







IBM TSM SQL Workshop (A Practical Approach)

Robert Brilmayer, PLCS/STORServer
Laura Buckley, STORServer
Andy Raibeck, IBM



Agenda

- ◆ Brief SQL tutorial
- ◆ Anatomy of a SQL Database
- ◆ TSM's SQL Interface
 - SELECT Syntax
 - Clauses, Operators and Functions
 - Basic Joins
- ◆ Challenge us...



2



A Brief SQL Tutorial

- ◆ Structured Query Language maintained by ANSI
- ◆ Used to communicate with databases
- ◆ Universal interface that can:
 - ◆ View (select)
 - ◆ Manipulate (fetch,update)
 - ◆ Add (insert) data to databases*

*TSM's SQL Interface only supports viewing data (select)



3



Anatomy of a Relational Database

- ◆ In a relational database, one piece of information relates to another
- ◆ Data is stored in columns in tables
- ◆ The data in a column in one table can relate to data in a column in another table
- ◆ A row represents a single entry in a table



4



Example of a Table

TABLE = ASSOCIATIONS

COLUMNS

ROWS

DOMAIN_NAME	SCHEDULE_NAME	NODE_NAME	CHG_TIME	CHG_ADMIN
EXCHANGE	DAILY_FULL	EXCH_STORMAIL	7/29/2003 13:21	ADMIN
MOBILE	DAILY_INCR	BEBO	5/7/2003 16:30	ADMIN
MOBILE	DAILY_INCR	ROSEBUD	5/7/2003 16:30	ADMIN
PRODUCTON	DAILY_INCR	SSHQ	5/7/2003 17:12	ADMIN



Relationships Between Tables

CLIENT SCHEDULES							
DOMAIN_NAME	SCHEDULE_NAME	DESCRIPTION	ACTION	OBJECTS	STARTDATE	STARTTIME	DURATION
EXCHANGE	DAILY_FULL		COMMAND	c:\progra~	7/29/2003	23:10:00	10
MOBILE	DAILY_INCR	Daily increment	INCREMENTAL		11/14/2001	0:00:00	23
PRODUCTION	DAILY_INCR	Daily increment	INCREMENTAL		11/14/2001	0:00:00	8
PRODUCTION	TEST	testing options	INCREMENTAL		7/11/2003	16:27:48	1
SQL	DAILY_FULL	MSSQL agent	COMMAND	c:\progra~	7/3/2003	23:30:00	2
SQL	STORBASE_FULL	MSSQL agent	COMMAND	c:\progra~	6/2/2003	22:00:00	15
STANDARD	ARCH-TEST		ARCHIVE	e:* d:*	5/30/2003	16:17:00	1
STANDARD	DAILY_INCR	Daily increment	INCREMENTAL		11/14/2001	0:00:00	8

ASSOCIATIONS				
DOMAIN_NAME	SCHEDULE_NAME	NODE_NAME	CHG_TIME	CHG_ADMIN
EXCHANGE	DAILY_FULL	EXCH_STORMAIL	7/29/2003 13:21	ADMIN
MOBILE	DAILY_INCR	BEBO	5/7/2003 16:30	ADMIN
MOBILE	DAILY_INCR	ROSEBUD	5/7/2003 16:30	ADMIN
PRODUCTION	DAILY_INCR	SSHQ	5/7/2003 17:12	ADMIN
PRODUCTON	DAILY_INCR	SSEXT	5/7/2003 17:12	ADMIN





TSM SQL Interface

- ◆ Supports the SQL SELECT query only
 - ◆ Requires a “minimum” of 4MB of free space in the database
 - ◆ Complicated queries may take a long time to complete and can interfere with server operations
 - ◆ You cannot issue SELECT queries from the server console (admin command line only)



7



TSM SQL Interface

- ◆ “Mostly” conforms to standard SQL
 - Subset of the SQL92 and SQL93 ANSI standards
- ◆ Does NOT support:
 - UNION
 - INTERSECT
 - EXCEPT
 - Correlated subqueries (returning multiple values)
 - Semicolon cannot be used as a command terminator



8



SELECT Syntax

```
SELECT column/expression [,n..]
FROM tablename {,n..}
```

- ◆ **column** refers to a column in a table (
 - * is allowed as a wildcard to select all columns in a table
- ◆ **expression** refers to functions that allow you manipulate the data being returned
- ◆ [,n..] indicates that you may specify one or more columns or expressions
- ◆ FROM clause indicates which table to search
- ◆ You can specify one or more **tablename**s



9




Simple SELECT Example

```
SELECT STARTTIME
FROM CLIENT_SCHEDULES
```

- ◆ Display the contents of the *STARTTIME* column from every row in the *CLIENT_SCHEDULES* table
- ◆ Column and table names cannot be abbreviated
- ◆ Column names are displayed in the order they are entered on the SELECT statement
- ◆ Much of the data in TSM is stored in uppercase and must be entered in uppercase in SELECT statements





10




TSM Database Catalog


- ◆ TSM has three system catalog tables so that you can view the tables, columns and enumerated data types available
 - SYSCAT.TABLES
 - SYSCAT.COLUMNS
 - SYSCAT.ENUMTYPES









11




Viewing the TSM System Catalog

- ◆ To view available table names:
SELECT * FROM SYCAT.TABLES
- ◆ To view all column names within tables
SELECT tablename, colname FROM SYSCAT.COLUMNS
- ◆ To view the valid values and order for enumerated types
SELECT * FROM SYSCAT.ENUMTYPES







12



Manipulating the Results

- ◆ You may not want all of the data in all of the columns all of the time so:
 - SQL provides ***clauses, operators, and functions***
 - These allow you to sort, order, filter, and compute the data on a select command



13



Clauses, Operators, and Functions

ALL	CURRENT_DATE	GROUP BY	ORDER BY
ANY	CURRENT_TIME	HAVING	POSITION
AVG	CURRENT_TIMESTAMP	IN	SOME
AS	CURRENT_USER	JOIN	SUBSTRING
BETWEEN	DISTINCT	LIKE	SUM
CASE	EXISTS	MAX	TRIM
CAST	EXTRACT	MIN	WHERE
COUNT	FROM	NULL	

*Supported by TSM SELECT



14



Clauses (not the Santa kind..)

- ◆ A **clause** is a part of a SQL statement (i.e. SELECT column1,column2)
- ◆ Clauses combine to form an entire SQL statement
- ◆ For example, you can combine the SELECT clause and FROM clause to form a statement
 - SELECT also refers to the statement itself



15




WHERE Clause

- ◆ The **WHERE** clause allows you to filter out rows from the results
 - I want this, this and this, but only where this condition is true
 - I want to see all the volumes on which the client called CARROLL has data:

```
SELECT NODE_NAME, VOLUME_NAME
FROM VOLUMEUSAGE
WHERE NODE_NAME= 'CARROLL'
```







16



Comparison Operators (use with WHERE)

Operator	Description	Example (WHERE)
IS	Equal (used with NULL)	NODE_NAME IS NULL
IS NOT	Not equal (used with NULL)	VERSION IS NOT NULL
=	Equal	NODE_NAME='CARROLL'
<>	Not Equal	NODE_NAME<>'CARROLL'
<	Less than	LOGICAL_MB < 5000
>	Greater than	LOGICAL_MB > 5000
<=	Less than or equal to	LOGICAL_MB <= 5000
>=	Greater than or equal to	LOGICAL_MB >= 5000




17






Logical Operators

- ◆ Logical operators separate two or more conditions in the WHERE clause
 - **LIKE** is used with the wildcard % to match all occurrences


```
SELECT * FROM NODES
WHERE NODE_NAME LIKE 'C%'
```
 - **AND** means that the expressions on both sides must be true to return TRUE


```
SELECT * FROM NODES WHERE
NODE_NAME='CARROLL' AND
PLATFORM_NAME='WinNT'
```




18



More Logical Operators

- ◆ You can use **OR** to sum up a series of conditions. If any of the comparisons is true, OR returns TRUE


```
SELECT * FROM NODES
WHERE NODE_NAME='CARROLL' OR 'DODSON'
```
- ◆ Use **IN** to replace multiple OR's


```
SELECT * FROM NODES WHERE NODE_NAME
IN('CARROL', 'DODSON', 'LEWIS', 'CHARLES')
```
- ◆ Use **BETWEEN** to get a range


```
SELECT NODE_NAME FROM OCCUPANCY
WHERE LOGICAL_MB BETWEEN 5000 AND 10000
```



19



ORDER BY Clause

- ◆ The **ORDER BY** clause is used to sort the rows prior to displaying them:


```
SELECT NODE_NAME, PLATFORM_NAME
FROM NODES
ORDER BY PLATFORM NAME
```
- ◆ You can specify that the results be sorted in ascending or descending order:


```
SELECT NODE_NAME, TYPE, FILESPACE_NAME,
LOGICAL_MB FROM OCCUPANCY
ORDER BY LOGICAL_MB DESC
```



20



Functions

- ◆ Functions allow you to aggregate data and operate on strings, numeric and date and time values
- ◆ Aggregate functions perform operations on values from selected rows to produce a single value.
 - They include COUNT(*), SUM, AVG, MAX, and MIN.
 - COUNT(*) is useful for finding the number of rows that match a query.



21



Timestamp and CAST Functions

- ◆ Example of date/time and cast function – displays nodes that have not accessed the server in a specified (\$1) number of days
- ◆ CAST's the timestamp as decimal for processing

```
SELECT NODE_NAME, LASTACC_TIME FROM NODES
WHERE - CAST((CURRENT_TIMESTAMP -
LASTACC_TIME)DAYS AS - DECIMAL) >= $1
```



22



GROUP BY Clause

- ◆ The GROUP BY clause allows you to combine the rows being selected into logical groups
- ◆ Normally used with aggregate functions


```
SELECT NODE_NAME, SUM(NUM_FILES) AS
#_OF_FILES, SUM(LOGICAL_MB) AS
TOTAL_MB FROM OCCUPANCY GROUP BY
NODE_NAME
```
- ◆ When using aggregate functions, you need to name the columns (i.e. AS #_OF_FILES)



23



HAVING Clause

- ◆ HAVING always follows the GROUP BY clause
- ◆ Use the HAVING clause to filter the results of the GROUP BY clause


```
SELECT NODE_NAME, SUM(NUM_FILES) AS
#_OF_FILES, SUM(LOGICAL_MB) AS
TOTAL_MB FROM OCCUPANCY GROUP BY
NODE_NAME HAVING SUM(LOGICAL_MB)>1000
```



24



Joining Tables

- ◆ Helps you see how data relates between tables
- ◆ There are different types of joins, depending on the data you are trying to relate
- ◆ Use an alias to specify which column you want to display when joining tables with columns of the same name
- ◆ The keyword **DISTINCT** specifies only unique rows will be retrieved and prevent duplicates



25



JOIN Example

- ◆ To see which schedules a node is associated with join ASSOCIATIONS with CLIENT_SCHEDULES
- ◆ DOMAIN_NAME and SCHEDULE_NAME are common columns
- ◆ We'll use the alias C.SCHEDULE_NAME, to indicate which SCHEDULE_NAME to return



26



Join Example – continued

```
SELECT DISTINCT NODE_NAME,C.SCHEDULE_NAME  
FROM ASSOCIATIONS A, CLIENT_SCHEDULES C  
WHERE A.SCHEDULE_NAME=C.SCHEDULE_NAME
```

- ◆ This is an example of an inner join or equi-join
- ◆ The goal is to match the values of a column in one table to the corresponding values in the second table (schedule_name)



27



Challenge Us

- ◆ That's was just an overview of TSM SQL and some basic examples
- ◆ There is a lot more you can do with the tool
- ◆ We are now going to present some of the solutions to questions you provided earlier in the week



28



SQL References

- ◆ TSM SQL Guide
 - More info on clauses, operators, and functions
 - More examples
 - Using ODBC
- ◆ ADSM.ORG
- ◆ SQL Books and self-study courses

