Types of DBMS

A DBMS is said to be a Relational DBMS or RDBMS if the database relationships are treated in the form of a table. There are three keys on relational DBMS: relation, domain and attributes. A network means it contains a fundamental constructs sets or records sets contains one to many relationship, records contains fields [statical table](http://en.wikipedia.org/w/index.php?title=Statical_table&action=edit&redlink=1" \o "Statical table (page does not exist)) that is composed of rows and columns is used to organize the database and its structure and is actually a two dimension array in the [computer memory](http://en.wikipedia.org/wiki/Computer_memory). A number of RDBMSs are available, some popular examples are [Oracle](http://en.wikipedia.org/wiki/Oracle), [Sybase](http://en.wikipedia.org/wiki/Sybase), [Ingress](http://en.wikipedia.org/wiki/Ingress), [Informix](http://en.wikipedia.org/wiki/Informix), [Microsoft SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server), and [Microsoft Access](http://en.wikipedia.org/wiki/Microsoft_Access).

**Creating indexes**

Before executing **create index**, turn on **select into**:

sp\_dboption,'select into', true

The simplest form of **create index**is:

create index *index\_name*

     on *table\_name* (*column\_name*)

To create an index on the *au\_id* column of the *authors* table, execute:

create index au\_id\_ind

on authors(au\_id)

The index name must conform to the rules for identifiers. The column and table name specify the column you want indexed and the table that contains it.

You cannot create indexes on columns with *bit*, *text*, or *image* datatypes.

<http://www.sybaseteam.com/differences-between-clustered-non-clustered-indexes-sybase-t-405.html>

**Why the table can only have one clustered index?**

Because the table only exists once. The clustered index, as you note, is the table. Yes, you can create secondary indexes with all the columns if you want, they're not the clustered index because SQL considers them copies of the data in the table, the clustered index contains the data in the table.  
  
The clustered index is the one whose keys are present in all no clustered indexes

**declare cursor examples**

The following **declare cursor** statement defines a result set for the *authors\_crsr* cursor that contains all authors who do not reside in California:

**declare** authors\_crsr **cursor**

**for** select au\_id, au\_lname, au\_fname

from authors

where state != 'CA'

**for update**

The following example defines an **insensitive** scrollable result set, of the **stores\_scrollcrsr**, containing bookstores in California, for:

**declare** storinfo\_crsr insensitive scroll **cursor**

**for** select stor\_id, stor\_name, payterms

from stores

where state = "CA"

To declare an **insensitive**, nonscrollable cursor called “C1,” enter:

declare C1 insensitive cursor for

select fname from emp\_tab

To declare an **insensitive**, scrollable cursor called “C3,”enter:

declare C3 insensitive scroll cursor for

select fname from emp\_tab

To fetch the first row, enter:

fetch first from <cursor\_name>

You can also fetch the columns of the first row from the result set. To place them in the variables you specify in *<fetch\_target\_list>*, enter:

fetch first from <cursor\_name> into <fetch\_target\_list>

You can fetch the 20th row in the result set directly, regardless of the cursor’s current position:

fetch absolute 20 from <cursor\_name> into <fetch\_target\_list>

*\*/*

**create** **procedure** increase\_price\_cursor

**as**

**declare** @price money

*/\* declare a cursor for the select from titles \*/*

**declare** curs **cursor** **for**

**select** price

**from** titles

**for** **update** **of** price

*/\* open the cursor \*/*

**open** curs

*/\* fetch the first row \*/*

**fetch** curs **into** @price

*/\* now loop, processing all the rows*

*\*\* @@sqlstatus = 0 means successful fetch*

*\*\* @@sqlstatus = 1 means error on previous fetch*

*\*\* @@sqlstatus = 2 means end of result set reached*

*\*/*

while (@@sqlstatus != 2)

**begin**

*/\* check for errors \*/*

if (@@sqlstatus = 1)

**begin**

print "Error in increase\_price"

**return**

**end**

*/\* next adjust the price according to the*

*\*\* criteria*

*\*/*

if @price > $60

**select** @price = @price \* 1.05

**else**

if @price > $30 **and** @price <= $60

**select** @price = @price \* 1.10

**else**

if @price <= $30

**select** @price = @price \* 1.20

*/\* now, update the row \*/*

**update** titles

**set** price = @price

**where** **current** **of** curs

*/\* fetch the next row \*/*

**fetch** curs **into** @price

**end**

*/\* close the cursor and return \*/*

**close** curs

**return**

# Sybase Adaptive Server Enterprise Retrieval Using Cursors

If you expect only a single row to exist in the employee table with the specified emp\_id, use the singleton SELECT. In a singleton SELECT, you specify the SELECT statement and destination variables in one concise SQL statement:

SELECT emp\_name, emp\_salary

INTO :emp\_name\_var, :emp\_salary\_var

FROM employee WHERE emp\_id = :emp\_id\_var;

However, when a SELECT may return multiple rows, you must:

1. Declare a cursor.
2. Open it (which conceptually executes the SELECT).
3. Fetch rows as needed.
4. Close the cursor.

#### Declaring and opening a cursor

Declaring a cursor is tightly coupled with the OPEN statement. The DECLARE specifies the SELECT statement to be executed, and the OPEN actually executes it.

Declaring a cursor is similar to declaring a variable; a cursor is a nonexecutable statement just like a variable declaration. The first step in declaring a cursor is to define how the result set looks. To do this, you need a SELECT statement. Since you must refer to the result set in subsequent SQL statements, you must associate the result set with a logical name.

Note**Multiple cursors** The CT-Lib API lets you declare and open multiple cursors without having to open additional database connections.

#### Example

Assume the SingleLineEdit sle\_1 contains the state code for the retrieval:

// Declare cursor emp\_curs for employee table retrieval.

DECLARE emp\_curs CURSOR FOR

SELECT emp\_id, emp\_name FROM EMPLOYEE

WHERE emp\_state = :sle\_1.text;

// Declare local variables for retrieval.

string emp\_id\_var

string emp\_name\_var

// Execute the SELECT statement with

// the current value of sle\_1.text.

OPEN emp\_curs;

// At this point, if there are no errors,

// the cursor is available for further

// processing.

#### Fetching rows

In the singleton SELECT, you specify variables to hold the values for the columns within the selected row. The FETCH statement syntax is similar to the syntax of the singleton SELECT. Values are returned INTO a specified list of variables.

This example continues the previous example by retrieving some data:

// Go get the first row from the result set.

FETCH emp\_curs INTO :emp\_id\_var, :emp\_name\_var;

If at least one row can be retrieved, this FETCH places the values of the emp\_id and emp\_name columns from the first row in the result set into the PowerScript data variables emp\_id\_var and emp\_name\_var. Executing another FETCH statement will place the variables from the next row into specified variables.

FETCH statements typically occur in a loop that processes several rows from a result set (one row at a time): fetch the row, process the variables, and then fetch the next row.

Note**What happens when the result set is exhausted?** FETCH returns +100 (not found) in the SQLCode property within the referenced transaction object. This is an informational return code; -1 in SQLCode indicates an error.

#### Example

This cursor example illustrates how you can loop through a result set. Assume the default transaction object (SQLCA) has been assigned valid values and a successful CONNECT has been executed.

The statements retrieve rows from the employee table and then display a message box with the employee name in each row that is found.

// Declare the emp\_curs.

DECLARE emp\_curs CURSOR FOR

SELECT emp\_name FROM EMPLOYEE

WHERE emp\_state = :sle\_1.text;

// Declare a destination variable for employee

// names.

string    emp\_name\_var

// Execute the SELECT statement with the

// current value of sle\_1.text.

OPEN emp\_curs;

// Fetch the first row from the result set.

FETCH emp\_curs INTO :emp\_name\_var;

// Loop through result set until exhausted.

DO WHILE SQLCA.sqlcode = 0

// Pop up a message box with the employee name.

MessageBox("Found an employee!",emp\_name\_var)

// Fetch the next row from the result set.

FETCH emp\_curs INTO :emp\_name\_var;

LOOP

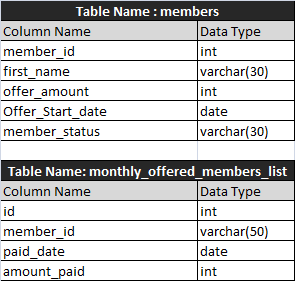
Sybase error codes 0 means no errors

@error<>0

<http://binabdu-fayaz.blogspot.com/2009/06/some-t-sql-interview-questions-and.html>

**1. How to use sql query to copy only structure?**  
Ans: select \* into table2 from table1 where 1 = 2  
  
**2. How do we handle Error?**  
Ans: I think we can use @@Error. Right after the query condition is executed we can check for @@Error <> 0, if it is not returning zero mean some error occured. Raiserror is another command for raising error We can also use Try catch block

**3. What is PatIndex?**  
Ans: Returns the starting position of the first occurrence of a pattern in a specified expression, or zeros if the pattern is not found  
Syntax - PATINDEX ( '%pattern%' , expression )  
Eg: USE AdventureWorks;  
USE AdventureWorks;  
GO  
SELECT PATINDEX('%ensure%',DocumentSummary)  
FROM Production.Document  
WHERE DocumentID = 3;  
GO  
  
**4. How to query a string contains %?**  
Ans: SELECT Name FROM tblPlayer WHERE Name Like '%[''%'']'  
  
**5. How to get values from a table with comma seperated?**  
Ans: declare @vName nvarchar(100)  
set @vName = ''  
select @vName = @vName + ','+ [Name] from HumanResources.Shift  
select @vName  
  
**6. How to update 'Yes' to 'No' and viceversa in a query?**  
Ans: Update tablename set ColumnName1 = (case ColumnName1 when 'Yes'  
then 'No'else 'Yes' end)  
  
**7. Consider you have a table with columns ID(primary key), Country and State.**  
**Now if you have some rows with combination of country and state repeating,**  
**ie, two rows with combination India, Kerala. Write a query for deleting**  
**duplicate records?**  
  
Ans: With T1 as  
(Select \*,Row\_Number() over (partition by Country, State order by ID)   
as 'RowNo' From TableName)  
Delete from T1 where RowNo > 1;  
  
**8. How to create temporary table? How do we apply noncluster index? What is nolock? When and where is nolock applied normally?**  
  
Ans. Two ways of creating temporary table with non clusterindex applied on it. Also example shows how to apply "nolock". nolock is normally applied while querying on production servers. This would make the records being queried sharable on the table. ie, will not prevent other queries from querying the same record parallely on same table. The risk will be nolock might return junk data some times because the select query might be querying the table while some other insertion or updation commands are performed on the table.  
  
1.  
CREATE TABLE #tmpTable  
(  
OfficeName varchar(50)   
, officeid int   
, CustID int   
, AgentID int  
, mlsid varchar(4)  
, RequestMoreDetails int null   
, Emails int null  
)  
CREATE NONCLUSTERED INDEX #IX\_DW\_Listings ON #DW\_AllListings(AgentID)  
  
2.  
select   
OfficeName   
, officeid   
, o.CustID   
, AgentID -  
, o.mlsid   
, PrintBrochure\_Views = null   
, RequestMoreDetails = null   
, Emails = null   
into #ForOffices from #Offices o   
LEFT JOIN dbo.planparts WITH (NOLOCK)  
ON bppa.officeid = o.RID   
CREATE NONCLUSTERED INDEX #IX\_DW\_Listings ON #ForOffices(AgentID)  
  
**9. Another simple example for using temporary table.**  
**Also how to split a single column in to multiple columns based on column value.**  
**Here date\_format function format date to yyyymm format.**  
**period\_diff function returns number of months between two date parameters passed. Also parameter expect value in either yymm or yyyymm format.**

[](http://1.bp.blogspot.com/-Z8kfhnLFB40/TwlBhw0Bk0I/AAAAAAAACvw/srqwzCC6SJY/s1600/Table+List.png)

**IN MYSQL**

CREATE TEMPORARY TABLE monthlyoffer( MemberName VARCHAR(50) ,January INT DEFAULT 0 ,February INT DEFAULT 0 ,March INT DEFAULT 0 ,April INT DEFAULT 0 ,May INT DEFAULT 0 ,June INT DEFAULT 0,July INT DEFAULT 0 ,August INT DEFAULT 0 ,September INT DEFAULT 0 ,October INT DEFAULT 0 ,November INT DEFAULT 0 ,December INT DEFAULT 0 ,PaidDate date ,CurrentYearPaid INT ,TotalPaid INT ,Balance INT);

Insert into monthlyoffer (MemberName ,January,February,March,April,May,June,July,August,September,October,November,December,PaidDate,CurrentYearPaid ,TotalPaid,Balance)

Select

  mem.first\_name,

  sum(case when  monthname(paid\_date)='January' and Year(mo.paid\_date) = 2012 then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='February' and Year(mo.paid\_date) = 2012 then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='March' and Year(mo.paid\_date) = 2012 then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='April' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='May' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='June' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='July' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='August' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='September' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='October' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='November' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  sum(case when  monthname(paid\_date)='December' and Year(mo.paid\_date) = 2012  then mo.amount\_paid else 0 end),

  max(paid\_date),

  sum(case when Year(mo.paid\_date) = 2012 then mo.amount\_paid else 0 end),

  sum(mo.amount\_paid),(mem.offer\_amount\*(1+PERIOD\_DIFF(DATE\_FORMAT(Now(), '%Y%m'),DATE\_FORMAT(mem.offer\_start\_date, '%Y%m'))))-sum(case when mo.amount\_paid is null then 0 else mo.amount\_paid end)

from members mem

  left join  monthly\_offered\_members\_list mo on mo.member\_id =  mem.member\_id and mo.approved = 1

where mem.offer\_amount is not null and member\_status = 'Active' and Year(mem.offer\_start\_date) <= 2012

group by mem.first\_name;

Select \* from monthlyoffer;

|  |
| --- |
| **In the table below, how would you retrieve the unique values for the employee\_location without using the DISTINCT keyword?** |
|  |
| |  | | --- | | employee | | |  |  | | --- | --- | | **employee\_name** | **employee\_location** | | Joe | New York | | Sunil | India | | Alex | Russia | | Albert | Canada | | Jack | New York | | Alex | Russia | |   We can actually accomplish this with the GROUP BY keyword. Here’s what the SQL would look like:   |  | | --- | | SELECT employee\_location from employee  GROUP BY employee\_location |   Running this query will return the following results:   |  | | --- | | **employee\_location** | | New York | | India | | Russia | | Canada |   So, you can see that the duplicate values for "Russia" and "Canada" are not returned in the results. |

<http://sybaseblog.com/sybasewiki/index.php?title=What_is_locking_and_how_many_types_of_locking_in_SYBASE_ASE%3F>

# What is locking and how many types of locking in SYBASE ASE?

Whenever any kind of operation perform in tables, SYBASE ASE implement a locking scheme in that table.So that other operation could not affect current operation and consistency of data in that table. While locking prevent another processes from accessing process data which some time leading blocking and deadlock.

Locking scheme affects tables in SYBASE ASE in three ways.

1. **Row level Locking**
2. **Page Level Locking**
3. **Table/All page Level Locking**

**Datarows locking** which locks only the data rows.Index rows and pages are not locked.

**Datapages locking** which locks only the data pages.

**Allpages locking** which locks datapages and index pages.

#### Page / Row level locking

There are three type of lock is made on page/row level locking :

**1.Shared locks**Shared lock is applied in a table for read operation . Many read transaction can be perform in a table or page while data modification will not allowed during shared lock. Data modification transaction wait for release existing shared lock.

**2.Exclusive locks**Exclusive lock applies during data modification. When a transaction acquire exclusive lock , no other transaction can acquire exclusive lock. Other transaction wait for releasing existing exclusive lock.

**3.Update locks** – Initial phase of data modification, ASE applies update lock in a data. Update lock allow for shared lock, while does not allow for exclusive lock.

#### Table level Locking

**1.Intent lock –**If any transaction is performing with page-level and row-level lock, and it require other row/page level lock for completion of transaction. Adaptive Server applies an intent table lock with each shared or exclusive page or row lock.

**2.Shared lock –** similar to a shared page or row lock, except that it affects the entire table. A create no clustered index command also acquires a shared table lock.

**3.Exclusive lock –** similar to an exclusive page or row lock, except that it affects the entire table.