LESSON 3.4

98-364 Database Administration Fundamentals

Delete Data

Lesson Overview

3.4 Delete data

In this lesson, you will review:

- `DELETE FROM`
- `TRANSACTIONS`
- `ROLLBACK`
- `COMMIT`

DELETE FROM

- The `DELETE` statement is used to delete rows in a table:

```
DELETE FROM table_name
WHERE column_name=variable
```

Note: the `WHERE` clause in the `DELETE` statement specifies which record or records should be deleted. Without the `WHERE` clause, all records will be deleted!

DELETE FROM (continued)

- It is possible to delete all rows in a table without deleting the entire table. *This means that the table structure, attributes, and indexes will be intact:*

```
DELETE FROM table_name
```

or

```
DELETE * FROM table_name
```

Be very careful when deleting records. You cannot undo this statement!

Transactions

- Transactions group a set of two or more statements into a single unit.
- If any of the tasks fail, the entire transaction fails, which prevents damage to the database.
- You can think of transactions as compiling a group of programming lines together.
- A transaction begins with the execution of a SQL-Data statement (`UPDATE/INSERT/DELETE`).
- All subsequent statements until a `COMMIT` or `ROLLBACK` statement become part of the transaction. Execution of a `COMMIT` statement or `ROLLBACK` statement completes the current transaction.
- **`COMMIT`** —if all statements are correct within a single transaction, all changes are recorded to the database.
- **`ROLLBACK`** —the process of reversing changes.

Transactions (continued)

- The simplest transaction in Microsoft® SQL Server® is a single data modification statement:

```
UPDATE authors SET au_fname = 'John'
WHERE au_id = '172-32-1176'
```

- It is an autocommit transaction.
- SQL Server first logs what it's going to do, and then it does the actual UPDATE statement.
- Finally, it logs that it has completed the UPDATE statement.
- If the server fails after a transaction has been committed and written to the log, SQL Server uses the transaction log to “roll forward” or redo that transaction when it restarts.

Transactions (continued)

- To be useful, transactions need to have two or more statements in them.
- These are called explicit transactions:

```
BEGIN TRAN
UPDATE authors SET au_fname = 'John'
WHERE au_id = '172-32-1176'
UPDATE authors SET au_fname = 'Marg'
WHERE au_id = '213-46-8915'
COMMIT TRAN
```

- *Note:* The **BEGIN TRAN** and **COMMIT TRAN** statements start and complete a transaction. Everything inside these statements is considered a logical unit of work. If any statement in the transaction fails, nothing in the database will be changed.

Transactions (continued)

- A transaction can be cancelled if it doesn't do what is expected:

```
BEGIN TRAN

UPDATE authors SET au_fname = 'John' WHERE au_id
  = '172-32-1176'

UPDATE authors SET au_fname = 'JohnY' WHERE city
  = 'Lawrence'

IF @@ROWCOUNT = 10 COMMIT TRAN

ELSE

  ROLLBACK TRAN
```

- *Note:* If `@@ROWCOUNT` (a SQL function) is 10, then the transaction commits; otherwise, it rolls back. The `ROLLBACK TRAN` statement "undoes" all the work since the `BEGIN TRAN` statement. Neither `UPDATE` statement is completed.

Transactions (continued)

- Most user transactions will occur in stored procedures:

```
Create Proc TranTest2  AS

BEGIN TRAN

INSERT INTO [authors]([au_id], [au_lname],
  [au_fname], [phone], [contract]) VALUES ('123-32-
  1176', 'Gates', 'Bill', '800-BUY-MSFT', 1)

IF @@ERROR <> 0 BEGIN ROLLBACK TRAN return 10 END

UPDATE authors SET au_fname = 'Johnzzz' WHERE
  au_id = '172-32-1176'

IF @@ERROR <> 0 BEGIN ROLLBACK TRAN return 11 END

COMMIT TRAN

GO
```

Transactions (continued)

An example with error checking:

```
BEGIN TRAN
INSERT INTO [authors]([authors_id],[authors_lname],
[authors _fname], [phone],[contract])
VALUES ('123-32-1176', 'Gates', 'Bill', '1-800-BUY-MSFT', 1)
IF @@ERROR <> 0 BEGIN ROLLBACK TRAN return 10 END
UPDATE authors SET au_fname = 'Johnzzz' WHERE au_id = '172-
  32-1176'
IF @@ERROR <> 0  BEGIN ROLLBACK TRAN return 11 END
COMMIT TRAN
```

Each statement is checked for failure. If a statement fails, then it rolls back the work performed to that point and uses the `RETURN` statement to exit the stored procedure.

Lesson Review

1. What is the purpose of the `DELETE` command?

2. What is the relationship between `TRANSACTIONS`, `ROLLBACK`, and `COMMIT` ?

3. Why is it important to check for error(s) after every statement?