7. Architektura klient/server a třívrstvá architektura

Ing. Vladimír Bartík, Ph.D.
RNDr. Marek Rychlý, Ph.D.

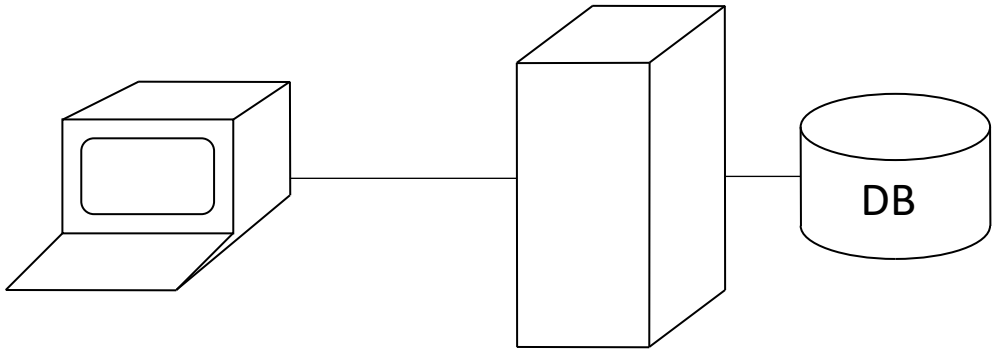
Osnova

7. Architektura klient/server a třívrstvá architektura

- **Základ kooperativního zpracování**
- **Faktory ovlivňující architekturu**
  - **požadavky na interoperabilitu zdrojů**
  - **růst velikosti zdrojů**
  - **růst počtu klientů**
- **Typy služeb v databázové technologii**
  - ***prezentační služby*** **- příjem vstupu, zobrazování výsledků**
  - ***prezentační logika*** **- řízení interakce (hierarchie menu, obrazovek)**
  - ***logika aplikace*** **- operace realizující algoritmus aplikace**
  - ***logika dat*** **- podpora operací s daty (integritní omezení, ...)**
  - ***datové služby*** **- akce s databází (definice a manipulace, transakční zpracování, ...)**
  - ***služby ovládání souborů*** **- vlastní V/V operace**

7.1. Varianty architektury

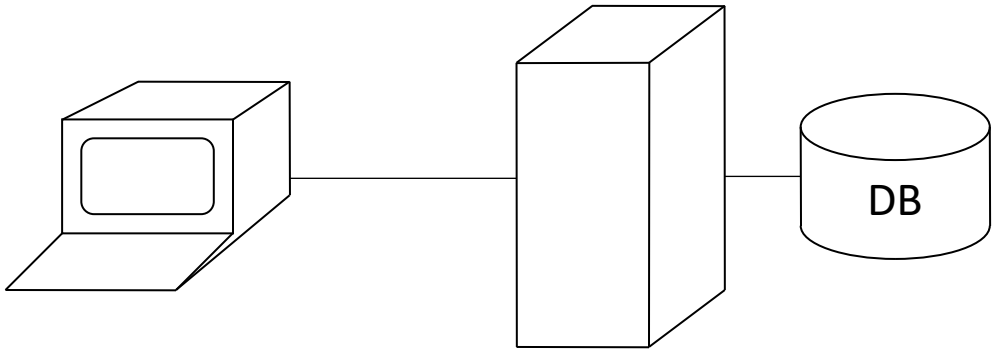
Klient/server se vzdálenými daty



- prezentační služby
- prezentační logika
- logika aplikace
- logika dat
- datové služby
- ovládání souborů

- komunikační zátěž, zatížení stanice

Klient/server se vzdálenou prezentací

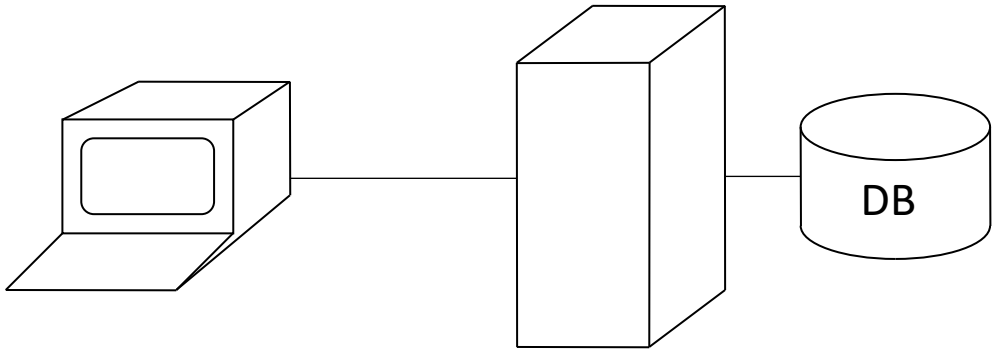


- prezentační služby
- prezentační logika
- logika aplikace
- logika dat
- datové služby
- ovládání souborů

- zatížení serveru

7.1. Varianty architektury

Klient/server s rozdělenou logikou



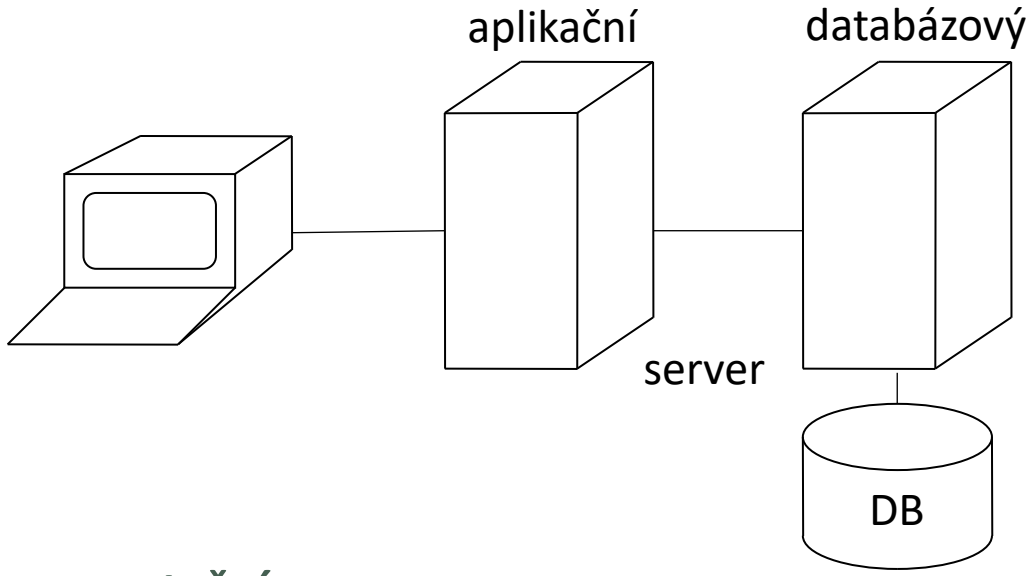
- prezentační služby
- prezentační logika
- logika aplikace
- logika dat

- logika aplikace
- logika dat
- datové služby
- ovládání souborů

\+ vyvážená zátěž

\- správa a údržba SW klientů

Třívrstvá architektura



- prezentační služby
- prezentační logika

- logika aplikace
- logika dat

- datové služby
- ovládání souborů

\+ správa aplikace, sdílené objekty aplikací, tenký klient, rozšiřitelnost

7.2. Přínos architektury klient/server a třívrstvé architektury

- **pružnější rozdělení práce**
- **lze použít horizontální (více serverů) i vertikální (výkonnější server) škálování**
- **aplikace mohou běžet na levnějších zařízeních**
- **standardizovaný přístup umožňuje zpřístupnit další zdroje**
- **centralizace dat podporuje účinnější ochranu**
- **u třívrstvé architektury centralizace údržby aplikace, možnost využití sdílených objektů (business objects) několika aplikacemi**

7.3. Podpora pro rozdělení zátěže v architektuře klient/server

- **deklarativní integritní omezení**
- **uložené podprogramy**
- **databázové triggery**

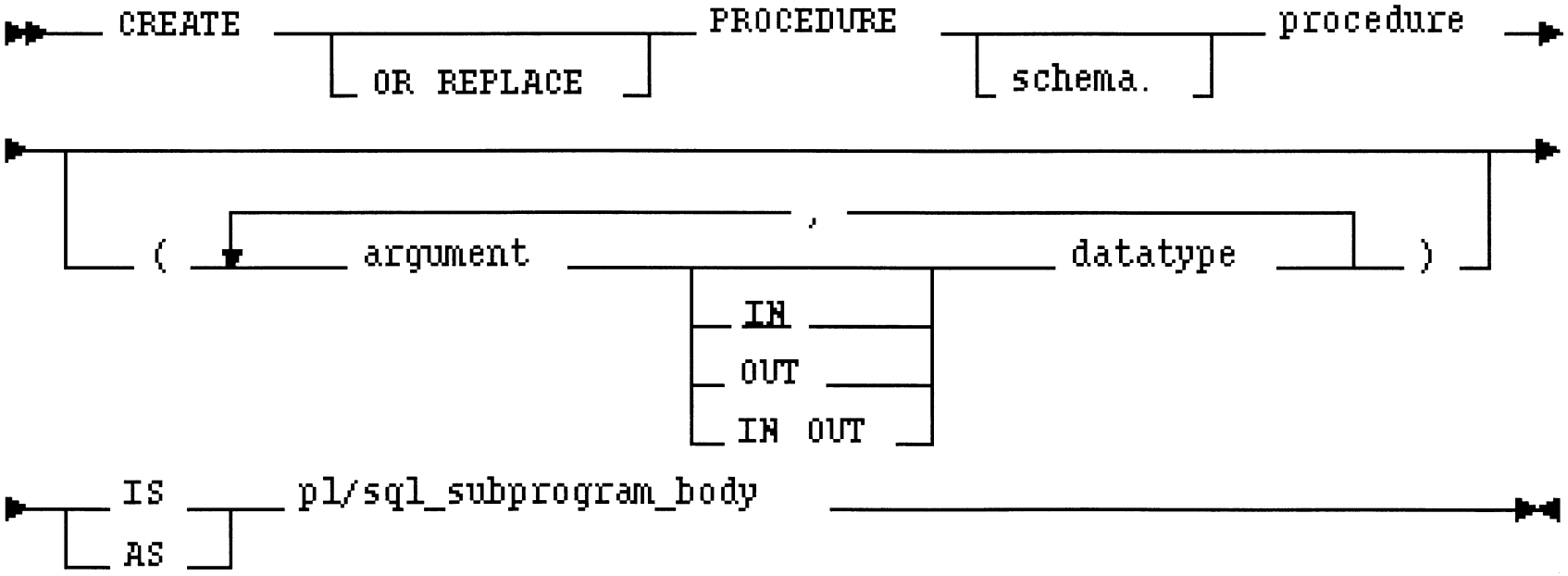
7.3.1. Uložené podprogramy v SQL

- **dodatek SQL-92/PSM (*Persistent Stored Modules*) – zavádí:**
  - **složený příkaz**
  - **řídicí struktury (IF, CASE, LOOP, WHILE, REPEAT, FOR)**
  - **výjimečné situace**
  - **uložené procedury a funkce**
- **databázový objekt obsahující kód prováděný na straně DB serveru**
- **příkazy**
  - **CREATE**
  - **DROP**
  - **ALTER**

7.3.1. Uložené podprogramy v SQL

Př.) Uložené podprogramy v Oracle

- Uložená procedura v Oracle (zjednodušeno)




```
CREATE PROCEDURE remove_emp (employee_id NUMBER) AS
BEGIN
      DELETE FROM employees
      WHERE employees.employee_id = remove_emp.employee_id;
END;
```

7.3.1. Uložené podprogramy v SQL

- **volání v PL/SQL jako jakékoliv jiné procedury/funkce**

  Př.) Uložená funkce v Oracle

```
CREATE FUNCTION get_bal (acc_no IN NUMBER)
  RETURN NUMBER
  IS acc_bal NUMBER(11,2);
  BEGIN
    SELECT order_total
    INTO acc_bal
    FROM orders
    WHERE customer_id = acc_no;
    RETURN (acc_bal);
  END;
```

- **u Oracle lze vytvářet i uložené moduly (PACKAGE)**

7.3.2. Databázové triggery

- není součástí SQL/92, je součástí SQL/99
- ***Databázový trigger*** je databázový objekt obsahující kód spouštěný specifikovanou událostí v databázi.
- Složky příkazu vytvoření databázového triggeru
  - jméno triggeru
  - jméno tabulky – `jméno [REFERENCING alias_pro_OLD_a_NEW]`
  - čas spuštění akce – `BEFORE/AFTER`
  - událost – `INSERT/DELETE/UPDATE [OF sloupec, …]`
  - spouštěná akce – `FOR EACH {ROW | STATEMENT}[WHEN podmínka] spouštěný_SQL_příkaz`

7.3.2. Databázové triggery

- **Typy databázového triggeru**
  - **BEFORE/AFTER – příkazy těla triggeru se provedou před/po provedení události, která trigger spouští.**
  - **INSTEAD OF – příkazy těla triggeru se provedou místo přímé modifikace neaktualizovatelného pohledu (uvedenou tabulkou musí být pohled, ne bázová tabulka).**
  - **Příkazový – je spuštěn jedenkrát pro příkaz, který způsobí zadanou událost, je-li splněna podmínka uvedená v klauzuli WHEN (když je uvedena).**
  - **Řádkový – je spuštěn jedenkrát pro každý řádek ovlivněný danou událostí, je-li splněna podmínka uvedená v klauzuli WHEN (když je uvedena).**

Poznámka: Databázový trigger je něco jiného, než trigger klientské části aplikace, který se spouští typicky událostí formuláře (např. stisk tlačítka) nebo tiskové sestavy (např. nová stránka)).

7.3.2. Databázové triggery

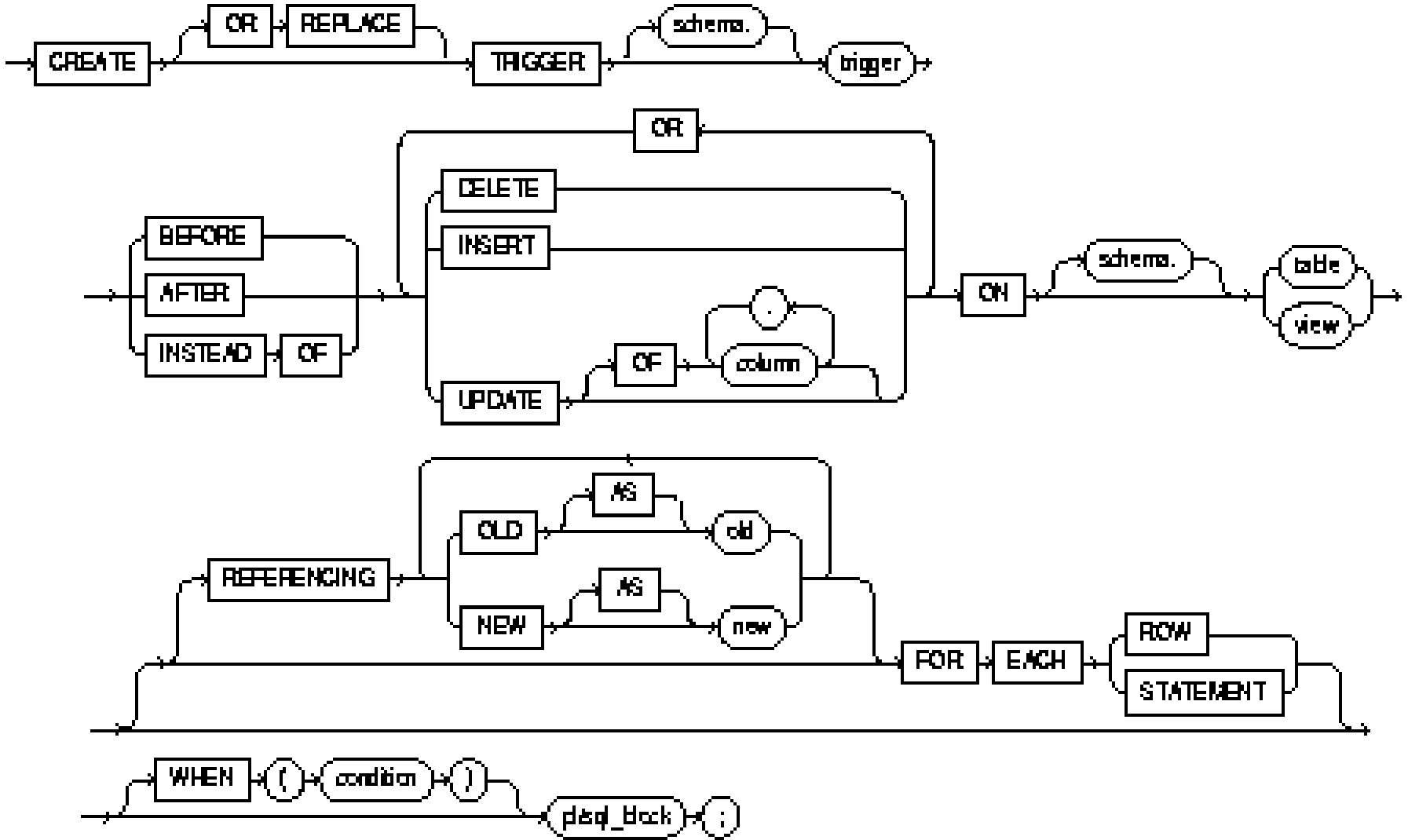
- **Typická použití databázových triggerů**
  - **Implementace integritních omezení**
  - **Implementace složitých bezpečnostních omezení (např. možnost modifikace dat jen v určitou dobu)**
  - **Implementace auditu (kdo jaké operace prováděl)**
  - **Provedení transparentních modifikací při nějaké události.**
  - **Výpočet hodnot pro sloupec s odvozenými hodnotami**

7.3.2. Databázové triggery

- **Zásady při použití databázových triggerů**
  - **Použij databázový trigger k zajištění, že při provedení konkrétní operace jsou provedeny odpovídající akce.**
  - **Použij databázový trigger jen pro centralizované globální operace, které by měly být spuštěny bez ohledu na to, který uživatel/aplikace operaci provede.**
  - **Nepoužívej databázový trigger k odmítnutí nesprávných dat, jestliže lze totéž provést použitím deklarativních integritních omezení.**
  - **Je-li tělo triggeru rozsáhlé (např. 60 řádků PL/SQL pro Oracle), vytvoř uloženou proceduru, která se bude volat z těla triggeru.**
  - **Nevytvářej rekurzivní triggery.**
  - **Používej databázové triggery uvážlivě. Provádí se pro každého uživatele/aplikaci pokaždé, když daná událost nastane.**

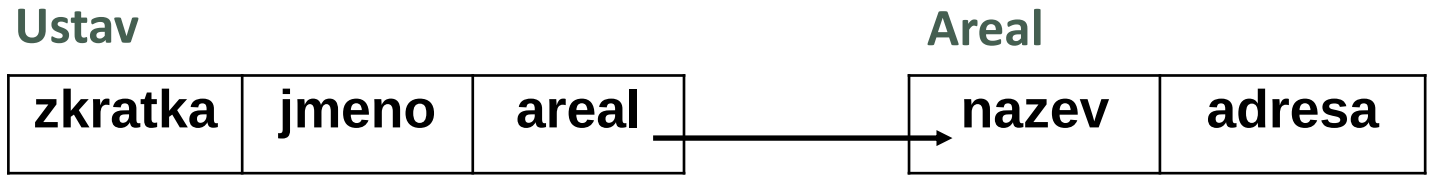
7.3.2. Databázové triggery

Př.) Databázové triggery v prostředí Oracle (zjednodušeno)



7.3.2. Databázové triggery

Př.)

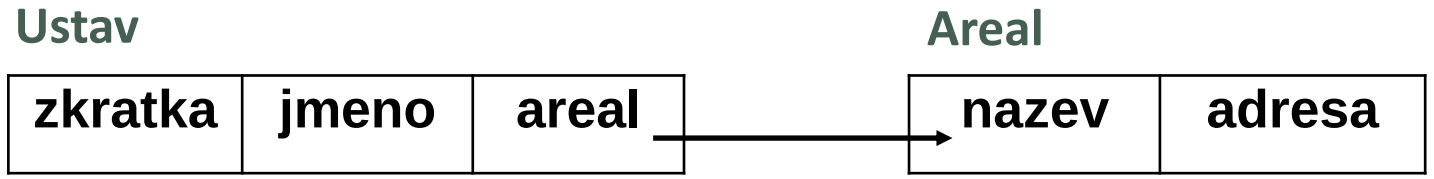


- **Možná implementace ON UPDATE CASCADE (Oracle podporuje jen ON DELETE CASCADE)**

```
CREATE OR REPLACE TRIGGER aktualizuj_ustav
     AFTER UPDATE OF nazev ON Areal
     REFERENCING OLD AS puvodni NEW AS novy
     FOR EACH ROW
BEGIN
     UPDATE Ustav SET areal = :novy.nazev
     WHERE areal = :puvodni.nazev;
END;
```

7.3.2. Databázové triggery

Př.)

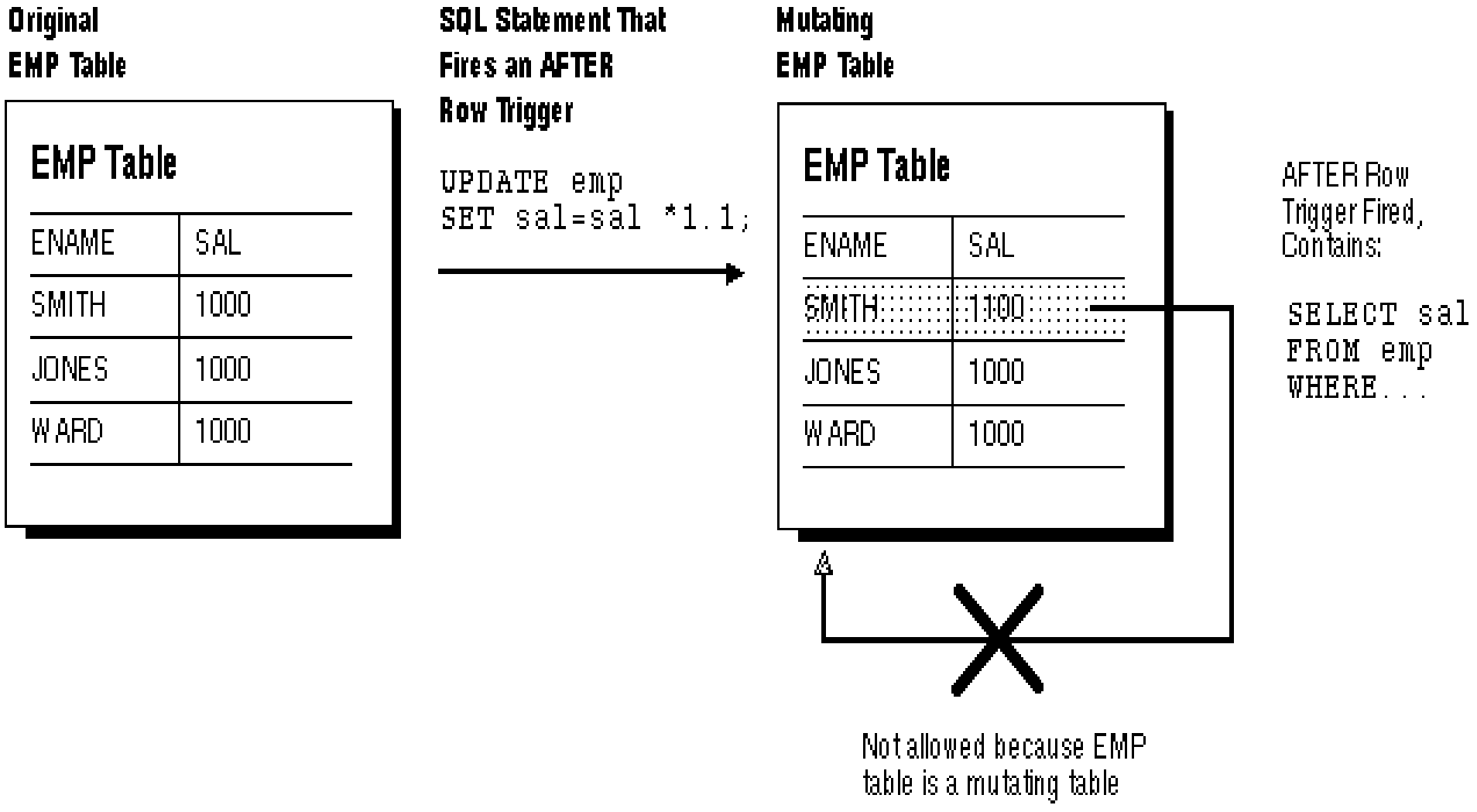


- **Možná implementace ON DELETE SET NULL (Oracle už podporuje)**

```
CREATE OR REPLACE TRIGGER nuluj_Ustav
     AFTER DELETE ON Areal
     FOR EACH ROW
BEGIN
     UPDATE Ustav SET areal = NULL
     WHERE areal = :old.nazev;
END;
```

7.3.2. Databázové triggery

Omezení příkazů v těle triggeru (Oracle)



Literatura

1. Silberschatz, A., Korth H.F., Sudarshan, S.: Database System Concepts. Fifth Edition. McGRAW-HILL. 2006, str. 241-248.
2. Lemahieu, W., Broucke, S., Baesens, B.: Principles of Database Management. The Practical Guide to Storing, Managing and Analyzing Big and Small Data. Cambridge University Press 2018, str. 121-137.

Skripty

1. **Ke kap. 7.3.2. Definice dat:**
   - ***banka_trigger.sql* – ukázky použití databázového triggeru.**