

# **Baze podataka II**

## **Modul 6 – Jezik SQL**

Uvod u SQL/DML

Komande za modifikovanje podataka

# Summary

- Dodavanje zapisa – INSERT;
- Brisanje zapisa – DELETE;
- Modifikovanje zapisa – UPDATE;
- Osnove o transakcijama;



# Lekcija 1: Dodavanje zapisa - INSERT

- Osnove komande INSERT;
- Dodavanje zapisa preko liste vrijednosti;
- INSERT – SELECT kombinacija;
- SELECT INTO kombinacija;
- Parcijalno dodavanje zapisa;
- Dodavanje u kombinaciji sa „default“ vrijednostima;
- Dodavanje vrijednosti u IDENTITY kolonu;



# Osnove komande INSERT

- **INSERT je DML komanda za dodavanje novih zapisa;**

```
INSERT [INTO] Tabela (lista_kolona)
        (Lista_vrijednosti)
```

- `INTO` je ključna riječ za specificiranje lokacije
- `Tabela` je naziv objekta (tabele)
- `Lista_kolona` je lista atributa u koje želimo pohraniti podatke
- `Lista_vrijednosti` su podaci koje želimo pohraniti
- **INSERT se može koristiti u mnogim kombinacijama**
  - (dodavanje jednog zapisa, više zapisa sa jednom komandom, podupit, kombinacija sa TOP i sl.

# Dodavanje zapisa preko liste vrijednosti

- Obratiti pažnju na constraints (ograničenja) unutar tabele;
  - Tip, opseg, pravila, FK i sl.
- Redoslijed liste vrijednosti treba pratiti listu sa nazivima kolona;

```
INSERT INTO Osobe  
(Prezime, Ime, Telefon, Email)  
VALUES  
( 'Azemović', 'Jasmin', '111-111-111', 'jasmin@edu.fit.ba' )
```

# INSERT – SELECT kombinacija

- Radi se o podupitu u INSERT komandi;
- Dodaju se svi zapisi koje vrati komanda SELECT ;
  - Provjeriti destinacijsku tabelu
  - Osigurati kompatibilnost podataka;
  - Provjeriti default i NULL vrijednosti u destinaciji;

```
INSERT INTO Osobe
SELECT AP.LastName, AP.FirstName,
       APP.PhoneNumber, AEA.EmailAddress
FROM AdventureWorks2014.Person.Person AS AP
INNER JOIN AdventureWorks2014.Person.PersonPhone AS APP
ON AP.BusinessEntityID = APP.BusinessEntityID
INNER JOIN AdventureWorks2014.Person.EmailAddress AS AEA
ON APP.BusinessEntityID = AEA.BusinessEntityID
```

# SELECT INTO kombinacija

- **Kreira novu tabelu i dodaje zapise iz upita**
  - Može raditi i sa temp (# ili ##) tabelama
  - Tipovi podataka se definišu na osnovu izvornih podataka

```
SELECT Title + ', ' + FirstName + ' ' + LastName AS Customer  
    INTO #tempOsobeTitule  
FROM AdventureWorks2014.Person.Person  
WHERE Title IS NOT NULL
```

```
SELECT Title + ', ' + FirstName + ' ' + LastName AS Customer  
    INTO OsobeTitule  
FROM AdventureWorks2014.Person.Person  
WHERE Title IS NOT NULL
```

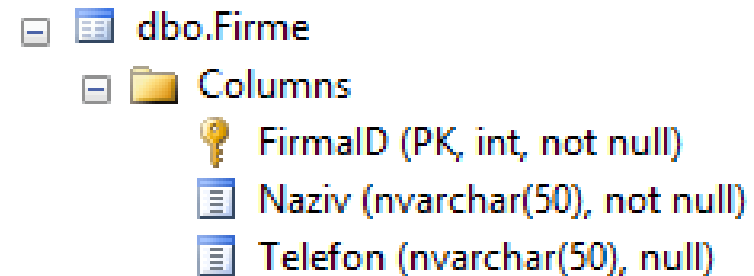
# Parcijalno dodavanje zapisa

- Dodavanje novog zapisa bez navođenja kompletne liste kolona;

```
INSERT INTO Firme (Naziv)  
VALUES ('Pujdo Inc.')
```

- Provjera

```
SELECT * FROM Firme  
WHERE Naziv = 'Pujdo Inc.'
```



FirmaID	Naziv	Telefon
1	Pujdo Inc.	NULL

(1 row(s) affected)



# Dodavanje u kombinaciji sa „default“ vrijednostima

- **Ključna riječ DEFAULT:**

- Ubacuje podrazumijevane (default) vrijednosti u specificirane kolone
- Mandatorno je da kolone koje se koriste na takav način imaju DEFAULT ili NULL osobinu

```
INSERT INTO Firme (Naziv, Telefon)  
VALUES ('New Pujdo Inc.', DEFAULT)
```

- **Ključna riječ DEFAULT VALUES**

- Ubacuje podrazumijevane (default) vrijednosti za sve kolone

# Dodavanje vrijednosti u IDENTITY kolonu

- U posebnim slučajevima je potrební dodati specifičnu vrijednost u IDENTITY kolonu
- Koristi se SET IDENTITY\_INSERT ON

```
SET IDENTITY_INSERT Firme ON;  
GO  
INSERT INTO Firme (FirmaID, Naziv)  
VALUES (99, 'Firma sa posebnim ID')
```

## Lekcija 2: Brisanje zapisa - DELETE

- Osnove komande DELETE;
- Primjeri sa komandom DELETE;
- Brisanje preko posredne tabele;
- Korišćenje TRUNCATE komande;



# Osnove komande DELETE

- Komande DELETE briše jedan ili više zapisa
- Prilikom rada sa ovom komandom, obratite pažnju na:
  - Moguće je obrisati jedan ili više zapisa istovremeno
  - Brisanje bez WHERE klauzule vodi ka gubitku svih zapisa iz tabele
    - Ovo neće važiti ako postoji FK ograničenje
    - ...ali ako je opcija kaskadno brisanje ON, tada dolazi do lančanog brisanja
  - Samo WHERE garantuje da će biti obrisani specifični zapisi
    - Najbolje preko PK atributa
  - Kada se brišu velike količine podataka, DELETE može kreirati LOCK nad tabelom

# Primjeri sa komandom DELETE

```
DELETE FROM Firme  
WHERE FirmaID = 99
```

```
DELETE FROM Firme
```



```
DELETE FROM Orders  
WHERE DATEDIFF(MONTH, ShippedDate, GETDATE()) >= 6
```

# Brisanje preko posredne tabele

- **Mnogo efikasniji način za brisanje umjesto da se piše više DELETE komandi;**
  - Tamo gdje je to potrebno
- **Izvodi se preko dodatne FROM klauzule**
  - Prvi FROM je tabela koja se modifikuje
  - Drugi FROM se ponaša kao JOIN veza i restrikcija prilikom brisanja;

```
DELETE FROM [Order Details]
FROM Orders AS O
INNER JOIN [Order Details] AS OD
ON O.OrderID = OD.OrderID
WHERE OrderDate = '4/14/1998'
```

# Korištenje TRUNCATE komande

- **TRUNCATE TABLE briše sve zapise, ali bez operacije logiranja**

```
TRUNCATE TABLE Person.Person
```



- **Permanetno briše sve zapise iz Person.Person tabele**
- **TRUNCATE neće raditi u svim slučajevima**
  - FK referenca na drugu tabelu(e)
  - Kada se tabela koristi u replikacijama
  - Kada je dio indeksiranog view objekta

# Lekcija 3: Modifikovanje zapisa – UPDATE

- Osnove komande UPDATE
- Primjeri sa komandom UPDATE
- Modifikovanje zapisa preko posredne tabele





# Osnove komande UPDATE

- Komanda UPDATE modifikuje podatke unutar kolona za jedan ili više zapisa
- Klauzula SET sadrži naziv kolone i novu vrijednost
- FROM klauzula omogućava korištenje posredne tabele
- WHERE specificira uslov modifikovanja

```
UPDATE Tabela  
  SET Kolona(e) = NovaVrijednost(i)  
  FROM Posredna_tabela  
  WHERE Uslov_modifikovanja
```

# Primjeri sa komandom UPDATE

- Korištenje bez WHERE može biti „opasno“

```
UPDATE Person.Person  
SET FirstName = 'New Ken'
```



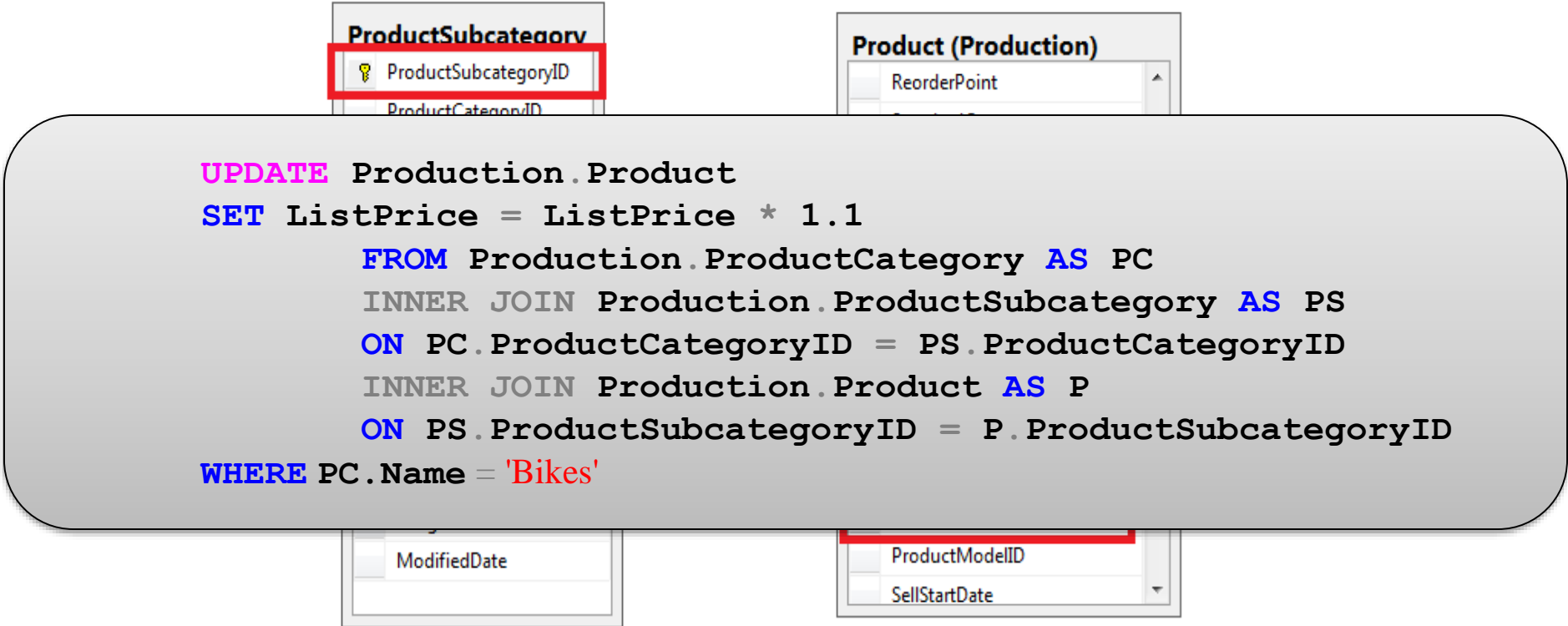
- Sve osobe sada imaju isto prezime

```
UPDATE Person.Person  
SET FirstName = 'New Ken'  
WHERE BusinessEntityID = 1
```

- U ovome slučaju samo jedna osoba (ako je u uslovu PK)

# Modifikovanje zapisa preko posredne tabele

- Specijalan slučaj komande UPDATE je poduput



```
UPDATE Production.Product
SET ListPrice = ListPrice * 1.1
FROM Production.ProductCategory AS PC
INNER JOIN Production.ProductSubcategory AS PS
ON PC.ProductCategoryID = PS.ProductCategoryID
INNER JOIN Production.Product AS P
ON PS.ProductSubcategoryID = P.ProductSubcategoryID
WHERE PC.Name = 'Bikes'
```

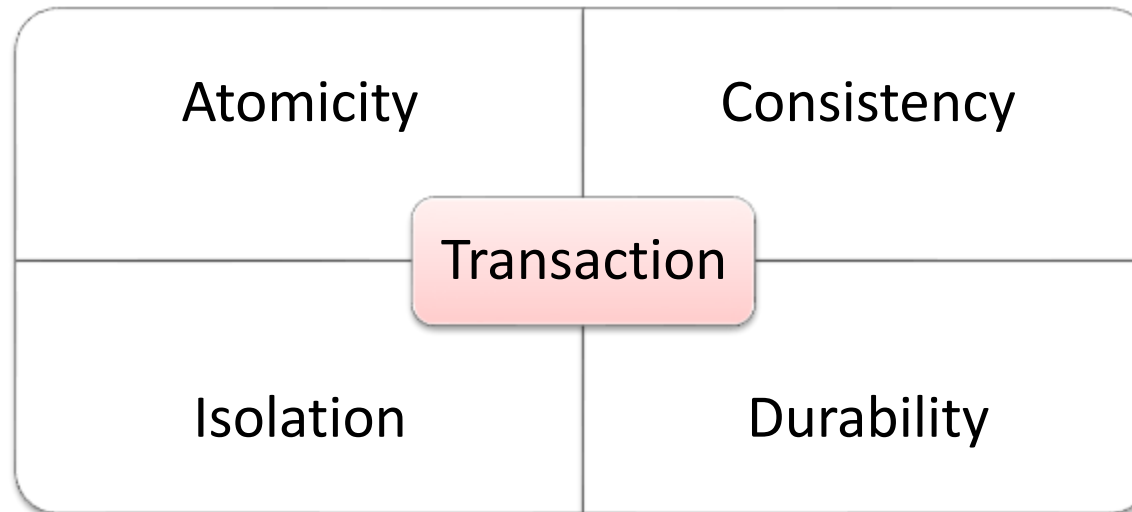
# Lekcija 4: Osnove o transakcijama

- Šta su transakcije
- Transakcije unutar SQL Server okruženja
- Komande za transakcije



# Šta su transakcije

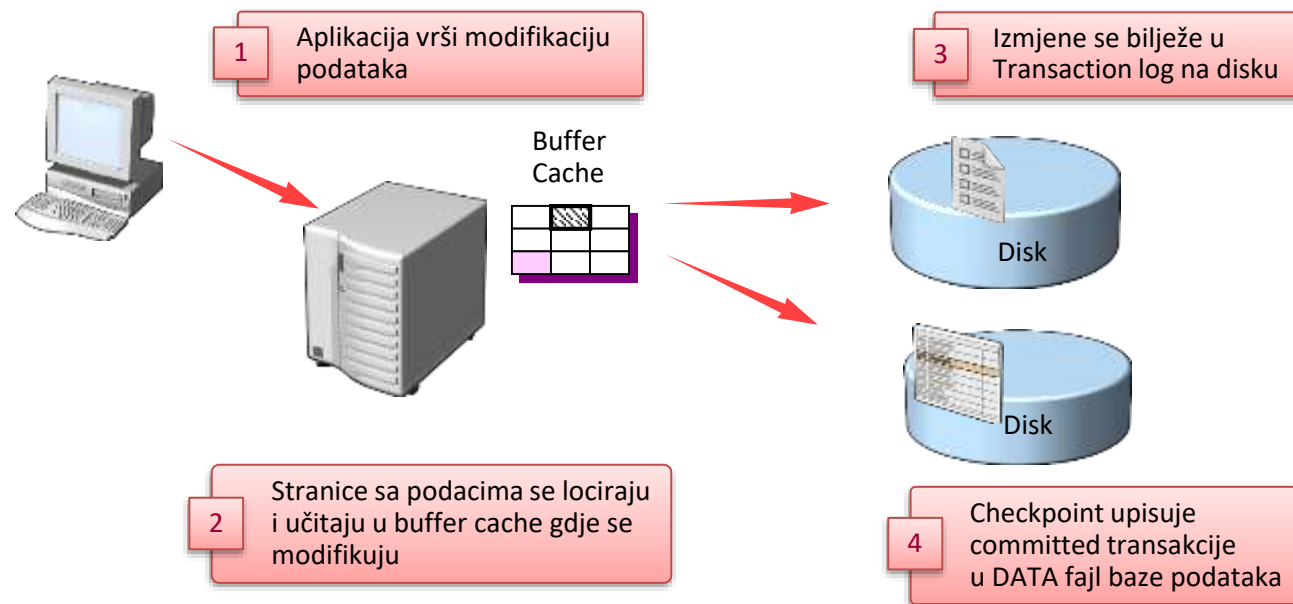
- Svrha transakcije jeste da osigura konzistentnost podataka.
  - Kako?
- Odgovor je ACID.
- Kada se transakcija pokrene, neophodno je da budu ispoštovani sljedeći uslovi:



- Ako bilo šta od navedenog zakaže, onda se radi rollback

# Transakcije unutar SQL Server okruženja

- SQL Server funkcioniše na principu WAL
  - Write Ahead Log



# Komande za transakcije

- **BEGIN TRANSACTION** – je tačka odakle sve što slijedi treba da bude konzistentno.
- **COMMIT TRANSACTION** – sve prije ove tačke se smatra validnim i upisuje se u DATA fajl baze podataka
- **ROLLBACK TRANSACTION** – Ako bilo koji ACID parametar zakaže sve djelomične operacije se vraćaju u početno stanje tj. do BEGIN TRANSACTION komande
  - *Rollback transakcije nije moguć nakon COMMIT*

# Pitanja

