Data modification

Jaroslav Porubän, Miroslav Biňas,
Milan Nosáľ (c) 2011 - 2016

`INSERT` statement

- Inserts a new row to the table
- syntax 1 (order and number of attributes is significant):

```
INSERT INTO table
VALUES (value1, ..., valueN);
```

- syntax 2 (order is insignificant):

```
INSERT INTO table
(col_name1, ..., col_nameN)
VALUES (value1, ..., valueN);
```

INSERT example

```
INSERT INTO
student (id, name, surname)
VALUES (1,'Janko','Hraško');

INSERT INTO
student (name,surname, id)
VALUES ('Jožko','Mrkvička',2);

INSERT INTO student
VALUES (3, 'Hanka', 'Veselá');
```

UPDATE statement

- Updates data in the table
- syntax:

```
UPDATE table
SET atr1=new_value1 [,...
atrN=new_valueN]
[WHERE condition];
```

- example:

```
UPDATE student
SET name='Ferko' WHERE id=3;
```

DELETE statement

- Removes row/rows from table
- syntax:

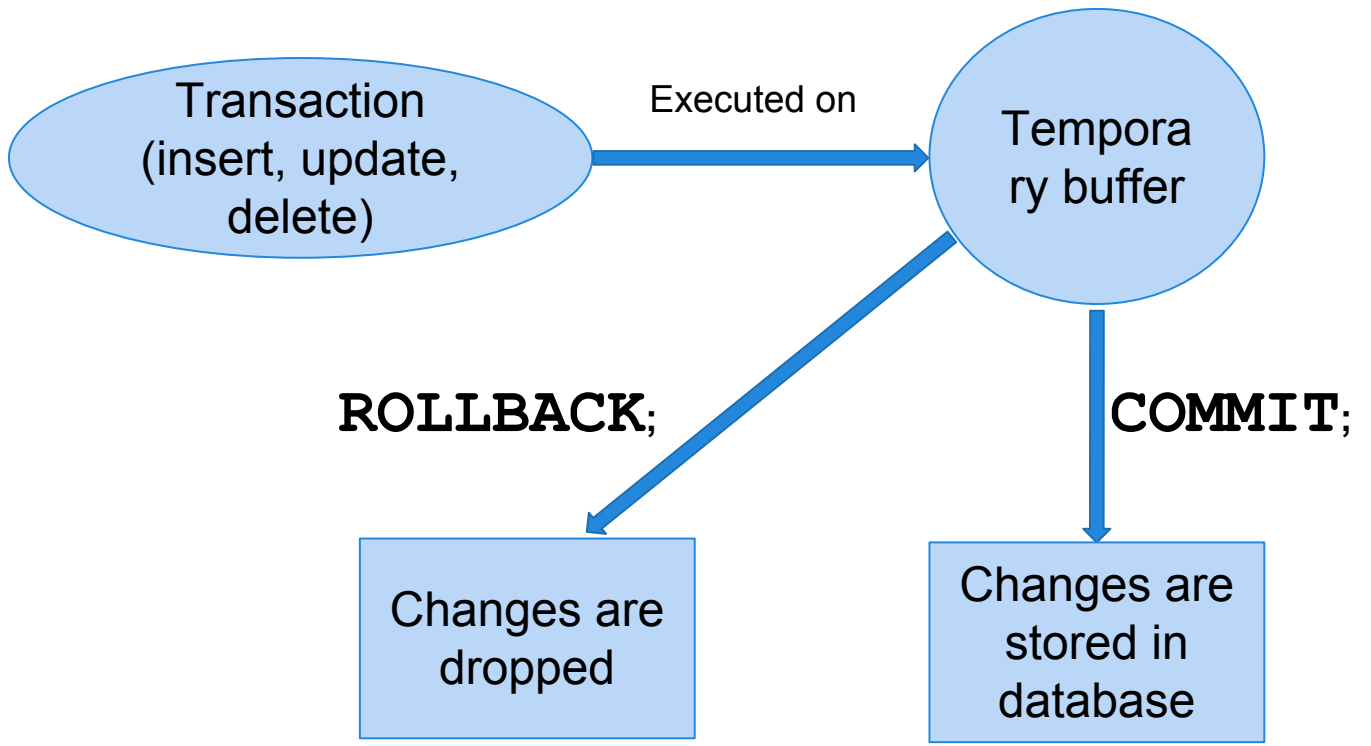
```
DELETE FROM table
 [WHERE condition];
```

- príklad:

```
DELETE FROM student
WHERE id=3;
```

Transactional execution

- Sequence of modifying statements executed as a whole - transaction
  - Either all changes are stored
  - Or none is stored



`TRUNCATE TABLE` statement

- Removes all rows from table
- syntax:

```
TRUNCATE TABLE table;
```

- example:

```
TRUNCATE TABLE student;
```

- Same result with (although `TRUNCATE` is not transactional):

```
DELETE FROM tabuľka;
```

Enforcing referential integrity I.

- Referential integrity violations
  - Removing a tuple that is referenced by another tuple
  - Insertion/update of a tuple referencing a non-existent tuple
- Approaches:
  - **Cascade**
  - **Restrict**
  - **Set null**
- Default behavior (*Oracle*)

```
ON DELETE NO ACTION
```

Enforcing referential integrity II.

- `CASCADE` - changes are cascaded
  - If the referenced tuple is deleted, the referencing tuple is deleted as well
- `RESTRICT (NO ACTION)` - changes are forbidden
  - Referenced tuple cannot be deleted, if there is referencing tuple
- `SET NULL`
  - If the referenced tuple is deleted/updated, the foreign key in referencing tuple is set to NULL

Enforcing referential integrity - Example

```
CREATE TABLE course_enlist (
 id_course INT,
 id_student INT,
 PRIMARY KEY (id_course,
id_student),
 FOREIGN KEY(id_course)
 REFERENCES course(id) ON DELETE
CASCADE,
 FOREIGN KEY(id_student)
 REFERENCES student(id) ON DELETE
CASCADE
);
```

Questions?