SQL-LAB03-DBS201-CREATE

**Purpose:**

* **To create a collection (or database)**
* **To create tables for your collection using SQL**
* **To designate an attribute(s) as primary key to a table**
* **To apply a few constraints to the columns in a table**
* **To insert rows of data into the tables**
* **To see the effect of constraints on data**

 </div> <div class="MsoNormal">

</div> <div class="MsoNormal">**You should be signed on and in SQL using iNavigator.**</div>

This may be a long lab, so please get started on it. It can be done in several stages.



## 1 CREATE A COLLECTION

In this step you will create a **collection** or **database** similar to the Premiere collection.  On the iSeries platform a collection is similar to a library or a database with other RDMS products.  All collection or library names must be unique.

(NOTE: If you attempt to create a collection with the name PAYROLL and it is already defined in the system catalog you would get the following message:  **PAYROLL in \*N type \*LIB already exists**.  You would also need proper authority in order to see this PAYROLL collection that someone else has created.)

A maximum of 10 characters can be used in a collection name.

Have you done the global replace yet? You should. It will make it easier for you.

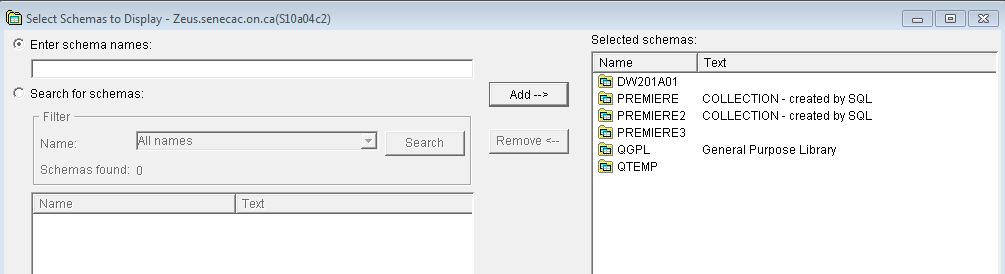
Type

**CREATE   COLLECTION PREMA01;** (where A01 should have been replaced by the last three characters of your ID) and run the query.

You can't see the collection yet. You will need to make it visible later on the page.

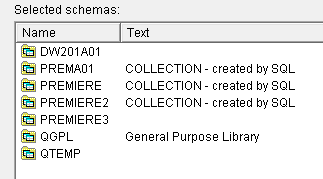
The creation of your collection may take several minutes. Some of the time used in creating a collection is devoted to the creation and setup of some special objects called a **journal** and **journal receiver**.  These objects record the before image and after image of any record that is changed in the collection. This allows recovery of the database due to hardware or software failures and provides an audit trail when investigating unauthorized changes to tables.

In order to see the collection you will need to **“Select schemas to display**” from bottom right corner of iSeries navigator window and this will appear.

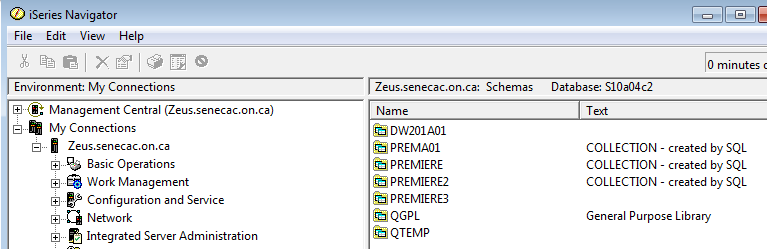


Enter **PREMA01** and add it to the selected schemas.

It will appear in the right pane



It will also appear in the iNavigator right pane as well



## 2 DROP A COLLECTION

If you want to **delete** or remove your newly created collection entirely, you would use the **DROP** command as follows:

## DROP COLLECTION PREMA01

<div class="MsoNormal">Again, you may have a bit of a wait. Also note this is different than removing it from the pane which you might do if there were lots of collections that you were not currently working on and you wanted to keep those out of view.

After it is dropped you will get a message confirming the action.

You may have to refresh the iNavigator screen to see that it is gone or you may have to log off and log n again.

Now put the collection back by creating it again. Remember you can't see it unless you select schemas to display.

 </div>

## 3 CREATE A TABLE

To create a few simple tables you will need to enter the following command

<div class="MsoNormal">You are going to create a few test tables and then two tables called CUSTOMER and SALESREP

Both SALESREP and CUSTOMER need to have the same column definitions in your collection as the ones in the PREMIERE collection. This will make copying of the data easier in a later lab.</div>

Unfortunately as of November 2010 iNavigator does not support the SQL DESCRIBE statement which would make things easier for us to see how the table structure was created.

## How to create a table

Ask these questions.

What do you want to do? ➔ CREATE

What do you want to create? ➔ TABLE

What name? ➔ CUSTOMER

The statement that does this is CREATE TABLE PREMA01.CUSTOMER, but with tables it is necessary to also define the columns in the table.

The next set of questions to answer is as follows:

What is the data called (column names)

What is the type of data and what is the length of the data in that column?

Finally, what constraints do you want to apply to ensure data integrity?

Example 1: Here is an example of a simple table with no constraints. You can cut and paste it, but you will not get any practice by doing that.

**CREATE TABLE PREMA01.TESTTABLE**

**(TESTID INT,**

**TESTNAME CHAR (15)**

**);**

 Enter the following:

**SELECT \***

**FROM PREMA01.TESTTABLE;**  
    
 Did you see this screen?



Since there is no data loaded into the table only column headings are displayed.

Example 2: An example applying some restrictions on the data that can be entered

**CREATE TABLE PREMA01.TESTTABLE2**

**(TESTID INT NOT NULL,**

**TESTNAME CHAR (15) NOT NULL WITH DEFAULT,**

**TESTSCORE DECIMAL (5, 2) NOT NULL WITH DEFAULT 1.25**

**);**

In example 2 the constraint on the data is that the INSERT statement to insert data can not enter a nothing value in the TESTID column. The NOT NULL constraint forces data to be entered at INSERT time. This makes sense as there needs to be an ID for the row. If this was a C program you would have to write the IF statement to check that data was entered. This is done for you at table definition time. Also it checks that the data is an integer.

The same applies to TESTNAME, but in this case if there is no data entered then a default value is inserted into the column. In the case of numeric data the default would be 0 (zero) and for character data it would be space.

Note that in the third column TESTSCORE the default is 1.25 if no data is provided.

Example 3: Defining a PRIMARY KEY

**CREATE TABLE Prema01.TESTTABLE3**

**(TESTID INT PRIMARY KEY,**

**TESTNAME CHAR(15) NOT NULL WITH DEFAULT 'Unknown',**

**TESTSCORE DECIMAL (5,2) NOT NULL WITH DEFAULT 0.00**

**);**

Notice in this example if a name is not entered at insert time then the value Unknown is stored.

Example 4: Adding more constraints and data types. Just cut and paste these tables one at a time..

**CREATE TABLE PREMA01.STUDENT**

**(SID INT PRIMARY KEY,**

**SNAME CHAR(15) NOT NULL**

**);**

Notice the comma at the end of each column definition except the last one of course. For the above a PRIMARY KEY constraint was applied. The PRIMARY KEY is the same as saying UNIQUE and NOT NULL which are constraints that you can apply to any of the attribute names.

We can also define the constraints at the end of the create table process and also after the table has been created.

Example 5: In this example the primary key is defined after all the attributes were defined.

**CREATE TABLE PREMA01.COURSE**

**(CID INT,**

**CNAME CHAR(15) NOT NULL,**

**CONSTRAINT PREMA01.COURSE\_PK**

**PRIMARY KEY (CID)**

**);**

Example 6: In this case because it is a composite key the constraint must be applied at the end. If the Primary key was applied to each attribute as it was defined the CID would generate an error as the primary key was already defined to apply to SID in the table below.

**CREATE TABLE PREMA01.SCORE**

**(SID INT,**

**CID INT,**

**CONSTRAINT PREMA01.SCORE\_PK**

**PRIMARY KEY (SID, CID)**

**);**

## 4 DROP A TABLE

Get rid of the table from the collection

Example: **DROP TABLE PREMA01.TESTTABLE2**

Check to see if it has gone by refreshing the iSeries Navigator window

(Note: If you can not drop the table because of a message the table is in use, then you probably have done a select on the table. You need to clear the history so the table can be dropped. Select Edit – Clear results)

## 5 ALTER TABLE

Remember ALTERing the table is about changing the structure of the table and the constraints and **NOT the data**.

Now set up the FOREIGN KEY constraint. You can do this when the table is being created, or it can be added after. You have already created a table and deleted (dropped) a table, the only thing else you can do with a table is modify it. This next step modifies a table.

Adding a constraint

First ask what you want to do ➔ alter a table

Next ask what altering are you going to do? ➔ ADD or DROP

➔ Constraint or column

**EXAMPLES:** Do not enter them as there is no CUSTOMER table to alter

Adding a foreign key constraint

ALTER TABLE PREMA01.CUSTOMER

ADD CONSTRAINT PREMA01.CUST\_SALES\_REP\_FK

FOREIGN KEY (SALES\_REP\_NUMBER)

REFERENCES PREMTARR.SALESREP (SALES\_REP\_NUMBER)

ON DELETE NO ACTION ← you can ignore these 2 lines

ON UPDATE NO ACTION ←

The table score looks like this SCORE [SID (fk), CID (fk), MARK]

There is a need to add the foreign key constraints

**ALTER TABLE PREMA01.SCORE**

**ADD CONSTRAINT PREMA01.SCORE\_CID\_FK**

**FOREIGN KEY (CID)**

**REFERENCES PREMA01.COURSE (CID);**

**ALTER TABLE PREMA01.SCORE**

**ADD CONSTRAINT PREMA01.SCORE\_SID\_FK**

**FOREIGN KEY (SID)**

**REFERENCES PREMA01.STUDENT (SID);**

You tell it what to do – ALTER a table

You tell it what type of altering you want – ADD a constraint

Since it is a foreign key you need to tell the system what field to apply it to (SID) and what is the parent table and the field in that table that it must check before allowing the data to be inserted.

Adding a column

Example syntax—do NOT enter it:

ALTER TABLE PREMA01.CUSTOMER

ADD COLUMN PREMA01.COMM\_RATE DECIMAL (3, 2)

When you add a column you need the type and length of the data to be stored just as in a create table

Alter the SCORE table to add a column called mark

**ALTER TABLE PREMA01.SCORE**

**ADD COLUMN MARK DECIMAL (3, 1);**

Do not do this next command.

There is some difficulty removing the column sometimes in the interactive mode.

Add another column

ALTER TABLE PREMA01.SCORE

ADD COLUMN comm\_rate DECIMAL (3, 1);

Do a select to see it was added

Removing a column

You can delete a column using the DROP COLUMN clause of the ALTER TABLE statement.

ALTER TABLE PREMA01.SCORE

DROP COLUMN COMM\_RATE;

Do a select to see that it was removed.

If there was data in the column (there isn't in our tables yet) all the data would be gone as well. Adding, changing and dropping tables is not something the company wants to do a lot of with its day-to-day data. That is why it is important to design it properly in the first place. However sometimes changes need to be made. Also, you will be working in the developing side of the business and you will need to add, change and delete tables all the time.

It is a good idea to do a SELECT \* to look at what the table contains and what columns exist now before doing a DROP.

# DATA IN THE TABLE

## 6 Adding data to a table

Since this table is not built yet you can not insert the data. This is an example for discussion only.

Insert data into table example.

INSERT INTO PREMA01.CUSTOMER VALUES

('100', 'LAST', 'FIRST','100 STREET','TORONTO', 'ON', 'M9W1A4', 5000, 1234.56, '6')

Notice

1. That the first value is in single quotes. This must be because the first column was defined originally as character data when the table was built. It would be better to define keys as numeric. Just like any other language character data is in quotes. For SQL, character data is in single quotes. Only the alternate name for columns in the select clause has double quotes.

2. The values entered must be in the same order and data type as the table.

3. If the column allows NULL, then NULL must be used as in the example below.

INSERT INTO PREMTARR.CUSTOMER VALUES

('100', 'LAST', 'FIRST', NULL,'TORONTO', NULL, 'M9W1A4', 5000, 1234.56, '6');

NULL is not a character value and does not need quotes. If you were to insert this row of data it will not work for 2 reasons

1) A primary key would have been defined on the CUSTOMER table and the previous INSERT already used '100'

2) Last name and province were defined as not allowing NULL values.

**Deleting data from the table**

**All the data**

DELETE FROM PREMA01.CUSTOMER ← Since there was no condition all the rows were deleted

**Some of the rows**

Delete only one of the rows by adding a condition.

DELETE FROM PREMA01.CUSTOMER

WHERE CUSTOMER\_NUMBER = '100';

**Changing data in a table**

UPDATE PREMA01.CUSTOMER

SET STREET = 'Carnation Rd'

WHERE CUSTOMER\_NUMBER ='100' ;

If there is not a WHERE condition, then all rows will be updated to the same address.

Now we will insert some data into the previously created tables, STUDENT, COURSE and SCORE.

INSERT without specifying the column

**INSERT INTO PREMA01.STUDENT**

**VALUES (100, 'Ron Tarr');**

**INSERT INTO PREMA01.STUDENT**

**VALUES (200, 'Jane Barber');**

**INSERT INTO PREMA01.STUDENT**

**VALUES (300, 'Peter Piper');**

If you don't specify the column then the data must match what the table description requires.

INSERT with specifying the column

**INSERT INTO PREMA01.STUDENT (SNAME, SID)**

**VALUES ('Ron Tarr2', 400);**

Notice that the order of the data being inserted did not matter since the name of the columns was specified in the same order. Of course this would not be a normal practice. Naming of the columns would occur if a table had 30 columns and only 4 were specified as NOT NULL and the insert only had those 4 pieces of data.

Check the data has been entered

SELECT \*

FROM PREMA01.STUDENT;

Testing the constraints by trying to insert incorrect data

**INSERT INTO PREMA01.STUDENT**

**VALUES ('500', 'Ron Tarr');**

Notice this inserted the data inside the quotes. This is the same as in C programming. It is called CASTING.

Try these

**INSERT INTO PREMA01.STUDENT**

**VALUES ('ABC', 'Ron Tarr');**

This did not allow casting. ABC as character data could not be converted to a number like the characters 123 could be in the previous example. So casting has to make sense.

INSERT INTO PREMA01.STUDENT

VALUES (NULL, 'Ron Tarr');

Error: INSERT or UPDATE not allowed by CHECK constraint. Cause . . . . . :

The value being inserted or updated does not meet the criteria of CHECK constraint. The data can not be NULL in the first column.

Now add data to the COURSE table.

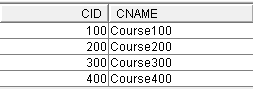
**INSERT INTO PREMA01.COURSE VALUES (100, 'Course100');**

**INSERT INTO PREMA01.COURSE VALUES (200, 'Course200');**

**INSERT INTO PREMA01.COURSE VALUES (300, 'Course300');**

**INSERT INTO PREMA01.COURSE VALUES (400, 'Course400');**

Your output should look like this



Insert data into the score table.

**INSERT INTO PREMA01.SCORE VALUES (100, 200, 66);**

-- inserting student 100 into course 200

You must have a value for every column in the table.

Try this

**INSERT INTO PREMA01.SCORE VALUES (400, 200);**

Error: Statement contains wrong number of values

Try this

**INSERT INTO PREMA01.SCORE VALUES (400, 200, NULL);**

In this case there were sufficient number of values being entered and the 3rd column allowed NULL

The select \* should show the following



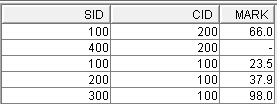
Insert some additional data into SCORE

**INSERT INTO PREMA01.SCORE VALUES (100, 100, 23.5);**

**INSERT INTO PREMA01.SCORE VALUES (200, 100, 37.9);**

**INSERT INTO PREMA01.SCORE VALUES (300, 100, 98);**

Run a select on SCORE and see that it looks like this



Notice that the rows appear in the order they were first stored. When displaying data humans tend to like things in some ORDER.

Enter the following:

**SELECT \***

**FROM PREMA01.SCORE**

**ORDER BY SID;**

**Testing the PK constraint**

Since there is already an entry for student 100 course 200 we will try to insert another row with the same primary key

**INSERT INTO PREMA01.SCORE VALUES (100, 200, 75.99);**

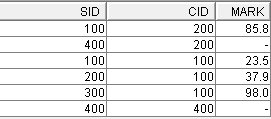
ERROR: The operation cannot be performed because one or more values would have produced a duplicate key in the unique index or constraint.

**Inserting only SOME values**

In order to do this or to insert values in another order than the way the columns are stored means you must specify the column names.

**INSERT INTO PREMA01.SCORE (SID, CID)**

**VALUES (400, 400);**



The other way was to not name the columns but use NULL instead as we did on the previous page.

**INSERT INTO PREMA01.SCORE (CID, SID)**

**VALUES (300, 400);**

ASIDE: If you do the select it will be the last row added. Because of the key values chosen in both tables of 100, 200 etc it is hard to tell. It might have been better when testing to use 900, 910, 920 etc for courses to see the difference.

**Testing the default values**

You should still have a table that was created by the following SQL

CREATE TABLE PREMA01.TESTTABLE3

(TESTID INT PRIMARY KEY,

TESTNAME CHAR(15) NOT NULL WITH DEFAULT 'Unknown',

TESTSCORE DECIMAL (5,2) NOT NULL WITH DEFAULT 0.00 );

Insert the following data

**INSERT INTO PREMA01.TESTTABLE3 (TESTID)**

**VALUES (111);**

You should see the following after issuing a select command



Notice that since nothing was entered for TESTNAME the value Unknown was inserted. The value 0.00 was also the default value if nothing was entered for TESTSCORE.

## 7 Changing Data

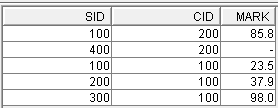
Change the mark for SID 100 and CID 200 from 66 to 85.8

**UPDATE PREMA01.SCORE**

**SET MARK = 85.8**

**WHERE SID = 100**

**AND CID = 200;**



Notice that it was necessary to use an AND condition since there were two entries for student 100.

**Test foreign key restraints** on the SCORE table as follows:

Try to insert CID 999 which does not exist in the Course table

**INSERT INTO PREMA01.SCORE VALUES (100, 999, 75.9);**

Error: Operation not allowed by referential constraint SCORE\_CID\_FK in PREMA01

It even tells you what constraint effects thee insert. The reason why we use a naming convention for constraints is so that it becomes obvious what the problem is. The table SCORE, the column CID has a foreign key constraint FK → *SCORE\_CID\_FK* in PREMA01

SUBMISSION:



**1 Create the SALES\_REP table as follows**

**CREATE TABLE PREMA01.SALES\_REP**

**( SID NUMERIC (2) PRIMARY KEY CHECK (SID > 10),**

**LAST VARCHAR (20) NOT NULL,**

**FIRST VARCHAR (20) NOT NULL,**

**STREET VARCHAR (60) NOT NULL,**

**CITY VARCHAR (20) NOT NULL with default 'Toronto',**

**PROV CHAR (2) NOT NULL with default 'ON',**

**PCODE CHAR (6) NOT NULL,**

**TOTAL\_COMM DECIMAL (7,2) CHECK (TOTAL\_COMM >= 0.00),**

**COMM\_RATE DECIMAL (3,2) CHECK (COMM\_RATE BETWEEN 0.01 AND 0.05)**

**);**

**2 Insert this data – One at a time so that you can see what happens.**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (10, '10LAST', '10FIRST', '10 TEN STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.01);**

Did it work? If not why didn't it work? **NO. SID IS 10**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (11, '11LAST', '11FIRST', '11 ELEVEN STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.02);**

Did it work? If not why didn't it work? **YES**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (12, '12LAST', '12FIRST', '12 TWELVE STREET', 'TORONTO', 'ON' ,'M9W3S4', 0.0, 0.02);**

Did it work? If not why didn't it work? **YES**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (11, '11LAST', '11FIRST', '11 ELEVEN STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.02);**

Did it work? If not why didn't it work? **NO. DUPLICATE**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (21, '21LAST', '21FIRST', '21 TWO-ONE STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.07);**

Did it work? If not why didn't it work? **NO. COMM\_RATE HAS TO BE BETWEEN 0.01 AND 0.05**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (21, '21LAST', '21FIRST', '21 TWO-ONE STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.05);**

Did it work? If not why didn't it work? **YES**

**INSERT INTO PREMA01.SALES\_REP**

**VALUES (12, '12LAST', '12FIRST', '12 TWELVE STREET', 'TORONTO', 'ON','M9W3S4', 0.0, 0.02);**

Did it work? If not why didn't it work? **NO. DUPLICATE**

**3 Using the chart below create the CUSTOMER table.**

We now want to create the CUSTOMER table using the following chart that is from the entity attribute list in the design phase.

Given the following from our design

CUSTOMER [**CID**, LAST, FIRST, STREET, CITY, PROV, PCODE, BALANCE, CREDIT\_LIMIT, **SID**]

The next step is to use the following chart to

Describe the layout of each table to be contained in the database

**Physical Schema** shown below

Table name: CUSTOMER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column | Data Type | SIZE | PK | FK and reference | NN | UK | CK and  Default |
| CID | NUMERIC | 3 | Y |  |  |  |  |
| LAST | VARCHAR | 20 |  |  | Y |  |  |
| FIRST | VARCHAR | 20 |  |  | Y |  |  |
| STREET | VARCHAR | 60 |  |  | Y |  |  |
| CITY | VARCHAR | 20 |  |  | Y |  | Toronto |
| PROV | CHAR | 2 |  |  | Y |  | ON |
| PCODE | CHAR | 6 |  |  | Y |  |  |
| BALANCE | DECIMAL | 7.2 |  |  |  |  |  |
| CREDIT\_LIMIT | DECIMAL | 7.0 |  |  |  |  |  |
| SID | NUMERIC | 2 |  | SALES\_REP (SID) | Y |  |  |

CREATE TABLE PREME19.CUSTOMER(

CID NUMERIC (3) PRIMARY KEY,

LAST VARCHAR (20) NOT NULL,

FIRST VARCHAR (20) NOT NULL,

STREET VARCHAR (60) NOT NULL,

CITY VARCHAR (20) NOT NULL with default 'Toronto',

PROV CHAR (2) NOT NULL with default 'ON',

PCODE CHAR (6) NOT NULL,

BALANCE DECIMAL (7,2),

CREDIT\_LIMIT DECIMAL (7,0),

SID NUMERIC (2) NOT NULL

);

ALTER TABLE PREME19.CUSTOMER

ADD CONSTRAINT PREME19.CUSTOMER\_SID\_FK

FOREIGN KEY (SID)

REFERENCES PREME19.SALES\_REP (SID);

**4 Insert the following data**

**INSERT INTO PREMA01.CUSTOMER**

**VALUES (661,**

**'Last 661',**

**'First Name 661',**

**'70 Pond Street',**

**'Toronto',**

**'ON',**

**'M9W1A9',**

**1000,**

**1000,**

**12);**

Notice that it might be easier if you were doing a series of inserts to see that you have data for every column by writing it this way.

**INSERT INTO PREMA01.CUSTOMER**

**VALUES (662, 'Last 662','First Name 662','662 Pond Street','Toronto','ON','M9W1A9',550.00,3000,12);**

**INSERT INTO PREMA01.CUSTOMER**

**VALUES (663, 'Last 663','First Name 663','663 Pond Street', 'Toronto', 'ON', 'M9W1A9', 155.27, 1000, 12);**

**INSERT INTO PREMA01.CUSTOMER**

**VALUES (664, 'Last 664','First Name - 664','664 Pond Street', 'Toronto', 'ON', 'M9W1A9', 5000, 3000, 12);**

**INSERT INTO PREMA01.CUSTOMER**

**VALUES (665, 'Last 665','First Name - 665','665 Pond Street', 'Toronto', 'ON', 'M9W1A9', 1123.75, 10000, 11);**

**5 Enter the following**

**SELECT CID, FIRST||', '||LAST, SID**

**FROM PREMA01.CUSTOMER;**

