

---

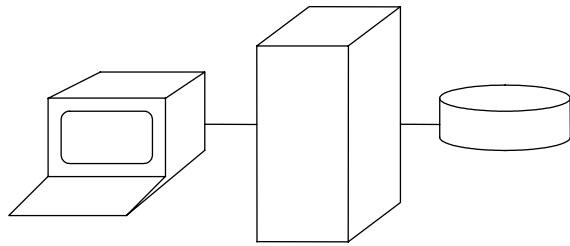
## 10. Architektura klient/server a třívrstvá architektura

10.1. Varianty architektury .....	3
10.2. Přínos architektury klient/server a třívrstvé architektury .....	5
10.3. Podpora pro rozdělení zátěže v architektuře klient/server .....	6
10.3.1. Podpora uložených podprogramů v SQL .....	6
10.4. Uložené podprogramy v prostředí Oracle .....	12
Literatura.....	13

- 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

## 10.1. Varianty architektury

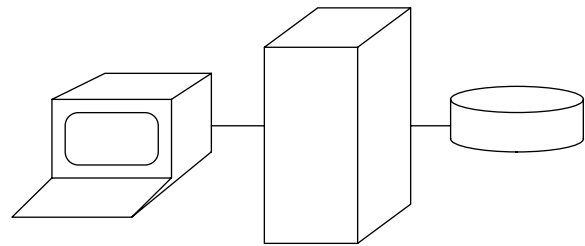
### Klient/server se vzdálenými daty



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

⊖ komunikační zátěž, zatížení stanice

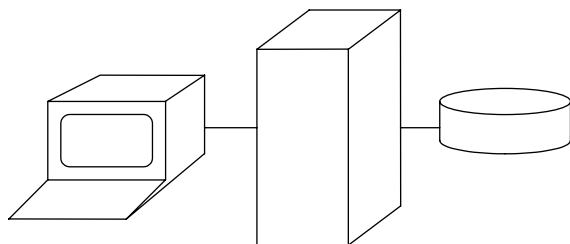
### Klient/server se vzdálenou prezentací



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

⊖ zatížení serveru

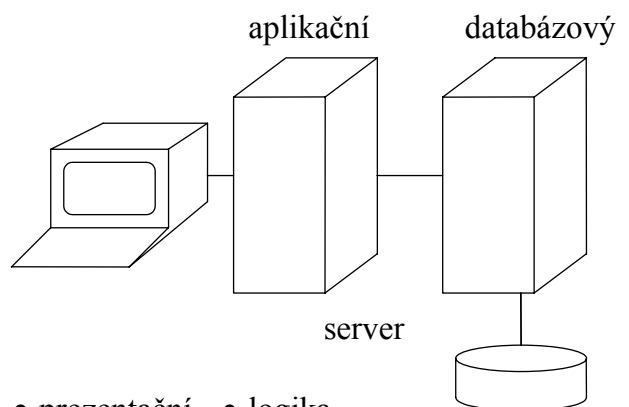
### Klient/server s rozdělenou logikou



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

⊕ vyvážená zátěž  
⊖ horší rozšířitelnost

### Třívrstvá architektura



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

⊕ správa aplikace, sdílené objekty aplikací, tenký klient, rozšířitelnost

## 10.2. 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
- na klientských stanicích lze používat oblíbený prezentační software
- 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

## 10.3. Podpora pro rozdělení zátěže v architektuře klient/server

- deklarativní integritní omezení
- databázové triggery
- uložené podprogramy

### 10.3.1. Podpora uložených podprogramů v SQL

- dodatek SQL-92/PSM (Persistent Stored Modules):
- Procedury a funkce s několika příkazy:

```
Př) /* varianta pro SQL/92 */
void main {
/* vložení informace o studentovi a zápisu do předmětu */
EXEC SQL INSERT INTO Studenti VALUES (100, 'Jan',
                                         'Novák', ...);
/* teď by následoval test SQLSTATE na bezchybné provedení */
EXEC SQL INSERT INTO Zapis VALUES (100, 'INS', ...);
/* teď by následoval test SQLSTATE na bezchybné provedení */
}
```

## ➤ složený příkaz

Př) /\* varianta pro SQL-92/PSM \*/

```
void main {  
EXEC SQL  
    BEGIN    // pro atomický ještě ATOMIC  
        INSERT INTO Studenti VALUES (100, 'Jan', 'Novák',...);  
        INSERT INTO Zapis VALUES (100, 'INS', ...);  
    END;  
/* teď by následoval test SQLSTATE */  
}
```

## ➤ lokální proměnné bloku, včetně kurzorů

Př)

```
BEGIN  
    DECLARE os_cislo_studenta DECIMAL(5);  
    DECLARE jmeno_studenta CHAR VARYING(15);  
    DECLARE prijmeni_studenta CHAR VARYING(20);  
    DECLARE cStudenti CURSOR FOR  
        SELECT os_cislo, jmeno, prijmeni FROM Studenti;  
    OPEN cStudenti;  
    FETCH cStudenti INTO os_cislo_studenta, jmeno_studenta,  
        prijmeni_studenta;  
    ...  
    CLOSE cStudenti;  
END;
```

## ➤ zpracování výjimečných situací (podmínky, handlers, akce)

Př)

```
BEGIN
  DECLARE i INTEGER DEFAULT 1;
  DECLARE ok INTEGER DEFAULT 0;
  WHILE ok = 0 DO
    BEGIN ATOMIC
      DECLARE chyba_usporadatelnosti
        CONDITION FOR SQLSTATE VALUE '40001';
      DECLARE UNDO HANDLER
        FOR chyba_usporadatelnosti
        BEGIN
          IF i>3 THEN RESIGNAL; /* výjimka je poslána dál*/
          SET i = i + 1;
        END
      INSERT INTO Studenti ... ;
      INSERT INTO Zapis ...;
      SET ok = 1;
    END;
  END WHILE
END;
```

## ➤ řídicí struktury

- přiřazení
- IF, CASE
- LOOP, WHILE, REPEAT, FOR

Př)

```
FOR z AS SELECT * FROM Zapis WHERE os_cislo=:student_ID
  DO ...
END FOR;
```

- Uložené podprogramy - procedury a funkce

Př)

Podle SQL/92 (hostitelská verze) - provádění na straně klienta:

```
EXEC SQL SELECT alias
  INTO :alias INDICATOR :alias_ind
  FROM Studenti
  WHERE os_cislo = :student_ID;
```

## Podle SQL/92 (jazyk modulů) - provádění na straně klienta:

```
/* volání v programu v C */
    alias_studenta(&SQLDATE, student_ID, &alias, &alias_ind);
}
MODULE /* Modul SQL klienta*/
PROCEDURE alias_studenta(SQLSTATE, kdo, alias, alias_ind);
    SELECT alias ...;
```

## Uložená procedura prováděná na straně serveru (SQL-92/PSM):

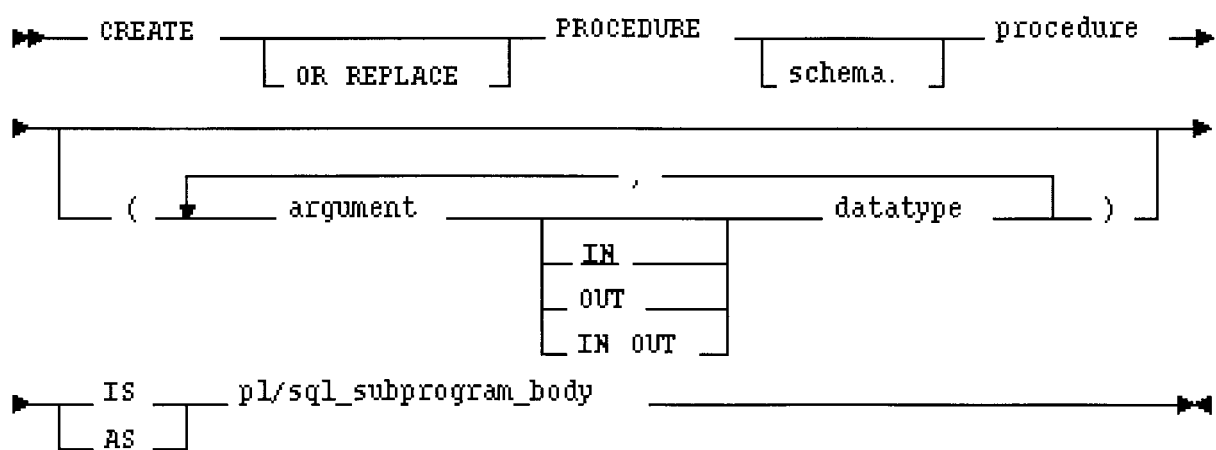
```
EXEC SQL CREATE PROCEDURE alias_studenta (IN kdo, OUT alias,
                                           OUT alias_ind);

    BEGIN ... END;
EXEC SQL CALL alias_studenta(...)
```

### • Uložené moduly

```
EXEC SQL CREATE MODULE ... END MODULE;
```

## 10.4. Uložené podprogramy v prostředí Oracle



- volání v PL/SQL jako jakékoliv jiné procedury
- uložené funkce, moduly (PACKAGE)

## **Literatura**

- 1. Silberschatz, A., Korth H.F, Sudarshan, S.: Database System Concepts. Fourth Edition. McGRAW-HILL. 2001, str. 565 – 680.**
- 2. Pokorný, J.: Databazová abeceda. Science, Veletiny, 1998, str. 57 – 60, 69 – 72, 187 – 190, 217 – 220.**