII VALUSE OF A CHARACTER**

DESCRIPTION: To retrieve the ASCII value of character. If a string is supplied as an argument the ASCII value of the first character is returned

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select ascii("a")  2> go | select ascii('abc') from sysibm.sysdummy1 |
| RESULT  -----------  97 | 97 |
| QUERY  1> select ascii("abc")  2> go | select ascii('abc') from sysibm.sysdummy1 |
| RESULT  -----------  97  (1 row affected) | 97 |

CONCLUSION Supported by both Sybase and DB2.

#### **FETCHING THE CHARACTER OF A GIVEN ASCII VALUE**

DESCRIPTION Inputs an integer as the ASCII value and outputs the character represented by it

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select char(97)  2> go | select chr(97) from sysibm.sysdummy1 |
| RESULT  -  a  (1 row affected) | a |

CONCLUSION Supported by both Sybase and DB2.

## **SEARCHING FOR A SUBSTRING WITHIN ANOTHER**

DESCRIPTION: Searching for the index of a sub-string in a string. Returns the index where the first occurrence of the string is found. Returns 0 if the string is not found.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select charindex("bcd","abcdef")  2> go | select posstr('abcdef','bcd') from sysibm.sysdummy1 |
| RESULT  -----------  2 | 2 |

CONCLUSION Equivalent function is available in DB2.

#### **LENGTH OF A STRING EXPRESSION**

DESCRIPTION: Calculates the length of a character or varchar expression

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select char\_length("database")  2> go | select length('database') from sysibm.sysdummy1 |
| RESULT  -----------  8  (1 row affected) | 8 |

CONCLUSION Both Sybase and DB2 support this though with different names.

## **COMPARING THE SOUNDEX VALUES OF TWO STRINGS**

DESCRIPTION: Obtains the difference between the soundex values of two string

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select difference("Smith","smythe")  2> go | select difference('Smith','smyhte') from sysibm.sysdummy1 |
| RESULT  -----------  4  (1 row affected) | 4 |

CONCLUSION Supported by both Sybase and DB2 with the same name.

## **CONVERTING A STRING TO LOWER CASE**

DESCRIPTION: Changes the case of a given string to lower

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select lower("DATABASE")  2> go | select lcase('DATABASE') from sysibm.sysdummy1; |
| RESULT  -----------  database  (1 row affected) | database |

CONCLUSION Supported by both Sybase and DB2 by the name “lower”.DB2 also has calls it by “lcase”.

#### **TRIMMING A STRING**

DESCRIPTION: Removes leading spaces in an expression.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select ltrim(" database")  2> go | select ltrim(' database') from sysibm.sysdummy1; |
| RESULT  ---------------  database  (1 row affected) | database |

CONCLUSION Supported by both Sybase and DB2 by the name “ltrim”.

#### **SEARCHING FOR A PATTERN**

DESCRIPTION Checks for the occurrence of a particular pattern in an expression

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select patindex("%def%","abcdefgh")  2> go | select posstr('abcdefg','def') from sysibm.sysdummy1; |
| RESULT  -----------  4  (1 row affected) | 4 |

CONCLUSION Supported by both Sybase and DB2.The % wildcard character must precede and follow *pattern*, except when searching for first or last characters. Returns an integer representing the starting position of the first occurrence of *pattern* in the specified character expression; returns **0** if *pattern* is not found. DB2 doesn’t allow for % wildcard character. Also the source string and searched string are in the reverse order than is in “patindex”

## **REPLICATING A STRING**

DESCRIPTION: Repeats a string a specified number of times

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select replicate("database",2)  2> go | select repeat('database',2) from sysibm.sysdummy1; |
| RESULT  ----------------  databasedatabase  (1 row affected) | databasedatabase |

CONCLUSION Supported by both Sybase and DB2.

## **REVERSING A STRING**

DESCRIPTION: Reverses the sequence in which characters occur in a string.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create proc get\_employee\_details @lastname varchar(40),  @firstname varchar(20) as  select au\_lname, au\_fname from  department  where au\_fname = @firstname  and au\_lname = @lastname  select reverse("@lastname") | QUERY  CREATE PROCEDURE get\_employee\_details (v\_lastname VARCHAR(40),   v\_firstname VARCHAR(20) )  LANGUAGE SQL  BEGIN  DECLARE temp\_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT   FOR SELECT au\_lname,  au\_fname  FROM department  WHERE v\_firstname IS NULL  AND au\_fname IS NULL  OR au\_fname = v\_firstname  AND v\_lastname IS NULL  AND au\_lname IS NULL  OR au\_lname = v\_lastname;  DECLARE temp\_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT   FOR SELECT CAST (SYB.REVERSE('@lastname') AS VARCHAR(9))  FROM (VALUES 1) temp\_table;  OPEN temp\_cursor;  OPEN temp\_cursor1;  END |
| DESCRIPTION:  In the above example reverse function is used to reverse the contents of the variable lastname. | DESCRIPTION:  There is no direct equivalent of this function in DB2 but by converting the code through MTK then MTK creates a function SYB.REVERSE which is highlighted in red. |

CONCLUSION Supported by Sybase ONLY.

## **EXTRACTING A SUBSTRING FROM A STRING**

DESCRIPTION: Fetches a substring from the right of a specified index of string.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select right("database",3)  2> go | select right('database',3) from sysibm.sysdummy1 |
| RESULT  ---  ase  (1 row affected) | ase |

CONCLUSION: Supported by both Sybase and DB2.

## **TRIMMING A STRING**

DESCRIPTION: Removing trailing spaces from a string.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select rtrim("database ")  2> go | select rtrim('database ') from sysibm.sysdummy1 |
| RESULT  ---------------  database | database |

CONCLUSION Supported by both Sybase and DB2.

## **SOUNDEX VALUES**

DESCRIPTION Obtains the SOUNDEX values of a string.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select soundex("smith")  2> go | select soundex('smith') from sysibm.sysdummy1 |
| RESULT  -----  S530  (1 row affected) | S530 |

CONCLUSION Supported by both Sybase and DB2 .

## **CREATING A STRING OF SPACES**

DESCRIPTION: To create a string composed of the space character.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select space(10)  2> go | select space(10) from sysibm.sysdummy1 |
| RESULT  ----------  (1 row affected) |  |

CONCLUSION Supported by both Sybase and DB2 .

## **CONVERTING A FLOATING POINT NUMBER TO A STRING**

DESCRIPTION: Represents the specified floating point number in a character string

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select str(1400.23564,9,3)  2> go | select char(1400.23564,'.') from sysibm.sysdummy1 |
| RESULT  ---------  1400.236  (1 row affected) | 1400.23564 |

CONCLUSION: Supported by Sybase. Partially supported by DB2.Sybase uses the function “str” and DB2

uses “char”.

## **STUFFING ONE STRING WITHIN ANOTHER**

DESCRIPTION: Takes two string as input and stuffs one into another

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select stuff("abcdef",3,2,"xy")  2> go | select INSERT('DINING', 3, 1, 'VID') from sysibm.sysdummy1 |
| RESULT  ------  abxyef  (1 row affected) | DIVIDING |

CONCLUSION: Sybase stuffs one string in another through the function “stuff”.  
 DB2 uses “insert” in a modified way.

#### **EXTRACTING A SUB STRING FROM ANYWHERE IN A STRING**

DESCRIPTION: Retrieves a sub-string starting from a given index and ending in another, within a string.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select substring("database",1,4)  2> go | select substr('database',1,4) from sysibm.sysdummy1 |
| RESULT  ----  data  (1 row affected) | DATA |

CONCLUSION Sybase uses the function “substring” and DB2 uses “substr”.

### **Text Functions**

3.2.3.1 RETREIVING A TEST FIELD

DESCRIPTION: Sybase employs pointer to retrive text fields.Whereas in DB2 an equivalent CLOB field can be specified directly in the SELECT clause, like any other fields.

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  declare @val binary(16)  select @val = textptr(copy) from blurbs  where au\_id = "486-29-1786"  readtext blurbs.copy @val 1 5 | select copy from blurbs  where au\_id = "486-29-1786" |

3.2.3.2 **TEST IF A POINTER IS VALID**

DESCRIPTION: Only Sybase employs POINTERS in retrieving TEXT fields.Therefor no such function is available in DB2.

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  select textvalid (“blurbs..copy”, @val)  Returns 1 if the pointer is valid , and 0 otherwise | Not supported by DB2. |

### **Aggregate Functions**

Supported by both Sybase and DB2

CASE1:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select avg(price \* 2) from title | QUERY:  SELECT AVG(price \* 2) FROM titles |
| RESULT:    ------------------------  236.26  (1 row affected) | RESULT:  1  ---------------------------------  236.26000  1 record(s) selected. |

CONCLUSION: Results are same.

CASE 2:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select avg(au\_lname) from authors | QUERY:  SELECT AVG(au\_lname) FROM authors |
| RESULT:  Implicit conversion from datatype 'VARCHAR' to 'INT' is not allowed. Use the CONVERT function to run thisquery. | RESULT:  SQL0440N No authorized routine named "AVG" of type "FUNCTION" having compatible arguments was found. SQLSTATE=42884 |

CONCLUSION: You cannot use the average function with text columns.

**SUM FUNCTION:**

Supported by both Sybase and DB2

CASE1:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select sum(total\_sales) from titles | QUERY:  SELECT SUM(total\_sales) FROM titles |
| RESULT:  -----------  97446  (1 row affected) | RESULT:  1  -----------  97446  1 record(s) selected. |

CONCLUSION: Results are same.

CASE2:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select sum(au\_lname) from authors | QUERY:  SELECT SUM(au\_lname) FROM authors |
| RESULT:  The SUM OR AVERAGE AGGREGATE operation cannot take a VARCHAR datatype as an argument. | RESULT:  SQL0440N No authorized routine named "SUM" of type "FUNCTION" having compatible arguments was found. SQLSTATE=42884 |

CONCLUSION: You cannot use the SUM function with text columns.

NOTE: You can use sum and avg with numeric columns only— decimal, numeric, float, int, smallint and money.

**MIN FUNCTION:**

You can use min (minimum) to find the lowest value—the one closest to the beginning of the alphabet—in a character type column:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:    select min (au\_lname) from authors | QUERY:  select min (au\_lname) from authors |
| RESULT:  -----------------  Bennet  (1 row affected) | RESULT:  1  -----------  Bennet  1 record(s) selected |

CONCLUSION: Results are same.

**MAX FUNCTION:**

Supported by both Sybase and DB2

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:    select max (au\_lname) from authors | QUERY:  select max (au\_lname) from authors |
| RESULT:    ----------------------------------------  del Castillo  (1 row affected) | RESULT:  1  -----------------  Yokomoto  1record(s)selected |

CONCLUSION: In DB2, if we have given the collate using identity then the max function will take the value in bye to byte fashion else if we have given using the collate using system then it will take the values in ascending order. This two options collate using identity and system can be given at the time of creating the database and once given they can’t be edited afterwards.

NOTE: In the above case the option is collate using system and hence the values is given out by AMX function in ascending order. Thus the results differ. In Sybase it is always by byte to byte (ASCII).

**COUNT(\*) FUNCTION:**

Supported by both Sybase and DB2

Count(\*) finds the total number of rows in a table.

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select count(\*) from publishers | QUERY:  SELECT COUNT(\*)FROM publishers |
| RESULT:  -----------  4  (1 row affected) | RESULT:  1  -----------  4  1 record(s) selected. |

**COUNT FUNCTION:**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select count(advance) from titles | QUERY:  select count(advance) from titles |
| RESULT:  -----------  16  (1 row affected) | RESULT:  ------------------------  16  (1 record selected) |

CONCLUSION: Results are same.

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:    select count(title) from titles | QUERY:    select count(title) from title |
| RESULT:  -----------  18  (1 row affected) | RESULT:  1  -----------  18  1 record(s) selected |

CONCLUSION: Results are same.

NOTE: The count() function ignores the NULL value among the total no. of rows in the specified column.

**Difference b\w the count() and count(\*) function :**  
 While count function finds the number of non-null values to expression, count (\*) finds the total number of rows in a table

**COUNT DISTINCT FUNCTION:**

Supported by both Sybase and DB2

**No duplicate results are displayed if we are using the Count distinct.**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:    select count(distinct city) from authors | QUERY:  SELECT COUNT(DISTINCT city)FROM authors |
| RESULT:  -------------  16  (1 row affected) | RESULT:  1  -----------  16  1 record(s) selected |

CONCLUSION: Results are same.

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:    select count(city) from authors | QUERY:  SELECT COUNT(city)FROM authors |
| RESULT:  -------------  23  (1 row affected) | RESULT:  1  -----------  23  1 record(s) selected |

CONCLUSION: Results are same.

NOTE: Without Distinct the same query was different.

**NULL VALUES AND THE AGGREGATE FUNCTIONS:**

Supported by both Sybase and DB2

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select count(distinct title) from titles where type = "poetry” | QUERY:  SELECT COUNT(DISTINCT title) FROM titles WHERE type = 'poetry’ |
| RESULT:  **-------------**  0  (1 row affected | RESULT:  1  -----------  0  1 record(s) selected |

CONCLUSION: If all the values of a column are null, count returns 0. If no rows meet the conditions specified in the where clause, count returns 0.

### **Date Functions**

#### **CURRENT DATE**

DESCRIPTION: To view the Current date

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select getdate()  2> go | select current date from sysibm.sysdummy1 |
| RESULT  --------------------------  Oct 20 2005 2:47PM  (1 row affected) | Oct 20,2005 |

CONCLUSION: Both Sybase and DB2 has mechanisms to fetch the Current Date.

But they use different syntax to do so.

#### **DIFFERENCE BETWEEN TWO DATES**

DESCRIPTION: Gets the number of days between two days

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datediff(dd,"2005-10-20","2005-10-25")  2> go | select days('2005-10-25')-days('2005-10-20') from sysibm.sysdummy1; |
| RESULT  -----------  5  (1 row affected) | 5 |

CONCLUSION: Sybase has direct functions to obtain the difference between two dates but there is no such function in DB2. However it can be achieved in DB2 by date arithmetic.

#### **DAY OF WEEK FUNCTION (NAME)**

DESCRIPTION Fetches the day of week from the date specified.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datename(dw,"2005-10-21")  2> go | select dayname(current date) from sysibm.sysdummy1; |
| RESULT  ------------------------------  Friday  (1 row affected) | Friday |

CONCLUSION DB2 provides the function “dayofweek()” and Sybase uses “datename()”.

#### **ADDING TO A DATE**

DESCRIPTION: Add specified number of days to the specified date.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select dateadd(mm,4,"2005-10-20")  2> go | select date ( days('2005-10-20') + 120) from sysibm.sysdummy1; |
| RESULT  --------------------------  Feb 20 2006 12:00AM  (1 row affected) | Feb 17,2006 |

CONCLUSION: In Sybase “dateadd()” functions is used to achieve the purpose.  
 In DB2 date arithmetic should be used with the “days()” function.

#### **DAY OF WEEK FUNCTION (INTEGER)**

DESCRIPTION: Fetches the day of week from a date as an integer between 1 and 7 signifying Sunday to Saturday.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(dw,getdate())  2> go | select dayofweek(current date) from sysibm.sysdummy1; |
| RESULT  -----------  6  (1 row affected) | 6 |

CONCLUSION Both Sybase and DB2 can report the day of a week as an integer. However the usage is quite

different.

#### **DAY OF YEAR FUNCTION.**

DESCRIPTION: From the specified date, the function extracts the day of the year the date signifies.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(dy,getdate())  2> go | select dayofyear(current date) from sysibm.sysdummy1; |
| RESULT  -----------  294  (1 row affected) | 294 |

CONCLUSION: Sybase and DB2 both have individual ways of doing this.

#### **THE QUARTER OF THE YEAR FUNCTION**

DESCRIPTION: Extracts the quarter of year from the specified date.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(qq,getdate())  2> go | select quarter(current date) from sysibm.sysdummy1; |
| RESULT  -----------  4  (1 row affected) | 4 |

CONCLUSION: Sybase and DB2 both have individual ways of doing this.

#### **THE WEEK OF THE YEAR FUNCTION**

DESCRIPTION Extracts the week of the year the specified date falls within. Returns an integer ranging

from 1 to 54.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(wk,getdate())  2> go | select week(current date) from sysibm.sysdummy1; |
| RESULT  -----------  44  (1 row affected) | 44 |

CONCLUSION Sybase and DB2 both have individual ways of doing this.

#### **THE DAY OF MONTH FUNCTION**

DESCRIPTION: Returns an integer between 1 and 31 to state which day of month the specified date

signifies.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(dd,getdate())  2> go | select day(current date) from sysibm.sysdummy1; |
| RESULT  -----------  24  (1 row affected) | 24 |

CONCLUSION: Sybase and DB2 both have individual ways of doing this.

#### **THE HOUR OF DAY FUNCTION**

DESCRIPTION: Extracts the information about the hour of the day the specified date contains

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(hh,getdate())  2> go | DB2 has a HOUR function that can be invoked with a timestamp or time value.  DB2 **can** return a value between -99 and 99 if the argument is a **time duration**. |
| RESULT  -----------  10  (1 row affected) |  |

CONCLUSION: No direct function in DB2 which takes a date as an input.

#### **THE MINUTE OF AN HOUR FUNCTION**

DESCRIPTION: Extracts the minute of an hour as an integer ranging from 0 to 59, from a date.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(**mi**,getdate())  2> go | DB2 has a MINUTE function that can be invoked with either a timestamp or time value.  Also Sybase returns a value between 0 and 59 where as DB2 can return a value between -99 and 99 if the argument is a time duration |
| RESULT  -----------  26  (1 row affected) |  |

CONCLUSION: No direct function in DB2 which takes a date as an input.

#### **THE SECOND OF A MINUTE**

DESCRIPTION Extracts the second of a minute as an integer ranging from 0 to 59, from a date.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(ss,getdate())  2> go | DB2 has the SECOND function that can be invoked with either a timestamp or time value.  Also Sybase returns a value between 0 and 59 where as DB2 **can** return a value between -99 and 99 if the argument is a **time duration.** |
| RESULT  -----------  45  (1 row affected) |  |

CONCLUSION: No direct function in DB2 which takes a date as an input.

#### **THE MONTH OF THE YEAR FUNCTION (INTEGER)**

DESCRIPTION: Fetches the month of the year from a date as an integer ranging from 1 to 12.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datepart(mm,getdate())  2> go | select month(current date) from sysibm.sysdummy1; |
| RESULT  -----------  10  (1 row affected) | 10 |

CONCLUSION: Sybase and DB2 both have individual ways of doing this.

#### **THE MONTH OF THE YEAR FUNCTION (MONTH NAME)**

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select datename(mm,getdate())  2> go | select MONTHNAME(current date) from sysibm.sysdummy1; |
| RESULT  ------------------------------  October  (1 row affected) | October |

CONCLUSION: Sybase and DB2 both have individual ways of doing this.

## **Joins**

### **3.3.1 Datatypes in Joins**

The columns being joined must have the same or compatible datatypes. Use the convert function when comparing columns whose datatypes cannot be implicitly converted. Columns being joined need not have the same name, although they often do. If the datatypes used in the join are compatible, Adaptive Server automatically converts them.

For example,

Adaptive Server converts among any of the numeric type columns—int, smallint, tinyint, decimal, or float, and among any of the character type and date columns—char, varchar, unichar, univarchar, nchar, nvarchar, and datetime.

**Joins and textand imagecolumns:**

You cannot use joins for columns containing text or image values. You can,

however, compare the lengths of text columns from two tables with a

where clause.

For example:

where datalength(textab\_1.textcol) >

**Note**: You cannot use aggregate functions other than count(\*) with text and image datatypes.

### **3.3.2 Equi/Natural Joins**

Joins based on equality (=) arecalled **equijoins**.

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select \* from authors**,** publishers where authors.city = publishers.city | SYNTAX:  SELECT \* FROM authors, publishers WHERE authors.city = publishers.city |
| OUTPUT:  au\_id au\_lname au\_fname  phone address  city state country postalcode pub\_id  pub\_name city state  ----------- ---------------------------------------- --------------------  ------------ ----------------------------------------  -------------------- ----- ------------ ---------- ------  ---------------------------------------- -------------------- -----  238-95-7766 Carson Cheryl  415 548-7723 589 Darwin Ln.  Berkeley CA USA 94705 1389  Algodata Infosystems Berkeley CA  409-56-7008 Bennet Abraham  415 658-9932 6223 Bateman St.  Berkeley CA USA 94705 1389  Algodata Infosystems Berkeley CA  (2 rows affected) | OUTPUT:  AU\_ID AU\_LNAME AU\_FNAME  PHONE ADDRESS CITY ST  ATE COUNTRY POSTALCODE PUB\_ID PUB\_NAME CITY  STATE  -------------------- ---------------------------------------- ------------------  -- ------------ ---------------------------------------- -------------------- --  --- ------------ ---------- ------ ---------------------------------------- ----  ---------------- -----  409-56-7008 Bennet Abraham  510 658-9932 6223 Bateman St. Berkeley CA  USA 94705 1389 Algodata Infosystems Berk  eley CA  238-95-7766 Carson Cheryl  510 548-7723 589 Darwin Ln. Berkeley CA  USA 94705 1389 Algodata Infosystems Berk  eley CA  2 record(s) selected. |

### 

### **3.3.3 Joins based on Equality**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select pub\_name, publishers.state, au\_lname, au\_fname, authors.statefrom publishers, authors where authors.state > publishers.state and pub\_name = "New Age Books" | SYNTAX:  SELECT pub\_name,publishers.state,au\_lname,au\_fname,authors.state FROM publishers,Authors WHERE authors.state > publishers.state AND pub\_name = 'New Age Books' |
| OUTPUT:  pub\_name state  au\_lname au\_fname state  ---------------------------------------- -----  ---------------------------------------- -------------------- -----  New Age Books MA  Greene Morningstar TN  New Age Books MA  Blotchet-Halls Reginald OR  New Age Books MA  del Castillo Innes MI  New Age Books MA  Panteley Sylvia MD  New Age Books MA  Ringer Anne UT  New Age Books MA  Ringer Albert UT  (6 rows affected) | OUTPUT:  PUB\_NAME STATE AU\_LNAME  AU\_FNAME STATE  ---------------------------------------- ----- ------------------  ------- -------------------- -----  New Age Books MA Ringer  Albert UT  New Age Books MA Ringer  Anne UT  New Age Books MA Panteley  Sylvia MD  New Age Books MA Greene  Morningstar TN  New Age Books MA del Castillo  Innes MI  New Age Books MA Blotchet-Halls  Reginald OR  6 record(s) selected. |

### **3.3.4 Self Joins**

**Self Join:**  Joins that compare values within the same column of one table are called self-joins.

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select au1.au\_fname, au1.au\_lname, au2.au\_fname, au2.au\_lname from authors au1, authors au2 where au1.city = "Oakland" and au2.city = "Oakland" and au1.state = "CA" and au2.state = "CA" and au1.postalcode = au2.postalcode | SYNTAX:  SELECT au1.au\_fname,au1.au\_lname,au2.au\_fname,au2.au\_lnameFROM authors,authors WHERE au1.city = 'Oakland' AND au2.city = 'Oakland' AND au1.state = 'CA' AND au2.state = 'CA' AND au1.postalcode = au2.postalcode |
| OUTPUT:  au\_fname au\_lname  au\_fname au\_lname  -------------------- ----------------------------------------  -------------------- ----------------------------------------  Marjorie Green  Marjorie Green  Dick Straight  Dick Straight  Dick Straight  Dirk Stringer  Dick Straight  Livia Karsen  Dirk Stringer  Dick Straight  Dirk Stringer  Dirk Stringer  Dirk Stringer  Livia Karsen  Stearns MacFeather  Stearns MacFeather  Livia Karsen  Dick Straight  Livia Karsen  Dirk Stringer  Livia Karsen  Livia Karsen  (11 rows affected) | OUTPUT:  AU\_FNAME AU\_LNAME AU\_FNAME  AU\_LNAME  -------------------- ---------------------------------------- --------------  -- ----------------------------------------  Marjorie Green Marjorie  Green  Dirk Stringer Dirk  Stringer  Dick Straight Dirk  Stringer  Livia Karsen Dirk  Stringer  Dirk Stringer Dick  Straight  Dick Straight Dick  Straight  Livia Karsen Dick  Straight  Dirk Stringer Livia  Karsen  Dick Straight Livia  Karsen  Livia Karsen Livia  Karsen  Stearns MacFeather Stearns  MacFeather  11 record(s) selected. |

**Note:** In Db2 the table should be aliased twice with different names*.*

### **3.3.5 Not equal Joins**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select distinct au\_lname, authors.city from publishers, authors where au\_lname like "[ABC]%" and publishers.city != authors.city | SYNTAX:  SELECT DISTINCT au\_lname, authors.city FROM publishers, authors WHERE SYB.isLike(au\_lname, '[ABC]%', '') = 1 AND publishers.city <> authors.city |
| OUTPUT:  au\_lname city  ---------------------------------------- --------------------  Bennet Berkeley  Carson Berkeley  Blotchet-Halls Corvallis  (3 rows affected) | OUTPUT:  AU\_LNAME CITY  ---------------------------------------- --------------------  Bennet Berkeley  Carson Berkeley  Blotchet-Halls Corvallis  3 record(s) selected. |

### **3.3.6 Outer Joins**

**Inner and Outer tables:**

The terms **outer table** and **inner table** describe the placement of the tables in an outer join:

• In a *left join*, the **outer table** and **inner table** are the left and right tables respectively. The outer table and inner table are also referred to as the row-preserving and null-supplying tables, respectively.

• In a *right join*, the outer table and inner table are the right and left tables respectively.

For example, in the queries below, T1 is the outer table and T2 is the inner table:

T1 left join T2

T2 right join T1

Or, using Transact-SQL syntax:

T1 \*= T2

T2 =\* T1

**Outer join restrictions**

If a table is an inner member of an outer join, it cannot participate in *both* an outer join clause and a regular join clause. The following query fails because the salesdetail table is part of both the outer join and a regular join clause:

select distinct sales.stor\_id, stor\_name, title from sales, stores, titles, salesdetail where qty > 500

and salesdetail.title\_id =\* titles.title\_id

and sales.stor\_id = salesdetail.stor\_id

and sales.stor\_id = stores.stor\_id

Msg 303, Level 16, State 1:

Server ’FUSSY’, Line 1:

The table ’salesdetail’ is an inner member of an outer-join clause. This is not allowed if the table

also participates in a regular join clause.

If you want to know the name of the store that sold more than 500 copies of a book, you would have to use a second query. If you submit a query with an outer join and a qualification on a column from the inner table of the outer join, the results may not be what you expect. The qualification in

the query does not restrict the number of rows returned, but rather affects which rows contain the null value. For rows that do not meet the qualification, a null value appears in the inner table’s columns of those rows.

**Views using Left Outer Joins:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create view viewtest as select a.title\_id,a.title,a.pub\_id,b.pub\_name from titles a left outer join publishers b on a.pub\_id=b.pub\_id  Select \* from viewtest | SYNTAX:  create view viewtest as select a.title\_id,a.title,a.pub\_id,b.pub\_name fro  m titles a left outer join publishers b on a.pub\_id=b.pub\_id  Select \* from viewtest |
| OUTPUT:  title\_id  title  pub\_id pub\_name  --------  --------------------------------------------------------------------------------  ------ ----------------------------------------  BU1032  The Busy Executive's Database Guide  1389 Algodata Infosystems  BU1111  Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  BU2075  You Can Combat Computer Stress!  0736 New Age Books  BU7832  Straight Talk About Computers  1389 Algodata Infosystems  MC2222  Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  MC3021  The Gourmet Microwave  0877 Binnet & Hardley  MC3026  The Psychology of Computer Cooking  0877 Binnet & Hardley  PC1035  But Is It User Friendly?  1389 Algodata Infosystems  PC8888  Secrets of Silicon Valley  1389 Algodata Infosystems  PC9999  Net Etiquette  1389 Algodata Infosystems  PS1372  Computer Phobic and Non-Phobic Individuals: Behavior Variations  0877 Binnet & Hardley  PS2091  Is Anger the Enemy?  0736 New Age Books  PS2106  Life Without Fear  0736 New Age Books  PS3333  Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  PS7777  Emotional Security: A New Algorithm  0736 New Age Books  TC3218  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean  0877 Binnet & Hardley  TC4203  Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  TC7777  Sushi, Anyone?  0877 Binnet & Hardley  (18 rows affected) | OUTPUT:  TITLE\_ID TITLE  PUB\_ID PUB\_NAME  -------------------- -----------------------------------------------------------  --------------------- ------ ----------------------------------------  PS7777 Emotional Security: A New Algorithm  0736 New Age Books  PS3333 Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  BU2075 You Can Combat Computer Stress!  0736 New Age Books  PS2091 Is Anger the Enemy?  0736 New Age Books  PS2106 Life Without Fear  0736 New Age Books  MC2222 Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  TC7777 Sushi, Anyone?  0877 Binnet & Hardley  TC4203 Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  MC3021 The Gourmet Microwave  0877 Binnet & Hardley  TC3218 Onions, Leeks, and Garlic: Cooking Secrets of the Mediterra  nean 0877 Binnet & Hardley  MC3026 The Psychology of Computer Cooking  0877 Binnet & Hardley  PS1372 Computer Phobic and Non-Phobic Individuals: Behavior Variat  ions 0877 Binnet & Hardley  PC8888 Secrets of Silicon Valley  1389 Algodata Infosystems  BU1032 The Busy Executive's Database Guide  1389 Algodata Infosystems  BU1111 Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  PC1035 But Is It User Friendly?  1389 Algodata Infosystems  BU7832 Straight Talk About Computers  1389 Algodata Infosystems  PC9999 Net Etiquette  1389 Algodata Infosystems  18 record(s) selected. |

**Views using Right Outer Joins:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create view viewtest as select a.title\_id,a.title,a.pub\_id,b.pub\_name from titles a right outer join publishers b on a.pub\_id=b.pub\_id  Select \* from viewtest | SYNTAX:  create view viewtest as select a.title\_id,a.title,a.pub\_id,b.pub\_name from titles a right outer join publishers b on a.pub\_id=b.pub\_id  Select \* from viewtest |
| OUTPUT:  title\_id  title  pub\_id pub\_name  --------  --------------------------------------------------------------------------------  ------ ----------------------------------------  PS7777  Emotional Security: A New Algorithm  0736 New Age Books  PS3333  Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  BU2075  You Can Combat Computer Stress!  0736 New Age Books  PS2091  Is Anger the Enemy?  0736 New Age Books  PS2106  Life Without Fear  0736 New Age Books  MC2222  Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  TC7777  Sushi, Anyone?  0877 Binnet & Hardley  TC4203  Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  MC3021  The Gourmet Microwave  0877 Binnet & Hardley  TC3218  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean  0877 Binnet & Hardley  MC3026  The Psychology of Computer Cooking  0877 Binnet & Hardley  PS1372  Computer Phobic and Non-Phobic Individuals: Behavior Variations  0877 Binnet & Hardley  PC8888  Secrets of Silicon Valley  1389 Algodata Infosystems  BU1032  The Busy Executive's Database Guide  1389 Algodata Infosystems  BU1111  Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  PC1035  But Is It User Friendly?  1389 Algodata Infosystems  BU7832  Straight Talk About Computers  1389 Algodata Infosystems  PC9999  Net Etiquette  1389 Algodata Infosystems  (18 rows affected) | OUTPUT:    TITLE\_ID TITLE  PUB\_ID PUB\_NAME  -------------------- -----------------------------------------------------------  --------------------- ------ ----------------------------------------  PS7777 Emotional Security: A New Algorithm  0736 New Age Books  PS3333 Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  BU2075 You Can Combat Computer Stress!  0736 New Age Books  PS2091 Is Anger the Enemy?  0736 New Age Books  PS2106 Life Without Fear  0736 New Age Books  MC2222 Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  TC7777 Sushi, Anyone?  0877 Binnet & Hardley  TC4203 Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  MC3021 The Gourmet Microwave  0877 Binnet & Hardley  TC3218 Onions, Leeks, and Garlic: Cooking Secrets of the Mediterra  nean 0877 Binnet & Hardley  MC3026 The Psychology of Computer Cooking  0877 Binnet & Hardley  PS1372 Computer Phobic and Non-Phobic Individuals: Behavior Variat  ions 0877 Binnet & Hardley  PC8888 Secrets of Silicon Valley  1389 Algodata Infosystems  BU1032 The Busy Executive's Database Guide  1389 Algodata Infosystems  BU1111 Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  PC1035 But Is It User Friendly?  1389 Algodata Infosystems  BU7832 Straight Talk About Computers  1389 Algodata Infosystems  PC9999 Net Etiquette  1389 Algodata Infosystems  18 record(s) selected. |

**ANSI Left Outer Join:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select au\_fname, au\_lname, pub\_name from publishers left join authors on authors.city = publishers.city  Select \* from viewtest | SYNTAX:  SELECT au\_fname,au\_lname,pub\_name  FROM publishers LEFT OUTER JOIN authors ON authors.city = publishers.city |
| OUTPUT:  au\_fname au\_lname  pub\_name  -------------------- ----------------------------------------  ----------------------------------------  NULL NULL  New Age Books  NULL NULL  Binnet & Hardley  Abraham Bennet  Algodata Infosystems  Cheryl Carson  Algodata Infosystems  (4 rows affected) | OUTPUT:  AU\_FNAME AU\_LNAME PUB\_NAME  -------------------- ---------------------------------------- ------------------  ----------------------  - - New Age Books  - - Binnet & Hardley  Cheryl Carson Algodata Infosyste  ms  Abraham Bennet Algodata Infosyste  ms  4 record(s) selected. |

**ANSI Right Outer Joins:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select au\_fname, au\_lname, pub\_name from publishers right join authors on authors.city = publishers.city | SYNTAX:  SELECT au\_fname, au\_lname,pub\_name FROM publishers RIGHT OUTER JOIN authors ON authors.city = publishers.city |
| OUTPUT:  au\_fname au\_lname  pub\_name  -------------------- ----------------------------------------  ----------------------------------------  Johnson White  NULL  Marjorie Green  NULL  Cheryl Carson  Algodata Infosystems  Michael O'Leary  NULL  Dick Straight  NULL  Meander Smith  NULL  Abraham Bennet  Algodata Infosystems  Ann Dull  NULL  Burt Gringlesby  NULL  Chastity Locksley  NULL  Morningstar Greene  NULL  Reginald Blotchet-Halls  NULL  Akiko Yokomoto  NULL  Innes del Castillo  NULL  Michel DeFrance  NULL  Dirk Stringer  NULL  Stearns MacFeather  NULL  Livia Karsen  NULL  Sylvia Panteley  NULL  Sheryl Hunter  NULL  Heather McBadden  NULL  Anne Ringer  NULL  Albert Ringer  NULL  (23 rows affected) | OUTPUT:  AU\_FNAME AU\_LNAME PUB\_NAME  -------------------- ---------------------------------------- ----------  ----------------------  Johnson White -  Marjorie Green -  Cheryl Carson Algodata I  ms  Michael O'Leary -  Dick Straight -  Meander Smith -  Abraham Bennet Algodata I  ms  Ann Dull -  Burt Gringlesby -  Chastity Locksley -  Morningstar Greene -  Reginald Blotchet-Halls -  Voilet Yokomoto -  Innes del Castillo -  Michel DeFrance -  Dirk Stringer -  Stearns MacFeather -  Livia Karsen -  Sylvia Panteley -  Sheryl Hunter -  Heather McBadden -  Anne Ringer -  Albert Ringer -  23 record(s) selected. |

**TSQL Outer Joins:**

**Left Outer Join:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select au\_fname, au\_lname, pub\_name from authors, publishers  where authors.city \*= publishers.city | SYNTAX:  select au\_fname, au\_lname, pub\_name from authors left outer join publishers on authors.city = publishers.city |
| OUTPUT:  au\_fname au\_lname  pub\_name  -------------------- ----------------------------------------  ----------------------------------------  Johnson White  NULL  Marjorie Green  NULL  Cheryl Carson  Algodata Infosystems  Michael O'Leary  NULL  Dick Straight  NULL  Meander Smith  NULL  Abraham Bennet  Algodata Infosystems  Ann Dull  NULL  Burt Gringlesby  NULL  Chastity Locksley  NULL  Morningstar Greene  NULL  Reginald Blotchet-Halls  NULL  Akiko Yokomoto  NULL  Innes del Castillo  NULL  Michel DeFrance  NULL  Dirk Stringer  NULL  Stearns MacFeather  NULL  Livia Karsen  NULL  Sylvia Panteley  NULL  Sheryl Hunter  NULL  Heather McBadden  NULL  Anne Ringer  NULL  Albert Ringer  NULL  (23 rows affected) | OUTPUT:  AU\_FNAME AU\_LNAME PUB\_NAME  -------------------- ---------------------------------------- ------------------  ----------------------  Johnson White -  Marjorie Green -  Cheryl Carson Algodata Infosyste  ms  Michael O'Leary -  Dick Straight -  Meander Smith -  Abraham Bennet Algodata Infosyste  ms  Ann Dull -  Burt Gringlesby -  Chastity Locksley -  Morningstar Greene -  Reginald Blotchet-Halls -  Akiko Yokomoto -  Innes del Castillo -  Michel DeFrance -  Dirk Stringer -  Stearns MacFeather -  Livia Karsen -  Sylvia Panteley -  Sheryl Hunter -  Heather McBadden -  Anne Ringer -  Albert Ringer -  23 record(s) selected. |

**Note:** The \* sign used in the joins is not recognised by db2 .So we need to use the ANSI syntax for joins.

**RIGHT OUTER JOIN:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select au\_fname, au\_lname, pub\_name from authors, publishers where authors.city =\* publishers.city | SYNTAX:  SELECT au\_fname,au\_lname,pub\_name FROM authors RIGHT OUTER JOIN publisher  s ON authors.city = publishers.city |
| OUTPUT:  au\_fname au\_lname  pub\_name  -------------------- ----------------------------------------  ----------------------------------------  NULL NULL  New Age Books  NULL NULL  Binnet & Hardley  Abraham Bennet  Algodata Infosystems  Cheryl Carson  Algodata Infosystems  (4 rows affected) | AU\_FNAME AU\_LNAME PUB\_NAME  -------------------- ---------------------------------------- ------------------  ----------------------  - - New Age Books  - - Binnet & Hardley  Cheryl Carson Algodata Infosyste  ms  Abraham Bennet Algodata Infosyste  ms  4 record(s) selected. |

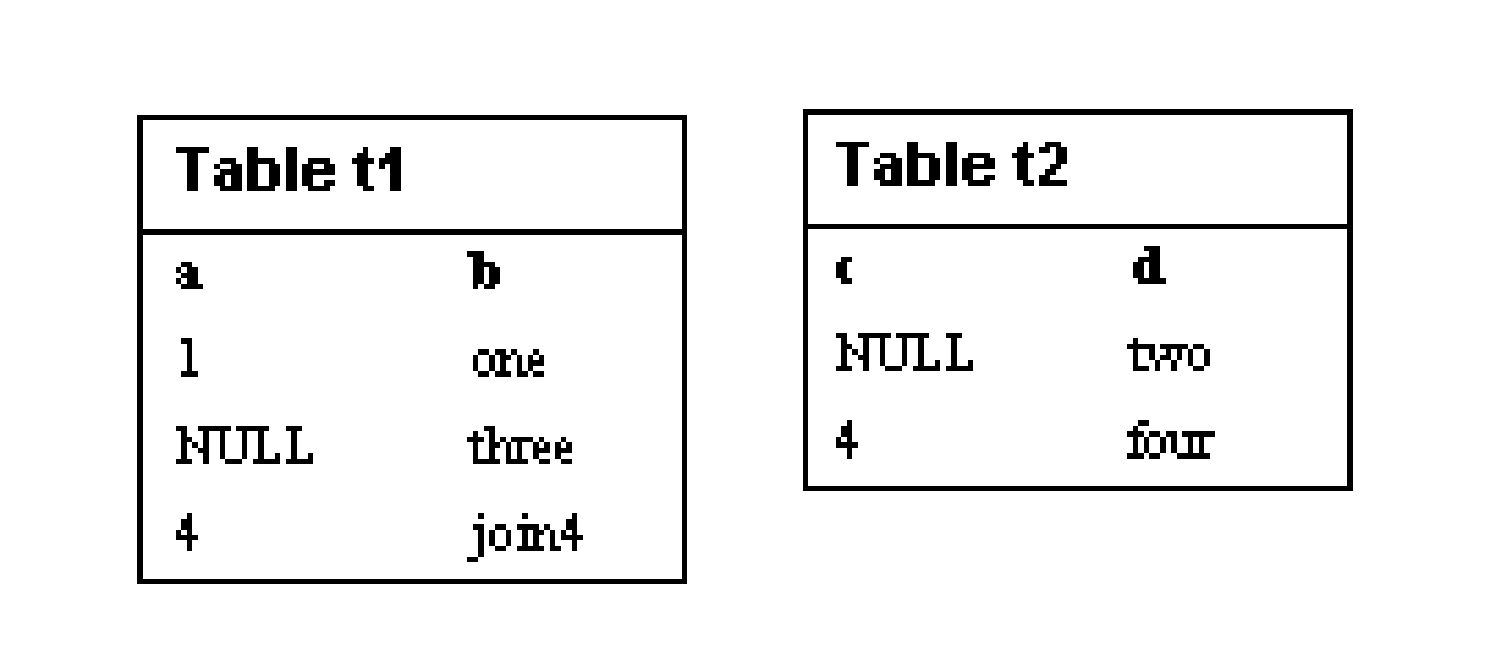
**Note:** The \* sign used in the joins is not recognised by db2 .So we need to use the ANSI syntax for joins.

**Null values and outer joins:**

Null values in tables or views being joined will never match each other. Since bit columns do not permit null values, a value of 0 appears in an outer join when there is no match for a bit column that is in the inner table. The result of a join of NULL with any other value is NULL. Because null values represent unknown or inapplicable values, there is no reason to believe that one unknown value matches another.

You can detect the presence of null values in a column from one of the tables being joined only by using an outer join. Here are two tables, each of which has a NULL in the column that will participate in the join. A left outer join displays the null value in the first table.

**Figure: Null values in outer join**



| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select \* from t1, t2 where a \*= c | SYNTAX:  SELECT \* FROM T1 LEFT OUTER JOIN T2 ON T1.a = T2.c |
| OUTPUT:  a b c d  ----------- ------ ----------- ------  1 one NULL NULL  NULL three NULL NULL  4 join4 4 four  (3 rows affected) | OUTPUT:  A B C D  ----------- ---------- ----------- ----------  1 one - -  - three - -  4 join4 4 four  3 record(s) selected. |

**Note:** The \* sign used in the joins is not recognised by db2 .So we need to use the ANSI syntax for joins.

## **Stored Procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure [*owner*.]*procedure\_name*[;*number*]  [[(]@*parameter\_name*  *datatype* [(*length*) | (*precision* [, *scale* ])]  [= *default*] [output]  [, @*parameter\_name*  *datatype* [(*length*) | (*precision* [, *scale*])]  [= *default*] [output]]...[)]]  [with recompile]  as {*SQL\_statements* | external name *dll\_name*} | SYNTAX:  create procedure *procedure\_name*(IN/OUT/INOUT *parameter\_ name data-type*)  SPECEFIC *specific name*  DYNAMIC RESULT SETS *integer*  MODIFIES SQL DATA/NO SQL/CONTAINS SQL/READS SQL DATA  NON DETERMINISTIC/ DETERMINISTIC  LAUNGUAGE *C/JAVA/COBOL/OLE* |
| EXAMPLE:  create proc get\_employee\_details @lastname varchar(40),  @firstname varchar(20) as  select au\_lname, au\_fname from  department  where au\_fname = @firstname  and au\_lname = @lastname | EXAMPLE:  create procedure get\_employee\_details (v\_lastname varchar(40), v\_firstname varchar(20) ) language sql begin declare temp\_cursor cursor with return   for select au\_lname,  au\_fname  from department  where au\_fname = v\_firstname  au\_lname = v\_lastname; open temp\_cursor; end @ |

CONCLUSION

We cannot retrieve more than one row in DB2, in order to do that we need to declare a cursor with return option as shown in the example.

NOTE

If you want to compile the stored procedure through the command prompt then “@” terminator needs to be included after the end

### **Using Default Parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set v\_a = a;  set v\_b = b;  set c = v\_a + v\_b;  end poc @ |
| DESCRIPTION:  In the above example two parameters with default values have been assigned. | DESCRIPTION:  In DB2 we cannot assign default values directly to the parameters. A workaround for this could be declaring a local variable and assigning default values |

### **Null as Default Parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = null  as  declare @c int  select @c=@a+@b | SYNTAX: Not Supported  WORKAROUND :  CREATE PROCEDURE add\_proc (IN a INTEGER, IN b INTEGER,  OUT c INTEGER)  POC: BEGIN  DECLARE v\_a INT DEFAULT 10;  DECLARE v\_b INT DEFAULT NULL;  SET a = v\_a;  SET b = v\_b;  SET c = a + b;  END POC |
| DESCRIPTION:  The above example show how nulls can be assigned to parameters. | DESCRIPTION:  The equivalent functionality is not available in DB2.Workaround is possible. |

### **Wildcard as Default Parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure showind4  @table varchar(30) = "sys%" as  select table\_name = sysobjects.name,index\_name = sysindexes.name,index\_id = indid  from sysindexes,sysobjects  where sysobjects.name like @table  and sysobjects.id = sysindexes.id | SYNTAX: Not Supported  WORKAROUND :  CREATE PROCEDURE showind4 (tableprefix VARCHAR(30) ) POC: BEGIN  DECLARE v\_tableprefix VARCHAR(30);    SET v\_tableprefix = 'sys%';  SET tableprefix = v\_tableprefix;  DECLARE temp\_cursor CURSOR WITH RETURN TO CLIENT FOR SELECT sysobjects.name AS table\_name,sysindexes.name AS index\_name, indid AS index\_id FROM sysindexes,sysobjects WHERE SYB.isLike(sysobjects.name, tableprefix, '') = 1  AND sysobjects.id = sysindexes.id;  OPEN temp\_cursor;  END POC |
| DESCRIPTION:  The above example show how wildcard characters can be assigned to parameters. | DESCRIPTION:  The equivalent functionality is not available in DB2.Workaround is possible. |

### **Procedure Groups**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  drop proc orders | Not Supported  WORKAROUND:  We need to identify the stored procedures and their dependent child procedures and then need to drop them explicitly one by one or in a script file. |
| DESCRIPTION:  The optional semicolon and integer number after the name of the procedure in the create procedure and execute statements allow you to group procedures of the same name so that they can be dropped together with a single drop procedure statement. Procedures used in the same application are often grouped this way. For example, you might create a series of procedures called orders;1, orders;2,and so on.  The above syntax will drop the entire group of stored procedures. | DESCRIPTION:  The equivalent functionality is not available in DB2. |

### **Using Recompile**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = null  with recompile  as  declare @c int  select @c=@a+@b | Not Supported |
| DESCRIPTION:  Here WITH RECOMPILE option can be used which will recompile the SP before execution and will create optimal access paths for the sql statements in the SP. | DESCRIPTION:  In DB2 we cannot explicitly mention the recompile option inside a SP but recompilation is done automatically. |

### **Using Recompile in Execute**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  exec storeid\_proc "7066" with recompile | Not Supported |
| DESCRIPTION:  In the above example WITH RECOMPILE can be also used with an exec statement. | DESCRIPTION:  The equivalent functionality is not available in DB2. |

### **Nesting Procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b  exec get\_val @d | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set v\_a = a;  set v\_b = b;  set c = v\_a + v\_b;  call get\_val(in d int);  end poc @ |
| DESCRIPTION:  In the above example get\_val is the child SP which is called by add\_proc. | DESCRIPTION:  In the above example exec is been replaced by call statement. |

### **Temp Tables**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure inv\_proc as  create table #tempstores  (stor\_id char(4), amount money) | SYNTAX:  create procedure inv\_proc() poc begin  declare global temporary table session."#tempstores"  (stor\_id char(4) not null,  amount decimal(19,4) not null  ) with replace on commit preserve rows not logged;  end poc @ |
| DESCRIPTION:  In the above example #tempstores is the temp table which is persistent until the current session of that SP. | DESCRIPTION:  In the above example session.#tempstores is the temp table which is persistent until the current session of that SP. |

### **Setting options in stored procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  Select pub\_id from titles set rowcount 2 | SYNTAX:  Not Supported  WORKAROUND 1(FOR SELECT):  Select pub\_id from titles fetch first 2 rows only;  WORKAROUND 2(FOR UPDATE):  create procedure set\_rowcount\_test(IN p\_name char(30))  poc: Begin  Declare v\_counter int default 1;  Declare v\_name varchar(6);  Declare v\_sr\_no int;  Declare v\_commit\_freq int default 0;  Declare insrt\_c1 cursor with hold  for select sr\_no,name from  data\_commit;  open insrt\_c1;  While (v\_counter <= 1000) do  Fetch insrt\_c1 into v\_sr\_no,v\_name;  Update data\_commit  Set name = 'pradeep'  where sr\_no = v\_sr\_no;  Set v\_counter = v\_counter +1;  Set v\_commit\_freq = v\_commit\_freq + 1;  If (v\_commit\_freq = 3) then  Commit;  Set v\_commit\_freq = 0;  End if;  End while;  end poc |
| DESCRIPTION:  Here if there are n number of pub\_id’s then only 2 rows will be returned | DESCRIPTION:  There is no mapping of this functionality in DB2. |

CONCLUSION

The above functionality can be implemented in DB2 by adding commit logic in the application code.In the above example workaround2 commit frequency is set to 3. i.e. for every 3 records the transaction will be committed.In DB2 if you want to fetch the first two records then example given in workaround1 would be fine.

### **Arguments for stored procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b  exec get\_val @d | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set a = v\_a;  set b = v\_b;  set c = a + b;  call get\_val(in d int);  end poc @ |
| DESCRIPTION:  Here the arguments passed are 10 and 20. | DESCRIPTION:  We cannot directly pass arguments to stored procedures instead we can set some values to the parameters in the code itself. |

### **Executing procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  execute procname(parm1,parm2,parm3...) | SYNTAX:  call procname(in parm1 ,in parm2,? …); |
| DESCRIPTION:  In Sybase execute is made use for calling a child stored procedure. | DESCRIPTION:  The equivalent in DB2 is a call statement. |

### **Cross database references**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  select pub\_id from pubs1..titles | SYNTAX:  create nickname r\_*titles* for *db2server.salesdata.europe* |
| DESCRIPTION:  we are trying to access titles table which is residing in pubs1. | DESCRIPTION:  In DB2 we need to create a nickname first in order to access the remote d database object.  R\_*Titles* → A unique nickname used to identify the table or view.*DB2SERVER.SALESDATA.EUROPE* → A three part identifier for the remote object. |

### **Executing procedures after a time delay**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  begin  waitfor delay "0:00:05"  select \* from titles  end  go | SYNTAX:  begin  select \* from titles;  call syb.waitfor('delay','0:00:05');  end; |
| DESCRIPTION:  In the above example waitfor will defer the execution of the select statement for 5 seconds. | DESCRIPTION:  Here after translation thru MTK syb.waitfor function is used to give the same output. |

### **Executing procedures remotely**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  You can execute procedures on a remote Adaptive Server from your local Adaptive Server. Once both servers are properly configured, you can execute any procedure on the remote Adaptive Server simply by using the server name as part of the identifier.  Example:  exec gateway.remotedb.dbo.remoteproc | DESCRIPTION:  If you are migrating applications that invoke stored procedures through nicknames, you need to modify your applications. DB2 Information Integrator does not currently support the ability to invoke a stored procedure using a nickname.  Using **Federated System** which is a special type of distributed database management system (DBMS) we can access tables/views which reside on a remote server.  The power of a DB2 federated system is in its ability to:   * Join data from local tables and remote data sources, as if all the data is stored locally in the federated database * Update data in relational data sources, as if the data is stored in the federated database v * Replicate data to and from relational data sources * Take advantage of the data source processing strengths, by sending requests to the data sources for processing * Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server |

### **Return status**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  declare @status int  execute @status = byroyalty 50  select @status | SYNTAX:  Not Supported  WORKAROUD:  Create procedure checkcontract (v\_titleid tid)  poc: begin  if (select contract from titles  where title\_id = v\_titleid) = 1  then  return 1;  else  return 2;  end if;  end poc @ |
| DESCRIPTION:  Stored procedures report a **return status** that indicates whether or not they completed successfully, and if they did not, the reasons for failure. This value can be stored in a variable when a procedure is called, and used in future Transact-SQL statements. Adaptive Server-defined return status values for failure range from -1 through -99; users can define their own return status values outside this range. | DESCRIPTION:  The equivalent functionality is not available in DB2, however user generated return values can be used in the application to serve the same purpose. |

### **Reserved return status values**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Adaptive Server reserves 0, to indicate a successful return, and negative values from -1 through -99, to indicate the reasons for failure. Numbers 0 and -1 through -14 are currently used in version 12 | DESCRIPTION:  The equivalent functionality is not available in DB2. |

### **User-generated return values**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create proc checkcontract @titleid tid  as  if (select contract from titles where  title\_id = @titleid) = 1  return 1  else  return 2 | SYNTAX:  Create procedure checkcontract (v\_titleid tid)  poc: begin  if (select contract from titles  where title\_id = v\_titleid) = 1  then  return 1;  else  return 2;  end if;  end poc @ |
| DESCRIPTION:  You can generate your own return values in stored procedures by adding a parameter to the return statement. | DESCRIPTION:  The equivalent in DB2 is the same. |

### **Return parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b  exec get\_val @d | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set a = v\_a;  set b = v\_b;  set c = a + b;  call get\_val(in d int);  end poc @ |
| DESCRIPTION:  Here the return parameter is @c, which will return the value of summation of @a + @b. | DESCRIPTION:  The summation of a + b is returned in c. The return parameter here is c. |

### **Passing values in parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  exec myproc @a = @myvara in, @b = @myvarb out  exec get\_details ‘Albert’,115619 | SYNTAX:  call myproc (a,?)    call get\_details(‘Albert’,115619) |
| DESCRIPTION:  In the first example there are two parameters one input and another output. We are passing the value of the variable @myvara as the first parameter and getting the out value in @myvarb.  In the second example we are passing two values to the parameters. one is a character and the other an integer. | DESCRIPTION:  In the first example the output parameter is represented by a ‘?’ symbol. In DB2 we cannot directly assign a variable to a parameter.  The second example is similar to the Sybase one. |

### **Output variable**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b  exec get\_val @d | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set a = v\_a;  set b = v\_b;  set c = a + b;  call get\_val(in d int);  end poc @ |
| DESCRIPTION:  Here the return parameter is @c, which will return the value of summation of @a + @b. | DESCRIPTION:  The summation of a + b is returned in c. The return parameter here is c. |

### **Qualifying names inside procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure get\_name  as  select emp\_name from  finance.employee  where emp\_id = 115619 | SYNTAX:  create procedure get\_name  poc: begin  select emp\_name from  finance.employee  where emp\_id = 115619 end poc @ |
| DESCRIPTION:  In the above example finance. Is the qualifying name. | DESCRIPTION:  In the above example finance. Is the qualifying name. |

### **Renaming objects referenced by procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  sp\_rename *objname*, *newname*  EXAMPLE:  sp\_rename friends\_etc, infotable | SYNTAX:  db2 rename table/view <source\_name> to <target\_identifier>  db2 rename tablespace <source\_tablespace\_name> to <target\_ tablespace\_name >  EXAMPLE:  db2 rename table employee\_details to emp\_details  db2 rename tablespace master\_data to mst\_data |
| DESCRIPTION:  To rename tables and other database objects like columns, constraints, datatypes, views, indexes, rules, defaults, procedures, and triggers we make use of sp\_rename. You must own an object to rename it. You cannot change the name of system objects or system datatypes. The Database Owner can change the name of any user’s objects. Also, the object whose name you are changing must be in the current database.  The above example is to change the name of friends\_etc to infotable. | DESCRIPTION:  The RENAME statement renames an existing table, index and tablespace.We cannot use RENAME for any other database object.  The above two examples are for renaming a table and a tablespace. |

## **TRIGGERS**

## **Creating Triggers**

DESCRIPTION

A trigger defines a set of actions that are activated, or triggered, by an insert, update, or delete on specified base table. Triggers are powerful tools that can be used for several purposes.

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  create trigger test\_trigger  on titles  for insert  as  raiserror 9999 “Titleauthor need to be updated.” | create trigger test\_trigger  AFTER INSERT  on titles  REFERENCING NEW\_TABLE as new  FOR EACH STATEMENT  MODE DB2SQL  raiserror 9999 “Titleauthor need to be updated.” |
| Result: |  |

CONCLUSION: Sybase triggers are “AFTER” triggers whereas DB2 triggers can be “AFTER or BEFORE” tiggers.

## **Insert, update triggers**

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  create trigger trig01  on table01 for insert, update  as  begin  declare @savecount int  select @savecount = @@rowcount  --Trigger is not fired if the table is maintained by the maint user  if suser\_name() like "%maint"  return  if @savecount = 0  return  if not exists (select \* from table02,inserted where  col01 = "AA" and col02 = “BB”)  begin  raiserror 88888 "BB does not exist.”  rollback transaction  return  end  if not exists (select \* from table02,inserted where  col01 = "AA" and col02 = “CC”)  begin  raiserror 99999 "CC does not exist."  rollback transaction  return  end  end | For Insert:  CREATE TRIGGER trig01ins  AFTER INSERT (1)  ON table01  REFERENCING NEW\_TABLE as new (2)  FOR EACH STATEMENT (3)  MODE DB2SQL (4)  WHEN  ((select USER from sysibm.sysdummy1 where USER like '%maint') is null  (5)  and  ( select count(\*) from new) > 0)) (6)  BEGIN ATOMIC (7)  VALUES (  CASE WHEN (select count(\*) from table02, new where  col01 = 'AA' and col02 = ‘BB’) = 0  THEN raise\_error ('88888', 'BB does not exist.') (8)  WHEN (select count(\*) from table02,new where  col01 = 'AA' and col02 = ‘CC’) = 0  THEN raise\_error ('99999', 'CC does not exist.')  else 0  end);  end  For Update:  CREATE TRIGGER trig01upd  AFTER UPDATE  ON table01  REFERENCING NEW\_TABLE as new  FOR EACH STATEMENT  MODE DB2SQL  WHEN  ((select USER from sysibm.sysdummy1 where USER like '%maint') is null  and  ( select count(\*) from new) > 0))  BEGIN ATOMIC  VALUES (  CASE WHEN (select count(\*) from table02, new where  col01 = 'AA' and col02 = ‘BB’) = 0  THEN raise\_error ('88888', 'BB does not exist.')  WHEN (select count(\*) from table02,new where  col01 = 'AA' and col02 = ‘CC’) = 0  THEN raise\_error ('99999', 'CC does not exist.')  else 0  end);  end |

CONCLUSION: In Sybase, both insert and update statements can be in a single trigger where DB2 can have only one of them.

## **Delete Triggers**

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  create trigger trg03  on table04 for delete  as  begin  if suser\_name() like "%maint"  return  delete table05  from table05 a,deleted b  where a.col01 = b.col01  end | CREATE TRIGGER trg03  AFTER DELETE ON table04  REFERENCING OLD\_TABLE AS oldtable  FOR EACH STATEMENT MODE DB2SQL  WHEN ((select USER from sysibm.sysdummy1 where USER like '%maint') is null )  BEGIN ATOMIC  delete from table05  where col01= (select col01 from oldtable);  END |

CONCLUSION: DB2 does not allow joined tables in deletes; this needed to be replaced by a subquery.

## **if update (column name)**

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  create trigger trg04 on table06 for update as  begin  declare @count int  select @count = count(\*) from inserted  if @count = 0  return  if update(col01) or update(col02)  begin  rollback transaction  raiserror 666666 'can not update col01 or col02'  return  end  insert table07 (col01,col02)  select getdate(), ‘tried to update col01 or col02’ from inserted  end | CREATE TRIGGER trg04  AFTER UPDATE OF col01, col02 ON table06 (1)  REFERENCING NEW\_TABLE AS new  FOR EACH STATEMENT MODE DB2SQL  WHEN ((select count(\*) from new) > 0)  BEGIN ATOMIC  SIGNAL SQLSTATE '66666' ('can not update col01 or col02'); (2)  insert into table07 (col01,col02)  (select CURRENT TIMESTAMP,’tried to update col01 or col02’ from new);  END |

## **Difference between ROLLBACK TRANSACTION and ROLLBACK TRIGGER**

DESCRIPTION

Rollback trigger rolls back only the effect of the trigger and the statement that caused the trigger to fire whereas rollback transaction rolls back the entire transaction.

EXAMPLE

**Rollback Trigger:**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create trigger trg\_publishers  on publishers  for insert as  if ( select count(\*) from inserted  where inserted.city = "Delhi" ) = 1  begin  rollback trigger with raiserror 88858 "City cannot be  delhi."  end    begin tran  insert into publishers  values("9901","Sharma publication","Bangalore","KA")  insert into publishers  values("9902","Surjit publication","Delhi","HA")  commit tran  select \* from publishers | Not supported. |
| RESULT  pub\_id pub\_name city state  ------ ------------------------------- ------------------ -----  0736 New Age Books Boston MA  0877 Binnet & Hardley Washington DC  1389 Algodata Infosystems Berkeley CA  9901 Sharma publication Bangalore KA |  |

* + 1. **Rollback Transaction**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create trigger trg\_publishers  on publishers  for insert as  if ( select count(\*) from inserted  where inserted.city = "Delhi" ) = 1  begin  rollback tran  end    begin tran  insert into publishers  values("9901","Sharma publication","Bangalore","KA")  insert into publishers  values("9902","Surjit publication","Delhi","HA")  commit tran  Select \* from publishers | create trigger trg\_publishers  after insert  on publishers  referencing NEW as new  for each row  mode DB2SQL  when( new.city = 'Delhi' )  SIGNAL SQLSTATE ’9901’ SET MESSAGE\_TEXT=’City cannot be Delhi’  Select \* from publishers |
| RESULT  pub\_id pub\_name city state  ------ ---------------------------------- ------------------ -----  0736 New Age Books Boston MA  0877 Binnet & Hardley Washington DC  1389 Algodata Infosystems Berkeley CA | PUB\_ID PUB\_NAME CITY STATE  ------ ------------------------------------ -------------- -----  0736 New Age Books Boston MA  0877 Binnet & Hardley Washington DC  1389 Algodata Infosystems Berkeley CA |

* + 1. **Nesting Triggers**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create trigger trg1\_publishers  on publishers  for update as  if update(pub\_id)  begin  update test\_publishers  set test\_publishers.pub\_id=inserted.pub\_id  from test\_publishers,inserted,publishers  where test\_publishers.pub\_id = publishers.pub\_id  create trigger trg1\_test\_publishers  on test\_publishers  for update as  if update(pub\_id)  begin  update test\_publishers1  set test\_publishers1.pub\_id=inserted.pub\_id  from test\_publishers1,inserted,test\_publishers  where test\_publishers1.pub\_id = test\_publishers.pub\_id  end | create trigger trg\_publishers  after insert  on publishers  referencing NEW as new  for each row  mode DB2SQL  when( new.city = 'Delhi' )  insert into publishers1 values(‘9999’,’Test’,’sd’,’MA’)  create trigger trg\_publishers1  after insert  on publishers1  referencing NEW as new  for each row  mode DB2SQL  when( new.city = 'Delhi' )  insert into publishers2 values(‘9999’,’Test’,’sd’,’MA’) |

* + 1. **Sybase Restrictions**

| **SYBASE** | **DB2** |
| --- | --- |
| Restrictions  Followings are not allowed in a trigger  • All create commands, including create database, create table, create  index, create procedure, create default, create rule, create trigger, and  create view  • All drop commands  • alter table and alter database  • truncate table  • grant and revoke  • update statistics  • reconfigure  • load database and load transaction  • disk init, disk mirror, disk refit, disk reinit, disk remirror, disk unmirror  • select into | Followings are not allowed in a trigger  • All create commands, including create database, create table, create  index, create procedure, create default, create rule, create trigger, and  create view  • All drop commands  • alter table and alter database  • truncate table  • grant and revoke  • update statistics  • reconfigure  • load database and load transaction  • disk init, disk mirror, disk refit, disk reinit, disk remirror, disk unmirror  • select into |

* + 1. **Self Recursive Triggers**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create trigger budget\_change  on new\_budget  for update as  if exists (select \* from inserted  where parent\_unit is not null)  begin  set self\_recursion on  update new\_budget  set new\_budget.budget = new\_budget.budget +  inserted.budget - deleted.budget  from inserted, deleted, new\_budget  where new\_budget.unit = inserted.parent\_unit  and new\_budget.unit = deleted.parent\_unit  end | Not supported. |

* + 1. **Disabling Triggers**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  Alter table publishers  disable trigger trg\_publishers | UPDATE db2admin.trigger\_state SET active='Y'  WHERE trigschema='DB2ADMIN' and trigname='TRIG1' |

## 

## **Datatypes**

### **User Defined Datatypes**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SP\_ADDTYPE DATATYPE NAME,  PHYSTYPE[ (length) | (precision [, scale] ) ]  [,"IDENTITY" | NULLTYPE ]  EXAMPLES:  SP\_ADDTYPE IdentType, "NUMERIC(4,0)",  "IDENTITY"  ============================================    SP\_ADDTYPE hk\_id,"CHAR(10)", "NOT NULL"      ============================================  SP\_ADDTYPE hk\_id, "CHAR(10)", "NULL” | SYNTAX:  CREATE DISTINCT TYPE distinct type  AS source datatype WITH COMPARISONS  EXAMPLES:  CREATE DISTINCT TYPE IdentType AS DECIMAL(4,0) WITH COMPARISONS  ===================================  CREATE DISTINCT TYPE hk\_id  AS  CHAR(10)  WITH COMPARISONS  ===================================  CREATE DISTINCT TYPE hk\_id  AS  CHAR(10)  WITH COMPARISONS |
| DESCRIPTION:  A Transact-SQL enhancement to SQL allows you to name and design your own datatypes to supplement the system datatypes. A user-defined datatype is defined in terms of system datatypes. You can give one name to a frequently used datatype definition. This makes it easy for you to custom fit datatypes to columns. | DESCRIPTION:  The CREATE DISTINCT TYPE statement defines a distinct data type which is most frequently used in the application, we can give our own names to the data type defined. The distinct type is always sourced on one of the built-in data types. |

NOTE:

Irrespective of whether the user defined data type is null, not null, identity, in DB2 it can not be created.

As a workaround

1. We have to identify the declarations of UDT in Sybase.
2. Make a note of whether “IDENTITY”, “NULL” OR “NOT NULL” constraint is added.
3. Where ever a column is declared with the user defined type, we have to specify it with the respective constraint found in the UDT declaration.

### **System Defined Datatypes**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  A datatype defines the kind of information each column in a table holds, and how that information is stored. You can use Adaptive Server system datatypes when you are defining columns, or you can create and use userdefined datatypes. | DESCRIPTION:  The smallest unit of data that can be manipulated in SQL is called a *value*.  Values are interpreted according to the data type of their source. Sources  include:   * Constants * Columns * Host variables * Functions * Expressions * Special registers.   DB2 supports a number of built-in data types. |
| LIST:  TINYINT  SMALLINT  INT  BIGINT  NUMERIC(P,S)  DECIMAL(P,S)  FLOAT  DOUBLE PRECISION  REAL  SMALL MONEY  MONEY  SMALLDATETIME  DATETIME  CHAR(n)  CHAR(n) NULL  VARCHAR(n)  NVARCHAR(n)  NCHAR(n)  NCHAR(n) NULL  BINARY(n)  BINARY(n) NULL  VARBINARY(n)  BIT  IMAGE  TEXT  NTEXT  SYSNAME  SQL\_VARIANT  UNIQUE IDENTIFIER | LIST:  SMALLINT  INTEGER  BIGINT  DECIMAL(P,S)  DECIMAL(P,S)  DOUBLE  DOUBLE  REAL  DECIMAL(10,4)  DECIMAL(19,4)  TIMESTAMP  TIMESTAMP  CHAR(n)  VARCHAR(N)  VARCHAR(n)  VARGRAPHIC(n)  GRAPHIC(n)  VARGRAPHIC(n)  CHAR(n) FOT BIT DATA  VARCHAR(n) FOR BIT DATA  VARCHAR(n) FOR BIT DATA  SMALLINT  BLOB(2G)  CLOB(2G)  DBCLOB  VARCHAR(30)  VARCHAR(800)  CHAR(16) FOR BIT DATA |

## **Tables**

### **Creating Table**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  Create table test\_syb (  id char(4) NOT NULL,  name varchar(40) NOT NULL,  age smallint )  sp\_help test\_syb | Create table test\_table (  id char(4) NOT NULL,  name varchar(40) NOT NULL,  age smallint )  describe test\_table |
| RESULT:  Column\_name Type Length Nulls  ----------- ---- ----------- -----------  char 4 NULL 0 NULL  name varchar 40 0 NULL  Age smallint 2 0 NULL | Column name Type name Length Nulls  ------------------- ---------------- ------- -------  ID CHARACTER 4 No  NAME VARCHAR 40 No  AGE SMALLINT 2 Yes |

**Conclusion:** If not declared, Sybase takes the value of a column as NOT NULL whereas DB2 takes as NULL.

### **Adding, Dropping, Modifying Columns**

* **Adding Column**

Example

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  alter table test\_table  add ms char(1) NULL | alter table test\_table  add ms char(1) |
| RESULT:  Column added successfully | Column added successfully |

**Conclusion:** Alter table is used in both Sybase and DB2 to add columns.

**NOTE:** if the column property is NOT NULL, then a value should be provided using default clause.

* **Dropping Column**

Example

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  alter table test\_table  drop age | Export the data, recreate the table with the dropped column and reload the data. You need to extract the DDL completely along with all the dependencies like indexes, views, SPs, triggers, user auth etc and current statistics. then export data twice and drop/recreate table and reload data, followed by statistics. Any static packages need to be rebound.  Offline dropping of column works with BETA DB2 version 9. This functionality would be expected this year in version 9. |
| **Result**  Column dropped successfully |  |

**Conclusion:** Drop column is supported in Sybase but not in DB2.

* **Modifying Column**

Example

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  alter table titles  modify type varchar(20) null | **Query:**  db2 ALTER TABLE MOHM.ALTER\_VARCHAR ALTER COLUMN COL1 SET DATA TYPE VARCHAR ( 50 ) |
| **Result**  Warning: a default (object id 368001311) is defined on column 'type' being modified. Check the validity of the default value after this ALTER TABLE operation.  Non-clustered index (index id = 2) is being rebuilt.  Non-clustered index (index id = 3) is being rebuilt.  Non-clustered index (index id = 4) is being rebuilt.  Warning: The schema for table 'titles' has changed. Drop and re-create each trigger on this table that uses the 'if update(column\_name)' clause.  (18 rows affected) | **Result**  0000IThe SQL command completed uccessfully. |

**Conclusion:** Drop column is supported in Sybase but not in DB2.

### **Using Select Into**

CONCLUSION:

The select into command lets us create a new table based on the columns specified in the select statement’s select list and the rows chosen in the where clause. The into clause is useful for creating test tables, new tables as copies of existing tables, and for making several smaller tables out of one large table.

Example

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  select \* into test\_select\_into  from authors | CREATE TABLE test\_select\_into AS (SELECT \*   FROM authors) DEFINITION ONLY  INSERT INTO test\_select\_into SELECT \* FROM authors |
| **Result:**  Authors table copied to test\_select\_into successfully. | Authors table copied to test\_select\_into successfully. |

**Conclusion:** In Sybase, only one statement can create a copy of and existing table, Whereas DB2 requires two statements to make a copy of an existing table.

### **User Defined Datatypes**

DESCRIPTION:

It allows users to name and design their own datatypes to supplement the system datatypes. A user-defined datatype is defined in terms of system datatypes.

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  sp\_addtype test\_datatype, "char(6)", "not null" | CREATE DISTINCT TYPE test\_datatype  AS CHAR(6) WITH COMPARISONS |
| **Result:**  Datatype added successfully. | Datatype added successfully. |

**Conclusion: “**sp\_addtype” is used to create user datatypes whereas “CREATE DISTINCT TYPE” is used in DB2

## **Cursors**

* + 1. **Body of a Cursor:**

| **SYBASE** | **DB2** |
| --- | --- |
| FLOWCHART:    Declaration:  declare *cursor\_name* cursor  for *select\_statement*  [for {read only | update [of  *column\_name\_list*]}]  Example:  declare business\_crsr cursor  for select title, title\_id  from titles  where type = "business"  for update of price  =====================================  Opening:  open *cursor\_name*  Example:  open business\_crsr  *=====================================*  Fetching:  fetch business\_crsr  Running fetch business\_crsr a second time displays the title and identification number of the next business book in titles.    =====================================  Closing:  Close *cursor\_name*  Example:  close business\_crsr | SYNTAX:  Declaration:  declare *cursor-name* cursor  with hold  with return  to client  to caller  for *select-statement/*  *update-statement*  Example:  declare businees\_crsr cursor  for select title, title\_id  from titles  where type = ‘business’;  *======================================*  Opening:  open *cursor\_name*  Example:  openbusiness\_crsr;  *======================================*  Fetching:  fetch business\_crsr into v\_title,title\_id  Each time this cursor is fetched the host variables will be holding the values of the next business book in titles.  ======================================  Closing:  close business\_crsr  Example:  close business\_crsr; |

* + 1. **Examples of Cursors**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure prc\_read\_table  as  declare @ret\_code int  select @ret\_code = 0  select \* from table01  if @@error != 0  select @ret\_code = @@error  return @ret\_code  =======================================  declare pubs\_crsr cursor  for select pub\_name, city, state  from publishers  for update of city, state | SYNTAX:  create procedure prc\_read\_table()  result set 1 language sql  begin  declare sqlcode integer default 0;  declare var\_sqlcode integer default 0;  declare cur1 cursor with hold with return  for select \*  from table01;  declare continue handler for sqlexception,sqlwarning,not found  set var\_sqlcode = sqlcode;  open cur1;  return(var\_sqlcode);  end  =========================================  declare pubs\_crsr cursor  for select pub\_name,city,state from publishers  for update; |
| DESCRIPTION:  In the first example a simple result set is fetched .  ===========================================  In the second example a cursor is declared for update of two columns city and state. | DESCRIPTION:  We cannot retrieve more than one record by using a simple select statement. The equivalent of the same in DB2 would be declaring a cursor with return option and opening it.  ============================================  The equivalent in DB2 is shown in the above example. |

* + 1. **Cursors and locking**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  For Reading Data:  Statements that read data (such as select or readtext) use shared locks on each data page to avoid reading changed data from an uncommitted transaction.  For Updating Data:  Update statements use exclusive locks on each  page they change. To reduce deadlocks and improve concurrency, Adaptive Server often precedes an exclusive lock with an update lock.  For updatable cursors, Adaptive Server uses update locks by default when scanning tables or views referenced with the for update clause of declare cursor.  NOTE:  Any exclusive locks acquired by a cursor in a transaction are held until the end of that transaction. This also applies to shared or update locks when you use the holdlock keyword or the set isolation level 3 option. However, if you do not set the close on endtran option, the cursor remains open past the end of the transaction, and its current page lock remains in effect. It can  also continue to acquire locks as it fetches additional rows. | DESCRIPTION:  If a cursor is declared with hold option. All locks are released, except locks protecting the current cursor position of open WITH HOLD cursors. The locks held include the locks on the table, and for parallel environments, the locks on rows where the cursors are currently positioned. Locks on packages and dynamic SQL sections (if any) are held.  RECOMMENDATIONS:  By making use of proper isolation levels we can avoid locking. Refer to Isolation topic for more information. |

* + 1. **Cursors and Performance**

| **SYBASE** | **DB2** |
| --- | --- |
| RECOMMENDATIONS:  General:  Applications should attempt to avoid updating join columns or columns with search clauses and other predicates to change their value when cursor scans are in progress.  For Updating Data:  Update statements use exclusive locks on each  page they change. To reduce deadlocks and improve concurrency, Adaptive Server often precedes an exclusive lock with an update lock.  For updatable cursors, Adaptive Server uses update locks by default when scanning tables or views referenced with the for update clause of declare cursor.  NOTE:  Any exclusive locks acquired by a cursor in a transaction are held until the end of that transaction. This also applies to shared or update locks when you use the holdlock keyword or the set isolation level 3 option. However, if you do not set the close on endtran option, the cursor remains open past the end of the transaction, and its current page lock remains in effect. It can  also continue to acquire locks as it fetches additional rows. | RECOMMENDATIONS:  Row blocking reduces database manager overhead for cursors by retrieving a *block* of rows in a single operation.  Note:The block of rows that you specify is a number of pages in memory. It is not a multi-dimensional (MDC) table block, which is physically mapped to an extent on disk.  Row blocking levels are specified by the following arguments to the BIND or  PREP commands:  **UNAMBIG**  Blocking occurs for read-only cursors and cursors not specified as  “FOR UPDATE OF”. Ambiguous cursors are treated as updateable.  **ALL**  Blocking occurs for read-only cursors and cursors not specified as  “FOR UPDATE OF”. Ambiguous cursors are treated as read-only.  **NO**  Blocking does not occur for any cursors. Ambiguous cursors are treated as read-only. |

* + 1. **Halloween Problem**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  The Halloween problem occurs when a client updates a column through a cursor, and that column defines the order in which the rows are returned from the base tables (that is, a unique indexed column). For example, if Adaptive Server accesses a base table using an index, and the index key is updated by the client, the updated index row can move within the index and be read again by the cursor. The row seems to appear twice in the result set: when the index key is updated by the client and when the updated index row moves farther down the result set. |  |
| EXAMPLE:  mytable has more than one unique index for example, on columns c3 and c4, you need to specify one unique index in the for update clause as follows:  declare mycursor cursor  for select c1, c2, c3  from mytable  for update of c1, c2, c3  RECOMMENDATIONS:  The Halloween problem can be avoided by creating tables with the unique auto\_identity index database option set to on. |  |

* + 1. **Declaring cursor in outer procedure and using inside sub procedure and trigger calls**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  create procedure au\_sales (@author\_id id)  as  declare author\_sales\_csr cursor for  select ta.title\_id, t.title, t.total\_sales  from titleauthor ta, titles t  where ta.title\_id = t.title\_id  and ta.au\_id = @author\_id  exec au\_sales\_subproc, @author\_id  end  create procedure au\_sales\_subproc (@author\_id id)  as  declare @title\_id tid  declare @title varchar(80)  declare @ytd\_sales int  declare @msg varchar(120)  open author\_sales  fetch author\_sales  into @title\_id, @title, @ytd\_sales  if (@@sqlstatus = 2)  begin  print "We do not sell books by this author."  close author\_sales  return  end  while (@@sqlstatus = 0)  begin  if (@ytd\_sales = NULL)  begin  select @msg = @title + " -- Had no sales this year."  print @msg  end  else if (@ytd\_sales < 500)  begin  select @msg = @title +" -- Had poor sales this year."  print @msg  end  else if (@ytd\_sales < 1000)  begin  select @msg = @title +" -- Had mediocre sales this year."  Chapter 17 Cursors: Accessing Data Row by Row  Transact-SQL User’s Guide 609  print @msg  end  else  begin  select @msg = @title +" -- Had good sales this year."  print @msg  end  fetch author\_sales into @title\_id, @title,  @ytd\_sales  end | CREATE PROCEDURE au\_sales (IN medianSalary varchar(8))  RESULT SETS 1  LANGUAGE SQL  BEGIN  declare author\_sales\_csr cursor for  select ta.title\_id, t.title, t.total\_sales  from titleauthor ta, titles t  where ta.title\_id = t.title\_id  and ta.au\_id = @author\_id;  END  CREATE PROCEDURE AU\_SALES\_SUBPROC (IN author\_id varchar(8))  RESULT SETS 1  LANGUAGE SQL  BEGIN  DECLARE title\_id varchar(11);  DECLARE title varchar(80);  DECLARE ytd\_sales int;  DECLARE @msg varchar(120);  OPEN author\_sales\_csr;  WHILE v\_counter < (v\_numRecords / 2 + 1)  DO  FETCH author\_sales INTO medianSalary;  SET v\_counter = v\_counter + 1;  END |

* + 1. **Cursor stage usage**

| **SYBASE** | | | | **DB2** |
| --- | --- | --- | --- | --- |
| | Values | Meanings | | --- | --- | | 0 | Indicates successful completion of the fetch statement. | | 1 | Indicates that the fetch statement resulted in an error. | | 2 | Indicates that there is no more data in the result set. This warning can occur if the current cursor position is on the last row in the result set and the client submits a fetch statement for that cursor. | | | To translate @@SQLSTATUS to SQL PL, the value of SQLCODE can be checked after each FETCH operation. If SQLCODE has a value that is less than 0, it means that an error has occurred and the statement has not been processed. If SQLCODE has a value greater than 0, it means a warning has been issued, but the statement was still processed.  Below is an example that shows how to gather the status of a FETCH statement in DB2 UDB.    DECLARE v\_sqlstatus INTEGER DEFAULT 0;  DECLARE c1 CURSOR FOR -- [1]  SELECT author\_id, name FROM authors;  DECLARE CONTINUE HANDLER FOR NOT FOUND -- [2]  SET v\_sqlstatus = -1;  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION -- [3]  SET v\_sqlstatus = -1;  OPEN c1;  WHILE 0 = v\_sqlstatus DO -- [4]  FETCH c1 INTO ...  ...  END WHILE  CLOSE c1;    It should be noted that DB2 UDB throws an exception when a warning or an error is encountered during processing. In the variable declaration part of the code [1], a CONTINUE HANDLER is defined for the case that no data is found [2] and another CONTINUE HANDLER is defined [3] for any other exception that arises. In both CONTINUE HANDLERs, a flag variable is set, which is checked in the WHILE loop condition [4]. If the flag is set by one of the CONTINUE HANDLERs, cursor processing (the WHILE loop processing) ceases. | | |

## **Views**

### **Creating View**

DESCRIPTION:

A view is an alternative way of looking at the data in one or more tables.

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  CREATE VIEW test\_view\_authors  AS SELECT \* FROM authors  where state = "UT" | CREATE VIEW test\_view\_authors AS SELECT \* FROM authors  WHERE state = 'UT' |
| **Result:**  View Created Successfully. | View Created Successfully. |

**Conclusion:** CREATE VIEW is used to create views in both SYBASE and DB2.

### **Modifying Underlying Objects**

DESCRIPTION:

The name of a view’s underlying objects can be changed.

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  CREATE VIEW test\_view  AS SELECT au\_id, au\_fname, city  FROM test\_authors  where state = "UT"  sp\_rename test\_authors, test\_authors1 | CREATE VIEW test\_view\_authors AS SELECT au\_id, au\_fname, city  FROM test\_authors  WHERE state = 'UT'  RENAME TABLE test\_authors to test\_authors1 |
| **Result:**  Test\_authors renamed to test\_authors1 | Renaming test\_authors not allowed |

**Conclusion:** In Sybase the table referenced by a view can be renamed whereas it is not allowed in DB2.

### **Modifying Data thru Views**

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  CREATE VIEW test\_view  AS SELECT city  FROM test\_authors  where state = "UT"  update test\_view  set city=”Bangalore” | CREATE VIEW test\_view  AS SELECT city  FROM test\_authors  WHERE state = 'UT'  Update test  set city=’Bangalore’ |
| **Result:**  Updated successfully. | Updated Successfully. |

**Conclusion:** Data can be updated using views in both SYBASE and DB2.

### **Updating Multitable Views**

DESCRIPTION:

We can insert or update a multitable view in SYBASE if:

• The view has no with check option clause.

• All columns being inserted or updated belong to the same base table.

Example:

| **SYBASE** | **DB2** |
| --- | --- |
| **Query:**  create view test\_multitable\_view  as select title, type, titles.pub\_id, state  from titles, publishers  where titles.pub\_id = publishers.pub\_id  update test\_multitable\_view  set type = "user\_friendly"  where type = "popular\_comp" | create view test\_multitable\_view  as select title, type, titles.pub\_id, state  from titles, publishers  where titles.pub\_id = publishers.pub\_id  update test\_multitable\_view  set type = ‘user\_friendly’  where type = ‘popular\_comp’ |
| **Result:**  Updated successfully. | Update not permitted. |

**Conclusion:** Multitable can be updated or inserted in SYBASE but not in DB2.

**NOTE:** The update or insert statement in a multitable view fails if it affects columns from both the

Tables

## **Rules**

### **Creating**

Supported in Sybase only

A rule lets you specify what users can or cannot enter into a particular

column or any column with a user-defined datatype.

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create rule [owner.]rule\_name as condition\_expression | It has check constraint. |
| EXAMPLE:  create rule pub\_idrule  as @pub\_id in ("1389", "0736", "0877", "1622", "1756")  or @pub\_id like "99[0-9][0-9]"  insert into publishers values("2444", "Ashish", "Alaska", "USA") | EXAMPLE:  CREATE TABLE EMP ( AGE BIGINT NOT NULL , EMP\_NO INTEGER NOT NULL , EMP\_NAME CHARACTER (20) NOT NULL , CONSTRAINT CC1129698164052 CHECK (in (age(21,22,23,24,25)))  insert into emp values(1,1,'ash') |
| RESULT:  Msg 552, Level 16, State 1:  Server 'PSNSYBD1', Line 1:  A column insert or update conflicts with a rule bound to the column. The command is aborted. The conflict occured in database 'pubs2', table 'publishers', rule 'pub\_idrule', column 'pub\_id'.  Command has been aborted.  (0 rows affected) | RESULT:  DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:  SQL0545N The requested operation is not allowed because a row does not satisfy the check constraint "ASHISHN.EMP.CC1129698164052". SQLSTATE=23513 |

CONCLUSION: Both are working in the same fashion.

NOTE:The as clause contains the name of the rule’s argument, prefixed with “@”, and the definition of the rule itself. You can use anyname for the argument, but the first character must be “@.”

### **Binding to Columns & User Defined Datatypes**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  sp\_bindrule rulename, objname[, futureonly] | Option not available. |
| DESCRIPTION:  rulename --is the name of the rule created with create rule.  Objname --is the name of the table and column, or of the user-defined datatype to  which the rule is to be bound if the parameter is not of the from table.column, it is assumed to be a user datatype.  Futureonly-- when binding a rule to a userdefined datatype. | DESCRIPTION:  Not available. |

Note: The check constraint in Db2 doesn’t required to get bind on the column, once created it works for that column whenever we insert a value not mentioned in the constraint.

### **Rules & Nulls**

You cannot define a column to allow nulls, and then override this definition with a rule that prohibits null values.

NOTE: For DB2, we can specify in the constraint whether it is nullable or not.

## **Defaults**

### **Creating**

| **SYBASE** | **DB2** |
| --- | --- |
| CREATE SYNTAX:  create default [*owner.]default\_name as constant\_expression* | CREATE SYNTAX:  create table table\_name (column\_name datatype with default ‘default\_value’) |

### **Binding and Unbinding**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  sp\_bindefault advancedflt | SYNTAX:  ALTER TABLE table\_name ALTER COLUMN col\_name SET DEFAULT default\_value  Before this command is executed, DB2 requires the column name and the default value which is specified in default\_name. |

**Note:** No Equivalent found for binding defaults in db2. Workaround is possible as shown above.

## **Control Flow**

### **If else case**

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  select title, "Contract Status" =  case  when contract = 1 then "Contract"  when contract = 0 then "No Contract"  end  from titles | Query:  SELECT title,  CASE   WHEN contract = 1 THEN 'Contract'  WHEN contract = 0 THEN 'No Contract'  END AS Contract\_Status  FROM titles |
| RESULT:  title  Contract Status  ----- ---------------  The Busy Executive's Database Guide  Contract  Cooking with Computers: Surreptitious Balance Sheets  Contract  You Can Combat Computer Stress!  Contract  Straight Talk About Computers  Contract  Silicon Valley Gastronomic Treats  Contract  The Gourmet Microwave  Contract    The Psychology of Computer Cooking  No Contract  But Is It User Friendly?  Contract  Secrets of Silicon Valley  Contract  Net Etiquette  No Contract  Computer Phobic and Non-Phobic Individuals: Behavior  Variations Contract  Is Anger the Enemy?  Contract  Life Without Fear  Contract  Prolonged Data Deprivation: Four Case Studies  Contract  Emotional Security: A New Algorithm  Contract  Onions, Leeks, and Garlic: Cooking Secrets of the  Mediterranean Contract  Fifty Years in Buckingham Palace Kitchens  Contract  Sushi,Anyone?  Contract | RESULT:  TITLE  CONTRACT\_STATUS  --------------------------------------------------------------------------------  ---------------  The Busy Executive's Database Guide  Contract  Cooking with Computers: Surreptitious Balance Sheets  Contract  You Can Combat Computer Stress!  Contract  Straight Talk About Computers  Contract  Silicon Valley Gastronomic Treats  Contract  The Gourmet Microwave  Contract  The Psychology of Computer Cooking  No Contract  But Is It User Friendly?  Contract  Secrets of Silicon Valley  Contract  Net Etiquette  No Contract  Computer Phobic and Non-Phobic Individuals: Behavior Variations  Contract  Is Anger the Enemy?  Contract  Life Without Fear  Contract  Prolonged Data Deprivation: Four Case Studies  Contract  Emotional Security: A New Algorithm  Contract  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean  Contract  Fifty Years in Buckingham Palace Kitchens  Contract  Sushi, Anyone?  Contract  18 record(s) selected. |

CONCLUSION: Results are same**.**

NOTE: In Sybase the name of the case is mentioned in beginning while in Db2 it is in the end after placing the condition.

**Case and division by zero:**

Case expression allows you to write queries that avoid division by zero (exception avoidance).

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, total\_sales, avance,total\_sales/advance from titles | QUERY:  Select title\_id, total\_sales, advance, total\_sales/advance from titles |
| RESULT:  title\_id total\_sales advance  -------- ----------- ------------------------ ------------------------  BU1032 4095 5,000.00 0.82  BU1111 3876 5,000.00 0.78  BU2075 18722 10,125.00 1.85  BU7832 4095 5,000.00 0.82  Divide by zero occurred.  (4 rows affected) | RESULT:  TITLE\_ID TOTAL\_SALES ADVANCE 4  ---------- ----------- --------------------- ---------------------------------  BU1032 4095 5000.0000 0.8190000000000000  BU1111 3876 5000.0000 0.7752000000000000  BU2075 18722 10125.0000 1.8490864197530864  BU7832 4095 5000.0000 0.8190000000000000  SQL0801N Division by zero was attempted. SQLSTATE=22012 |

CONCLUSION: Results are same**.**

**NOTE:** You can use a case expression to avoid this by not allowing the zero to figure in the equation when the query comes across the zero, it returns a predefined value, rather than performing the division:

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  select title\_id, total\_sales, advance, "Cost Per Book" =case when advance != 0 then convert(char, total\_sales/advance) else "No Books Sold" end from titles | Query:  SELECT title\_id, total\_sales, advance, CASE WHEN advance <>0 THEN CAST (SYB.FLOAT\_TO\_CHAR(total\_sales / advance) AS CHAR(30)) ELSE 'No Books Sold' END AS Cost\_Per\_Book FROM titles |
| RESULT:  title\_id total\_sales advance Cost Per Book  -------- ----------- ------------------------ -------------  BU1032 4095 5,000.00 0.82  BU1111 3876 5,000.00 0.78  BU2075 18722 10,125.00 1.85  BU7832 4095 5,000.00 0.82  MC2222 2032 0.00 No Books Sold  MC3021 22246 15,000.00 1.48  MC3026 NULL NULL No Books Sold  PC1035 8780 7,000.00 1.25  PC8888 4095 8,000.00 0.51  PC9999 NULL NULL No Books Sold  PS1372 375 7,000.00 0.05  PS2091 2045 2,275.00 0.90  PS2106 111 6,000.00 0.02  PS3333 4072 2,000.00 2.04  PS7777 3336 4,000.00 0.83  TC3218 375 7,000.00 0.05  TC4203 15096 4,000.00 3.77  TC7777 4095 8,000.00 0.51  (18 rows affected) | RESULT:  TITLE\_ID TOTAL\_SALES ADVANCE COST\_PER\_BOOK  ---------- ----------- --------------------- ------------------------------  BU1032 4095 5000.0000 .81899999999999995  BU1111 3876 5000.0000 .7752  BU2075 18722 10125.0000 1.8490864197530863  BU7832 4095 5000.0000 .81899999999999995  MC2222 2032 0.0000 No Books Sold  MC3021 22246 15000.0000 1.4830666666666665  MC3026 - - No Books Sold  PC1035 8780 7000.0000 1.2542857142857142  PC8888 4095 8000.0000 .51187499999999997  PC9999 - - No Books Sold  PS1372 375 7000.0000 .053571428571428499  PS2091 2045 2275.0000 .89890109890109893  PS2106 111 6000.0000 .018499999999999999  PS3333 4072 2000.0000 2.036  PS7777 3336 4000.0000 .83399999999999996  TC3218 375 7000.0000 .053571428571428499  TC4203 15096 4000.0000 3.774  TC7777 4095 8000.0000 .51187499999999997  18 record(s) selected. |

CONCLUSION: Results are same**.**

NOTE: The syb.float java function has come in scene when translating it.

**Case and value comparisons:**

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  select title, pub\_id, "Publisher" =case pub\_id when "0736" then "New Age Books" when "0877" then "Binnet & Hardley" when "1389" then "Algodata Infosystems" else "Other Publisher" end from title\_test order by pub\_id | Query:  SELECT title, pub\_id, CASE pub\_id WHEN '0736' THEN 'New Age Books' WHEN '0877' THEN 'Binnet & Hardley' WHEN '1389' THEN 'Algodata Infosystems' ELSE 'Other Publisher' END AS Publisher FROM titles ORDER BY pub\_id |
| RESULT:  title pub\_id Publisher  ----- ------ ---------  Life Without Fear  0736 New Age Books  Is Anger the Enemy?  0736 New Age Books  You Can Combat Computer Stress!  0736 New Age Books  Emotional Security: A New Algorithm  0736 New Age Books  Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  Sushi, Anyone? 0877  Binnet & Hardley  The Gourmet Microwave  0877 Binnet & Hardley  Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  The Psychology of Computer Cooking  0877 Binnet & Hardley  Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  Computer Phobic and Non-Phobic Individuals: Behavior  Variations 0877 Binnet & Hardley  Onions, Leeks, and Garlic: Cooking Secrets of the  Mediterranean  0877 Binnet & Hardley  Net Etiquette 1389  Algodata Infosystems  But Is It User Friendly?  1389 Algodata Infosystems  Secrets of Silicon Valley  1389 Algodata Infosystems  Straight Talk About Computers  1389 Algodata Infosystems  The Busy Executive's Database Guide  1389 Algodata Infosystems  Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  (18 rows affected) | RESULT:  TITLE  PUB\_ID PUBLISHER  -----------------------------------------------------------------  ------ ---------------------  You Can Combat Computer Stress!  0736 New Age Books  Is Anger the Enemy?  0736 New Age Books  Life Without Fear  0736 New Age Books  Prolonged Data Deprivation: Four Case Studies  0736 New Age Books  Emotional Security: A New Algorithm  0736 New Age Books  Silicon Valley Gastronomic Treats  0877 Binnet & Hardley  The Gourmet Microwave  0877 Binnet & Hardley  The Psychology of Computer Cooking  0877 Binnet & Hardley  Computer Phobic and Non-Phobic Individuals: Behavior Variations  0877 Binnet & Hardley  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean  0877 Binnet & Hardley  Fifty Years in Buckingham Palace Kitchens  0877 Binnet & Hardley  Sushi, Anyone?  0877 Binnet & Hardley  The Busy Executive's Database Guide  1389 Algodata Infosystems  Cooking with Computers: Surreptitious Balance Sheets  1389 Algodata Infosystems  Straight Talk About Computers  1389 Algodata Infosystems  But Is It User Friendly?  1389 Algodata Infosystems  Secrets of Silicon Valley  1389 Algodata Infosystems  Net Etiquette  1389 Algodata Infosystems  18 record(s) selected. |

CONCLUSION: Results are same**.**

**Case expression requires at least one non-null result:**

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  select price,  case  when title\_id like "%" then NULL  when pub\_id like "%" then NULL  end from titles | Query:  SELECT price,  CASE  WHEN title\_id LIKE '%' THEN CAST (NULL AS VARCHAR(1)) WHEN pub\_id LIKE '%' THEN CAST (NULL AS VARCHAR(1)) END FROM titles |
| RESULT:  All result expressions in a CASE expression must not be NULL. | RESULT:  PRICE 2  --------------------- -  159.9200 -  95.6000 -  23.9200 -  159.9200 -  159.9200 -  23.9200 -  - -  183.6000 -  160.0000 -  - -  172.7200 -  87.6000 -  56.0000 -  159.9200 -  63.9200 -  167.6000 -  95.6000 -  119.9200 -  18 record(s) selected. |

CONCLUSION: Results differ.

**NOTE**: The converted query in DB2 is treating NULL as varchar(1).

### **Begin end**

| **SYBASE** | **DB2** |
| --- | --- |
| Query:  if (select avg(price) from titles) < $15  begin  update titles  set price = price \* 2  select title, price from titles where price > $28  end | Query:  BEGIN ATOMIC DECLARE temp\_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR  SELECT title, price FROM titles WHERE price > 28.000 UPDATE titles SET price = price \* 2 OPEN temp\_cursor; END |
| RESULT:  title price  ----- ------------------------  Secrets of Silicon Valley 40.00  The Busy Executive's Database Guide 39.98  Prolonged Data Deprivation: Four Case Studies 39.98  Silicon Valley Gastronomic Treats 39.98  Sushi, Anyone? 29.98  But Is It User Friendly? 45.90  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean 41.90  Straight Talk About Computers 39.98  Computer Phobic and Non-Phobic Individuals: Behavior Variations 43.18  (9 rows affected) | RESULT:  Top-level IF statements are not translated.  SQL0104N An unexpected token "WITH" was found following "HOLD". Expected tokens may include: "FOR". SQLSTATE=42601  DB21028E The cursor "TEMP\_CURSOR" has not been declared. |

CONCLUSION:. Whatever we give in the Begin –end Synatx is taken as a part of the procedure by DB2. It can be done manually by using all these statements within the procedure or by editing the syntax as per DB2 criteria.

### **While break continue**

| **SYBASE** | **DB2** |
| --- | --- |
| while (select avg(price) from titles) < $30  begin  select title\_id, price  from titles  where price > $20  update titles  set price = price \* 2  end | CREATE PROCEDURE MOHM.WHILE\_TEST (  )     DYNAMIC RESULT SETS 1 ------------------------------------------------------------------------ -- SQL Stored Procedure  ------------------------------------------------------------------------ POC: BEGIN        DECLARE V\_AVG\_PRICE DECIMAL(14,2);     DECLARE V\_TITLE\_ID INTEGER;     DECLARE V\_PRICE DECIMAL(14,2);          -- Declare cursor     DECLARE cursor1 CURSOR WITH HOLD FOR         SELECT title\_id, price from titles where price > 20 FOR UPDATE;         SELECT AVG(price) INTO V\_AVG\_PRICE FROM TITLES;        OPEN cursor1;                  WHILE (V\_AVG\_PRICE < 30) DO                               FETCH cursor1 INTO V\_TITLE\_ID,V\_PRICE;                                    UPDATE TITLES SET PRICE = PRICE \* 2                         WHERE CURRENT OF cursor1;                                     SELECT AVG(price) INTO V\_AVG\_PRICE FROM TITLES;                               END WHILE;                    CLOSE cursor1;    END POC |
| Result:  (0 rows affected)  title\_id price  ------ -------  PC1035 22.95  PS1372 21.59  TC3218 20.95  (3 rows affected)  (18 rows affected)  (0 rows affected)  title\_id price  ------ -------  BU1032 39.98  BU1111 23.90  BU7832 39.98  MC2222 39.98  PC1035 45.90  PC8888 40.00  PS1372 43.18  PS2091 21.90  PS3333 39.98  TC3218 41.90  TC4203 23.90  TC7777 29.98  (12 rows affected)  (18 rows affected)  (0 rows affected) | Result:  Run started.  Data returned in result sets is limited to the first 100 rows.  Data returned in result set columns is limited to the first 20 bytes or characters.  Calling the stored procedure.  Returned 0  Run completed. |

CONCLUSION:The while loop can be executed in DB2 by creating a stored procedure for it.

### **Declare goto return**

**DECLARE:**

| **SYBASE** | **DB2** |
| --- | --- |
| create proc ash.declare @a\_dtStart DATETIME, @a\_dtEnd DATETIME, @a\_iYears INTEGER OUTPUT, @a\_iDays INTEGER OUTPUT AS  DECLARE @dtWork1 DATETIME  DECLARE @dtWork2 DATETIME  DECLARE @iDays INTEGER | CREATE PROCEDURE ap\_qds\_datediff\_calc\_r (v\_a\_dtStart TIMESTAMP,   v\_a\_dtEnd TIMESTAMP, OUT v\_a\_iYears INTEGER, OUTv\_a\_iDaysINTEGER )LANGUAGE SQL BEGIN DECLARE SQLSTATE CHAR(5) DEFAULT '00000'; DECLARE l\_error CHAR(5) DEFAULT '00000'; DECLARE v\_dtWork1 TIMESTAMP;  DECLARE v\_dtWork2 TIMESTAMP;  DECLARE v\_iDays INTEGER; |

CONCLUSION: The parameters are declared using v\_parametername in DB2. For input parameters it uses IN and for Output parameters it takes OUT in the syntax for every parameter.

**GOTO:**

| **SYBASE** | **DB2** |
| --- | --- |
| IF @@error != 0 BEGIN SELECT @sErrMsg = "Could not split range. Please try again" GOTO errormsg END | IF l\_error <> '00000' THEN  SET v\_sErrMsg = 'Could not split range. Please try again'; GOTO errormsg; END IF; |

CONCLUSION: : The syntax is same in both the cases. The above syntax works only when it is part of the procedure else it won’t get translated

**RETURN:**

| **SYBASE** | **DB2** |
| --- | --- |
| if (select contract from titles where title\_id = @titleid) = 1return 1else return 2 | IF (SELECT contract FROM titles WHERE v\_titleid IS NULL AND title\_id IS NULL OR title\_id = v\_titleid) = 1 THEN RETURN 1; ELSE RETURN 2; END IF; |

CONCLUSION**:** The return syntax is the same in both the case.

NOTE: The above syntax works only when it is part of the procedure else it won’t get translated.

### **Waitfor print raise error**

**WAIT EXPRESSION:**

The waitfor command delays execution of a stored procedure at a specified time or until a specified amount of time has passed.

| **SYBASE** | **DB2** |
| --- | --- |
| waitfor time "16:23" | CALL SYB.waitfor('time','16:23') |

CONCLUSION: The SYB.waitfor java function is used by MTK.

**PRINT:**

| **SYBASE** | **DB2** |
| --- | --- |
| declare @count smallint  select @count = 1 restart: print "yes" select @count = @count + 1 while @count <=4 goto restart | The equivalent for the print statement is put\_line. |

CONCLUSION:Sybase code is not getting translated through the MTK

NOTE: IN DB2;

Syntax: VALUES (PUT\_LINE('debug message')) INTO dummy\_variable;

To write a number, the PUT\_LINE function is used as follows:

Syntax: VALUES (PUT\_LINE(dubug number) INTO dummy\_variable

To write a number and a message, use PUT\_LINE as follows:

Syntax: VALUES (PUT\_LINE(dubug number),'dubug message')) INTO dummy\_variable;

**RAISE ERROR:**

| **SYBASE** | **DB2** |
| --- | --- |
| IF @id = -1 BEGIN RAISEERROR 40001, "Error occured while calculating identity for table ipa\_vault\_leg"  ROLLBACK TRANSACTION  RETURN  END | IF v\_id = -1 THEN GOTO raiserror\_4000111; END IF; SET l\_error = '00000'; |

CONCLUSION: Instead of RAISEERROR it uses GOTO raiseerror in DB2.

### **Local Variables**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = 20  as  declare @c int  select @c=@a+@b  exec get\_val @d | SYNTAX:  create procedure add\_proc (in a integer,  in b integer,  out c integer)  poc: begin  declare v\_a int default 10;  declare v\_b int default 20;  set v\_a = a;  set v\_b = b;  set c = v\_a + v\_b;  call get\_val(in d int);  end poc @ |
| DESCRIPTION:  In the above example @c is the local variable used. | DESCRIPTION:  In the above example v\_a,v\_b are the local variables used. |

### **Global Variables**

**Comparison:**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Sybase has many global variables, which are system-defined variables that Sybase server updates on an ongoing basis.You can query global variables to monitor system activities or get connection information. For some global  variables, you can also set a value using the SET command to control query  processing of the current session. | DESCRIPTION:  To monitor system activities, you can use the Snapshot Monitor in DB2 UDB.DB2 UDB also has special registers, which are also updated on an ongoing basis by DB2 UDB. For some special registers, you can use the SET command to update the value. Some Sybase global variables can be mapped to the data element which you can obtain using the Snapshot Monitor in DB2, others can be mapped to DB2 special registers or different features of DB2 UDB depending on the purpose  of the global variable. In this section, we show you how you can implement the functionality of the following global variables in DB2 UDB: |

**Commonly used Global Variables:**

@@connections

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@connections | SYNTAX:  Not Supported directly  WORKAROUND:  $ db2 get snapshot for database on dbname | grep ‘Application connects’  Application connects = 24 |
| DESCRIPTION:  The @@connections global variable returns the number of logins or attempted logins to the Sybase server since it was started. | DESCRIPTION:  In DB2, you can execute the Snapshot Monitor and get the ‘Connects Since Database Activation’ element, which is indicated as ‘Application connects’ in the output, as the following example: |

@@error/@@sqlstatus

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@error  SELECT @@sqlstatus | SYNTAX:  Not Supported directly  WORKAROUND:  CREATE PROCEDURE MOHM.GLB\_TEMP\_TABLE\_DEBUGGING (IN P\_NAME VARCHAR(50) DEFAULT NULL,IN P\_EMP\_ID INT,OUT P\_ERR\_MSG VARCHAR(500),OUT E\_RET\_VAL CHAR(5))  ------------------------------------------------------------------------  -- SQL Stored Procedure  -- CREATED BY : XYZ  -- CREATION DATE : 26/09/2005  -- PURPOSE : TO MAKE USE OF SQLSTATE FOR DEBUGGING  -- TABLES USED : MOHM.TEST  -- INPUT PARAMETERS : NAME OF EMPLOYEE,EMPLOYEE\_ID  -- OUTPUT PARAMETERS : ERROR\_MESSAGE,SQLSTATE  ------------------------------------------------------------------------  DBCOE : BEGIN  -- DECLARATION OF LOCAL VARIABLES --  DECLARE V\_STMT\_NO INT;  DECLARE SQLSTATE CHAR(5) DEFAULT '00000';  DECLARE V\_RET\_VAL CHAR(5) DEFAULT '00000';  SET E\_RET\_VAL = V\_RET\_VAL;  SET P\_ERR\_MSG = 'EXECUTION SUCCESSFULL';  BEGIN  DECLARE EXIT HANDLER FOR SQLSTATE '23505'    BEGIN  DECLARE EXIT HANDLER FOR SQLSTATE 'T1000'  BEGIN  GET DIAGNOSTICS EXCEPTION 1 P\_ERR\_MSG = MESSAGE\_TEXT ;  SET E\_RET\_VAL = SQLSTATE;    BEGIN  DECLARE GLOBAL TEMPORARY TABLE MOHM\_GLBL\_TABLE1(TEMP\_COL1 VARCHAR(500)) WITH replace on commit preserve rows not logged IN GLBL\_TBSPC;  SET P\_ERR\_MSG = 'PRIMARY KEY VIOLATION TRYING TO INSERT DUPLICATE VALUES';  BEGIN  DECLARE TEMP\_CUR CURSOR WITH RETURN FOR SELECT \* FROM SESSION.MOHM\_GLBL\_TABLE1;  INSERT INTO SESSION.MOHM\_GLBL\_TABLE1 VALUES(P\_ERR\_MSG);  OPEN TEMP\_CUR;  END;  END;  END;    IF V\_STMT\_NO = 100 THEN  SIGNAL SQLSTATE 'T1000'  SET MESSAGE\_TEXT = 'PRIMARY KEY VIOLATION TRYING TO INSERT DUPLICATE VALUES';  END IF;    END; -- END OF BEGIN DECLARE EXIT HANDLER FOR SQLSTATE 'T1000'  -- INSERTION LOGIC --  BEGIN  SET V\_STMT\_NO = 100;  INSERT INTO MOHM.TEST VALUES(P\_NAME,P\_EMP\_ID);  END;  END; -- END OF BEGIN DECLARE EXIT HANDLER FOR SQLSTATE '23505'  END DBCOE |
| DESCRIPTION:  In Sybase, you can query the @@error or @@sqlstatus global variables to check  whether the previous statement succeeded or not. The @@error has the error  status of the most recently executed statement. The @@sqlstatus has the  status from the last FETCH statement. Its value is 0 (success), 1 (error), or 2  (no more data). | DESCRIPTION:  In DB2 UDB, return codes from SQL statements are handled with SQLCODE or SQLSTATE. You can use either of them in applications. The SQLCODE conforms to ISO/ANSI SQL standard, and the SQLSTATE is based on the ISO/ANSI SQL92  standard. |

@@rowcount

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@rowcount | SYNTAX:  Not Supported directly  WORKAROUND:  CREATE PROCEDURE sqlprocg ( IN deptnbr VARCHAR(3),OUT rcount INTEGER)  LANGUAGE SQL  BEGIN  DECLARE SQLSTATE CHAR(5)  UPDATE CORPDATA.PROJECT  SET PRSTAFF = PRSTAFF + 1.5  WHERE DEPTNO = deptnbr;  GET DIAGNOSTICS rcount = ROW\_COUNT;  END  This stored procedure issues the UPDATE statement and the GET DIAGNOSTICS statement, and returns the affected row count. |
| DESCRIPTION:  In Sybase, the @@rowcount variable contains the number of rows affected by  the last query. The value represents the number of rows of a cursor result set  returned to the client, up to the last fetch request. | DESCRIPTION:  In DB2 UDB, you can use the GET DIAGNOSTICS statement with the ROW\_COUNT option in stored procedures. If the previous statement is the PREPARE statement, ROW\_COUNT identifies the estimated number of result rows in the prepared statement. If the previous statement is the DELETE, INSERT, or UPDATE statement, ROW\_COUNT identifies the number of rows deleted, inserted, or updated by that statement, excluding rows affected by either triggers or  referential integrity constraints. |

## **Transactions**

DESCRIPTION

A transaction is a series of one or more data statements that either takes place as a unit or are aborted as a unit.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  begin tran  update accounts  set balance=balance-@amount  where acc\_id=@from  update accounts  set balance=balance+@amount  where au\_id=@to  commit tran | update accounts  set balance=balance-amount  where acc\_id=from  update accounts  set balance=balance+amount  where au\_id=to |

CONCLUSION: In Sybase transactions are implemented using the begin tran and commit tran

whereas DB2 implements transactions implicitly.

### **Isolation Levels**

DESCRIPTION

These are the levels in which data are locked so that transactions do not interfere with one another.

EXAMPLE:

| **SYBASE** | **DB2** | **Description** |
| --- | --- | --- |
| isolation level 0 | UR | Uncommitted Reads |
| isolation level 1 | CS | Cursor Stability, prevents uncommitted or ‘dirty’ reads |
| isolation level 2 | RS | Read Stability prevents rows from a transaction from being  read by another transaction until committed |
| isolation level 3 | RR | Read prevents phantom reads in addition to providing Read Stability |

CONCLUSION: In Sybase transactions levels are set using “set transaction” with one of the isolation levels whereas in DB2, the same is implemented by using “CHANGE ISOLATION”.

### **Modes**

DESCRIPTION

A transaction mode specifies how the server should define transactions.

CONCLUSION: Sybase has two transaction modes viz., chained and unchained whereas DB2 works in only one mode which does not require the explicit declaration of begin and commit statements.

### **Begin Commit**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  begin tran  update accounts  set balance=balance-@amount  where acc\_id=@from  update accounts  set balance=balance+@amount  where au\_id=@to  commit tran | Done implicitly. |

CONCLUSION: In Sybase, begin and commit statements are used explicitly in unchained mode whereas DB2 performs the begin and commit operation implicitly.

### **Transaction States**

DESCRIPTION

Transaction state gives the current state of a transaction.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  Select @@transtate | Doesn’t exist. |
| RESULT:  -----------  1 |  |

CONCLUSION: @@transtate stores the current state of transaction in SYBASE and whereas it doesn’t exist in DB2.

### **Deadlocks**

DESCRIPTION

Deadlocks occur when, in the process of serializing multi-threaded operations ‘impossible situations’ develop where different threads wait for one another to release resources. The result is clients are left hanging in a deadlocked state.

| **SYBASE** | **DB2** |
| --- | --- |
| With proper application coding. | With proper application coding. |
| Converting table schema from all  pages-->data pages-->data rows. | Converting table schema from all  pages-->data pages-->data rows. |
| Using isolation levels | Using isolation levels |
| Resubmitting transactions once deadlock occurs | Resubmitting transactions once deadlock occurs |

CONCLUSION: Both handle deadlock avoidance in the same way.

* + 1. **Nested Transactions**

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  begin tran  select @@trancount  /\* @@trancount = 1 \*/  begin tran  select @@trancount  /\* @@trancount = 2 \*/  begin tran  select @@trancount  /\* @@trancount = 3 \*/  commit tran  commit tran  commit tran | There is no direct equivalent to the @@TRANCOUNT function in SQL PL, since transactions are handled differently in DB2 UDB. Converting the @@TRANCOUNT function to DB2 UDB essentially amounts to ensuring that transactional statements including COMMIT and ROLLBACK are included in the appropriate places in the SQL PL code. For instance, in T-SQL, database updates that are  performed when @@TRANCOUNT is zero are not part of any transaction, hence a COMMIT statement should be issued in SQL PL immediately after such updates. |

* + 1. **Transaction Modes**

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  Set chained on  Set chained off | db2 –c  db2 +c |

## **Temp Tables**

**Types of Temp Tables:**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Sybase has two types of user defined temporary tables. One type is local temporary tables and the other type is global temporary tables. Local  temporary tables (created in tempdb) are visible only from the current user who has created the temporary tables, and persist within the current application’s connection. If local temporary tables are created in a stored procedure, they persist within a stored procedure’s execution. The global temporary table has persistency between connections, and stays in tempdb until restarting of Sybase server.Temp Tables reside in the tempdb database.They are used for sorting, performing some intermediate operations which do not require logging. | DESCRIPTION:  In DB2, you can create a temporary table using the DECLARE GLOBAL TEMPORARY TABLE statement when you need it as an intermediate table. This type of temporary tables are called declared temporary tables. A declared temporary table persists within the current application’s connection and is dropped automatically at the termination of the connection. |

**Temporary space comparison of Sybase and DB2 UDB:**

|  | **SYBASE** | **DB2** |
| --- | --- | --- |
| Temporary space for  DBMS is implemented  in | tempdb database | System temporary table  spaces |
| User defined temporary  tables are created in | tempdb database | User temporary table spaces |
| User defined temporary  tables | Local temporary tables | Declared temporary tables |
| Global temporary tables | N/A |

**Creating Temp Tables :**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  Creating a local temporary table  create #temptab(col01 char(5), col02 char(10))  Creating a global temporary table  select \* into table tempdb..temptab from tab1 where col01>’C0000’; | SYNTAX:  declare global temporary table temptab(col01 char(5),col02 char(10)) not logged;  declare global temporary table temptab  as ( select \* from tab1) definition only not logged  declare global temporary table temptab like tab1 not logged |
| DESCRIPTION:  Above two examples are syntaxes to create local temporary table and global temporary table. | DESCRIPTION:  Here the temp table is created in the default temporary table space, unless you specify a table space name with IN clause.  The NOT LOGGED option is mandatory for the DECLARE GLOBAL TERMPOARY TABLE statement. |

**Using Temp Tables:**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure inv\_proc as  create table #tempstores  (stor\_id char(4), amount money) | SYNTAX:  create procedure inv\_proc()  poc begin  declare global temporary table  session."#tempstores"  (stor\_id char(4) not null,  amount decimal(19,4) not null  ) with replace on commit preserve  rows not logged;  end poc @ |

**Temporary tables comparison of Sybase and DB2:**

|  | **SYBASE** | **DB2** |
| --- | --- | --- |
| Can set referencial constraints? | No | No |
| Can define rules (Sybase) or check  constraints (DB2 UDB)? | Yes | No |
| Can define the default value for a column? | Yes | Yes |
| Can create indexes on? | Yes | Yes |
| Can create views? | No | No |
| Can create triggers? | No | No |
| Can use user-defined type? | Yes | No |

## **System Procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  System procedures are shortcuts for retrieving information from the system tables and mechanisms for performing database administration and other tasks  that involve updating system tables. A System Administrator can allow direct updates of system tables by changing a configuration variable and issuing the reconfigure with override command. The names of system procedures begin with “sp\_”. They are created by the installmaster script in the sybsystemprocs database during Adaptive Server  installation.  EXAMPLE:  sp\_addgroup is used to add a group. | DESCRIPTION:  System procedures are defined with the schema SYSIBM.These perform database/system related activities.  EXAMPLE:  DB2LOAD is used whenever a LOAD command is executed. |
| LIST:  sp\_activeroles  sp\_add\_qpgroup  sp\_add\_resource\_limit  sp\_add\_time\_range  sp\_addalias  sp\_addengine  sp\_addexeclass  sp\_addextendedproc  sp\_addexternlogin  sp\_addgroup  sp\_addlanguage  sp\_addlogin  sp\_addmessage  sp\_addobjectdef  sp\_addremotelogin  sp\_addsegment  sp\_addserver  sp\_addtabledef  sp\_addthreshold  sp\_addtype  sp\_addumpdevice  sp\_adduser  sp\_altermessage  sp\_autoconnect  sp\_aux\_computeprivs  sp\_aux\_get\_qpgroup  sp\_aux\_getsize  sp\_aux\_parse\_entity  sp\_aux\_sargs\_qpgroup  sp\_aux\_text  sp\_bindcache  sp\_bindefault  sp\_bindexeclass  sp\_bindmsg  sp\_bindrule  sp\_cacheconfig  sp\_cachestrategy  sp\_changedbowner  sp\_changegroup  sp\_checknames  sp\_checkreswords  sp\_checksource  sp\_chgattribute  sp\_chklangparam  sp\_clearpsexe  sp\_clearstats  sp\_cmp\_all\_qplans  sp\_cmp\_qplans  sp\_column\_privileges  sp\_columns  sp\_commonkey  sp\_config\_rep\_agent  sp\_configure  sp\_copy\_all\_qplans  sp\_copy\_qplan  sp\_countmetadata  sp\_create\_syscoordinations  sp\_cursorinfo  sp\_databases  sp\_datatype\_info  sp\_dboption  sp\_dbrecovery\_order  sp\_dbremap  sp\_default\_charset  sp\_defaultdb  sp\_defaultlanguage  sp\_defaultloc  sp\_depends  sp\_depends\_cols  sp\_deviceattr  sp\_diskdefault  sp\_displaylevel  sp\_displaylogin  sp\_displayroles  sp\_drop\_all\_qplans  sp\_drop\_qpgroup  sp\_drop\_qplan  sp\_drop\_resource\_limit  sp\_drop\_time\_range  sp\_dropalias  sp\_dropdevice  sp\_dropengine  sp\_dropexeclass  sp\_dropextendedproc  sp\_dropexternlogin  sp\_dropglockpromote  sp\_dropgroup  sp\_dropkey  sp\_droplanguage  sp\_droplockpromote  sp\_droplogin  sp\_dropmessage  sp\_dropobjectdef  sp\_dropremotelogin  sp\_droprowlockpromote  sp\_dropsegment  sp\_dropserver  sp\_droptabledef  sp\_dropthreshold  sp\_droptype  sp\_dropuser  sp\_dtx\_purge\_completedxacts  sp\_dumpoptimize  sp\_embt\_est\_tab\_space  sp\_engine  sp\_estspace  sp\_export\_qpgroup  sp\_extendsegment  sp\_extengine  sp\_familylock  sp\_find\_qplan  sp\_fix\_dtm\_ha\_roles  sp\_fixindex  sp\_fkeys  sp\_flushstats  sp\_forceonline\_db  sp\_forceonline\_object  sp\_forceonline\_page  sp\_foreignkey  sp\_freedll  sp\_gen\_login\_id  sp\_gen\_timerange\_id  sp\_getmessage  sp\_ha\_check\_certified  sp\_help  sp\_help\_qpgroup  sp\_help\_qplan  sp\_help\_rep\_agent  sp\_help\_resource\_limit  sp\_helpartition  sp\_helpcache  sp\_helpconfig  sp\_helpconstraint  sp\_helpdb  sp\_helpdevice  sp\_helpextendedproc  sp\_helpexternlogin  sp\_helpgroup  sp\_helpindex  sp\_helpjava  sp\_helpjoins  sp\_helpkey  sp\_helplanguage  sp\_helplog  sp\_helpobjectdef  sp\_helpremotelogin  sp\_helprotect  sp\_helpsegment  sp\_helpserver  sp\_helpsort  sp\_helptext  sp\_helpthreshold  sp\_helpuser  sp\_hidetext  sp\_ijdbc\_aux\_computeprivs  sp\_ijdbc\_class\_for\_name  sp\_ijdbc\_classes\_in\_jar  sp\_ijdbc\_columns  sp\_ijdbc\_convert\_datatype  sp\_ijdbc\_datatype\_info  sp\_ijdbc\_default\_charset  sp\_ijdbc\_escapeliteralforlike  sp\_ijdbc\_exportkey  sp\_ijdbc\_fkeys  sp\_ijdbc\_function\_escapes  sp\_ijdbc\_getbestrowidentifier  sp\_ijdbc\_getcatalogs  sp\_ijdbc\_getcolumnprivileges  sp\_ijdbc\_getcrossreferences  sp\_ijdbc\_getindexinfo  sp\_ijdbc\_getisolationlevels  sp\_ijdbc\_getprocedurecolumns  sp\_ijdbc\_getschemas  sp\_ijdbc\_gettableprivileges  sp\_ijdbc\_getudts  sp\_ijdbc\_getversioncolumns  sp\_ijdbc\_getxacoordinator  sp\_ijdbc\_importkey  sp\_ijdbc\_jar\_by\_name  sp\_ijdbc\_jar\_for\_class  sp\_ijdbc\_mda  sp\_ijdbc\_primarykey  sp\_ijdbc\_sql\_type\_name  sp\_ijdbc\_stored\_procedures  sp\_ijdbc\_tables  sp\_import\_qpgroup  sp\_indsuspect  sp\_inform  sp\_iqdbcheck  sp\_jdbc\_class\_for\_name  sp\_jdbc\_classes\_in\_jar  sp\_jdbc\_columns  sp\_jdbc\_computeprivs  sp\_jdbc\_convert\_datatype  sp\_jdbc\_datatype\_info  sp\_jdbc\_escapeliteralforlike  sp\_jdbc\_exportkey  sp\_jdbc\_fkeys  sp\_jdbc\_function\_escapes  sp\_jdbc\_getbestrowidentifier  sp\_jdbc\_getcatalogs  sp\_jdbc\_getcolumnprivileges  sp\_jdbc\_getcrossreferences  sp\_jdbc\_getindexinfo  sp\_jdbc\_getisolationlevels  sp\_jdbc\_getprocedurecolumns  sp\_jdbc\_getschemas  sp\_jdbc\_gettableprivileges  sp\_jdbc\_getudts  sp\_jdbc\_getversioncolumns  sp\_jdbc\_getxacoordinator  sp\_jdbc\_importkey  sp\_jdbc\_jar\_by\_name  sp\_jdbc\_jar\_for\_class  sp\_jdbc\_primarykey  sp\_jdbc\_stored\_procedures  sp\_jdbc\_tables  sp\_listsuspect\_db  sp\_listsuspect\_object  sp\_listsuspect\_page  sp\_lock  sp\_locklogin  sp\_logdevice  sp\_logiosize  sp\_lookup  sp\_makesuspect\_obj  sp\_markreport  sp\_mda  sp\_modify\_resource\_limit  sp\_modify\_time\_range  sp\_modifylogin  sp\_modifystats  sp\_modifythreshold  sp\_monitor  sp\_monitorconfig  sp\_monitormetadata  sp\_namecrack  sp\_object\_stats  sp\_objectsegment  sp\_passthru  sp\_password  sp\_pkeys  sp\_placeobject  sp\_plan\_dbccdb  sp\_poolconfig  sp\_primarykey  sp\_procqmode  sp\_procxmode  sp\_recompile  sp\_remap  sp\_remoteoption  sp\_remotesql  sp\_rename  sp\_rename\_qpgroup  sp\_renamedb  sp\_reportstats  sp\_reptostandby  sp\_role  sp\_sendmsg  sp\_server\_info  sp\_serverinfo  sp\_serveroption  sp\_servertype  sp\_set\_qplan  sp\_setlangalias  sp\_setlockpromote  sp\_setpglockpromote  sp\_setpsexe  sp\_setrepcol  sp\_setrepdefmode  sp\_setreplicate  sp\_setrepproc  sp\_setreptable  sp\_setrowlockpromote  sp\_setsuspect\_error  sp\_setsuspect\_granularity  sp\_setsuspect\_threshold  sp\_shmdumpconfig  sp\_shmdumpdisp  sp\_shmdumpdrop  sp\_shmdumpsize  sp\_showcontrolinfo  sp\_showexeclass  sp\_showplan  sp\_showpsexe  sp\_spaceused  sp\_special\_columns  sp\_sproc\_columns  sp\_sql\_type\_name  sp\_ssladmin  sp\_start\_rep\_agent  sp\_statistics  sp\_stop\_rep\_agent  sp\_stored\_procedures  sp\_syntax  sp\_sysmon  sp\_sysmon\_analyze  sp\_sysmon\_appmgmt  sp\_sysmon\_collect  sp\_sysmon\_dcache  sp\_sysmon\_dcache\_dtl  sp\_sysmon\_dcache\_sum  sp\_sysmon\_diskio  sp\_sysmon\_esp  sp\_sysmon\_hk  sp\_sysmon\_index  sp\_sysmon\_kernel  sp\_sysmon\_locks  sp\_sysmon\_maccess  sp\_sysmon\_mdcache  sp\_sysmon\_memory  sp\_sysmon\_netio  sp\_sysmon\_parallel  sp\_sysmon\_pcache  sp\_sysmon\_recovery  sp\_sysmon\_taskmgmt  sp\_sysmon\_wpm  sp\_sysmon\_xactmgmt  sp\_sysmon\_xactsum  sp\_table\_privileges  sp\_tables  sp\_transactions  sp\_unbindcache  sp\_unbindcache\_all  sp\_unbindefault  sp\_unbindexeclass  sp\_unbindmsg  sp\_unbindrule  sp\_validaltlang  sp\_validateconfigname  sp\_validlang  sp\_volchanged  sp\_who | LIST:  ADMIN\_SP\_C  ALTOBJ  AM\_DROP\_TASK  AM\_GET\_LOCK\_CHN\_TB  AM\_GET\_LOCK\_CHNS  AM\_GET\_LOCK\_RPT  AM\_GET\_RPT  AM\_SAVE\_TASK  ANALYZE\_LOG\_SPACE  CAPTURE\_STORAGEMGMT\_INFO  CREATE\_STORAGEMGMT\_TABLES  DB2LK\_CLEAN\_TABLE  DB2LK\_TBLNDEP  DB2LOAD  DROP\_STORAGEMGMT\_TABLES  GENERATE\_DISTFILE  GET\_DB\_CONFIG  GET\_DB2\_SQLROUTINE\_PREOPTS  GET\_DBSIZE\_INFO  GET\_SWRD\_SETTINGS  GET\_WRAP\_CFG\_C  HEALTH\_HI\_REC  LIST\_COL\_LONG\_OPTS  LIST\_FILES\_C  LIST\_NN\_LONG\_OPTS  LIST\_SRVR\_NODES  LIST\_SRVR\_OPTIONS  LIST\_SRVR\_OPTVALS  LIST\_SRVR\_TYPES  LIST\_SRVR\_VERSIONS  LIST\_WRAP\_OPTIONS  LIST\_WRAPPERS  NNSTAT  ODBC\_DISCOVERY  POLICY\_INSTALL  POLICY\_INSTALLOC  POLICY\_RETRIEVE  PSMD\_CLEAR\_SESSION  PSMD\_GET\_LEVEL  PSMD\_GET\_REPORT  PSMD\_INIT\_SESSION  PSMD\_PUT\_COMMAND  PSMD\_SET\_COMPILEMODE  PSMD\_SET\_SESSION  PSMD\_TERM\_SESSION  REBIND\_ROUTINE\_PACKAGE  REMOTE\_QUERY  REORGCHK\_IX\_STATS  REORGCHK\_TB\_STATS  SET\_DB2\_SQLROUTINE\_PREOPTS  SET\_ROUTINE\_OPTS  SET\_SWRD\_SETTINGS  SNAPSHOT\_FILEW  SQLCAMESSAGE  SQLCOLPRIVILEGES  SQLCOLUMNS  SQLFOREIGNKEYS  SQLGETTYPEINFO  SQLPRIMARYKEYS  SQLPROCEDURECOLS  SQLPROCEDURES  SQLSPECIALCOLUMNS  SQLSTATISTICS  SQLTABLEPRIVILEGES  SQLTABLES  SQLUDTS  STEPWISE\_REDISTRIBUTE\_DBPG  SYSINSTALLOBJECTS  SYSINSTALLROUTINES |

## **System Tables**

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Most tables in the *master* database are system tables. Some of these tables also occur in user databases. They are automatically created when the **create database** command is issued  EXAMPLE:  select \* from sysindexes will give all the information regarding the indexes existing in that particular database. | DESCRIPTION:  A set of system catalog tables is created and maintained for each database.  These tables contain information about the definitions of the database objects  (for example, tables, views, indexes, and packages), and security information  about the type of access that users have to these objects. These tables are  stored in the SYSCATSPACE table space.  These tables are updated during the operation of a database; for example,  when a table is created. You cannot explicitly create or drop these tables, but  you can query and view their content. When the database is created, in  addition to the system catalog table objects, the following database objects are  defined in the system catalog:  EXAMPLE:  select \* from sysibm.sysindexes will give you the name and creator of the index. |
| LIST:  sysalternates  sysattributes  syscertificates  syscharsets  syscolumns  syscomments  sysconfigures  sysconstraints  syscurconfigs  sysdatabases  sysdepends  sysdevices  sysengines  sysgams  sysindexes  sysjars  syskeys  syslanguages  syslisteners  syslocks  sysloginroles  syslogins  syslogs  syslogshold  sysmessages  sysmonitors  sysobjects  syspartitions  sysprocedures  sysprocesses  sysprotects  sysqueryplans  sysreferences  sysremotelogins  sysresourcelimits  sysroles  syssecmechs  syssegments  sysservers  syssessions  syssrvroles  sysstatistics  systabstats  systestlog  systhresholds  systimeranges  systransactions  systypes  sysusages  sysusermessages  sysusers  sysxtypes | LIST:  CHECK\_CONSTRAINTS  COLUMNS  COLUMNS\_S  PARAMETERS  PARAMETERS\_S  REF\_CONSTRAINTS  REFERENTIAL\_CONSTRAINTS  ROUTINES  ROUTINES\_S  SQLCOLPRIVILEGES  SQLCOLUMNS  SQLFOREIGNKEYS  SQLPRIMARYKEYS  SQLPROCEDURECOLS  SQLPROCEDURES  SQLSCHEMAS  SQLSPECIALCOLUMNS  SQLSTATISTICS  SQLTABLEPRIVILEGES  SQLTABLES  SQLTABLETYPES  SQLTYPEINFO  SQLUDTS  SYSATTRIBUTES  SYSBUFFERPOOLNODES  SYSBUFFERPOOLS  SYSCHECKS  SYSCODEPROPERTIES  SYSCOLAUTH  SYSCOLCHECKS  SYSCOLDIST  SYSCOLGROUPDIST  SYSCOLGROUPDISTCOUNTS  SYSCOLGROUPS  SYSCOLGROUPSCOLS  SYSCOLOPTIONS  SYSCOLPROPERTIES  SYSCOLUMNS  SYSCOLUSE  SYSCOMMENTS  SYSCONSTDEP  SYSDATATYPES  SYSDBAUTH  SYSDEPENDENCIES  SYSDUMMY1  SYSEVENTMONITORS  SYSEVENTS  SYSEVENTTABLES  SYSFUNCMAPOPTIONS  SYSFUNCMAPPARMOPTIONS  SYSFUNCMAPPINGS  SYSFUNCPARMS  SYSFUNCTIONS  SYSHIERARCHIES  SYSINDEXAUTH  SYSINDEXCOLUSE  SYSINDEXES  SYSINDEXEXPLOITRULES  SYSINDEXEXTENSIONMETHODS  SYSINDEXEXTENSIONPARMS  SYSINDEXEXTENSIONS  SYSINDEXOPTIONS  SYSJARCONTENTS  SYSJAROBJECTS  SYSKEYCOLUSE  SYSLIBRARIES  SYSLIBRARYAUTH  SYSLIBRARYBINDFILES  SYSLIBRARYVERSIONS  SYSNAMEMAPPINGS  SYSNODEGROUPDEF  SYSNODEGROUPS  SYSPARTITIONMAPS  SYSPASSTHRUAUTH  SYSPLAN  SYSPLANAUTH  SYSPLANDEP  SYSPREDICATESPECS  SYSPROCEDURES  SYSPROCOPTIONS  SYSPROCPARMOPTIONS  SYSPROCPARMS  SYSRELS  SYSREVTYPEMAPPINGS  SYSROUTINEAUTH  SYSROUTINEPARMS  SYSROUTINEPROPERTIESJAVA  SYSROUTINES  SYSSCHEMAAUTH  SYSSCHEMATA  SYSSECTION  SYSSEQUENCEAUTH  SYSSEQUENCES  SYSSERVEROPTIONS  SYSSERVERS  SYSSTMT  SYSTABAUTH  SYSTABCONST  SYSTABLES  SYSTABLESPACES  SYSTABOPTIONS  SYSTBSPACEAUTH  SYSTRANSFORMS  SYSTRIGGERS  SYSTYPEMAPPINGS  SYSUSERAUTH  SYSUSEROPTIONS  SYSVERSIONS  SYSVIEWDEP  SYSVIEWS  SYSWRAPOPTIONS  SYSWRAPPERS  SYSXMLOBJECTAUTH  SYSXMLOBJECTAUTHPERF  SYSXMLOBJECTPROPERTIES  SYSXMLOBJECTRELDEP  SYSXMLOBJECTS  SYSXMLOBJECTXMLDEP  SYSXMLPHYSICALCOLLECTIONS  SYSXMLQUERIES  SYSXMLRELATIONSHIPS  SYSXMLRSPROPERTIES  SYSXMLSTATS  TABLE\_CONSTRAINTS  TABLES  TABLES\_S  UDT\_S  USER\_DEFINED\_TYPES  VIEWS |

## **Utility Commands**

### **BCP**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  bcp [[database\_name.]owner.]table\_name[:slice\_number] {in | out} datafile  [-m maxerrors] [-f formatfile] [-e errfile]  [-F firstrow] [-L lastrow] [-b batchsize]  [-n] [-c] [-t field\_terminator] [-r row\_terminator]  [-U username] [-P password] [-I interfaces\_file] [-S server]  [-a display\_charset] [-q datafile\_charset] [-z language] [-v]  [-A packet size] [-J client character set]  [-T text or image size] [-E] [-g id\_start\_value] [-N] [-X]  [-M LabelName LabelValue] [-labeled]  [-K keytab\_file] [-R remote\_server\_principal]  [-V [security\_options]] [-Z security\_mechanism] [-Q]  EXAMPLE:  bcp pubs2.dbo.titles out titles.data –c –t”^” –Umiguser –SPSNSYBD1 | SYNTAX:  Not Supported.  WORKAROUND:  To load data into a table:  $ db2 “import from <filename> of del insert into <table\_name>”  $ db2 “load from <filename> of del insert into <table\_name>”  To extract data from a table:  $ db2 “export to <filename> of del select \* from <table\_name>” |
| DESCRIPTION:  bcp is a command which is used Transfer Data to and from Adaptive Server | DESCRIPTION:  Import/export/load commands can be used to load/unload data from tables. |

### **ISQL**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  isql [-b] [-e] [-F] [-p] [-n] [-v] [-X] [-Y] [-Q]  [-a display\_charset] [-A packet\_size]  [-c cmdend] [-D database]  [-E editor [-h header [-H hostname [-i inputfile]  [-I interfaces\_file] [-J client\_charset] [-K keytab\_file]  [-l login\_timeout] [-m errorlevel] [-M labelname labelvalue]  [-o outputfile] [-P password] [-R remote\_server\_principal]  [-s col\_separator] [-S server\_name] [-t timeout] [-U username]  [-V [security\_options]] [-w column\_width] [-z localename]  [-Z security\_mechanism]  EXAMPLE:  isql –U username –P password –S servername | SYNTAX:  db2 [option ...] [db2-command | sql-statement |  [? [phrase | message | sqlstate | class-code]]]  option:  -a, -c, -e{c|s}, -finfile, -lhistfile, -o, -n, -p, -rreport, -s,  -t, -td;, -v, -w, -x, -zoutputfile. |
| DESCRIPTION:  isql is a command used to connect to a Sybase Server. | DESCRIPTION:  Db2 is a command used to connect to a Sybase Server |

### **DEFNCOPY**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  defncopy  [-v]  [-X]  [-a <display\_charset>]  [-I <interfaces\_file>]  [-J [<client\_charset>]]  [-K <keytab\_file>]  [-P <password>]  [-R <remote\_server\_principal>]  [-S [<server\_name>]]  [-U <user\_name>]  [-V <security\_options>]  [-Z <security\_mechanism>]  [-z <language>]  { in <file\_name> <database\_name> |  out <file\_name> <database\_name> [<owner>.]<object\_name>  [[<owner>.]<object\_name>...] }  EXAMPLE:  defncopy -Usa -P -SMERCURY in new\_proc stagedb | SYNTAX:  Not Supported.  WORKAROUND:  $ db2 ”db2look -d department -u walid -z ianhe -e -o db2look.sql”  Here DDL statements for objects that have schema name ianhe, created by user walid and in database DEPARTMENT. The db2look output is sent to file db2look.sql |
| DESCRIPTION:  Copies definitions for specified views, rules, defaults, triggers, or procedures from a database to an operating-system file or from an operating-system file to a database | DESCRIPTION:  Using db2look utility we can have the same functionality of defncopy. |

## **NULL VALUES**

* + 1. **Expression is NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  SELECT \*  FROM test\_null  WHERE qty IS NULL | Select \* from project  Where majproj IS NULL |
| RESULT:  stor\_id ord\_num title\_id qty discount  ------- ------- -------- ----------- -----------  stor 1001 TIT1 NULL 34.0  stor 1004 TIT5 NULL 27.0 | STOR\_ID ORD\_NUM TITLE\_ID QTY DISCOUNT  ------- ------------ -------- ------ ------------------  stor 1001 TIT1 - +3.40000000000000E+001  stor 1004 TIT5 - +2.70000000000000E+001 |

CONCLUSION: Both SYBASE and DB2 acts the same in this case.

* + 1. **Expression = NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select \* from test\_null  where qty=NULL | select \* from test\_null  where qty= CAST(NULL AS smallint) |
| RESULT:  stor\_id ord\_num title\_id qty discount  ------- ---------------- -------- ------ --------------------  stor 1001 TIT1 NULL 34.000000  stor 1004 TIT5 NULL 27.000000 | STOR\_ID ORD\_NUM TITLE\_ID QTY DISCOUNT  ------- -------------------- -------- ------ -------- |

CONCLUSION: SYBASE returns columns with NULL whereas DB2 returns nothing.

* + 1. **Expression = @x where @x is a variable or parameter containing NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select price from titles  where price=@x | SELECT price FROM titles  WHERE price IS NULL. |
| RESULT:  price  --------------  NULL  NULL |  |

CONCLUSION: SYBASE returns columns with NULL when “expression=@x” but DB2 does not support “=NULL”. Workaround is as shown above.

* + 1. **Expression != n where n is a literal not containing NULL and expression evaluates to NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, qty  from test\_null  where qty!=35 | select title\_id, qty  from test\_null  where qty<>35 |
| RESULT:  title\_id qty  -------- ------  TIT1 27  TIT1 63  TIT7 21 | TITLE\_ID QTY  -------- ------  TIT1 27  TIT1 63  TIT7 21 |

CONCLUSION: Result is same in both Sybase and DB2.

* + 1. **Expression IS NOT NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, qty  from test\_null  where qty IS NOT NULL | select title\_id, qty  from test\_null  where qty IS NOT NULL |
| RESULT:  title\_id qty  -------- ------  TIT3 35  TIT1 27  TIT1 63  TIT7 21 | TITLE\_ID QTY  -------- ------  TIT3 35  TIT1 27  TIT1 63  TIT7 21 |

CONCLUSION: Result is same in both Sybase and DB2.

* + 1. **Expression != NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, qty  from test\_null  where qty != NULL | select title\_id, qty  from test\_null  where qty != CAST(NULL AS smallint) |
| RESULT:  title\_id qty  -------- ------  TIT3 35  TIT1 27  TIT1 63  TIT7 21 | TITLE\_ID QTY  -------- ------  0 record(s) selected. |

CONCLUSION: Sybase returns all not NULL columns whereas DB2 returns nothing.

* + 1. **Expression != @x where @x contains some value.**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY: [ @x = 35 ]  select title\_id, qty  from test\_null  where qty != @x | QUERY: [ @x = 35 ]  select title\_id, qty  from test\_null  where qty <> 35 |
| RESULT:  title\_id qty  -------- ------  TIT1 27  TIT1 63  TIT7 21 |  |

CONCLUSION: Sybase supports “!=@x” whereas DB2 does not support “<>@x”.

Where @x contains some value. Workaround is as shown above in DB2

* + 1. **Expression LIKE NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, price  from titles  where price like NULL | QUERY:  select title\_id, price  from titles  where price IS NULL |
| RESULT:  title\_id price  -------- ------------------------ | RESULT:  title\_id price  -------- ------------------------ |

CONCLUSION: Sybase returns nothing whereas DB2 doesn’t support. Workaround is as shown above.

* + 1. **Expression NOT LIKE NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, price  from titles  where price not like NULL | QUERY:  select title\_id, price  from titles  where price not IS NOT NULL |
| RESULT:  title\_id price  -------- ------------------------  BU1032 159.92  BU1111 95.60  BU2075 23.92  BU7832 159.92  MC2222 159.92  MC3021 23.92  MC3026 NULL  PC1035 183.60  PC8888 160.00  PC9999 NULL  PS1372 172.72 | RESULT:  title\_id price  -------- ------------------------  BU1032 159.92  BU1111 95.60  BU2075 23.92  BU7832 159.92  MC2222 159.92  MC3021 23.92  MC3026 NULL  PC1035 183.60  PC8888 160.00  PC9999 NULL  PS1372 172.72 |

CONCLUSION: Sybase returns columns with NULL also whereas DB2 doesn’t support it. Workaround is as shown above.

**3.18.10 False and Unknown**

* **False**

DESCRIPTION: False is the result of a failure in a select statement.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, qty  from test\_null  where qty = 50 | select title\_id, qty  from test\_null  where qty = 50 |
| RESULT:  title\_id qty  -------- ------ | TITLE\_ID QTY  -------- ------ |

CONCLUSION: Both SYBASE and DB2 returns same result.

* **Unknown**

DESCRIPTION: Whereas Unknown are those which are not known. A column having NULL value means that no value has been entered for that column or the value is unknown An operation on a column with NULL value is also UNKNOWN.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select qty, qty\*5 as “new qty”  from test\_null | select qty, qty\*5 as “new qty”  from test\_null |
| RESULT:  qty new qty  ------ -----------  NULL NULL  35 175  27 135  NULL NULL  63 315  21 105 | QTY new qty  ------ -----------  - -  35 175  27 135  - -  63 315  21 105 |

CONCLUSION: Both SYBASE and DB2 returns same result.

* + 1. **Substituting a value for NULL**

DESCRIPTION: ISNULL ( *column\_name*, default\_value ) is used in SYBASE to substitute a value for NULL whereas DB2 uses COALESCE( *column\_name, default\_value*).

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, ISNULL(qty,0) as “qty”  from test\_null | select title\_id, COALESCE(qty,0) as “qty”  from test\_null |
| RESULT:  title\_id qty  -------- ------  TIT1 0  TIT3 35  TIT1 27  TIT5 0  TIT1 63  TIT7 21 | TITLE\_ID qty  -------- -----------  TIT1 0  TIT3 35  TIT1 27  TIT5 0  TIT1 63  TIT7 21 |

CONCLUSION: Both SYBASE and DB2 returns same result.

* + 1. **Expression that evaluates to NULL**

DESCRIPTION: An expression with an arithmetic or bitwise operator evaluates to NULL if any of the operands is null. For example, this evaluates to NULL if column is NULL.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select qty, qty\*5 as “new qty”  from test\_null | select qty, qty\*5 as “new qty”  from test\_null |
| RESULT:  qty new qty  ------ -----------  NULL NULL  35 175  27 135  NULL NULL  63 315  21 105 | QTY new qty  ------ -----------  - -  35 175  27 135  - -  63 315  21 105 |

CONCLUSION: Both SYBASE and DB2 returns same result.

* + 1. **Concatenating Strings and NULL**

DESCRIPTION: An expression with an arithmetic or bitwise operator evaluates to NULL if any of the operands is null. For example, this evaluates to NULL if column is NULL.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select “Hello” + NULL +”World” | SELECT SYB.CONCAT(SYB.CONCAT('HELLO', CAST (NULL AS VARCHAR(1))), 'world')  FROM (VALUES 1) temp\_table |
| RESULT:  ----------  HelloWorld | 1  -----------  - |

CONCLUSION: Sybase returns the concatenated result without NULL whereas DB2 returns NULL.

* + 1. **System Generated and User Assigned NULLs**

DESCRIPTION: system-generated NULLs, such as those that result from a system function, behave differently than user-assigned NULLs.

* **User Assigned NULLs**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  Select “TRUE” where 1 != NULL | select 'TRUE' from sysibm.sysdummy1 where 1 IS NOT NULL |
| RESULT:  ----  TRUE | RESULT:  1  ----  TRUE  1 record(s) selected. |

CONCLUSION: In Sybase, comparison with an user assigned NULL results TRUE and DB2 doesn’t support comparison with the user assigned NULLs. Work around is as shown above.

* **System Generated NULLs**

DESCRIPTION: System-generated NULLs, such as those that result from a system function, behave differently than user-assigned NULLs.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  Select “TRUE”  where 1 != convert(integer, NULL) | select 'TRUE' from (VALUES 1) temp\_table  where 1 <> CAST(NULL AS integer) |
| RESULT:  ----  (0 rows affected) | ----  0 record(s) selected. |

CONCLUSION: Results are same in both SYBASE and DB2.

* 1. **KEY MIGRATION CHALLENGES**

### **Wildcard as Default Parameters**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure showind4  @table varchar(30) = "sys%" as  select table\_name = sysobjects.name,index\_name = sysindexes.name,index\_id = indid  from sysindexes,sysobjects  where sysobjects.name like @table  and sysobjects.id = sysindexes.id | SYNTAX: Not Supported  WORKAROUND :  CREATE PROCEDURE showind4 (tableprefix VARCHAR(30) ) POC: BEGIN  DECLARE v\_tableprefix VARCHAR(30);    SET v\_tableprefix = 'sys%';  SET tableprefix = v\_tableprefix;  DECLARE temp\_cursor CURSOR WITH RETURN TO CLIENT FOR SELECT sysobjects.name AS table\_name,sysindexes.name AS index\_name, indid AS index\_id FROM sysindexes,sysobjects WHERE SYB.isLike(sysobjects.name, tableprefix, '') = 1  AND sysobjects.id = sysindexes.id;  OPEN temp\_cursor;  END POC |
| DESCRIPTION:  The above example show how wildcard characters can be assigned to parameters. | DESCRIPTION:  The equivalent functionality is not available in DB2.Workaround is possible. |

### **Procedure Groups**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  drop proc orders | Not Supported  WORKAROUND:  We need to identify the stored procedures and their dependent child procedures and then need to drop them explicitly one by one or in a script file. |
| DESCRIPTION:  The optional semicolon and integer number after the name of the procedure in the create procedure and execute statements allow you to group procedures of the same name so that they can be dropped together with a single drop procedure statement. Procedures used in the same application are often grouped this way. For example, you might create a series of procedures called orders;1, orders;2,and so on.  The above syntax will drop the entire group of stored procedures. | DESCRIPTION:  The equivalent functionality is not available in DB2. |

### **Setting options in stored procedures**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  Select pub\_id from titles set rowcount 2 | SYNTAX:  Not Supported  WORKAROUND 1(FOR SELECT):  Select pub\_id from titles fetch first 2 rows only;  WORKAROUND 2(FOR UPDATE):  create procedure set\_rowcount\_test(IN p\_name char(30))  poc: Begin  Declare v\_counter int default 1;  Declare v\_name varchar(6);  Declare v\_sr\_no int;  Declare v\_commit\_freq int default 0;  Declare insrt\_c1 cursor with hold  for select sr\_no,name from  data\_commit;  open insrt\_c1;  While (v\_counter <= 1000) do  Fetch insrt\_c1 into v\_sr\_no,v\_name;  Update data\_commit  Set name = 'pradeep'  where sr\_no = v\_sr\_no;  Set v\_counter = v\_counter +1;  Set v\_commit\_freq = v\_commit\_freq + 1;  If (v\_commit\_freq = 3) then  Commit;  Set v\_commit\_freq = 0;  End if;  End while;  end poc |
| DESCRIPTION:  Here if there are n number of pub\_id’s then only 2 rows will be returned | DESCRIPTION:  There is no mapping of this functionality in DB2. |

CONCLUSION

The above functionality can be implemented in DB2 by adding commit logic in the application code.In the above example workaround2 commit frequency is set to 3. i.e. for every 3 records the transaction will be committed.In DB2 if you want to fetch the first two records then example given in workaround1 would be fine.

**Commonly used Global Variables:**

@@connections

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@connections | SYNTAX:  Not Supported directly  WORKAROUND:  $ db2 get snapshot for database on dbname | grep ‘Application connects’  Application connects = 24 |
| DESCRIPTION:  The @@connections global variable returns the number of logins or attempted logins to the Sybase server since it was started. | DESCRIPTION:  In DB2, you can execute the Snapshot Monitor and get the ‘Connects Since Database Activation’ element, which is indicated as ‘Application connects’ in the output, as the following example: |

@@error/@@sqlstatus

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@error  SELECT @@sqlstatus | SYNTAX:  Not Supported directly  WORKAROUND:  CREATE PROCEDURE MOHM.GLB\_TEMP\_TABLE\_DEBUGGING (IN P\_NAME VARCHAR(50) DEFAULT NULL,IN P\_EMP\_ID INT,OUT P\_ERR\_MSG VARCHAR(500),OUT E\_RET\_VAL CHAR(5))  ------------------------------------------------------------------------  -- SQL Stored Procedure  -- CREATED BY : XYZ  -- CREATION DATE : 26/09/2005  -- PURPOSE : TO MAKE USE OF SQLSTATE FOR DEBUGGING  -- TABLES USED : MOHM.TEST  -- INPUT PARAMETERS : NAME OF EMPLOYEE,EMPLOYEE\_ID  -- OUTPUT PARAMETERS : ERROR\_MESSAGE,SQLSTATE  ------------------------------------------------------------------------  DBCOE : BEGIN  -- DECLARATION OF LOCAL VARIABLES --  DECLARE V\_STMT\_NO INT;  DECLARE SQLSTATE CHAR(5) DEFAULT '00000';  DECLARE V\_RET\_VAL CHAR(5) DEFAULT '00000';  SET E\_RET\_VAL = V\_RET\_VAL;  SET P\_ERR\_MSG = 'EXECUTION SUCCESSFULL';  BEGIN  DECLARE EXIT HANDLER FOR SQLSTATE '23505'    BEGIN  DECLARE EXIT HANDLER FOR SQLSTATE 'T1000'  BEGIN  GET DIAGNOSTICS EXCEPTION 1 P\_ERR\_MSG = MESSAGE\_TEXT ;  SET E\_RET\_VAL = SQLSTATE;    BEGIN  DECLARE GLOBAL TEMPORARY TABLE MOHM\_GLBL\_TABLE1(TEMP\_COL1 VARCHAR(500)) WITH replace on commit preserve rows not logged IN GLBL\_TBSPC;  SET P\_ERR\_MSG = 'PRIMARY KEY VIOLATION TRYING TO INSERT DUPLICATE VALUES';  BEGIN  DECLARE TEMP\_CUR CURSOR WITH RETURN FOR SELECT \* FROM SESSION.MOHM\_GLBL\_TABLE1;  INSERT INTO SESSION.MOHM\_GLBL\_TABLE1 VALUES(P\_ERR\_MSG);  OPEN TEMP\_CUR;  END;  END;  END;    IF V\_STMT\_NO = 100 THEN  SIGNAL SQLSTATE 'T1000'  SET MESSAGE\_TEXT = 'PRIMARY KEY VIOLATION TRYING TO INSERT DUPLICATE VALUES';  END IF;    END; -- END OF BEGIN DECLARE EXIT HANDLER FOR SQLSTATE 'T1000'  -- INSERTION LOGIC --  BEGIN  SET V\_STMT\_NO = 100;  INSERT INTO MOHM.TEST VALUES(P\_NAME,P\_EMP\_ID);  END;  END; -- END OF BEGIN DECLARE EXIT HANDLER FOR SQLSTATE '23505'  END DBCOE |
| DESCRIPTION:  In Sybase, you can query the @@error or @@sqlstatus global variables to check  whether the previous statement succeeded or not. The @@error has the error  status of the most recently executed statement. The @@sqlstatus has the  status from the last FETCH statement. Its value is 0 (success), 1 (error), or 2  (no more data). | DESCRIPTION:  In DB2 UDB, return codes from SQL statements are handled with SQLCODE or SQLSTATE. You can use either of them in applications. The SQLCODE conforms to ISO/ANSI SQL standard, and the SQLSTATE is based on the ISO/ANSI SQL92  standard. |

@@rowcount

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  SELECT @@rowcount | SYNTAX:  Not Supported directly  WORKAROUND:  CREATE PROCEDURE sqlprocg ( IN deptnbr VARCHAR(3),OUT rcount INTEGER)  LANGUAGE SQL  BEGIN  DECLARE SQLSTATE CHAR(5)  UPDATE CORPDATA.PROJECT  SET PRSTAFF = PRSTAFF + 1.5  WHERE DEPTNO = deptnbr;  GET DIAGNOSTICS rcount = ROW\_COUNT;  END  This stored procedure issues the UPDATE statement and the GET DIAGNOSTICS statement, and returns the affected row count. |
| DESCRIPTION:  In Sybase, the @@rowcount variable contains the number of rows affected by  the last query. The value represents the number of rows of a cursor result set  returned to the client, up to the last fetch request. | DESCRIPTION:  In DB2 UDB, you can use the GET DIAGNOSTICS statement with the ROW\_COUNT option in stored procedures. If the previous statement is the PREPARE statement, ROW\_COUNT identifies the estimated number of result rows in the prepared statement. If the previous statement is the DELETE, INSERT, or UPDATE statement, ROW\_COUNT identifies the number of rows deleted, inserted, or updated by that statement, excluding rows affected by either triggers or  referential integrity constraints. |

### **BCP**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  bcp [[database\_name.]owner.]table\_name[:slice\_number] {in | out} datafile  [-m maxerrors] [-f formatfile] [-e errfile]  [-F firstrow] [-L lastrow] [-b batchsize]  [-n] [-c] [-t field\_terminator] [-r row\_terminator]  [-U username] [-P password] [-I interfaces\_file] [-S server]  [-a display\_charset] [-q datafile\_charset] [-z language] [-v]  [-A packet size] [-J client character set]  [-T text or image size] [-E] [-g id\_start\_value] [-N] [-X]  [-M LabelName LabelValue] [-labeled]  [-K keytab\_file] [-R remote\_server\_principal]  [-V [security\_options]] [-Z security\_mechanism] [-Q]  EXAMPLE:  bcp pubs2.dbo.titles out titles.data –c –t”^” –Umiguser –SPSNSYBD1 | SYNTAX:  Not Supported.  WORKAROUND:  To load data into a table:  $ db2 “import from <filename> of del insert into <table\_name>”  $ db2 “load from <filename> of del insert into <table\_name>”  To extract data from a table:  $ db2 “export to <filename> of del select \* from <table\_name>” |
| DESCRIPTION:  bcp is a command which is used Transfer Data to and from Adaptive Server | DESCRIPTION:  Import/export/load commands can be used to load/unload data from tables. |

### **DEFNCOPY**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  defncopy  [-v]  [-X]  [-a <display\_charset>]  [-I <interfaces\_file>]  [-J [<client\_charset>]]  [-K <keytab\_file>]  [-P <password>]  [-R <remote\_server\_principal>]  [-S [<server\_name>]]  [-U <user\_name>]  [-V <security\_options>]  [-Z <security\_mechanism>]  [-z <language>]  { in <file\_name> <database\_name> |  out <file\_name> <database\_name> [<owner>.]<object\_name>  [[<owner>.]<object\_name>...] }  EXAMPLE:  defncopy -Usa -P -SMERCURY in new\_proc stagedb | SYNTAX:  Not Supported.  WORKAROUND:  $ db2 ”db2look -d department -u walid -z ianhe -e -o db2look.sql”  Here DDL statements for objects that have schema name ianhe, created by user walid and in database DEPARTMENT. The db2look output is sent to file db2look.sql |
| DESCRIPTION:  Copies definitions for specified views, rules, defaults, triggers, or procedures from a database to an operating-system file or from an operating-system file to a database | DESCRIPTION:  Using db2look utility we can have the same functionality of defncopy. |

* 1. **OPEN RISKS/ISSUES**

#### **RETRIEVING THE CURRENT DATABASE NAME**

DESCRIPTION: To obtain the database name one is currently logged into.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select db\_name()  2> go | QUERY  VALUES CURRENT SERVER |
| RESULT  ------------------------------  pubs2  (1 row affected) | RESULT  1  ------------------  SAMPLE  1 record(s) selected. |

CONCLUSION: Supported by Sybase only. Workaround is shown above.

#### **RETRIEVING THE CURRENT DATABASE ID**

DESCRIPTION: To obtain the database id one is currently logged into.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select db\_id()  2> go | In DB2, there is no concept of Database ID. The name is the identifier |
| RESULT  ------  4  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING THE HOST ID**

DESCRIPTION: To retrieve the host id on which the database server is running

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select host\_id()  2> go | DB2 does not have the concept of host-ids as the database server is identified by its name or IP address(in conjunction with the port number) |
| RESULT  ---------------------  8746  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING THE SERVER NAME**

DESCRIPTION: Retrieves the name of the database server

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select host\_name()  2> go | QUERY  select host\_name from table(SYSPROC.ENV\_GET\_SYS\_INFO()) as systeminfo |
| RESULT  -----------------------  ps-server1  (1 row affected) | Can be checked only with FP (Fix Pack) 9 and higher. |

CONCLUSION Supported by Sybase only. Workaround is as shown above.

#### **RETRIEVING AN OBJECT ID**

DESCRIPTION: Retrieves the ID of any object like tables, indexes, views etc.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select object\_id("titles")  2> go | In DB2, objects are identified by their type and fully qualified name. For routines (like functions and procedures) we have the concept of routineid and for sequences we have seqid. |
| RESULT  -----------  224000798  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **RETRIEVING AN OBJECT NAME**

DESCRIPTION: Retrieves the name of any object like tables, indexes, views etc.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select object\_name( 224000798)  2> go | In DB2, objects are identified by their type and fully qualified name. For routines (like functions and procedures) we have the concept of routineid and for sequences we have seqid. |
| RESULT  ------------------------------  titles  (1 row affected) |  |

CONCLUSION Supported by Sybase only.

#### **FETCHING THE SERVER USER ID OF THE USER**

DESCRIPTION: Retrieves the server user id of the User.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select suser\_id("mohm")  2> go | In DB2, identifies users with their names. There isn’t any concept of server user id. |
| RESULT  -----------  23  (1 row affected) |  |

CONCLUSION Supported in Sybase Only.

#### **COMPARING TIME STAMP VALUES**

DESCRIPTION: Compares two timestamp values.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| DESCRIPTION:  Use the tsequal system function to compare timestamps when you are using browse mode in a front-end application. For example, the following  statement updates a row in publishers that has been browsed. It compares the timestamp column in the browsed version of the row with the hexadecimal timestamp in the stored version. If the two timestamps are not equal, you receive an error message, and the row is not updated.  EXAMPLE:  update publishers  set city = "Springfield"  where pub\_id = "0736"  and tsequal(timestamp,0x0001000000002ea8) | There is no such timestamp in hex format in DB2. In Sybase also this format is generated by the  db engine only for its own internal purpose. The user does not have to bother about that. after exporting also, this data can be loaded into  DB2 tables using varchar or char(20) data types. |

NOTE: Do not use the tsequal function in the where clause as a search argument.When you use tsequal, the rest of the where clause should match a single row uniquely. Use the tsequal function only in insert and update statements.

#### **FETCHING THE USER ID**

DESCRIPTION: Retrieves the database user id of the User

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select user\_id()  2> go | In DB2, the users are identified by their names. The Concept of user id does not exist. |
| RESULT  -----------  2 |  |

CONCLUSION Supported in Sybase. DB2 doesn’t support it.

#### **VALIDATING USER NAME**

DESCRIPTION: Checks to see if the user is authorised to the database. Returns 0 if the username is invalid. Else returns a non-zero value.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select valid\_name("mohm")  2> go | QUERY:  db2 “select grantee from syscat.dbauth where grantee=’MOHM’ “ | grep ‘record(s)’ | cut –f3 –d” “ |
| RESULT  -----------  1  (1 row affected) | RESULT  0 |

CONCLUSION: Supported in Sybase **Only.** Workaround is as shown above.

#### **VALIDATING USER THROUGH USER ID**

DESCRIPTION Checks to see if the user with the given user id is authorised to the database.

EXAMPLE

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY  1> select valid\_user(23)  2> go | In DB2, there isn’t any concept of userid. Thus, one cannot check for the validity based on user id’s. |
| RESULT  -----------  1  (1 row affected) |  |

CONCLUSION Supported in Sybase **Only.**

### **Using Recompile**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  create procedure add\_proc  @a int = 10, @b int = null  with recompile  as  declare @c int  select @c=@a+@b | Not Supported |
| DESCRIPTION:  Here WITH RECOMPILE option can be used which will recompile the SP before execution and will create optimal access paths for the sql statements in the SP. | DESCRIPTION:  In DB2 we cannot explicitly mention the recompile option inside a SP but recompilation is done automatically. |

### **Using Recompile in Execute**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  exec storeid\_proc "7066" with recompile | Not Supported |
| DESCRIPTION:  In the above example WITH RECOMPILE can be also used with an exec statement. | DESCRIPTION:  The equivalent functionality is not available in DB2. |

### **Binding to Columns & User Defined Datatypes**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  sp\_bindrule rulename, objname[, futureonly] | Option not available. |
| DESCRIPTION:  rulename --is the name of the rule created with create rule.  Objname --is the name of the table and column, or of the user-defined datatype to  which the rule is to be bound if the parameter is not of the from table.column, it is assumed to be a user datatype.  Futureonly-- when binding a rule to a userdefined datatype. | DESCRIPTION:  Not available. |

Note: The check constraint in Db2 doesn’t required to get bind on the column, once created it works for that column whenever we insert a value not mentioned in the constraint.

### **Binding and Unbinding**

| **SYBASE** | **DB2** |
| --- | --- |
| SYNTAX:  sp\_bindefault advancedflt | SYNTAX:  ALTER TABLE table\_name ALTER COLUMN col\_name SET DEFAULT default\_value  Before this command is executed, DB2 requires the column name and the default value which is specified in default\_name. |

**Note:** No Equivalent found for binding defaults in db2. Workaround is as shown above.

### **Transaction States**

DESCRIPTION

Transaction state gives the current state of a transaction.

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  Select @@transtate | There is no concept of transaction state in DB2 |
| RESULT:  -----------  1 |  |

CONCLUSION: @@transtate stores the current state of transaction in SYBASE and whereas it doesn’t exist in DB2.

**Expression = @x where @x is a variable or parameter containing NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select price from titles  where price=@x | SELECT price FROM titles  WHERE price IS NULL. |
| RESULT:  price  --------------  NULL  NULL |  |

CONCLUSION: SYBASE returns columns with NULL when “expression=@x” but DB2 does not support “=NULL”. Equivalent is as shown above.

**Expression != @x where @x contains some value.**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY: [ @x = 35 ]  select title\_id, qty  from test\_null  where qty != @x | select title\_id, qty  from test\_null  where qty<>35 |
| RESULT:  title\_id qty  -------- ------  TIT1 27  TIT1 63  TIT7 21 | RESULT:  title\_id qty  -------- ------  TIT1 27  TIT1 63  TIT7 21 |

CONCLUSION: Sybase supports “!=@x” whereas DB2 does not support “<>@x”.

Where @x contains some value.

**Expression LIKE NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, price  from titles  where price like NULL | QUERY:  select title\_id, price  from titles  where price IS NULL |
| RESULT:  title\_id price  -------- ------------------------ | RESULT:  title\_id price  -------- ------------------------ |

CONCLUSION: Sybase returns nothing whereas DB2 doesn’t support. Workarouns is as shown above.

**Expression NOT LIKE NULL**

EXAMPLE:

| **SYBASE** | **DB2** |
| --- | --- |
| QUERY:  select title\_id, price  from titles  where price not like NULL | QUERY:  select title\_id, price  from titles  where price IS NOT NULL |
| RESULT:  title\_id price  -------- ------------------------  BU1032 159.92  BU1111 95.60  BU2075 23.92  BU7832 159.92  MC2222 159.92  MC3021 23.92  MC3026 NULL  PC1035 183.60  PC8888 160.00  PC9999 NULL  PS1372 172.72 | RESULT:  title\_id price  -------- ------------------------  BU1032 159.92  BU1111 95.60  BU2075 23.92  BU7832 159.92  MC2222 159.92  MC3021 23.92  MC3026 NULL  PC1035 183.60  PC8888 160.00  PC9999 NULL  PS1372 172.72 |

CONCLUSION: Sybase returns columns with NULL also whereas DB2 doesn’t support it. Workaround is as shown above.

1. **glossary of terms**