



جامعة طرابلس كلية تقنية المعلومات



Advanced Databases قواعد البيانات المتقدمة ITSE312

د. عبدالسلام منصور الشريف

a.abdoessalam@uot.edu.ly

المحاضرة السادسة - المعاملات و أوامر التعامل مع البيانات

Transactions & Data Manipulation Language (DML)

Advanced Database Lecture 3

Contents

- ▶ Retrieving Data
 - ▶ INSERT
 - ▶ SELECT INTO
 - ▶ UPDATE
 - ▶ DELETE
 - ▶ Transactions

Advanced Database Lecture 3

Insert into Statement - General Structure

- ▶ Inserts one or more rows into a table. It has two forms as follows:
 - ▶ Without specifying columns.
 - ▶ When adding values to all columns.
 - ▶ Values should be introduced in the same order as the columns in the table.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

- ▶ By specifying columns
 - ▶ The values given must match columns' order, count and data type.
 - ▶ Use ' ' when supplying string values.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, value3, ...);
```



Insert into Statement

```
CREATE TABLE [dbo].[Departments] (
    Id TINYINT NOT NULL IDENTITY(1,1) PRIMARY KEY,
    Title NVARCHAR(50),
    PhoneNo CHAR(10)
```

```
);
GO
```

```
INSERT INTO [dbo].[Departments]
([Title],[PhoneNo])
VALUES
```

```
('General Department','0211158672'),
('Software Engineering','0213458978'),
('Networking','0212348478'),
('Web Technologies','0213515878'),
```

```
-----('Information Systems','0211158322');-----
```



Id	Title	PhoneNo
1	General Department	211158672
2	Software Engineering	213458978
3	Networking	212348478
4	Web Technologies	213515878
5	Information Systems	211158322

INSERT INTO SELECT- General Structure

- ▶ Inserts one or more rows into a table from another table. Both tables must exist.
 - ▶ Without specifying columns.
 - ▶ Columns from both tables should match

```
INSERT INTO table2
  SELECT * FROM table1
  WHERE condition;
```

- ▶ By specifying columns
 - ▶ The columns chosen must match columns' order, count and data type.
 - ▶ Not selected columns will be filled with NULL.

```
INSERT INTO table2 (column1, column2, ...)
  SELECT column1, column2, ...
  FROM table1 WHERE condition;
```



INSERT INTO SELECT

```
CREATE TABLE [dbo].[OtherDep] (
  Id TINYINT ,
  Title NVARCHAR(50),
  PhoneNo CHAR(10)
);
GO
-- Select all columns
INSERT INTO [dbo].[OtherDep]
SELECT * FROM [dbo].[Departments] -- TOP can be used

-- other columns will be null
INSERT INTO [dbo].[OtherDep] (Title)
SELECT Title
FROM [dbo].[Departments]
```



SQL SELECT INTO - General Structure

- ▶ Copies data from one table into a new table. The new table will be created with the column-names and types as defined in the old table. You can create new column names using the AS clause.
 - ▶ Without specifying columns.
 - ▶ All columns will be copied.

```
SELECT *
INTO newtable
FROM oldtable
WHERE condition;
```

- ▶ By specifying columns
 - ▶ Some columns will be copied.

```
SELECT column1, column2, column3, ...
INTO newtable
FROM oldtable
WHERE condition;
```



SQL SELECT INTO

```
-- All columns selected
SELECT *INTO [dbo].[AinZarahStudents]
FROM [dbo].[Students]WHERE Municipality = 'AinZarah';

-- Some columns selected
SELECT Name,BirthDate,GenderINTO [dbo].[PersonalData]
FROM [dbo].[Students]WHERE Semester = 1;
```



UPDATE - General Structure

- ▶ Modifying the existing records in a table.
 - ▶ If WHERE omitted, all records in the table will be updated!

```
UPDATE table
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
ALTER TABLE [dbo].[Departments]
ADD Status NVARCHAR(10) DEFAULT N'NOT ACTIVE';
```

```
UPDATE [dbo].[Departments]
SET Status = N'Active'
WHERE Id = 1;
```



DELETE - General Structure

- ▶ Deletes row(s) from a table permanently.
 - ▶ If WHERE omitted, all records in the table will be deleted!

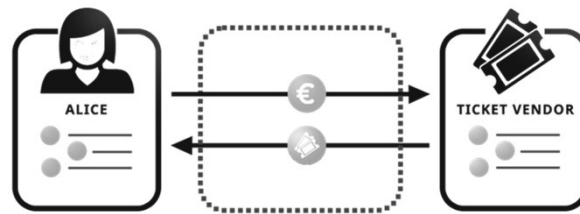
```
DELETE FROM table_name
WHERE condition;
```

```
DELETE FROM [dbo].[AinZarahStudents]
WHERE Gender = 'M'; -- then comment out
```



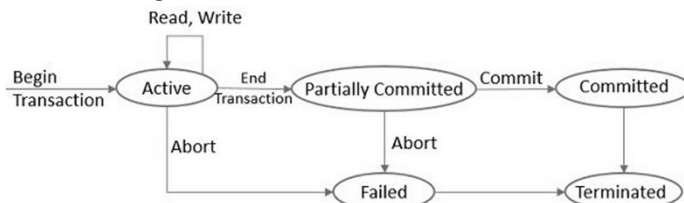
Transactions

- ▶ A transaction is a single unit of work. If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database. If a transaction encounters errors and must be canceled or rolled back, then all of the data modifications are erased.



Transaction States

- ▶ Transaction is atomic that either it should execute entirely or none should execute. Partial execution will affect the consistency of the data. Now whenever a transaction fails, the data affected by the transaction has to be rolled back.
- ▶ For the recovery purpose, the system must be aware that from where the transaction has started, where it has terminated and where it must be committed where it has been aborted. To keep track of the transaction the recovery manager must have to track the following transaction states.



Transaction States

- ▶ **Active State:** As soon as the transaction starts its execution it enters the active state (perform Read and Write).
 - ▶ **Partially Committed State:** When the transaction finishes executing Read and Writes operation it enters the partially committed state. If there is no interrupt (failure) the transaction will enter a committed state.
 - ▶ **Committed State:** The transaction enters the committed state if the transaction is executed successfully without any interruption.
 - ▶ **Failed State:** The transaction enters the failed state either from the partially committed state i.e. if the failure occurs after the execution of the transaction but before it is committed or from the active state i.e. when the transaction is in execution.
 - ▶ **Terminated State:** Entered when the changes done by the transaction are recorded on the database, or canceled in case of failure.
-

Transaction – ACID Test

Before considering a transaction as a successful transaction it must pass the ACID test.

- ▶ **Atomicity:** The atomicity test specifies that either all the transactions are carried out in their entirety or no transactions at all. The partial transaction is not accepted. The transaction failure or the incomplete transaction may be the result of the following reasons.
 - ▶ **System Failure:** hardware error (memory failure), software error (error in coding or logic) or network error (failure of connectivity).
 - ▶ **Transaction Error:** ex. integer overflow or division by the zero, erroneous parameter value, programming error or user has interrupted the transaction execution.
-

Transaction – ACID Test

- ▶ **Consistency:** It is the user's responsibility to maintain the consistency of the database. The user should code the transaction in such a way that it leaves the database in a consistent state after the transaction execution.
 - ▶ **Isolation:** Some transactions execute simultaneously but they must not interfere with each other executing concurrently. The execution of one transaction must not affect the execution of another transaction.
 - ▶ **Durability:** The modifications done to the data present in the database by a committed transaction must persist and must not be lost due to any kind of failure
-



Transaction Modes

- ▶ SQL Server operates in the following transaction modes:
 - ▶ **Autocommit transactions:**
Each individual statement is a transaction.
 - ▶ **Explicit transactions:**
Each transaction is explicitly started with the BEGIN TRANSACTION statement and explicitly ended with a COMMIT or ROLLBACK statement.
 - ▶ **Implicit transactions:**
A new transaction is implicitly started when the prior transaction completes, but each transaction is explicitly completed with a COMMIT or ROLLBACK statement.
-




Transaction – General Structure

```
BEGIN { TRAN | TRANSACTION }
  [ { transaction_name | @tran_name_variable } ]
[ ; ]

COMMIT [ { TRAN | TRANSACTION }
  [ transaction_name | @tran_name_variable ] ]
[ ; ]


ROLLBACK { TRAN | TRANSACTION }
  [ transaction_name | @tran_name_variable ]
[ ; ]
```



Transaction Examples

```
--
DELETE FROM [dbo].[Students]
  WHERE Id = '09223082';
COMMIT;

BEGIN TRANSACTION;
DELETE FROM [dbo].[Students]
  WHERE Id = '09223082';
COMMIT;
```



Transaction Examples

```
--
DELETE FROM [dbo].[Students]
    WHERE Id = '09223082';
COMMIT;
```

```
BEGIN TRANSACTION;
DELETE FROM [dbo].[Students]
    WHERE Id = '09223082';
COMMIT;
```

Transaction Examples

```
BEGIN TRAN AddStudent

INSERT INTO [dbo].[Students] (Id,[Name], [PhoneNo], [Semester],
    [Gender], [BirthDate], [Municipality], [DepartmentId], Enrolled)
VALUES ('09223365','Zaher Fouad Aljosh','+218912123482',2,'F',
    '09-01-2003','TripoliCenter',4,1);

ROLLBACK TRAN AddStudent;

INSERT INTO [dbo].[Students] (Id,[Name], [PhoneNo], Semester],[Gender],
    [BirthDate],[Municipality],[DepartmentId],Enrolled)
VALUES ('09223323','Mohamed Ahemd','+218912123482',2,'F',
    '09-01- 2003','TripoliCenter',4,1);

SELECT * FROM [dbo].[Students];
```
