

# ENKAPSULIRANI SQL

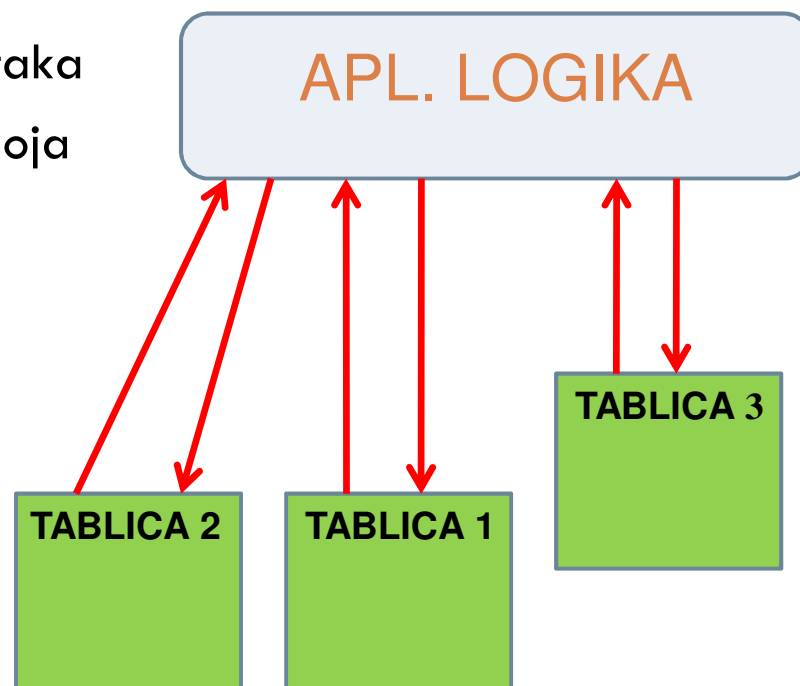
27.10.2013.

Ervin Jagatić

# Pregled predavanja

- Analiza SQLa u PL/SQL-u
- Zašto je SQL loš?
- Što ponuditi kao rješenje
- Primjeri i pregledi mogućnosti upotrebe enkapsuliranog SQLa

- Backend aplikacija
  - Stalni dohvati i ažuriranje podataka
  - Izravan dohvat iz aplikativnog sloja u bazni
  
- SQL upiti
  - Stalna potreba za izmjenama
  - Utječu na performanse
  - Izvori runtime errora



## □ Primjer

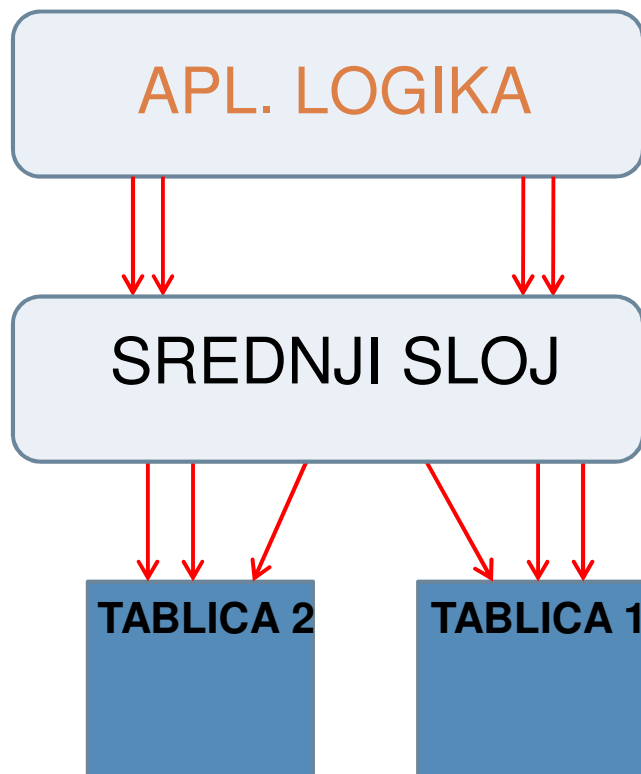
- Napišemo koplekasn upit, iskoristimo ga u funkciji, u izvještaju, na više različitih mjesta....
- Nakon nekog vremena shvatimo da moramo izmijeniti upit
- SQL isto kao i HardCode

# Rješenje

- NE pisati SQL u aplikacijskom kodu
  - ▣ Bilo da se radi o front ili backend dijelu aplikacij
  
- Izbjegavati ponavljanje upita
  - ▣ Ponavljanje upita uzrokuje teško održavanje koda
  
- PL/SQL samo kada je potreban
  - ▣ SQL brži, lakši

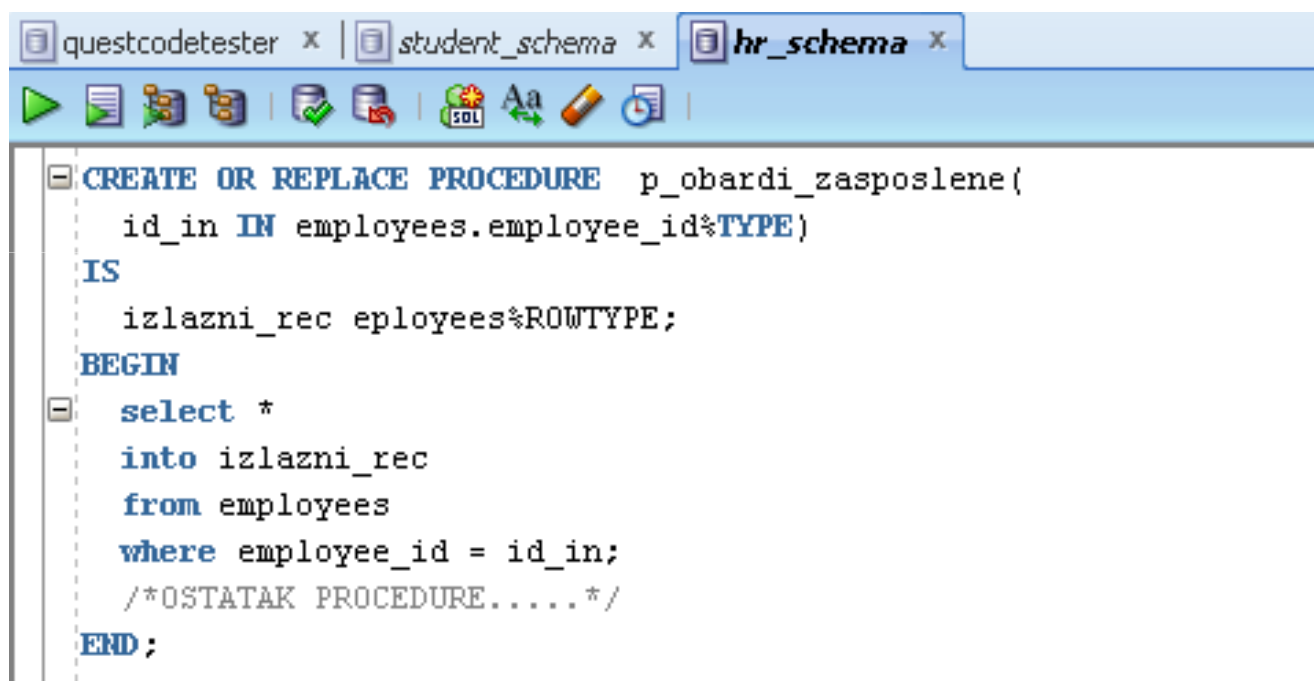
# Rješenje

- SQL u srednjem sloju smješten u funkcije i procedure, lako za održavati i ponovno koristiti



# Primjer

## □ Procedura obrade zaposlenika

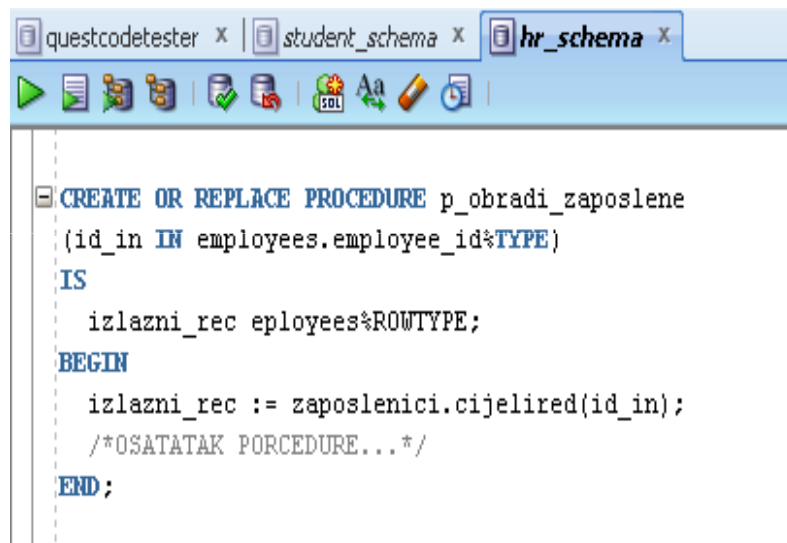


The screenshot shows a SQL IDE window with three tabs: 'questcodetester', 'student\_schema', and 'hr\_schema'. The 'hr\_schema' tab is active. The main area displays the following SQL code:

```
CREATE OR REPLACE PROCEDURE p_obardi_zasposlene(  
    id_in IN employees.employee_id%TYPE)  
IS  
    izlazni_rec employees%ROWTYPE;  
BEGIN  
    select *  
    into izlazni_rec  
    from employees  
    where employee_id = id_in;  
    /*OSTATAK PROCEDURE.....*/  
END;
```

# Primjer

## □ Izdvojen SQL u zasebnu funkciju



```
questcodetester x | student_schema x | hr_schema x
[Icons: Run, Stop, Refresh, Save, Undo, Redo, SQL, Aa, Erase, Print]

CREATE OR REPLACE PROCEDURE p_obradi_zaposlene
(id_in IN employees.employee_id%TYPE)
IS
    izlazni_rec employees%ROWTYPE;
BEGIN
    izlazni_rec := zaposlenici.cijelired(id_in);
    /*OSATATAK PORCEDURE...*/
END;
```

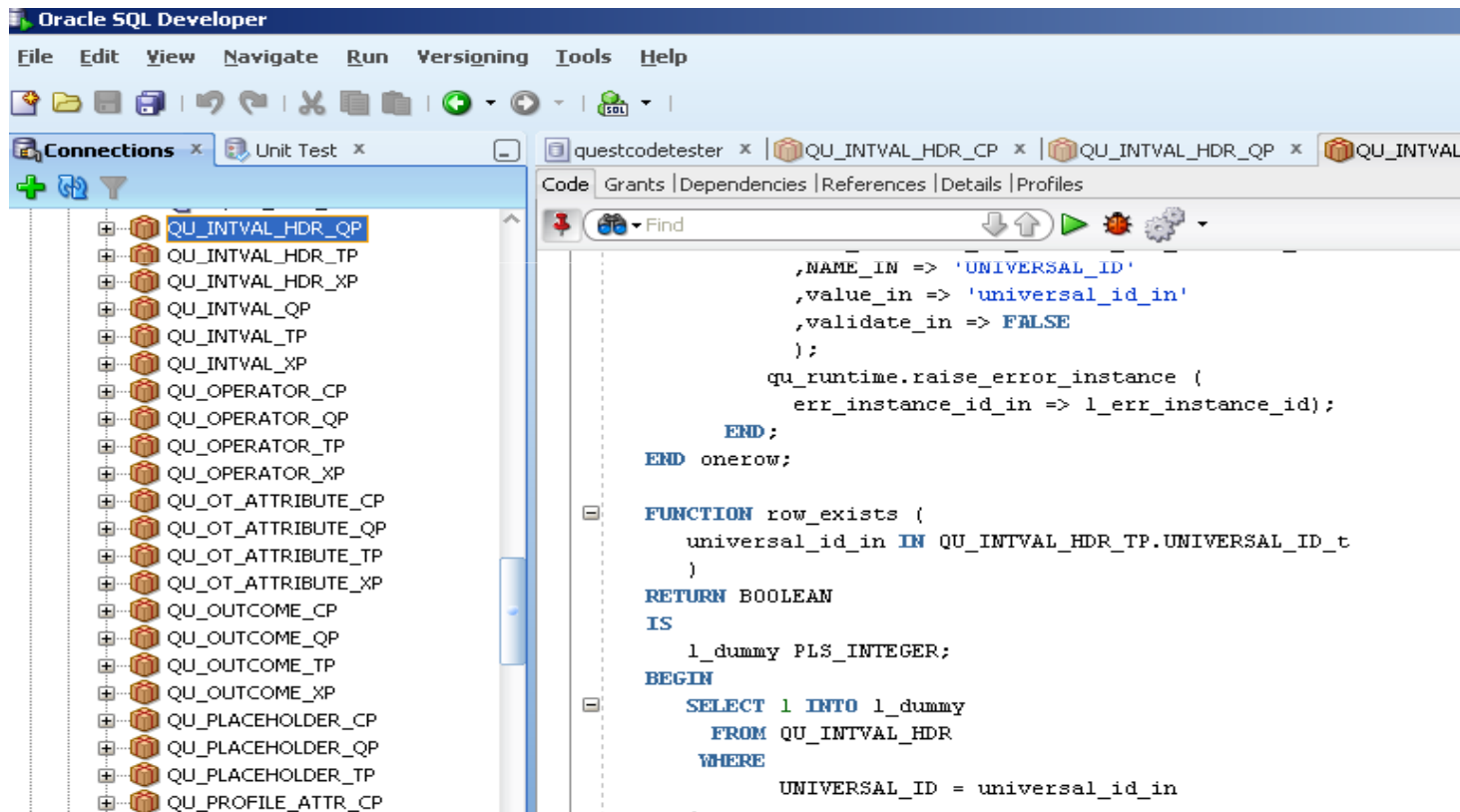
```
CREATE OR REPLACE PACKAGE zaposlenici
IS
    FUNCTION cijelired(id_in IN employees.employee_id%type)
        RETURN employees%ROWTYPE
        --result_cache;
END;

CREATE OR REPLACE PACKAGE BODY zaposlenici
IS
    FUNCTION cijelired(id_in IN employees.employee_id%type)
        RETURN employees%ROWTYPE
        result_cache relies_on(employees)
    IS
        izlazni_rec employees%ROWTYPE;
    BEGIN
        select *
        into izlazni_rec
        from employees
        where employee_id = id_in;
        /*OSTATAK PROCEDURE.....*/
    END;
```



# Primjer

## □ Formiranje paketa



The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a tree view of database objects, with 'QU\_INTVAL\_HDR\_QP' selected. The main editor window shows the 'Code' tab with the following SQL code:

```
        ,NAME_IN => 'UNIVERSAL_ID'  
        ,value_in => 'universal_id_in'  
        ,validate_in => FALSE  
    );  
    qu_runtime.raise_error_instance (  
        err_instance_id_in => l_err_instance_id);  
    END;  
END onerow;  
  
FUNCTION row_exists (  
    universal_id_in IN QU_INTVAL_HDR_TP.UNIVERSAL_ID_t  
)  
RETURN BOOLEAN  
IS  
    l_dummy PLS_INTEGER;  
BEGIN  
    SELECT 1 INTO l_dummy  
    FROM QU_INTVAL_HDR  
    WHERE  
        UNIVERSAL_ID = universal_id_in  
    .
```

# Primjer

- Upotreba enkapsuliranog SQLa kod reporta

# Zaključak

- Mijenjamo implementaciju aplikacije s minimalnom intervencijom u kodu
- Poboljšana obrada run-time grešaka vezanih uz SQL
- Moguća bolja implementacija poslovnih pravila

# Zaključak

- SQL se ne može izbjeći u aplikacijama na Oracle bazama
- Izbjegavati SQL izraze u aplikacijskoj logici aplikacija
- Sakriti SQL iz PL\SQL APIa