

# SQL FOR DB2

## Chapter 12 Updating Tables

# CHAPTER OBJECTIVES

- ① Demonstrate the use of the `INSERT` statement
- ① Use a `SELECT` statement to insert rows
- ① Demonstrate the use of the `UPDATE` statement
- ① Demonstrate the use to the `DELETE` statement

# THE INSERT STATEMENT

Using the  
**INSERT**  
to add a  
new row  
to a table

```
Insert Into Customer
      ( CustID,
        Name,
        ShipCity,
        Discount )
      Values ( 678987,
              'Atlas Inc.',
              '',
              Null );
```

# DEFAULT KEYWORD

Using the  
DEFAULT  
keyword  
to set the  
default  
value of a  
column

```
Insert Into Customer
      ( CustID,
        Name,
        ShipCity,
        Discount )
      Values ( 678987,
              'Atlas Inc.',
              '',
              Default )
```

# INSERTING DATA FROM OTHER TABLES

```
Insert Into Customer
    ( CustID,
      Name,
      ShipCity,
      Discount,
      ShipState )
Select CustID,
      Name,
      ShipCity,
      Discount,
      ShipState
From CustOld
```

# THE UPDATE STATEMENT

Updating a  
row using  
its primary  
key value  
in the  
**WHERE**  
clause

```
Update Customer
  Set Name    = 'Wood Products',
      Discount = Discount + .02
Where CustID = 905011
```

# UPDATING A SET OF ROWS

Updating a  
set of rows  
using a  
search  
condition  
in the  
**WHERE**  
clause

```
Update Customer
  Set Discount = .10
  Where ShipCity = 'Portland'
```

# SETTING A COLUMN TO NULL

```
Update Customer  
  Set Discount = Null  
  Where ShipCity = 'Portland'
```

# SETTING A COLUMN TO ITS DEFAULT VALUE

```
Update Customer  
Set Discount = Default  
Where ShipCity = 'Portland'
```

# UPDATING A COLUMN USING A SUBSELECT

```
Update Customer
  Set CreditLimit =
    (Select Avg(TotAmt) * 1.10
     From Sale)
```

# GROUPING COLUMN NAMES AND VALUES

```
Update Customer
  Set ( CustID,
        Name,
        ShipCity,
        Discount ) =
  ( 905011,
    'Wood Products',
    'Roseburg',
    Discount = Discount + .02)
Where CustID = 905011
```

# THE DELETE STATEMENT

```
Delete  
  From Customer  
  Where CustID = 905011
```

# DELETING A SET OF ROWS

```
Delete  
  From Customer  
  Where ShipCity = 'Portland'
```

# CLEARING ALL ROWS IN A TABLE

```
Delete  
From Customer
```

# CODING SUGGESTIONS

- ⦿ Be sure that a subselect used on the right-hand side of the SET clause in an UPDATE statement can never have more than one row.
- ⦿ Use an explicit list of columns in the INSERT statement to make clear which column each new value corresponds to.
- ⦿ Be careful to include a WHERE clause in an UPDATE or DELETE statement unless you intend to update or delete all rows in the table.
- ⦿ Whenever updating or inserting a row into a table, be certain to adhere to check constraints defined on the table.
- ⦿ Be sure to consider the effect of primary key, unique, and foreign key constraints when updating a primary key, unique, or foreign key column or when deleting a row in a table referenced by a foreign key.

# CHAPTER SUMMARY

- ③ Three SQL statements are used to manipulate table rows:
  - ③ INSERT
  - ③ UPDATE
  - ③ DELETE

# CHAPTER SUMMARY - INSERT

- ◎ The INSERT statement
  - ◎ Can insert a single row using the VALUES clause
  - ◎ Can insert multiple rows using a nested form of the SELECT statement

# CHAPTER SUMMARY - UPDATE

- ◎ The UPDATE statement
  - ◎ Changes the contents of one or more columns by specifying new values in a SET clause
  - ◎ The right-hand side of a SET clause assignment can be any expression, including a scalar subselect
  - ◎ Variations of the SET clause use column lists and/or a list of expressions or a multicolumn subselect.
  - ◎ Can update all rows from a table when the statement is specified with no WHERE clause

# CHAPTER SUMMARY - DELETE

- ◎ The DELETE statement
  - ◎ Can remove a single row from a table
  - ◎ Can remove multiple rows using a search condition that specifies more than one row
  - ◎ Can clear all rows from a table when the statement is specified with no WHERE clause