

---

# **DML-Anweisungen**

# Was ist DML?

- DML: Data Manipulation Language
  - Einfügen, Löschen, Verändern von Daten
- Einfügen mit **INSERT**
- Löschen mit **DELETE**
- Verändern mit **UPDATE**

Hier geht es um die Datensätze, nicht um die Tabelleneigenschaften!

# INSERT I


- Syntax:

```
INSERT [INTO] <Tabelle/View> [(Spaltenliste)]  
VALUES (Einträge)
```

## Beispiel

```
INSERT Mitarbeiter (ID, Vorname, Nachname)  
VALUES  
(10, 'Gerwin', 'Schnittjer'),  
(20, 'Tony', 'Stark');
```

Spaltenliste legt  
auch Reihenfolge  
fest



## INSERT II

- Die OUTPUT-Klausel gibt die eingefügten Daten in Tabellenform aus
  - Mit `inserted` kann zusätzlich nach Spalten gefiltert werden

```
INSERT Mitarbeiter (Nachname, Vorname)
OUTPUT inserted.Nachname, inserted.ID
VALUES
('Bunny', 'Bugs'),
('Duck', 'Donald');
```

	Nachname	ID
1	Bunny	6
2	Duck	7

## Zusatzinformationen

- Statt VALUES kann auch ein SELECT verwendet werden, die Ergebnisdaten vom SELECT-Statement werden dann eingefügt
- Die automatische Identitätsvergabe (IDENTITY) kann auch abgestellt werden

➤ SET IDENTITY\_INSERT [<Datenbankname>.[<Schema\_name>.]  
Tabelle {ON/OFF}

- Mit SELECT...INTO können Daten kopiert werden

➤ SELECT \* INTO Mitarbeiter2 FROM Mitarbeiter

# DELETE I

- DELETE entfernt Tabellen- oder Viewzeilen

```
➤ DELETE FROM <Tabellenname/View>  
  WHERE [Bedingung];
```

- ...oder alle Zeilen löschen

```
➤ DELETE FROM <Tabellenname/View>;
```

DELETE löscht die Inhalte, nicht die Tabelle selbst!

# WHERE I

Art des Operators	Operator	Bedeutung
Unär	+	Gibt den Wert des folgenden Ausdrucks als positiven Wert zurück
	-	Gibt den Wert des folgenden Ausdrucks als negativen Wert zurück
	~	Gibt den Wert der folgenden Ganzzahl bitweise umgekehrt zurück
Arithmetische Operatoren	+, -, *, /	Addition, Subtraktion, Multiplikation und Division
	%	Modulo, ganzzahliger Rest einer Division
Verknüpfung von Zeichenketten	+	5 + 9 erzeugt 14, 5 + '9' erzeugt ebenfalls 14, '5' + '9' erzeugt 59, '5.2' + 9 scheitert, '5.2' + 9.0 erzeugt 14.2, '5,2' + 9 scheitert
Bitweise Operatoren	&,  , ^	Bitweise AND, OR, XOR
Vergleichsoperatoren	=, <, >, <=, >=, <>, !=, !<, !>	Gleich, kleiner, größer, gleiner oder gleich, größer oder gleich, ungleich, ungleich, nicht kleiner, nicht größer

## WHERE II

Art des Operators	Operator	Bedeutung
Erweiterte Vergleichsoperatoren	BETWEEN	5 BETWEEN 3 AND 7 ergibt TRUE
	IN	5 IN (3, 5, 7) ergibt TRUE
	LIKE	Musterüberprüfung der Ausdrücke, kann Platzhalter (% , _) und Gruppenzeichen ([ , ^]) enthalten
	EXISTS	TRUE bei wenigstens einem Vorkommen; oft schneller als COUNT
	ALL, ANY, SOME	<erster Ausdruck> <Standardoperator> ALL ANY SOME (<Unterabfrage>) Liefert TRUE oder FALSE
Logische Operatoren	AND, OR	Gibt Wahrheitswerte zurück



## Beispiele

```
DELETE FROM Mitarbeiter;  --löscht alle Mitarbeiter

DELETE FROM Mitarbeiter
WHERE Nachname = 'Bunny'; -- löscht alle Mitarbeiter mit
                           Nachnamen Bunny

DELETE FROM Mitarbeiter
WHERE Nachname LIKE '%nn%'
AND Geburtsdatum !< '1998-01-01';
-- löscht alle Mitarbeiter, die in ihrem Nachnamen zwei 'n'
-- hintereinander haben und nicht vor dem 1. Januar 1998
-- geboren wurden
```

Riesiges Potential an Einschränkungsmöglichkeiten!

## Löschen mit TRUNCATE

- TRUNCATE kann nicht bei Tabellen verwendet werden, auf die mit einem FOREIGN KEY verwiesen wird
- DELETE löscht physikalisch, TRUNCATE markiert die Datenseiten als „frei“
- TRUNCATE ist schneller, DELETE ist sicherer
- TRUNCATE ist „all or nothing“; keine Einschränkungen

```
TRUNCATE TABLE Mitarbeiter;
```

## DELETE II

- OUTPUT-Klausel auch bei DELETE möglich

Beispiel:

```
DELETE FROM Mitarbeiter  
OUTPUT deleted.Vorname  
WHERE Vorname = 'Bugs';
```

# UPDATE I

Syntax:

```
UPDATE <Tabellenname>  
SET <Spalte 1 = Wert 1, Spalte 2 = Wert 2, ...>  
WHERE [Einschränkungen];
```

Beispiel:

```
UPDATE Abteilung  
SET Abteilungsname = 'Office Management'  
WHERE Abteilungsname = 'Tippsenzimmer';
```

## UPDATE II

- UPDATE kann auch mit OUTPUT-Klausel verwendet werden
- UPDATE vs. ALTER
  - UPDATE erlaubt das Verändern von Spalteninhalten einer Tabelle
  - ALTER erlaubt das Verändern von Tabelleneigenschaften

# MERGE I

- Eine Anweisung, welche die Fähigkeiten der Kommandos INSERT, UPDATE und DELETE vereinigt.
- MERGE ist auch bei anderen RDBMS zu finden

ID	Vorname	Nachname	Mail
1	Gerwin	Schnittjer	gs@drheuer.de
2	Eduard	Paul	ep@drheuer.de

ID	Vorname	Nachname	Mail
3	Andreas	Kolonko	ak@drheuer.de
2	Eduard	Paul	ep@drheuer.de



ID	Vorname	Nachname	Mail
1	Gerwin	Schnittjer	gs@drheuer.de
2	Eduard	Paul	ep@drheuer.de
3	Andreas	Kolonko	ak@drheuer.de

## MERGE II

```
MERGE <Zieltabelle>  
USING <Quelltabelle> ON <Übereinstimmungsbedingung>  
WHEN MATCHED THEN  
    UPDATE SET <Spalte Zieltabelle = Spalte Quelltabelle, ...>  
WHEN NOT MATCHED BY TARGET THEN  
    INSERT (Spalte 1, Spalte 2,...) VALUES (Wert 1, Wert2,...)  
WHEN NOT MATCHED BY SOURCE THEN  
    INSERT (Spalte 1, Spalte 2,...) VALUES (Wert 1, Wert2,...);
```