DEBRECENI EGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT

Dinamikus weblapok készítése Oracle PL/SQL nyelven

Témavezető: Dr. L. Nagy Éva
számítástechnikai munkatárs

Készítette: Tóth János
programozó matematikus

DEBRECEN
2007

1. Bevezetés

A Web fejlődésével egyre nagyobb az igény arra, hogy a különböző érdeklődésű, eltérő anyanyelvű felhasználók gyorsan érhessék el azokat az információkat, amelyek érdeklik őket, és ne kelljen az idejüket olyan dokumentumok megtekintésére pazarolni, amelyek nem, vagy csak kis részben érdeklik. A testre szabott web-szolgáltatások, például portál szolgáltatások ezt valósítják meg. A felhasználó elkészítheti a saját profilját, amelyben közli a szolgáltatóval, hogy mely információk érdeklik, ezt a web-szolgáltató tárolja, a felhasználóhoz pedig válogatott információkat küld el. Az ilyen web-szolgáltatások komoly méretű és összetett adatbázisokon alapulnak, a felhasználó részére a weboldalakat a beérkező kérésre reagálva, a profilnak megfelelően generálják, vagyis (majdnem) minden kérésre új dokumentumot hoznak létre. Ezek a dinamikus weboldalak. A dinamikus HTML a Netscape újítása volt, válasz arra az igényre, mely egyre nagyobb interaktivitást követelt a webes dokumentumok létrehozásakor. A HTML nyelvet, amelyben a web-dokumentumokat létrehozzuk, olyan kiegészítésekkel látták el, amelyek lehetővé tették a webes megjelenés programozását, animációkat, a fent említett testreszabást, automatikus installációt, stb. Ezek az eszközök: a Java, a JavaScript, a VBScript és a stíluslapok (CSS, Cascaded Style Sheets), amelyek bár eltérően viselkedhetnek az egyes web-böngészőkben, de alkalmasak a weboldalak egyénibbé tételére.

A dolgozatban a dinamikus weblapok Oracle PL/SQL nyelven való elkészítésével foglalkozok. Témaválasztásom fő oka dinamikus weblapok egyre nagyobb jelentősége. A dinamikus weblapok készítésének 2 módját mutatom be: az egyik a PL/SQL Web Toolkit alapeszközök segítségével, a másik a PL/SQL Server Page (PSP) technika. Részletesebben a PSP technika lesz tárgyalva. Mindkét technikát egy egyszerű példával mutatom be, amely a dolgozat utolsó fejezetében található.

2. A World Wide Web története, alapfogalmak

Először is a Web megszületéséről, majd pár alapfogalomról ejtenék szót.

A World Wide Web gyakorlatilag egy elosztott, platformfüggetlen információhalmaz. A Web 1990-ben született meg, amikor a CERN (Európai Részecskefizikai Kutatóközpont) egyik informatikusa, Tim Berners-Lee, hozzáfogott egy olyan kódrendszer - protokoll - megalkotásához, melynek segítségével az egyes számítógépek más számítógépekkel kapcsolatba léphetnek anélkül, hogy foglalkozniuk kellene a különböző adatbázisok összeférhetetlenségéből és a nehézkes bejelentkezési procedúrákból adódó problémákkal. A Web első, belső kutatói használatra szánt változata a CERN fizikusainak munkáját tette könnyebbé: adatokat, eredményeket, dokumentumokat oszthattak meg egymással a saját hálózatukon belül. A kutatóközpontból kifelé irányuló kapcsolat (például a CERN és a FermiLab között) és így az információ más tudósokkal való megosztása ekkor még nehéz feladatnak bizonyult. Berners-Lee-t 1991 januárjában pár hétre meghívták a Fermilabba. A látogatás ideje egybeesett egy hipertext konferenciával, ami Berners-Lee ötleteinek igen jó táptalajt szolgáltatott. Berners-Lee 1992-ben visszatért a Fermilabba. Ott-tartózkodása alatt született meg az első kapcsolat a CERN kiszolgáló számítógépével. Ezek után Berners-Lee elkezdett a World Wide Web számára protokollokat kidolgozni, amelyek segítségével nem csak a fizikusok kommunikálhattak egymással, hanem a világ minden tájáról elérhetővé váltak az információk.

Az információk elérését a következő mechanizmusok segítségével biztosítják a felhasználó számára:

URL (v. URI) - *Uniform Resource Locator (Identifier)*: egy általános szabálygyűjtemény az erőforrások megnevezésére. Minden erőforrás a WEB-en egy címmel rendelkezik, amelynek formátumát határozza meg az URL. Egy URL három részből áll:

1. az erőforrás eléréséhez használt módozat (pl.: `http, ftp, mailto`)
2. az erőforrást tároló számítógép címe (pl.: `www.szamitogep.hu`)
3. az erőforrás neve, elérési útként megadva (pl.: `…/data/data1101.html`)

HTTP - *HyperText Transfer Protocol*: protokoll, amely segítségével ezeket az erőforrásokat el lehet érni, a hypertext hálózaton való átvitelére szolgál. Amikor a böngésző segítségével egy távoli WEB-helyhez kapcsolódunk, akkor programunk a WEB-kiszolgálótól (szerver) lekéri az általunk megcímzett dokumentumot, erőforrást. A kiszolgáló erre elküldi a

megcímzett erőforrást a böngészőnek, amely aztán azt értelmezi és megjeleníti. A kiszolgáló és böngésző közti kommunikáció a HTTP protokoll segítségével zajlik.

HTML - *HyperText Markup Language*: Szabványosított, platformfüggetlen hypertext leíró nyelv. A HTML-fájlok egyszerű ASCII-szövegfájlok (leírókódok formájában megadott) beágyazott kódokkal, amelyek a formázást és a hiperszöveges hivatkozásokat jelölik. A HTML a következő lehetőségeket biztosítja a felhasználó számára:

- online (állandóan elérhető, számítógépen tárolt) dokumentumok létrehozása, címekkel, szöveggel, táblázatokkal, fotókkal, stb.
- online információk lekérése hypertext linkeken keresztül
- form-ok használata, távoli számítógépek által nyújtott szolgáltatások igénybevételére, mint például információ keresése, termékek megrendelése, stb.
- videoklipek, zene/hang és más alkalmazások csatolása a dokumentumokhoz.

Berners-Lee az SGML sablonjára építve tervezte meg a HTML-t. A HTML 0-s verziója a dokumentum tartalmára vonatkozó címkéket, valamint hiperhivatkozásokhoz, címsorokhoz, bekezdésekhez, listákhoz és menütételekhez használható jelölési definíciókat foglalt magában. A HTML 1-es verziója megtartotta a HTML 0-s verzió összes tulajdonságát, és kiegészítette azokat a sorokba illeszthető képek támogatásával, valamint a különböző karakterformázó képességekkel (pl. vastagítás, döntés). A HTML 2-es verziója ugyancsak megtartotta az előző verziók tulajdonságait, kiegészítve azokat a form-ok (űrlapok) létrehozásának lehetőségével. A HTML 3-as és 3.2-es verziók az előző verziók jellemzőit tovább bővítették az ábrák, táblázatok és vezérlőelemek képességeinek kiszélesítésével ill. az appletek, scriptek és színek támogatásával. A HTML 4.0 verzió 1997 végén jelent meg, mint hivatalos ajánlás. A megjelenéstől fogva a felhasználói programoknak és a dokumentumok szerzőinek ajánlatos az új verziót használni ill. az új verzió szerinti dokumentumokat előállítani, azonban a korábbi verziókkal (2.0 és 3.2) való kompatibilitás megőrzésének érdekében a régebbi elemeket is támogatniuk kell. A HTML 4.0 a következő mechanizmusokkal egészíti ill. bővíti ki a korábbi HTML verziót: stíluslapok, scripting, frame-ek, objektumok beágyazása, továbbfejlesztett szöveg és táblázatirány meghatározás, továbbfejlesztett táblázatok, form-ok és fogyatékos felhasználók számára elérhetőség.

XML - *Extensible Markup Language:* bővíthető leírókódnyelv. Olyan meta-leírókódnyelv, amelyek segítségével olyan szöveges formátumokat állíthatunk elő, amelyek alkalmasak adatok strukturált leírására. Strukturált adatok alatt olyan dolgokat értünk, mint egy táblázat,

címjegyzék, konfigurációs paraméterek, üzleti tranzakciók vagy műszaki rajzok. Az XML a számítógép számára megkönnyíti az adatok generálását, olvasását és biztosítja, hogy az adatszerkezet egyértelmű legyen. Elkerüli a programozási nyelvekben gyakran előforduló csapdákat: az XML bővíthető, platform-független és támogatja a nemzetköziesítést és a lokalizációt. Teljes egészében az Unicode-on alapul.

Ahogy a HTML is, az XML is az SGML-ből leszármaztatott jelölőnyelv. Annak ellenére, hogy mindkét jelölőnyelv ősatyja ugyanazon nyelv, mégis alapjaikban különböznek. Míg a HTML megadott elemhalmazból épül fel, addig az XML-ben saját magunk hozzuk létre az egyes elemeket. A HTML-hez hasonlóan az XML is tagokat (szavak '<' és '>' jelek között) és attribútumokat (név="érték" formátumban) használ. Azonban míg a HTML előre definiálja, hogy az egyes tagok és attribútumok mit jelentsenek, s hogy a tagok közötti szöveg a böngészőben miképp jelenjen meg, addig az XML a tagokat csak az egyes adatcsoportok elválasztására használja, az adatok értelmezését meghagyja az alkalmazások számára, amelyek az XML-t olvassák. A HTML-től eltérően az XML szabályai sokkal szigorúbbak. Egy lefelejtett lezáró tag vagy egy attribútum hiányzó idézőjelei az XML fájlt használhatatlanná teszik, míg a HTML-ben ilyen "könnyelműségek" megengedettek, sőt, esetenként kifejezetten engedélyezettek. Az XML lehetőséget nyújt arra, hogy új dokumentum-formátumokat meglévő formátumok újrahasznosításával hozzunk létre. Mivel két, egymástól függetlenül fejlesztett formátum esetlegesen használhatja ugyanazokat a tagokat vagy attribútumokat ugyanazokkal a nevekkel, ezekre ügyelni kell a formátumok kombinálásakor. Az elnevezési problémák kiküszöbölése céljából az XML az ún. névtér mechanizmust hívja segítségül.

Az XML fejlesztése 1996-ban kezdődött, és 1998 februárja óta már hivatalos W3C ajánlás, amiből esetleg arra is lehet következtetni, hogy ez egy meglehetősen kiforratlan technológia. Az XML fejlesztői csupán annyit csináltak, hogy vették az SGML legjobb részeit, hozzávéve a HTML-lel kapcsolatos tapasztalataikat, és előállítottak valamit, ami semmivel sem kevésbé hatékony, mint az SGML, ám mégis jóval szabályosabb és egyszerűbben használható.

Dinamikus weblap: Ezek alkotják az alapját az interaktív, adatbázis alapú webes információs rendszereknek. A weblapok dinamikusan vannak előállítva az input adatok és az adatbázis tartalma alapján.

3. Oracle HTTP Szerver, PL/SQL Gateway, mod_plsql

3.1 Oracle HTTP Szerver és PL/SQL Gateway

Az Oracle HTTP Szerver az Oracle adatbázis Apache infrastruktúrán alapuló, megbízható Web szerver komponense. Különböző részekből áll, amelyek ugyanazon folyamaton belül futnak. Ezek a komponensek a sajátosságaik terjedelmes listáját nyújtják, amit az Oracle HTTP Szerver szolgáltat, amikor lekezeli a kliens kéréseit.

- **HTTP Listener**: Az Oracle HTTP Szerver egy Apache HTTP figyelőn alapul, amelynek segítségével kiszolgálja a kilens kéréseket. Egy HTTP szerver figyelője lekezeli a beérkező kéréseket és elirányítja őket a megfelelő feldolgozó egységhez.
- **Modulok**: a modulok megvalósítják és kibővítik az Oracle HTTP Szerver alapvető funkcionalitásait, és biztosítják az integrációt az Oracle HTTP Szerver és más Oracle adatbázis komponensek között. Az alap Apache modulok közül sok megtalálható az Oracle HTTP Szerverben, plusz az Oracle még beépített különböző belső modulokat.
- **Perl Értelmező**: egy Perl futtatási környezet beépítve az Oracle HTTP Szerverbe a mod_perl-en keresztül.

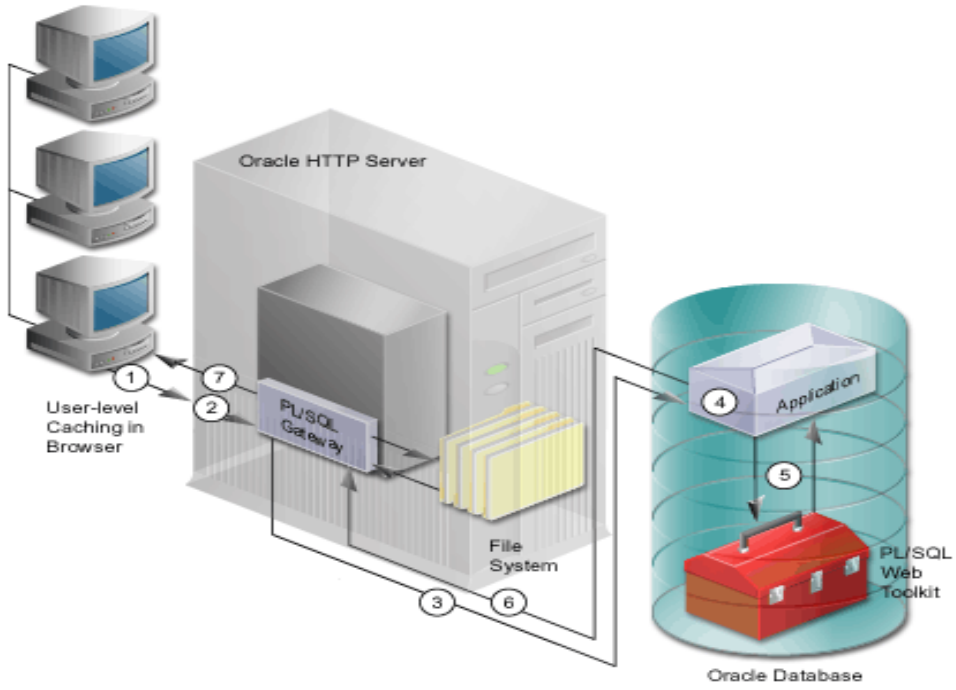
Az Oracle HTTP Szerver kezelését az Oracle Process Manager and Notification Server (OPMN) végzi. Mindig szükség van az OPMN használatára, ha el akarjuk indítani, le akarjuk állítani, vagy újra akarjuk indítani az Oracle HTTP Szervert. Az Oracle HTTP Szerver indítására a `startproc`, leállítására a `stopproc`, újraindítására `restartproc` parancsot használjuk.

Az Oracle HTTP Szerver Unix alatt az `ORACLE_HOME/Apache` könyvtárba, Windows alatt az `ORACLE_HOME\Apache` könyvtárba van installálva. Az `Apache` könyvtár a legfelső szinten helyezkedik el az `ORACLE_HOME` alatt. Ez alkönyvtárakat tartalmaz modulok konfigurálására, mint például a `mod_plsql`. Az `Apache`-on belül van még egy `Apache` könyvtár, ez az Oracle HTTP Szerver alap könyvtára. A fő konfigurációs fájl a `httpd.conf`. Ezt a fájlt más konfigurációs fájlokkal egyetemben a szerver használja, és a következő az elérési útvonala:

- UNIX: `ORACLE_HOME/Apache/Apache/conf`
- Windows: `ORACLE_HOME\Apache\Apache\conf`

Egy PL/SQL Web alkalmazás adatbázisban tárolt eljárásokból áll, amelyek kapcsolatba kerülnek a Web böngészővel az Oracle HTTP Szerver segítségével. Az Oracle HTTP Szerver PL/SQL Gateway része tartalmaz egy mod_plsql eszközt, amely lehetővé teszi ezt a kommunikációt: leképezi a böngésző *DAD_leíró* (Database Access Descriptor, Adatbázis-hozzáférési leíró) virtuális útvonallal jelzett kérését a *DAD_leíró* -hoz tartozó konfigurációs információk alapján az Oracle adatbázisban tárolt *PSP_eljárás* (PL/SQL Server Page, speciális PL/SQL eljárás, mely weblapot állít elő) hívásra, majd a *PSP_eljárás* eredményét a böngésző megkapja.

A következő ábra azt mutatja meg, hogy mi történik akkor, amikor a szerver kap egy kérést a klienstől:



1. Az Oracle HTTP Szerver fogadja a böngésző kérését.
2. Az Oracle HTTP Szerver továbbítja a kérést a PL/SQL Gateway -nek.
3. A PL/SQL Gateway felhasználva a megadott *DAD_leíró* konfigurációs információit, kapcsolatba lép az adatbázissal.
4. A PL/SQL Gateway előkészíti a *PSP_eljárás* hívását, majd meghívja a megadott sémában lévő PL/SQL eljárást.
5. A PL/SQL eljárás generál egy HTML lapot felhasználva az adatbázis adatokat és a PL/SQL Web Toolkit eszközöket.

6. A HTML lapot visszakapja a PL/SQL Gateway.
7. Az Oracle HTTP Szerver pedig visszaküldi azt a kliens böngészőjének.

Az eljárás, amelyet a mod_plsql segítségül hív, visszatér a HTTP válasszal a klienshez. Leegyszerűsítve: ez a folyamat, a mod_plsql be van építve a PL/SQL Web Toolkit-be, amely csomagok halmazát tartalmazza, amelyet owa csomagoknak nevezünk. Ezeket a csomagokat arra használjuk a tárolt eljárásunkban, hogy információkat kapjunk a kérésről, HTML tagokat készítsünk, és visszatérjünk fejrész információkkal a klienshez.

3.2 mod_plsql

Az Oracle HTTP Szervert kapcsolja az Oracle adatbázishoz; engedélyezi, hogy Web alkalmazásokat készítsünk tárolt eljárások használatával. Ez egy Oracle modul, amelyet a PL/SQL Gateway tartalmaz. Ahhoz, hogy hozzáférjünk egy Web-engedélyezett PL/SQL alkalmazáshoz, konfigurálni kell az adatbázis-hozzáférési leírót.

3.2.1. Adatbázis-hozzáférési leíró (DAD)

Mindegyik mod_plsql kérés össze van kapcsolva DAD leíróval, amely konfigurációs értékek halmaza. Ezek az értékek azt mondják meg, hogy a mod_plsql hogyan kapcsolódik az adatbázishoz, hogy teljesíteni tudja a beérkező HTTP kéréseket. Egy DAD fontos információkat tartalmaz, mint például:

- az adatbázis alias nevét (Oracle Net szolgáltatásnév)
- egy kapcsoló sztring, ha az adatbázis távoli
- egy eljárás a dokumentumok fel- és letöltésére

A DAD-ban meg kell adni a felhasználónévre és jelszóra vonatkozó információkat is. Ha ez nincs megadva, akkor a felhasználónak kell bevinni egy felhasználónevet és egy jelszót, amikor az URL-t meghívja.

A DAD létrehozásának lépései:

1) Szerkesszük az `ORACLE_HOME/Apache/modplsql/conf/dads.conf` állományt, amely a DAD konfigurációs állománya.

2) Adjunk hozzá egy DAD-ot, ahol a DAD a következő formátumú:

a) az Oracle HTTP Szerver `<Location>` direktívája definiál egy virtuális útvonalat, amely segítségével hozzáférünk a PL/SQL Web alkalmazásunkhoz. Ez a direktíva direktívák csoportját veszi körül, amelynek neve `Location`.

b) tartalmazhat egy vagy több mod_plsql specifikus direktívát. Például:

```
PlsqlDatabaseUsername név
PlsqlDatabasePassword jelszó
PlsqlDatabaseConnectString kapcsoló_sztring
PlsqlAuthenticationMode Basic
```

c) `</Location>` direktíva, amely lezárja a direktíváknak ezen csoportját.

3) Mentsük el a változtatásokat.

4) A DAD jelszó eltüntetése a `dadTool.pl` futtatásával, amely az `ORACLE_HOME/Apache/modplsql/conf` könyvtárban található.

5) Az Oracle HTTP Szerver újraindítása.

Egy új adatbázis-hozzáférési leíró felvételét az Oracle rendszergazda végzi a már említett `dads.conf` fájl szerkesztésével. Abban a nyító `<Location` */DAD_leíró*`>` és a záró `</Location>` direktívák között van megadva a *DAD_leíró* neve és a hozzá tartozó konfigurációs paraméterek. Az oktatáshoz a **`/pls/hallgatodad`** nevű *DAD_leíró* lett létrehozva (a **`/pls`** prefixet az Oracle9i még automatikusan és kötelezően használta, a 10g-ben már elhagyható lenne).

```
<Location /pls/hallgatodad >
...
PlsqlDatabaseUsername        hallgato
PlsqlDatabasePassword        ********
PlsqlDatabaseConnectString   oracle.inf.unideb.hu:1521
PlsqlAuthenticationMode      Basic
PlsqlDefaultPage
...
</Location>
```

Ezt a *DAD_leírót* használva az URL-ben, automatikusan megtörténik az oktatáshoz használt Oracle adatbázishoz való kapcsolódás és a bejelentkezés **`hallgato/********`** felhasználóként. A **`hallgato`** sémája üres, azt nem is fogjuk használni. A *PSP_eljárásokat* a saját **`tothjani`** sémámban hozom létre, majd futtatási jogot adok hozzá **`hallgato`**-nak.

3.2.2. mod_plsql meghívása

Ha Web böngészőben meg akarjuk hívni a mod_plsql-t, akkor írjuk be az URL-t a következő formában:

```
protocoll://web_server_név[:port]/DAD_leíró/[[!][séma.][csomag.]PSP_eljárás[?par1=ért1&par2=ért2 ...]]
```

A POST, GET és HEAD metódus:

A POST, GET és HEAD metódus tájékoztatja a böngészőket a HTTP protokollban, hogy hogyan adják át az alkalmazásoknak a paramétereket. A paraméter adatok HTML formok által generálódnak. A mod_plsql alkalmazások ezeket a metódusokat tudják használni. Mindegyik metódus olyan biztonságos, mint az alapját alkotó szállítási protokoll (HTTP vagy HTTPS).

1) Amikor POST metódust használunk, akkor a paraméterek a kérdés törzsében adódnak át. Ha nagy mennyiségű paraméteradatot adunk át a szervernek, akkor általában a POST metódust használjuk.
2) Amikor GET metódust használunk, akkor a paraméterek egy kérdés sztring használatával adódnak át. Ennek a metódusnak az a korlátozása, hogy az értékek hossza a név-érték párokban nem haladhatja meg a maximum hosszúságot egy környezeti változó miatt, ami az alapot alkotó operációs rendszer által van megadva. Ráadásul az operációs rendszereknek van egy limitje, hogy hány környezeti változót tudunk definiálni.
3) Amikor a HEAD metódust használjuk, akkor ugyanazok a funkcionalitások vannak, mint a GET metódusnál. Az egyedüli különbség, hogy csak a HTTP status line (állapotvonal) és a HTTP fejrész adódik vissza a böngészőnek, a tartalom nem.
4) Kevert mód: a mod_plsql-ben néhány paramétert átadhatunk kérdés sztringben és a maradékot POST adatként. Például, ha van egy

   ```
   pelda(a varchar2, b number)
   ```

   eljárásunk, és át akarjuk adni a “v” és “1” értékeket ‘a’-nak illetve ‘b’-nek, akkor háromféle módon tudjuk létrehozni az URL-t:

   a. Az összes érték a kérdés sztring részeként van megadva:

      ```
      http://host:port/pls/DAD/pelda?a=v&b=1
      ```

   b. Az összes érték a POST adat részeként van megadva:

      ```
      http://host:port/pls/DAD/pelda, POST data="a=v&b=1"
      ```

   c. Néhány paraméter URL-ben van megadva és a maradék a POST adatban:

      ```
      http://host:port/pls/DAD/pelda?a=v, POST data="b=1"
      ```

Tranzakció mód:

Amikor egy eljáráshívás számára feldolgoztunk egy URL kérést, és valamilyen hiba lép fel, akkor a mod_plsql végrehajt egy visszagörgetést. Egyébként végrehajtódik egy

véglegesítés. Ez a mechanizmus egy tranzakciónak nem engedi meg, hogy több HTTP kérést is átíveljen.

A támogatott adattípusok:

A HTTP csak karaktersorozatot támogat, ezért a mod_plsql a PL/SQL adattípusok következő részhalmazát támogatja:

- NUMBER
- VARCHAR2
- TABLE OF NUMBER
- TABLE OF VARCHAR2

A rekord nem támogatott.

3.2.3. Paraméterátadás

A mod_plsql a paraméterátadások következő fajtáit támogatja:

- **paraméterátadás név szerint**: megengedett a paraméterek túlterhelése. A túlterhelés megengedi több alprogramnak (eljárások vagy függvények), hogy ugyanazt a nevet használják, de különbözzenek a paraméterek számában, sorrendjében vagy az adattípusban. Amikor meghívunk egy túlterhelt alprogramot, a PL/SQL fordító határozza meg, hogy az alprogramot milyen fajta adattípus átadással kell meghívni.

A PL/SQL megengedi a helyi vagy csomag szintű alprogramok túlterhelését. Önálló alprogram nem terhelhető túl.

A túlterhelt alprogramok neveinek különbözőnek kell lenni, ha azt akarjuk, hogy ugyanannyi legyen a paraméterszámuk. A HTML adat nincs összekapcsolva az adattípussal, ezért a mod_plsql nem fogja tudni, hogy az alprogram melyik verzióját kell meghívni. Például: habár a PL/SQL megengedi, hogy két eljárásban ugyanazokat a paraméterneveket használjuk, hiba fog fellépni, ha ezt mod_plsql-lel használjuk.

```
-- legális PL/SQL-ben, de nem az a mod_plsql számára
CREATE PACKAGE my_pkg AS
PROCEDURE my_proc (val IN VARCHAR2);
PROCEDURE my_proc (val IN NUMBER);
END my_pkg;
```

A paraméterneveknek különbözni kell, hogy elkerüljük a hibát. Például:

```
-- legális PL/SQL-ben és mod_plsql számára is működik
CREATE PACKAGE my_pkg AS
PROCEDURE my_proc (valvc2 IN VARCHAR2);
PROCEDURE my_proc (valnum IN NUMBER);
END my_pkg;
```

- **rugalmas paraméterátadás**: az eljárások előtt egy ! karakter szerepel. Lehetnek HTML formjaink, amelyből a felhasználók bármennyi elemet ki tudnak választani. Ha ezek az elemek különböznek a nevükben, akkor készíthetünk túlterhelt eljárásokat, hogy lekezeljék mindegyik lehetséges kombinációt. Beszúrhatunk rejtett form elemeket, amelyek biztosítják, hogy a kérdés sztringben lévő nevek minden pillanatban konzisztensek legyenek, függetlenül attól, hogy a felhasználó mely neveket választotta ki. A mod_plsql ezt a műveletet könnyebbé teszi, hogy támogatja a rugalmas paraméterátadás kezelését HTML formokban, ahol a felhasználók bármennyi elemet ki tudnak választani.

Ha használni akarjuk a rugalmas paraméterátadást URL-alapú eljáráshívásban, akkor az URL-ben az eljárás neve elé egy felkiáltó jelet kell raknunk. Használhatunk kettő vagy négy paramétert. A kétparaméteres interfész javított teljesítményt nyújt a mod_plsql-lel. A négyparaméteres interfész a kompatibilitás miatt támogatott.

Kétparaméteres interfész:

```
procedure [proc_name]
(name_array IN [array_type],
value_array IN [array_type]);
```

Példa:

```
http://host:port/pls/dad/!user.proc?x=john&y=10&z=doe
```

A felkiáltójel prefix mondja meg a mod_plsql-nek, hogy rugalmas paraméterátadást alkalmazzon. Az pedig meghívja a `user.proc` eljárást, és átadja neki a következő argumentumokat:

```
name_array ==> ('x', 'y', 'z')
value_array ==> ('john', '10', 'doe')
```

Négyparaméteres interfész:

```
procedure [proc_name]
(num_entires IN NUMBER,
name_array IN [array_type],
value_array IN [array_type],
reserved in [array_type]);
```

A `num_entire` a kérdés sztringben lévő név-érték párok száma, a `reserved` pedig tartalék.

Példa, ahol az „x” kétszer szerepel:

```
http://host:port/pls/dad/!user.package.proc?x=a&y=b&x=c
```

A felkiáltójel prefix mondja meg a mod_plsql-nek, hogy rugalmas paraméterátadást alkalmazzon. Az pedig meghívja a `user.package.proc` eljárást, és átadja neki a következő argumentumokat:

```
num_entries ==> 3
```

```
name_array ==> ('x', 'y', 'x');
value_array ==> ('a', 'b', 'c')
reserved ==> ()
```

- **nagy (32K fölött) paraméterek átadása**: a nevek vagy értékek nagyobbak lehetnek, mint 32 kilobyte.

3.2.4 Megszorítások mod_plsql-ben

A következő megszorítások léteznek mod_plsql-ben:

- a HTTP cookie fejrészének maximális hossza 32000 byte lehet. Ha ennél nagyobb, akkor hiba keletkezik. Ez a limit megfelel a PL/SQL VARCHAR2 limitjének.
- a HTTP cookie-n belül bármely egyedül álló cookie maximális hossza 3990 byte. Ha ennél nagyobb, akkor hiba keletkezik.
- a PL/SQL Gateway nem támogatja az eljárások hívását OUT paraméterekkel. ORA-6502 hibát generál eredményként. Az ajánlott felfogás, hogy ne hívjunk meg olyan eljárást, amely OUT paraméterekkel rendelkezik.
- a név-érték párok - amelyek a PL/SQL eljárásnak adódhatnak át - maximális száma 2000.
- a mod_plsql 2000-ben limitálja azon paraméterek számát, amelyek átadódhatnak egy egyedi eljárásnak.
- a mod_plsql 32000 byte-ban limitálja az egyszerű paraméterek méretét, amelyek átadódhatnak egy egyedi eljárásnak.

Követelmények ellenőrzése:
Mielőtt futtatnánk a mod_plsql-t, ellenőrizzük a következő követelményeket:

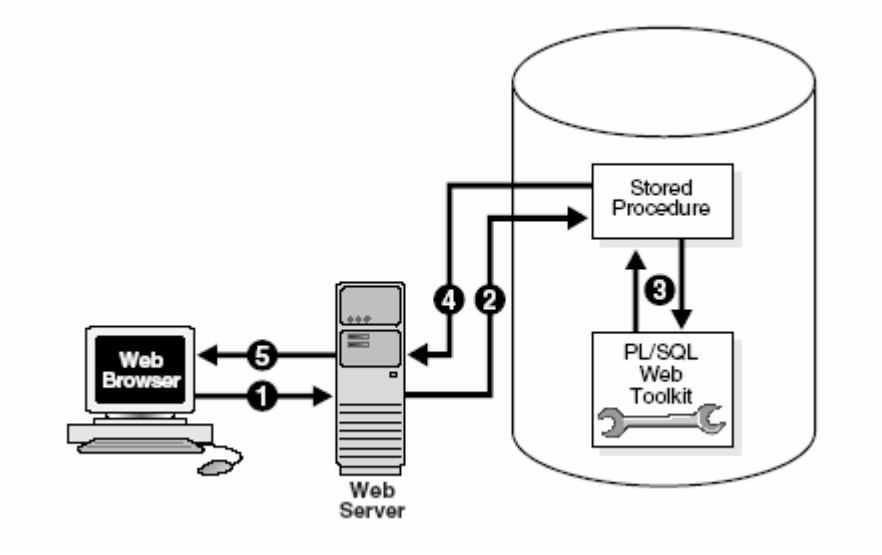
- szükség van egy SYS felhasználói jelszóra az adatbázisban, ahol megtervezzük azon PL/SQL Web Toolkit csomagok betöltését, amelyre a mod_plsql-nek szüksége van
- az adatbázisnak futnia kell, amelyben megtervezzük a mod_plsql kapcsolódását
- javasolt, hogy az OWA csomagok telepítve legyenek az adatbázisban, legalább a 9.0.4.0.1-es verzió

3.3. PL/SQL Web alkalmazás

Általában egy PL/SQL-ben írt web alkalmazás tárolt eljárások halmaza, amelyek a web böngészővel egymásra hatnak a HTTP-n keresztül. Egy PL/SQL web alkalmazás program folyamata hasonló egy CGI Perl szkriptéhez. A fejlesztők gyakran használnak CGI szkripteket, hogy dinamikus weboldalakat hozzanak létre, de néhány szkript hozzáférése az

Oracle adatbázishoz nem optimális. A web tartalom PL/SQL tárolt eljárásokkal való megadása az adatbázis feldolgozásának rugalmasságát nyújtja. Például: használhatunk DML-t, dinamikus SQL-t és kurzorokat.

Az alábbi ábra egy PL/SQL web alkalmazás generikus folyamatát illusztrálja:



A folyamat a következő lépéseket tartalmazza:

1. a felhasználó meglátogat egy weboldalt, követi a hypertext linket vagy a formban megadott adatot, amely azt idézi elő, hogy a böngésző egy HTTP kérést küld a HTTP szervernek az URL segítségével
2. a HTTP szerver meghív egy tárolt eljárást az Oracle adatbázisból az URL-be kódolt adatnak megfelelően. Az URL-ben lévő adat a form paramétereit adja át a tárolt eljárásnak.
3. A tárolt eljárás alprogramokat hív meg a PL/SQL Web Toolkit-ben. Tipikusan ez a HTP.PRINT, amely dinamikus weboldalakat generál. A generált weboldal függ az adatbázis tartalmától és az input paraméterektől.
4. Az alprogramok átadják a dinamikusan generált oldalt a Web szervernek.
5. A Web szerver kézbesíti az oldalt a kliensnek.

Egy web böngésző alapú alkalmazást implementálhatunk teljes egészében PL/SQL-ben az Oracle adatbázis komponensei segítségével.

4. PL/SQL Web Toolkit eszközök használata (1-es technika)

4.1. PL/SQL Web Toolkit

PL/SQL csomagok halmaza, generikus interfész, amely engedélyezi, hogy tárolt eljárásokat hívjunk meg futás közben a mod_plsql által. A böngésző kérésének megválaszolása közben egy PL/SQL eljárás az Oracle adatbázisból származó adatot a felhasználói bemenetnek megfelelően frissíti, vagy visszakeresi. Ez aztán generál egy HTTP választ a böngészőnek, általában HTML-ben, amelyet megjelenít. A Web Toolkit API engedélyezi a tárolt eljárások hatásának elvégzését, mint például a következők:

- információ beszerzése egy HTTP kérésről
- generálja a HTTP fejlécet
- beállítja a cookie-kat a böngészőben
- HTML oldalakat generál

Az egyik leggyakrabban használt PL/SQL Web Toolkit csomag a HTP. Ennek segítségével tudjuk generálni a HTML tagokat.

4.2. HTML kimenet generálása HTP csomagok segítségével

A PL/SQL Web alkalmazások eljáráshívásokat használnak, hogy generálják az összes HTML tagot. Ezek az eljárások a PL/SQL Web Toolkit csomagok része, amelyek az Oracle adatbázisból származnak. A következő példa azt illusztrálja, hogy hogyan generálunk egy egyszerű HTML oldalt HTP eljárások hívásával, amelyek mindegyike megfelel egy HTML tag-nek.

```
CREATE OR REPLACE PROCEDURE html_oldal
IS
BEGIN
HTP.HTMLOPEN; -- generálja a <HTML> -t
HTP.HEADOPEN; -- generálja a <HEAD> -t
HTP.TITLE('Hello'); -- generálja a <TITLE>Hello</TITLE> -t
HTP.HEADCLOSE; -- generálja a </HEAD>
-- generálja a <BODY TEXT="#000000" BGCOLOR="#FFFFFF"> -t
HTP.BODYOPEN( cattributes => 'TEXT="#000000" BGCOLOR="#FFFFFF"');
-- generálja a <H1>Cím a HTML fájlban</H1> -t
HTP.HEADER(1, 'Cím a HTML fájlban');
HTP.PARA; -- generálja a <P> -t
HTP.PRINT('Egy kis szöveg a HTML fájlban');
HTP.BODYCLOSE; -- generálja a </BODY> -t
HTP.HTMLCLOSE; -- generálja a </HTML> -t
END;
```

```
/
SHOW ERRORS
GRANT EXECUTE ON html_oldal TO hallgato;
```

Futtatása böngészőből:

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.html_oldal

Egy másik alternatíva a HTP.PRINT függvény használata, amely a szöveget és a tagot együtt írja ki. A következő példa ezt illusztrálja:

```
CREATE OR REPLACE PROCEDURE html_oldal2
IS
BEGIN
HTP.PRINT('<html>');
HTP.PRINT('<head>');
HTP.PRINT('<meta http-equiv="Content-Type" content="text/html">');
HTP.PRINT('<title>A HTML File címe</title>');
HTP.PRINT('</head>');
HTP.PRINT('<body TEXT="#ffffff" BGCOLOR="#000000">');
HTP.PRINT('<h1>Cím a HTML File-ban</h1>');
HTP.PRINT('<p>Szöveg a HTML file-ban.');
HTP.PRINT('</body>');
HTP.PRINT('</html>');
END;
/
SHOW ERRORS
GRANT EXECUTE ON html_oldal2 TO hallgato;
```

Futtatása böngészőből:

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.html_oldal2

4.3. Paraméterek átadása egy PL/SQL Web alkalmazásnak

Ahhoz, hogy különböző szituációk széles választékát kínálja, egy Web alkalmazásnak interaktívnak kell lenni. Hogy fenntartsa az internetezők figyelmét, a felhasználók választásainak egyszerűnek kell lenni. A paraméterek átadásának fő metódusa a következő:

- HTML űrlap (form) tagok használata. A felhasználó kitölti az űrlapot egy weboldalon, és az összes adat és választás átadódik egy tárolt eljárásnak, amikor a felhasználó a Submit (elküld) gombra kattint az oldalon.
- URL-be vannak kódolva a paraméterek. A felhasználó rákattint egy linkre, és az előredefiniált paraméterek halmaza átadódik egy tárolt eljárásnak. A weboldalakon

általában minden olyan lehetőséghez, amire a felhasználónak szüksége lehet, külön linkek szoktak lenni.

4.4. Hálózati műveletek elvégzése PL/SQL tárolt eljárásokon belül

Míg a beépített PL/SQL jellemzők hagyományos adatbázis műveletekre vannak fókuszálva, addig az Oracle adatbázis olyan csomagokat nyújt, amelyek utat nyitnak az internet programozás felé a PL/SQL programozók számára.

Email küldése PL/SQL-ből:

Egy PL/SQL programból vagy tárolt eljárásból e-mailt küldeni az UTL_SMTP csomag segítségével tudunk. A következő kód azt illusztrálja, hogy hogyan használjuk az SMTP csomagot arra, hogy egy alkalmazásból e-mailt küldjünk. Az alkalmazás kapcsolódik az SMTP szerverhez a 25-ös porton keresztül és elküld egy egyszerű szöveges üzenetet.

```
PROCEDURE send_test_message
IS
mailhost VARCHAR2(64) := 'mailhost.fictional-domain.com';
sender VARCHAR2(64) := 'me@fictional-domain.com';
recipient VARCHAR2(64) := 'you@fictional-domain.com';
mail_conn utl_smtp.connection;
BEGIN
mail_conn := utl_smtp.open_connection(mailhost, 25);
utl_smtp.helo(mail_conn, mailhost);
utl_smtp.mail(mail_conn, sender);
utl_smtp.rcpt(mail_conn, recipient);
-- If we had the message in a single string, we could collapse
-- open_data(), write_data(), and close_data() into a single call to data().
utl_smtp.open_data(mail_conn);
utl_smtp.write_data(mail_conn, 'This is a test message.' || chr(13));
utl_smtp.write_data(mail_conn, 'This is line 2.' || chr(13));
utl_smtp.close_data(mail_conn);
utl_smtp.quit(mail_conn);
EXCEPTION
WHEN OTHERS THEN
-- Insert error-handling code here
NULL;
END;
```

Egy host nevének vagy címének megszerzése PL/SQL-ből:

PL/SQL programban vagy tárolt eljárásban meg tudjuk határozni egy helyi gépnek a host nevét vagy egy adott host névhez tartozó IP címet az UTL_INADDR csomag használatával.

TCP/IP kapcsolat PL/SQL-ből:

TCP/IP kapcsolatot tudunk létrehozni a hálózaton lévő gépekhez, és a megfelelő socket-eket olvasni és írni tudjuk az UTL_TCP csomag használatával.

HTTP URL tartalom visszakeresése PL/SQL-ből:

Vissza tudjuk keresni egy HTTP URL tartalmait az UTL_HTTP csomag használatával. A tartalmak általában a HTML szöveg formjában vannak, de lehetnek sima szövegben, JPEG képben, vagy egyéb olyan fájlban, amelyet letölthetünk a Web szerverről. Az UTL_HTTP csomag a következőket tudja:

- vezérli a HTTP szolgálathasználó részletes adatait, beleértve a cookie-kat, proxy szervereket, azonosítókat és jelszókat védett oldalakhoz, és CGI paramétereket a GET vagy POST metóduson keresztül.
- URL-t készít és értelmezi az UTL_HTTP csomag használatával
- gyorsítja a többszöri hozzáférést ugyanahhoz a Web oldalhoz a HTTP 1.1 ismétlődő kapcsolat használatával.

Általában a fejlesztők Java-t vagy Perl-t használnak ezekhez a műveletekhez.

5. PL/SQL szerveroldalak fejlesztése, PSP technika (2-es technika)

A PL/SQL szerveroldalak (PL/SQL Server Pages [PSP]) szerveroldali szkriptek, amelyek a weboldalakon belül tartalmazzák a dinamikus tartalmat, beleértve a SQL kérések eredményeit. Ezeket a speciális PL/SQL eljárásokat az Oracle adatbázisban kell tárolni (szólóban vagy csomagban), híváskor az adatbázis-szerveren futnak le és weblapot, vagy weblap részletet állítanak elő. Alkalmasan felparaméterezett eljárásokkal dinamikus weblapok készíthetők. Mivel a PL/SQL nyelvben közvetlenül használhatók az SQL-DML utasítások, az EXECUTE IMMEDIATE natív dinamikus SQL utasítással pedig bármilyen SQL utasítás kiadható, így komplett Web-alkalmazások készíthetők. Az Oracle számos csomagot (PL/SQL Web Toolkit) biztosít ehhez a munkához.

A létrehozott PSP eljárást a `loadpsp` operációs rendszer szintű paranccsal tudjuk lefordítani, és az Oracle adatbázisban tárolni.

Az összetevőket telepítve, a PL/SQL szerveroldalak révén a következő előnyök származnak:

- a szerveroldalak a legegyszerűbb módja, hogy professzionális weboldalt készítsünk, amely tartalmaz adatbázis által generált összetevőt, és ez a PL/SQL segítségével a fejlesztők számára egy egyszerű és barátságos mód
- a PSP sokkal kényelmesebb, mint a HTP és HTF csomagok használata, vagyis hogy egy HTML tartalmat sorról sorra kiírjunk
- a feldolgozás az adatbázis szerveren van végrehajtva, a kliens böngésző egy egyszerű HTML oldalt kap speciális szkript tagok nélkül
- a hálózati forgalom hatékony, mert a PSP használata minimalizálja az adatbázishoz kapcsolódás számát
- könnyen tudunk összetevőket írni és követni egy gyors, ismétlődő fejlődési folyamatot. Karbantarthatjuk a szoftver központi vezérlését egyetlen web böngészővel, ami a kliens gépen van.

5.1. A PL/SQL szerveroldalak kifejlesztésének és telepítésének előfeltétele

Ahhoz, hogy kifejlesszünk és telepítsünk PL/SQL szerveroldalakat, szükségünk van a következő előfeltételekre:

- ahhoz, hogy írjunk egy PL/SQL szerveroldalt, szükségünk van egy szövegszerkesztőre vagy HTML szerkesztő eszközre, amivel megírjuk a szkriptet
- ahhoz, hogy betöltsük a PL/SQL szerveroldalt, szükség van:
    - hozzáférésre ahhoz az Oracle adatbázishoz, ahol futtatni akarjuk a szerveroldalt
    - futtatási jogra a `loadpsp` parancssori programra, amely a `$ORACLE_HOME/bin` könyvtárban van elhelyezve
- a szerveroldalak telepítéséhez használnunk kell a mod_plsql-t.

5.2. A PSP szkript és a HTP csomag

Engedélyezni tudjuk a felhasználók számára, hogy PL/SQL programegységeket futtassanak HTTP-n keresztül a következő esetekben:

- olyan HTML oldal írásakor, melybe PL/SQL kód van beágyazva és ennek PL/SQL szerveroldalra való lefordításakor. Eljárásokat lehet meghívni PL/SQL Web Toolkit-ból, de a HTML kimenetből nem generálódik minden sor.
- egy teljes tárolt eljárás írásakor, amely HTML-t állít elő a HTP csomag hívásakor a PL/SQL Web Toolkit-ben. Ez a technika a „HTML kimenet generálása PL/SQL segítségével” részben van leírva.

Választanunk kell, hogy milyen technikát alkalmazunk a Web alkalmazásunk megírásakor. A technikák közötti választás kulcstényezői a következők lehetnek:

- milyen forrást használunk kezdőpontként?
    - Ha olyan HTML oldalunk van, amelynek nagy a törzs része, és azt akarjuk, hogy legyen benne dinamikus tartalom, akkor válasszuk a PSP-t.
    - Ha olyan PL/SQL kódunk van, amelynek nagy a törzs része, és amely formázott kimenetet produkál, akkor sokkal kényelmesebb, ha meghívjuk a PL/SQL Web Toolkit HTP csomagját, hogy generálja a HTML tagokat.
- mi a leggyorsabb és legkényelmesebb szerkesztői környezet?
    - Ha a munka legnagyobb részében HTML szerkesztő eszközöket használunk, akkor használjuk a PSP technikát.
    - Ha olyan szerkesztő eszközöket használunk, amely PL/SQL kódot produkál, akkor kevésbé kényelmes megoldás a PSP használata.

5.3. PSP és egyéb szkript megoldások

A szkript megoldások lehetnek kliens vagy szerver oldalon. A JavaScript az egyik legnépszerűbb kliens oldali szkript nyelv. A PSP teljes mértékben támogatja a JavaScriptet. Mivel néhány tagot változatlanul át tudunk adni egy PL/SQL szerveroldalon keresztül a böngészőnek, ezért a JavaScriptet vagy más kliens oldali szkript kódot bele tudjuk foglalni egy PL/SQL szerveroldalba.

A Java Serves Pages (JSP) és az Active Server Pages (ASP) a két legnépszerűbb szerveroldali szkript megoldás.

- a Java szerveroldalak nagyon hasonlítanak a PSP oldalakhoz; a Java szervletek pedig a PL/SQL csomagokhoz. A PSP ugyanazt a szkript tag szintaktikát használja, mint a JSP.
- a PSP által használt szintaktika különbözik az ASP által használttól, habár nem teljesen.

5.4. PL/SQL szerveroldalak írása

Ahhoz, hogy PL/SQL szerveroldalakat írjunk, tudnunk kell már létező weboldalakat vagy tárolt eljárásokat indítani. Valamelyik módon, kis kiegészítéssel vagy változtatással létre tudunk hozni dinamikus weboldalakat, amelyek adatbázis műveleteket hajtanak végre és kiírják az eredményt.

A PL/SQL szerveroldal fájl kiterjesztésének `.psp`-nek kell lenni. Magában foglalja azt a tartalmat, amit választunk; szöveggel és tagokkal, teletűzdelve PSP direktívákkal, deklarációkkal stb. Egy szerveroldalt a következő alakokban tudunk megadni:

- a legegyszerűbb esetben ez egy HTML fájl. PL/SQL szerveroldalként szerkesztjük, tárolt eljárásokat készítünk, és kimenetként ugyanolyan HTML fájlt kapunk.
- a legbonyolultabb esetben ez egy PL/SQL eljárás, amely generálja a weboldal összes tartalmát, beleértve a „title”, „body” és „head” tag-eket.
- általános esetben ez keveréke a HTML-nek (ez nyújtja az oldal statikus részét) és a PL/SQL-nek (ez nyújtja a dinamikus tartalmat).

A PSP direktívák és deklarációk helye és rendezettsége általában nem fontos. Ez csak akkor válik fontossá, amikor más fájl is benne van. A karbantartás kényelme miatt javasolt, hogy helyezzük a direktívákat és deklarációkat együtt a fájl elejére.

Néhány speciális tag:

- `<%@ page … %>` : oldal direktíva. A PL/SQL szerveroldal jellemzői.

- `<%@ parameter … %>` : paramétert definiáló direktíva. Tartalmazza a paraméter nevét, és opcionálisan a típust és default értéket.
- `<%@ plsql … %>` : eljárás direktíva. Tárolt eljárás neve a PSP fájl által előállítva.
- `<%@ include … %>` : tartalmazási direktíva. Egy fájl neve, amely a PSP fájl egy speciális pontján meg lett nyitva.
- `<%! … %>` : deklarációs blokk. Globális változók deklarációja PL/SQL szintaxis szerint.
- `<%= … %>` : kifejezés blokk. PL/SQL kifejezés értékének behelyettesítése HTML szkriptbe.
- `<%-- … --%>` : megjegyzés PSP szkriptben.

5.5. Alapvető szerveroldal jellemzők meghatározása

Használjuk a `<%@ page … %>` direktívát, hogy meghatározhassuk a PL/SQL szerveroldal jellemzőit, mint például:

- milyen szkript nyelvet használunk
- mi az információ típusa, amit előállítunk stb.

A következő kód a `page` direktíva szintaktikáját mutatja (fontos, hogy a `contentType` és `errorPage` attribútumok eset-érzékenyek):

```
<%@ page [language="PL/SQL"] [contentType="content type string"]
charset="encoding" [errorPage="file.psp"] %>
```

A szkript nyelv meghatározása:

Ahhoz, hogy egy fájlt PL/SQL szerveroldalként azonosítsunk, tartalmaznia kell valahol a következő direktívát:

```
<%@ page language="PL/SQL" %>
```

Ez a direktíva más szkriptleíró környezetekkel is kompatibilis.

Adat visszaadása a kliensnek:

Ahhoz, hogy meghatározzuk a kliensböngészőnek visszaadott adat típusát, következő alapesetek állnak rendelkezésre:

1. visszatérés HTML-ként
2. visszatérés XML-ként, szövegként vagy egyéb dokumentumtípusként
3. visszatérés olyan oldalként, amely különböző karaktertípusokat tartalmaz

1. Visszatérés HTML-ként: egy PL/SQL szerveroldal PL/SQL része körül van véve speciális elhatároló jelekkel. Az összes többi tartalom betű szerint – beleértve a szóközöket -

át lesz adva a böngészőnek. Ahhoz, hogy megjelenítse a szöveget vagy HTML tagokat, ugyanúgy írjuk meg, ahogy egy normál weboldalt. Nincs szükség semmilyen kimeneti függvény meghívására. Példa:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>Példa</title>
</head>
<body>
…
<% FOR rekord IN dolgozo_kurzor LOOP %>
<tr>
<td> <% rekord.vezeteknev %> </td>
<td> <% rekord.keresztnev %> </td>
</tr>
<% END LOOP %>
…
</body>
</html>
```

2. Visszatérés XML-ként, szövegként vagy egyéb dokumentumtípusként: alapértelmezetten a PL/SQL Gateway a HTML dokumentumokat, mint fájlokat továbbítja, így a böngésző a HTML tagokat értelmezi. Ha azt akarjuk, hogy a böngésző a dokumentumot XML-ként, egyszerű szövegként (nem formázott), vagy valamely más dokumentumtípusként értelmezze, akkor a következő direktívát kell alkalmaznunk:

```
<%@ page contentType="MIMEtype" %>
```

Az attribútum neve eset-érzékeny, így legyünk pontosak a `contentType` helyes leírásában, a kis- és nagybetűk különböznek. Szúrjunk be text/html, text/xml, text/plain, image/jpeg vagy egyéb MIME típust, hogy a böngésző vagy más kliens program felismerje. A következő példa az Excel táblázatkezelőre vonatkozó direktívát mutatja:

```
<%@ page contentType="application/vnd.ms-excel" %>
```

3. Visszatérés olyan oldalként, amely különböző karaktertípusokat tartalmaz: alapértelmezetten a PL/SQL Gateway fájlokat továbbít a PL/SQL Gateway által definiált karakterkészlettel. Ahhoz, hogy konvertáljuk az adatot különböző karakterkészletű böngészők számára, a következő direktívát kell beszúrni:

```
<%@ page charset="encoding" %>
```

Pontosabban Shift_JIS, Big5, UTF-8 vagy egyéb kódolások léteznek, amelyet a kliens program felismer.

A karaktertípus beállításokat konfigurálnunk kell a PL/SQL Gateway adatbázis-hozzáférési leírójában (DAD). A felhasználók kiválaszthatják a megfelelő kódolást a böngészőjükben, hogy a megjelenített adatokat helyesen lássák. Például Japánban az adatbázis karaktertípusa lehet az EUC kódolás, de a Web böngészőben a Shift_JIS kódolást kell beállítani.

Szkript hibák kezelése:

Amikor PL/SQL szerveroldalakat írunk, figyelmesnek kell lennünk a következő hibatípusoknál:

- *HTML szintaktikai hibák:* a HTML hibák kezelése a böngésző feladata.
- *PL/SQL szintaktikai hibák:* ha a PL/SQL kódban szintaktikai hiba keletkezik, akkor a `loadpsp` segédprogram megáll és kiírja a sor számát, az oszlop számát és egy rövid üzenetet a hibáról. A hibát ki kell javítani, mielőtt tovább haladnánk. Fontos, hogy a tárolt eljárás bármely korábbi verziója törlődhet, ha megpróbáljuk lecserélni egy olyan szkripttel, ami szintaktikai hibát tartalmaz.
- *Futási hibák:* hogy kezelni tudjuk azokat az adatbázis hibákat, amelyek akkor jelennek meg, mikor a szkript fut, a PSP fájlba be kell szúrnunk PL/SQL kivételkezelő kódot és a nem kezelt kivételek számára létre kell hozni egy speciális PL/SQL szerveroldalt. Használjuk a `<%@ page … %>` direktíva `errorPage` attribútumát, hogy specifikáljuk ennek az oldalnak a nevét.

  A nem kezelt kivételek számára létrehozott oldal egy PL/SQL szerveroldal `.psp` kiterjesztéssel. A hibakezelő eljárás nem kap semmilyen paramétert, így ahhoz, hogy meghatározza a hiba okát, meg kell hívni az SQLCODE és SQLERRM függvényeket.

A következő példa egy direktívát mutat, amely specifikálja az `errors.psp`-t olyan oldalként, amely lefut, amikor hibák keletkeznek:

```
<%@ page language="PL/SQL" contentType="text/html" errorPage="errors.psp" %>
```

5.6. Felhasználói bemenet elfogadása

Hogy beállítsuk a PL/SQL szerveroldalak számára a paraméter átadást, egy direktívát kell beszúrni a következő szintaktikával:

```
<%@ plsql parameter="paraméter neve" [type="PL/SQL típus"]
[default="érték"] %>
```

Alapértelmezetten a paraméter típusa VARCHAR2. Ha egy ettől különböző típust akarunk használni, akkor kell a `type="PL/SQL típus"` attribútumot beleírni a direktívába, mint például ahogyan a következő példa mutatja:

```
<%@ plsql parameter=”p_belepo” type=”NUMBER” %>
```

Hogy beállítsunk egy alapértelmezett értéket, a `default=”kifejezés”` attribútumot be kell szúrni a direktívába. Egy PL/SQL állításban az attribútum közvetlenül helyettesítődik a hozzá tartozó értékkel, így minden jelsorozatnak idézőjelben kell szerepelnie, és használhatunk speciális értékeket, mint a null, ahogyan a következő példában is:

```
<%@ plsql parameter=”p_belepo” default=”null” %>
```

A felhasználói bemenet az URL-ben kódolva jelenik meg, amelyet a HTML oldalból nyerünk.

Példa:

```
<%@ page language="PL/SQL" %>
<%@ page contentType="text/html" %>
<%@ plsql parameter="p_Mozi_kod" default="null" type="VARCHAR2" %>
<%@ plsql procedure="mozikat_kilistaz" %>
<%!
CURSOR mozi_kurzor IS
SELECT Varos, Nev
FROM Mozi
WHERE Mozi_kod = p_Mozi_kod
ORDER BY Nev;
%>
```

Ha a PL/SQL Gateway konfigurálva volt, akkor futtatni tudjuk az eljárásokat `http://www.host.com/pls/proc_name` meghívásával, ahol a `proc_name` az eljárás neve, és ezután a `p_Mozi_kod`–nak át tudunk adni SZO01-et paraméterként a következő módon:

```
http://www.host.com/pls/mozikat_kilistaz?p_Mozi_kod=SZO01
```

5.7. PL/SQL tárolt eljárások elnevezése

A szerveren belül mindegyik legmagasabb szintű PL/SQL szerveroldal megegyezik egy tárolt eljárással. Amikor a `loadpsp`-vel betöltjük az oldalt, a segédprogram létrehoz egy PL/SQL tárolt eljárást. Alapértelmezetten az eljárás ugyanazt a nevet kapja, mint a PSP szkript kivéve, hogy a `.psp` kiterjesztés lemarad. Így, ha a szkriptünk neve `select_musor_psp.psp`, akkor alapértelmezetten a segédprogram egy olyan eljárást hoz létre, amelynek a neve `select_musor_psp`.

Ha az eljárásnak egy olyan nevet akarunk adni, amelyik különbözik a szkript nevétől, akkor szúrjuk be a következő direktívát, ahol a `’eljárásnév’` az eljárás neve:

```
<%@ plsql procedure=”eljárásnév” %>
```

Így bármit is adunk meg névként a psp fájlunknak, az eljárás neve az lesz, ami a `'eljárásnév'` helyén áll. Fontos, hogy ez az eljárás neve, és nem a psp szkript neve, amelyet az URL-be szúrunk be.

5.8. Más fájlok tartalmának beszúrása

Be tudjuk állítani egy beszúró mechanizmussal, hogy más fájlok tartalma is jelen legyen, általában valamilyen statikus HTML tartalom vagy PL/SQL szkript kód. Szúrjuk be a következő direktívát arra a pontra, ahol más fájl tartalmát szeretnénk megjeleníteni, lecserélve a fájlnevet a beszúrandó fájl nevére:

```
<%@ include file="fájlnév" %>
```

A beszúrt fájl kiterjesztésének másnak kell lenni, mint `.psp`. Pontosan ugyanazt a nevet kell megadni mindkét `include` direktívában és a `loadpsp` parancsban, beszúrva tetszés szerinti relatív utat, mint például `../include/`.

Mivel a fájlok fel vannak dolgozva, amikor az adatbázisba betöltjük a tárolt eljárást, ezért a helyettesítés csak egyszer lesz elvégezve, nem annyiszor, ahányszor az oldal kiszolgál. Ezért a változtatások a beszúrt fájlokban – amelyek megtörténnek, miután az oldal betöltődött az adatbázisba - nincsenek megjelenítve, amikor az eljárás lefutott.

Jegyezzük meg a beszúrt fájlok következő jellemzőit:

- bármilyen nevet és kiterjesztést használhatunk a beszúrt állományok számára.
- ha a beszúrt fájlok PL/SQL szkript kódot tartalmaznak, akkor nincs szükségük saját direktíva halmazra, hogy megállapítsák az eljárás nevét, karakterkészletét és így tovább.
- amikor a fájlok nevét megadjuk a `loadpsp` segédprogramnak, akkor az összes beszúrt fájl nevét is be kell szúrnunk. A beszúrt fájlok nevét hamarabb határozzuk meg, mint a `.psp` fájlok nevét.

5.9. Globális változók deklarálása PSP szkriptben

Használjuk a `<%! … %>` direktívát, hogy definiálni tudjunk PL/SQL változók halmazát, amelyek az oldalon mindenhonnan láthatók, nemcsak a következő BEGIN / END blokkon belül. Ez az elem általában több soron keresztül helyezkedik el, egyedi PL/SQL változó deklarációkkal, és mindegyik végén pontosvesszővel. Ennek a direktívának a szintaktikája a következő:

```
<%! PL/SQL deklaráció;
[ PL/SQL deklaráció;] … %>
```

A szokásos PL/SQL szintaktika blokkon belül megengedett. Gyorsításként meg van engedve, hogy elhagyjuk a `DECLARE` kulcsszót. A következő egy kurzor deklarációja:

```
<%!
CURSOR mozi_kurzor IS
SELECT Varos, Nev
FROM Mozi
ORDER BY Nev;
%>
```

Meghatározhatunk többszörös deklarációs blokkokat; belsőleg, az összes össze lesz vonva egy egyszerű blokkba, amikor a PSP fájl létre lesz hozva egy tárolt eljárásként.

Használhatunk explicit `DECLARE` blokkokat a `<% … %>` határolójeleken belül. Ezek a deklarációk csak a következő BEGIN / END blokkban láthatók.

5.10. Futtatható utasítások meghatározása PSP szkriptben

A `<% … %>` kódblokk direktívát tudjuk használni PL/SQL utasítások halmazának futtatására, amikor a tárolt eljárás fut. A következő kód a futtatható utasítások szintaktikáját mutatja:

```
<% PL/SQL utasítás;
[ PL/SQL utasítás; ] … %>
```

Ez az elem tipikusan több sorba írható, az egyes PL/SQL utasítások végén pontosvesszővel. Az utasítások lehetnek teljes blokkok, ahogyan a következő példában, amelyik meghívja az `OWA_UTIL.TABLEPRINT` eljárást:

```
<% OWA_UTIL.TABLEPRINT(CTABLE => 'hr.dolgozo', CATTRIBUTES => 'border=2',
CCOLUMNS => 'vezeteknev,keresztnev', CCLAUSES => 'WHERE dolgozo_azonosito >
100'); %>
```

Az utasításoknak szintén lehet IF / THEN / ELSE vagy BEGIN / END része. Amikor egy kódblokkot szétosztunk többszörös direktívákba, közéjük berakhatunk HTML vagy más direktívákat, és a középső darabok feltételek mellett futnak le, amikor lefut a tárolt eljárás. A következő kódrészlet ezt a technikát mutatja be:

```
<% FOR ITEM IN (SELECT product_name, list_price, catalog_url
FROM product_information
WHERE list_price IS NOT NULL
ORDER BY list_price DESC) LOOP
IF item.list_price > p_minprice THEN
v_color := '#CCCCFF';
```

```
ELSE
v_color := '#CCCCCC';
END IF;
%>
<TR BGCOLOR="<%= v_color %>">
<TD><A HREF="<%= item.catalog_url %>"><%= item.product_name %></A></TD>
<TD><BIG><%= item.list_price %></BIG></TD>
</TR>
<% END LOOP; %>
```

A blokkon belül az összes szokásos PL/SQL szintaktika megengedett. A későbbiek során az összes deklaráció rendelkezésre áll a kódban.

5.11. Kifejezés eredményének helyettesítése PSP szkriptben

Egy kifejezés direktíva kimenete egy egyszerű PL/SQL kifejezés, mint például egy sztring, aritmetikai kifejezés, függvényhívás vagy ezek kombinációja. Az eredmény a HTML oldal ezen pontján helyettesítődik egy sztringgel a tárolt eljárás által előállítva. A kifejezés eredményének sztringnek kell lenni vagy sztringgé konvertálhatónak. Néhány típusnál nem lehet implicit módon konvertálni, mint például a DATE, ekkor át kell adni az értéket a `PL/SQL TO CHAR` függvénynek.

E kifejezés direktíva szintaktikája a következő, ahol a *kifejezés*-t helyettesíteni kell a kívánt kifejezéssel:

```
<%= kifejezés %>
```

Fontos, hogy a PL/SQL kifejezés végén nincs szükség pontosvesszőre.

```
<%= rekord.Nev %>
```

A konkatenációt ugyanúgy tudjuk használni, mint PL/SQL-ben, a `||` jel használatával. A következő direktíva egy példa a konkatenációra:

```
<%= 'A mozi neve: ' | |  rekord.Nev %>
```

5.12. Idézőjeles és escape sztringek PSP szkriptben

A PSP attribútumok kettős idézőjelet használnak, hogy adatot határoljanak körül. Amikor meghatározott értékű PSP attribútumokat használunk PL/SQL műveletekben, ezek pontosan ugyanúgy lesznek átadva, ahogy a PSP fájlban meghatároztuk őket. Így, ha a PL/SQL egyszeres idézőjeles sztringet kér, akkor pontosan meg kell határozni a sztringet egyszeres idézőjelekkel körülvéve, és utána körülvenni az egészet kétszeres idézőjelekkel.

Például, a PL/SQL eljárásunk egy változó alapértelmezett értékeként a `Debrecen Plaza Cinema City` sztringet használja. Azért, hogy a sztringet használhassuk PL/SQL-ben, körbe

kell venni egyszeres idézőjelekkel így: 'Debrecen Plaza Cinema City'. Ha megadjuk ezt az egyszeres idézőjeles sztringet egy PSP direktíva default attribútumában, akkor körbe kell venni kétszeres idézőjellel, ahogyan a következő példa mutatja:

```
<%@ plsql parameter="p_Nev" default="'Debrecen Plaza Cinema City'" %>
```

Meg lehet azt is tenni, hogy aposztrófos sztringbe beleágyazunk egy másik aposztrófos sztringet. Ebben az esetben a beágyazott sztringet körbe kell venni a \' karaktersorozattal. Például:

```
<%@ plsql parameter="p_Jatekos" default="'Kovács \'Kokó\' István'" %>
```

A legtöbb karaktert és karaktersorozatot beszúrhajuk a PSP fájlba anélkül, hogy a PSP loader kicserélné őket. Ha a %> jelsorozatot akarjuk beszúrni, akkor azt ily módon kell: %\>. A <% jelsorozatot pedig <\% módon. Például:

```
<%= 'A %\> jelsorozat használata a következő szkript nyelvben: ' || nyelv_neve %>
<%= 'A <\% jelsorozat használata a következő szkript nyelvben: ' || nyelv_neve %>
```

5.13 Megjegyzések beszúrása PSP szkriptbe

Ha a PL/SQL szerveroldal HTML részébe megjegyzést akarunk beszúrni, hogy a PSP forráskód könnyebben olvasható legyen, akkor használjuk a következő szintaktikát:

```
<%-- PSP megjegyzés szövege --%>
```

A megjegyzések nem jelennek meg a HTML kimenetben, és a USER_OBJECTS-ben, amikor lekérdezzük a PL/SQL forráskódot.

Ha olyan megjegyzést akarunk írni, amely látható legyen a HTML kimenetben és a USER_OBJECTS forrásban, akkor helyezzük a megjegyzést a HTML-be és használjuk a normál HTML megjegyzés szintaktikáját:

```
<!-- HTML megjegyzés szövege -->
```

Ha egy PSP-ben megjegyzést egy PL/SQL blokkban akarunk elhelyezni, és láthatatlanná akarjuk tenni a HTML kimenetben, de láthatóvá a USER_OBJECTS-ben, akkor használjuk a normál PL/SQL megjegyzés szintaktikáját, ahogyan a következő példában:

```
-- Megjegyzés PL/SQL kódban
```

5.14. Egy PL/SQL szerveroldal betöltése az adatbázisba

Használjuk a loadpsp segédprogramot, amely az $ORACLE_HOME/bin –ben található, hogy be tudjunk tölteni egy vagy több PSP fájlt az adatbázisba tárolt eljárásként. Mindegyik .psp fájl megfelel egy tárolt eljárásnak. Az oldalak egy lépésben lesznek lefordítva és

betöltve, meggyorsítva ezzel a feldolgozási folyamatot. A `loadpsp` segédprogram szintaktikája a következő:

```
loadpsp [ –replace ] –user username/password[@connect_string] [
include_file_name … ] [ error_file_name] psp_file_name …
```

Ha a `CREATE OR REPLACE` szintakszissal akarunk létrehozni eljárásokat, akkor használjuk a `–replace` jelzőt. Amikor egy PSP fájlt betöltünk, a betöltő a következő tevékenységeket végzi el:

1. Bejelentkezik az adatbázisba a megadott felhasználói névvel, jelszóval és hálózati szolgáltatás névvel
2. Létrehozza a tárolt eljárást a felhasználó sémájában

Szúrjuk be az összes beszúrandó fájl nevét, mielőtt felsoroljuk a PL/SQL szerveroldalak neveit. Szintén szúrjuk be annak a fájlnak a nevét, amelyet a `page` direktíva `errorPage` attribútumában adtunk meg. Ezeknek a fájlneveknek a `loadpsp` parancssorában pontosan meg kell egyezni azokkal a nevekkel, amelyeket a PSP fájl `include` és `page` direktívájában adtunk meg, esetleg beszúrhatunk valamilyen relatív utat, mint például `../include/` . Példa egy PSP betöltő parancsra:

```
loadpsp –replace –user hr/hr@orcl banner.inc error.psp display_order.psp
```

Jegyezzük meg az előző példa jellegzetességeit:

- a tárolt eljárás az `orcl` adatbázisban jön létre. Az adatbázishoz a `hr` felhasználó `hr` jelszavával fér hozzá, amikor a tárolt eljárás létre lesz hozva és akkor, amikor a tárolt eljárás futtatva lesz.
- a `banner.inc` egy fájl, amely tartalmaz sablonszöveget és szkript kódot, amely be van szúrva a `.psp` fájlba. A beépítés akkor történik meg, amikor a PSP fájl betöltődik az adatbázisba, nem amikor a tárolt eljárás lefut.
- az `error.psp` egy fájl, amely tartalmaz kódot, szöveget, vagy mindkettőt, és amely akkor dolgozódik fel, amikor előfordul egy nem kezelt kivétel.
- a `display_order.psp` tartalmazza a fő kódot és szöveget a weboldal számára. Alapértelmezetten a megegyező tárolt eljárás neve `display_order`.

PSP forráskód lekérdezése:

Miután egy PSP fájl betöltődött, meg tudjuk nézni a forráskódot az adatszótárban a `USER_SOURCE` vagy `DBA_SOURCE` táblákban. Például, tegyük fel, hogy betöltünk egy szkriptet a következő paranccsal:

```
loadpsp –replace –user tothjani/”jelszó” select_musor_psp.psp
```

Ha belépünk az adatbázisba `tothjani` felhasználóként, akkor a következő kérdést tudjuk futtatni SQL*Plus-ban, hogy a PSP forráskódját meg tudjuk nézni.

```
SELECT * FROM user_objects WHERE object_type = 'PROCEDURE';
SELECT text FROM user_source
     WHERE name='SELECT_MUSOR_PSP' AND type='PROCEDURE';
```

A `loadpsp` által generált kód különbözik a forrásfájlban lévő kódtól. A `loadpsp` segédprogram még hozzáad egyéb kódot is, főleg hívásokat HTP csomagra. A HTP csomag generálja a HTML tagokat weboldal számára.

5.15. PL/SQL szerveroldal futtatása URL-en keresztül

Miután a PL/SQL szerveroldal átalakult egy tárolt eljárássá, az eljárást le tudjuk futtatni egy Web böngészőből a HTTP URL-en keresztül. Az URL-ben lévő virtuális út függ a PL/SQL Gateway-ben konfigurált úttól.

A tárolt eljárás paraméterei át lesznek adva a HTTP protokoll POST vagy GET metódusának. A POST metódussal a paraméterek egy HTML form-ból közvetlenül átadódnak, és nem láthatók az URL-ben. A GET metódussal a paraméterek név-érték párokként adódnak át az URL kérdés sztringjében, elválasztva & karakterekkel, a legtöbb nem alfanumerikus karakter pedig kódolt formában (például a szóköz %20-ként). Ha egy HTML formból meg akarunk hívni egy PSP oldalt, akkor a GET metódust tudjuk használni, vagy használhatunk hardkódolt (hard-coded) linket, hogy meghívjuk a tárolt eljárást megadott paraméterekkel.

A GET metódus használatával az URL szintaktikája a következőképpen néz ki:

```
http://oldalnév/sémanév/eljárásnév?paraméternév1=érték1&paraméternév2=érték2
```

A POST metódus használatával az URL szintaktikájában nem szerepelnek paraméterek:

```
http://oldalnév/sémanév/eljárásnév
```

A GET metódus formátum sokkal kényelmesebb hibakeresés szempontjából és megengedi a felhasználóknak, hogy pontosan ugyanazokat a paramétereket adják át, amikor visszatérnek az oldalra egy könyvjelzőn keresztül.

A POST metódus formátum megengedi a paraméteradatok nagyobb, állományokból álló kötetét, és alkalmas érzékeny információk átadására, amelyeket az URL-ben nem lehet megjeleníteni.

5.16. PL/SQL szerveroldalak problémáinak nyomonkövetése

Miután elkezdünk kísérletezni a PSP-vel, és miután átalakítjuk az első egyszerű oldalainkat egy sokkal bonyolultabbá, tartsuk észben ezeket az irányelveket, mikor problémákkal találkozunk:

1) az első lépés az összes PL/SQL szintaktika és PSP direktíva helyességének ellenőrzése. Ha itt egy hibát ejtünk, akkor a fájl nem fordul le.
   - bizonyosodjunk meg, hogy használunk pontosvesszőt sorok lezárására, ahol szükséges
   - ha egy értéknek idézettnek kell lenni, akkor legyen idézett. Lehet, hogy szükség van aposztrófos értékre az idézőjelesen belül.
   - a PSP direktívákban bekövetkező hibákról általában PL/SQL szintaktikai üzeneteken keresztül kapunk jelentést. Ellenőrizzük, hogy a direktíváinkban helyes szintaktikát használunk, és ezek a direktívák helyesen le vannak-e zárva.
   - a PSP atrribútumnevek eset-érzékenyek. A legtöbb kisbetűsen van megadva; a `contentType` és az `errorPage` viszont kevert módon.
2) Amikor egy URL-t használunk, hogy PSP-t kérjünk, lehetséges, hogy kapunk egy hibát, hogy a fájl nem található. Ebben az esetben a következőket tegyük:
   - győződjünk meg, hogy a helyes virtuális utat kértük, az út függ a Web Gateway konfigurálásától. Tipikusan, az út tartalmazza a host nevet, opcionálisan egy portot, a séma nevét és a tárolt eljárás nevét (`.psp` kiterjesztés nélkül).
   - ha használjuk a `-replace` opciót, amikor fordítjuk a fájlt, a tárolt eljárás régi változata törlődik. Így egy hibás fordítás után ki kell javítani a hibát vagy az oldal nem lesz elérhető. Lehetséges az is, hogy az új szkripteket egy külön sémában teszteljük, és ezután töltjük be őket abba a sémába, amelyben dolgozunk.
   - ha a fájlt egy másik fájlból másoltuk, akkor ne felejtsük el lecserélni az eljárásnév direktívákat a forrásban, hogy egyezzen az új fájlnévvel.
   - amikor kapunk egy „`fájl nem található`" hibaüzenetet, akkor bizonyosodjunk meg, hogy következő alkalommal az oldal legutolsó verzióját kérjük le. Az „`error page`" lehet, hogy a böngésző cache-ében van. Ekkor szükség van az oldal frissítésére, hogy a cache-t figyelmen kívül hagyjuk.
3) Amikor a PSP szkript fut, és az eredmények visszaérkeznek a böngészőhöz, használjuk a standard nyomkövetési technikákat, hogy ellenőrizzük és javítsuk a hibás kimenetet. A bonyolultabb rész az, hogy beállítsuk az interfészt különböző HTML formok, szkriptek, és

CGI programok között úgy, hogy a helyes értékek adódjanak át az oldalunknak. Lehetséges, hogy az oldal hibával tér vissza, mert a paraméterek nem megfelelőek. Jegyezzük meg a következő ötleteket:

- Hogy pontosan meghatározzuk mi is adódott át az oldalunknak, használjuk a `METHOD=GET`-et a form hívásában, így a paraméterek látszanak az URL-ben.
- Legyünk biztosak benne, hogy a form vagy CGI program, amelyet az oldalunk hívott meg, megfelelő számú paramétert ad át, és a formon a `NAME=` attribútum által meghatározott nevek megegyeznek a PSP fájlban található paraméter nevekkel. Ha a form beszúr rejtett bemeneti mezőket vagy használja a `NAME=` attribútumot a Submit vagy Reset gomboknál, akkor a PSP fájlban deklarálni kell azonos értékű paramétereket.
- Legyünk biztosak benne, hogy a paraméterek konvertálva vannak sztringből a megfelelő PL/SQL típusra. Például, ne szúrjunk be alfabetikus karaktereket, ha a PSP fájlban a paraméterek NUMBER-ként vannak deklarálva.
- Bizonyosodjunk meg, hogy az URL kérdés sztringje név-érték párokból áll elválasztva egyenlőség jelekkel.
- Ha sok paraméter adatot adunk át, mint például nagy sztringeket, lehetséges, hogy meghaladja az értéket, amit a `METHOD=GET`-tel át tudunk adni. A meghívó formban át tudunk váltani `METHOD=POST`-ra a PSP fájl megváltoztatása nélkül.
- Habár a `loadpsp` parancs helyesen számol be a sorok számáról, amikor szintaktikai hiba van a forrás fájlban, a megadott sorok száma fordítási hibáknál hivatkozik a forrás átalakított változatára, és nem egyezik az eredeti forrásban lévő sorok számával. Amikor hibákkal találkozunk, a várt weboldal helyett létrejön egy hibakövető, kivételkezelőkön keresztül szükség lesz a hiba meghatározására, és a hibakövetés kimenetre való kiíratására.

5.17. PL/SQL szerveroldalak beszúrása termékbe

Mielőtt beszúrnánk a PSP alkalmazásunkat termékbe, figyelembe kell venni különböző problémákat, mint használhatóság és letöltési sebesség:

- a böngészőben az oldalakat gyorssá tudjuk tenni, ha a `HEIGHT=` és `WIDTH=` attribútumokat az összes képre megadjuk. Egységesíteni lehet a képméreteket, vagy tárolni az adatbázisban a képek szélességét és magasságát az adattal vagy URL-lel egyetemeben.

- Azon böngészők számára, akik kikapcsolják a grafikákat, szúrjunk be a fontos képekhez leírást az `ALT=` attribútum használatával. A leírást a képpel együtt tárolhatjuk az adatbázisban.
- Habár egy HTML táblázat az adatok megjelenítésének egy jó módja, egy nagy táblázat lassúvá tudja tenni az alkalmazásunkat. Gyakran a felhasználó egy üres oldalt néz addig, amíg az egész táblázat be nem töltődik. Ha egy HTML táblázatban lévő adatok mennyisége nagy, gondoljuk át a kimenet több táblázatra való szétvágását.
- Ha beállítjuk a szöveget, karakterkészletet vagy a háttérszínt, teszteljük az alkalmazást a böngésző színbeállításainak különböző kombinációjával:
    - Teszteljük, mi történik, ha a böngészőben felülírjuk az előtér színét, vagy csak a háttér színét, vagy mindkettőt.
    - Általában, ha beállítunk egy színt (mint például az előtérben lévő szöveg színét), be kellene állítani az összes színt a `<BODY>` tagon keresztül, elkerülve ezzel a nehezen olvasható kombinációkat, mint például a fehér szöveg fehér háttéren.
    - Ha háttérnek egy képet használunk, határozzunk meg egy hasonló háttérszínt, hogy megfelelő kontrasztot nyújtsunk azoknak a böngészőknek a számára, akik nem töltik be a grafikákat.
    - Ha kritikus az információk közvetítése a különböző színek által, gondoljuk meg más technika használatát. Például lehetséges, hogy egy táblázatban ikont helyezzünk a speciális adatok mellé. Sok felhasználó nézheti az oldalunkat fekete-fehér színű monitoron, vagy böngészőkkel, amelyek nem jelenítenek meg különböző színeket.
- Az is megeshet, hogy az üres mezőkbe a felhasználók rossz értékeket írnak be. Ahol lehetséges, használjunk listát a választható értékek kiválasztására. Minden mezőbe beírt szöveget érvényesítsünk, mielőtt továbbítjuk az SQL-nek.

6. Mintafeladat

Ebben a fejezetben egy egyszerű mintafeladat segítségével kerül bemutatásra a fentebb tárgyalt két technika. A mintafeladat moziműsorok nyilvántartása lesz. Az első része a 4-es fejezetben tárgyalt technika lesz, amikor PL/SQL Web Toolkit eszközöket használunk a dinamikus weblap elkészítésére. A második részben pedig az 5-ös fejezetben bővebben kifejtett PSP technika található.

A feladatban 3 táblát használok:

- `mozi`, amely tartalmazza a mozik adatait
- `film`, amely tartalmazza a filmek adatait
- `musor`, amely tartalmazza a műsorokra vonatkozó adatokat

A táblák létrehozása CREATE TABLE utasítással történik:

```
CREATE TABLE mozi (
  Mozi_kod     VARCHAR2(5) PRIMARY KEY,
  Nev          VARCHAR2(40) NOT NULL,
  Varos        VARCHAR2(30) NOT NULL,
  Cim          VARCHAR2(80)
);

CREATE TABLE film (
  Filmcim      VARCHAR2(80) NOT NULL,
  Kulfoldi_cim VARCHAR2(80) NOT NULL,
  Gyartasi_ev  NUMBER(4) NOT NULL,
  Mufaj        VARCHAR2(20) NOT NULL,
  Hossz        NUMBER(3) NOT NULL,
  Szereplok    VARCHAR2(80) NOT NULL,
  CONSTRAINT film_pk PRIMARY KEY (Filmcim, Gyartasi_ev)
);

CREATE TABLE musor (
  Mozi_kod     VARCHAR2(5),
  Filmcim      VARCHAR2(80) NOT NULL,
  Gyartasi_ev  NUMBER(4) NOT NULL,
  Melyik_nap   VARCHAR2(20),
  Mikor        VARCHAR2(5),
  Belepo       NUMBER(4),
  CONSTRAINT musor_pk PRIMARY KEY (Mozi_kod, Melyik_nap, Mikor),
  CONSTRAINT musor_fk1 FOREIGN KEY (Mozi_kod)
        REFERENCES mozi(Mozi_kod),
  CONSTRAINT musor_fk2 FOREIGN KEY (Filmcim, Gyartasi_ev)
        REFERENCES film(Filmcim, Gyartasi_ev)
);
```

A feltöltésre az INSERT utasítás használható. Itt nem adnám meg az összes bevitt adatot, hanem mindegyik táblában csak párat:

Adatok felvitele a mozi táblába:

```
INSERT INTO mozi VALUES (
  'SZO01', 'Szolnok Plaza Cinema City', 'Szolnok', '5000 Szolnok, Ady Endre
u. 28.');

INSERT INTO mozi VALUES (
  'SZO02', 'Tallin Filmszínház', 'Szolnok', '5000 Szolnok, Kandó Kálmán tér
1.');

INSERT INTO mozi VALUES (
  'DEB01', 'Debrecen Plaza Cinema City', 'Debrecen', '4026 Debrecen,
Péterfia u. 18.');

INSERT INTO mozi VALUES (
  'MIS01', 'Miskolc Hollywood Multiplex', 'Miskolc', '3527 Miskolc,
Szinvapark, Bajcsy-Zsilinszky u. 2-4.');
```

Adatok felvitele a film táblába:

```
INSERT INTO film VALUES (
  '300','300', 2007, 'kaland', 117,'Gerard Butler, Lena Headey, David
Wenham, Dominic West');

INSERT INTO film VALUES (
  'T4xi','Taxi 4/ T4xi', 2007, 'akcióvígjáték', 91,'Samy Naceri, Frédéric
Diefenthal, Bernard Farcy, Emma Sjöberg');

INSERT INTO film VALUES (
  'A szellemlovas', 'Ghost Rider', 2007, 'thriller', 114,'Nicolas Cage, Eva
Mendes, Peter Fonda');
```

Adatok felvitele a musor táblába:

```
INSERT INTO musor VALUES (
  'SZO01','300', 2007, '2007. április 18.', '17:00', 990);

INSERT INTO musor VALUES (
  'SZO01','300', 2007, '2007. április 18.', '19:00', 990);

INSERT INTO musor VALUES (
  'SZO02','Hannibál ébredése', 2007, '2007. április 18.', '20:00', 890);

INSERT INTO musor VALUES (
  'SZO02','A szellemlovas', 2007, '2007. április 17.', '19:00', 990);

INSERT INTO musor VALUES (
  'DEB01','Mr. Bean nyaral', 2007, '2007. április 17.', '19:00', 890);

INSERT INTO musor VALUES (
  'DEB01','300', 2007, '2007. április 19.', '20:00', 990);

INSERT INTO musor VALUES (
  'MIS01','300', 2007, '2007. április 16.', '20:00', 990);
```

A `mozi` tábla induló tartalma:

MOZI_KOD	NEV	VAROS	CIM
SZO01	Szolnok Plaza Cinema City	Szolnok	5000 Szolnok, Ady Endre u. 28.
SZO02	Tallin Filmszínház	Szolnok	5000 Szolnok, Kandó Kálmán tér 1.
SZO03	Tisza Art mozi	Szolnok	5000 Szolnok, Templom u. 4.
DEB01	Debrecen Plaza Cinema City	Debrecen	4026 Debrecen, Péterfia u. 18.
DEB02	Apolló mozi	Debrecen	4025 Debrecen, Miklós u. 1.
PÉC01	Pécs Plaza Cinema City	Pécs	7631 Pécs, Megyeri u. 76.
MIS01	Miskolc Hollywood Multiplex	Miskolc	3527 Miskolc, Szinvapark, Bajcsy-Zsilinszky u. 2-4.

A `film` tábla induló tartalma:

FILMCIM	KULFOLDI_CIM	GYARTASI_EV	MUFAJ	HOSSZ	SZEREPLOK
300	300	2007	kaland	117	Gerard Butler, Lena Headey, David Wenham, Dominic West
T4xi	Taxi 4/ T4xi	2007	akcióvígjáték	91	Samy Naceri, Frédéric Diefenthal, Bernard Farcy, Emma Sjöberg
Hannibál ébredése	Hannibal Rising	2007	thriller	117	Gaspard Ulliel, Li Gong, Helena Lia Tachovska
A szellemlovas	Ghost Rider	2007	thriller	114	Nicolas Cage, Eva Mendes, Peter Fonda
Fekete Dália	The Black Dahlia	2006	krimi	121	Josh Hartnett, Scarlett Johansson, Aaron Eckhart, Hilary Swank
Mr. Bean nyaral	Mr. Beans Holiday	2007	vígjáték	80	Rowan Atkinson, Willem Dafoe, Karel Roden

Napfény	Sunshine	2007	thriller	108	Chris Evans, Cillian Murphy, Rose Byrne, Michelle Yeoh

A `musor` tábla induló tartalma:

MOZI_KOD	FILMCIM	GYARTASI_EV	MELYIK_NAP	MIKOR	BELEPO
SZO01	300	2007	2007. április 18.	17:00	990
SZO01	300	2007	2007. április 18.	19:00	990
SZO01	T4xi	2007	2007. április 16.	20:00	990
SZO01	T4xi	2007	2007. április 17.	19:00	990
SZO02	Hannibál ébredése	2007	2007. április 18.	20:00	890
SZO02	A szellemlovas	2007	2007. április 17.	19:00	990
SZO03	Mr. Bean nyaral	2007	2007. április 16.	19:00	890
DEB01	Mr. Bean nyaral	2007	2007. április 17.	19:00	890
DEB01	300	2007	2007. április 19.	20:00	990
DEB01	T4xi	2007	2007. április 18.	19:00	990
DEB02	Napfény	2007	2007. április 18.	20:00	990
PÉC01	300	2007	2007. április 16.	20:00	990
PÉC01	T4xi	2007	2007. április 17.	20:00	990
PÉC01	Napfény	2007	2007. április 18.	20:00	990
MIS01	300	2007	2007. április 16.	20:00	990
MIS01	Mr. Bean nyaral	2007	2007. április 18.	20:00	990

6.1. PL/SQL Web Toolkit eszközök használata

Itt egy egyszerű példát készítettem, amely kilistázza, hogy milyen műsorok vannak. Annyiban különbözik a `musor` tábla egyszerű kilistázásától, hogy ha csak azt listáznánk ki, akkor nem tudnánk, hogy melyik moziban megy az adott film, hacsak nem tudjuk az adott mozi kódját. Ugyanis a `musor` táblában csak a mozi kódja van, a neve nem található meg, ezt a `mozi` táblában találhatjuk meg. A feladatot úgy készítettem el, hogy a HTP csomag P eljárását használom, amely megegyezik a PRINT eljárással. A HTP csomagnak ez az eljárása a szövegeket és a tagokat együtt listázza ki.

`select_musor.sql` tartalma:

```
CREATE OR REPLACE PROCEDURE select_musor
AS
  PROCEDURE kiir
  AS
      rekord2 Mozi%ROWTYPE;
  BEGIN
      htp.p('<HTML> <HEAD> <TITLE>Mozimusorok listaja</TITLE> </HEAD> <BODY
      BGCOLOR=#000000>');
      htp.p('<P> <FONT COLOR=#ffffff SIZE=4> <B>Mozimusorok listaja:</B>
      </FONT> </P>');

      /*elkészítem a táblázat fejlécét*/
      htp.p('<TABLE BORDER="1" WIDTH="100%" CELLPADDING="1" CELLSPACING="2"
      ALIGN="center">');
      htp.p('<TR BGCOLOR="#aaaabf"> <TD ALIGN=center> </TD> <TD> <FONT
      FACE=arial>');
      htp.p('Mozi kod');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Varos');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Mozi neve');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Film');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Melyik nap?');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Mikor?');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      htp.p('Belepo');
      htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');

      /*kilistázom a musor tábla tartalmát, és közben folyamatosan épül fel
      a táblázat*/

      FOR rekord IN ( SELECT * FROM musor ) LOOP
           htp.p('<TR BGCOLOR="#bfaabf"> <TD ALIGN=center> </TD> <TD>
           <FONT FACE=arial>');
           htp.p( rekord.Mozi_kod );

           /*kikeresem a Mozi táblából a Mozi_kod-hoz tartozó város- és
           mozinevet*/

           SELECT * INTO rekord2 FROM Mozi WHERE Mozi_kod=rekord.Mozi_kod;
           IF rekord2.Mozi_kod=rekord.Mozi_kod THEN
            htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
            htp.p( rekord2.Varos);
            htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
            htp.p( rekord2.Nev);
           END IF;

           htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
           htp.p( rekord.Filmcim );
           htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
           htp.p( rekord.Melyik_nap );
           htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
           htp.p( rekord.Mikor );
           htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
```

```
            htp.p( rekord.belepo );
            htp.p('</FONT> </TD> <TD> <FONT FACE=arial>');
      END LOOP;
    htp.p('</TABLE> </BODY> </HTML>');
  END;

BEGIN
  kiir;
END;
/
SHOW ERRORS

GRANT EXECUTE ON select_musor TO hallgato;
```

Az eljárás először elkészíti a táblázat fejlécét, majd egy FOR ciklusban beolvastatom a `musor` tábla adatait. A FOR cikluson belül keresem ki a `mozi` táblából az adott mozi kódjához tartozó nevet, és hogy melyik városban található. A végén futtatási jogot kell adni a `hallgato`-nak.

Futtatás böngészőből:

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.select_musor

6.2. PSP technika

A leírásomban ez a rész jóval bővebb volt. Ezt a technikát akkor előnyösebb alkalmazni, ha sok HTML kódunk van, és viszonylag kevés a PL/SQL kód. A HTML szövegbe szúrunk be PSP direktívákat.

Itt két egyszerű példát készítettem. Az első ugyanolyan, mint az előző fejezetben ismertetett: a `musor` tábla tartalmát listázom ki, vagyis azt tudjuk meg, hogy melyik filmet mikor tudjuk megnézni.

`select_musor_osszes_psp.psp` tartalma:

```
<%@ page language="PL/SQL" %>
<HTML>
<HEAD><TITLE>Musorok listaja</TITLE>
</HEAD>
<BODY bgcolor=#000000>
<TABLE    BORDER="1"    WIDTH="100%"    CELLPADDING="1"    CELLSPACING="2"
ALIGN="center">
  <TR BGCOLOR="#77aabf"><TD> <FONT FACE=arial> Mozi kod </FONT> </TD>
  <TD> <FONT FACE=arial> Film </FONT> </TD>
  <TD> <FONT FACE=arial> Melyik nap? </FONT> </TD>
  <TD> <FONT FACE=arial> Mikor? </FONT> </TD>
  <TD> <FONT FACE=arial> Belepo </FONT> </TD>
  <% FOR rekord IN ( SELECT * FROM musor) LOOP %>
    <LI>
    <TR BGCOLOR="#bfaabf"> <TD> <FONT FACE=arial> <%= rekord.Mozi_kod %>
</FONT> </TD>
    <TD> <FONT FACE=arial> <%= rekord.Filmcim %> </FONT> </TD>
    <TD> <FONT FACE=arial> <%= rekord.Melyik_nap %> </FONT> </TD>
```

```
    <TD> <FONT FACE=arial> <%= rekord.Mikor %> </FONT> </TD>
    <TD> <FONT FACE=arial> <%= rekord.Belepo %> </FONT> </TD>
    </UL>
<% END LOOP; %>
</BODY>
</HTML>
```

Itt a HTML szövegbe beszúrtam egy FOR ciklust, amely kilistázza a `musor` táblát. Ezt el kell tárolni az adatbázisban. Erre szolgál a `loadpsp` parancs:

```
loadpsp –replace –user tothjani/”jelszó”@oracle.inf.unideb.hu
select_musor_osszes_psp.psp
```

A következő üzenetet kapjuk, ha nincs hiba:

Procedure "select_musor_osszes_psp" created.

Ezután a `hallgato`-nak futtatási jogot kell adni iSQL*Plus-ban, hogy futtatni lehessen ezt az eljárást.

```
GRANT EXECUTE ON select_musor_osszes_psp TO hallgato;
```

Futtatás böngészőből:

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.select_musor_osszes_psp

A második példa arról szól, hogy egy formba beírjuk, hogy mely városban lévő mozik műsorait akarjuk megnézni, és ezt egy táblázatban listázza ki az eljárás. Ezt 2 fájlban valósítottam meg. Az első fájl tartalmazza a formot, a második a listázó eljárást.

`select_musor_psp_form.psp` tartalma:

```
<!-- select_musor_psp eljárás meghívása -->
<%@ page language="PL/SQL" %>
<HTML>
<HEAD><TITLE>Select Musorhoz Form</TITLE>
</HEAD>
<BODY bgcolor=#000000>
<!-- mozilista kiíratása -->
<H1><FONT color=#ffffff>Mozik listaja:</FONT></H1>
<TABLE BORDER="1" WIDTH="100%" CELLPADDING="1" CELLSPACING="2"
ALIGN="center">
  <TR BGCOLOR="#aaaabf"><TD> <FONT FACE=arial> Varos </FONT> </TD>
  <TD> <FONT FACE=arial> Mozi neve </FONT> </TD>
  <% FOR rekord IN (SELECT * FROM mozi) LOOP %>
      <TR BGCOLOR="#bfaabf"> <TD> <FONT FACE=arial> <%=rekord.Varos %>
</FONT> </TD>
      <TD> <FONT FACE=arial> <%=rekord.Nev %> </TD>
  <% END LOOP; %>
</TABLE>
<FORM method="POST" action="tothjani.select_musor_psp">
<!-- POST: az elküldött paraméterek nem jelennek meg az URL-ben -->
<p><FONT color=#ffffff>Add meg  azt a varost, amelyben levo mozik musorait
akarod megnezni:</FONT>
<INPUT type="text" name="p_varos">
<INPUT type="submit" value="Elkuld">
```

```
</FORM>
</BODY>
</HTML>
```

Először kiíratok egy listát, hogy mely városban melyik mozi található, és ezután egy formban bekérem a város nevét. Ezt a városnevet pedig a `p_varos` paraméterben átadom a `select_musor_psp` eljárásnak.

`select_musor_psp.psp` tartalma:

```
<%@ page language="PL/SQL" %>
<%@ plsql parameter="p_varos" type="VARCHAR2" default="'Szolnok'" %>
<HTML>
<HEAD><TITLE>Musorok listaja</TITLE>
</HEAD>
<BODY bgcolor=#000000>
<TABLE BORDER="1" WIDTH="100%" CELLPADDING="1" CELLSPACING="2"
ALIGN="center">
  <TR BGCOLOR="#aaaabf"><TD> <FONT FACE=arial> Mozi kod </FONT> </TD>
  <TD> <FONT FACE=arial> Mozi neve </FONT> </TD>
  <TD> <FONT FACE=arial> Film </FONT> </TD>
  <TD> <FONT FACE=arial> Melyik nap? </FONT> </TD>
  <TD> <FONT FACE=arial> Mikor? </FONT> </TD>
  <TD> <FONT FACE=arial> Belepo </FONT> </TD>
  <% FOR rekord IN ( SELECT * FROM mozi WHERE Varos=p_varos) LOOP %>
  <LI>
    <% FOR rekord2 IN (SELECT * FROM musor WHERE Mozi_kod=rekord.Mozi_kod)
LOOP %>
      <TR BGCOLOR="#bfaabf"> <TD> <FONT FACE=arial> <%=rekord2.Mozi_kod %>
</FONT> </TD>
      <TD> <FONT FACE=arial> <%=rekord.Nev %> </FONT> </TD>
      <TD> <FONT FACE=arial> <%= rekord2.Filmcim %> </FONT> </TD>
      <TD> <FONT FACE=arial> <%=rekord2.Melyik_nap %> </FONT> </TD>
      <TD> <FONT FACE=arial> <%=rekord2.Mikor %> </FONT> </TD>
      <TD> <FONT FACE=arial> <%=rekord2.Belepo %> </FONT> </TD>
  <% END LOOP; %>
<% END LOOP; %>
</UL>
</TABLE>
</BODY>
</HTML>
```

Ez az eljárás a paraméterben megkapott városnévhez tartozó mozik műsorait listázza ki. Felépítem a táblázat fejlécét, majd egy FOR ciklusban kikeresem a `mozi` táblából azokat a mozikat, amelyek a paraméterül kapott városban vannak. Ezen a FOR cikluson belül pedig egy másik FOR ciklusban kikeresem a városban található mozikban futó műsorokat.

Ezután el lehet tárolni őket az adatbázisban, majd futtatási jogot adni. Futtatás böngészőből:

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.select_musor_psp_form

http://oracle.inf.unideb.hu/pls/hallgatodad/tothjani.select_musor_psp

Ha alapból a select_musor_psp-t hívom meg, és nem a formon keresztül, akkor alapértelmezetten a Szolnokon található mozik műsorait listázza ki.

Eredménye:

Mozi kod	Mozi neve	Film	Melyik nap?	Mikor?	Belepo
SZO01	Szolnok Plaza Cinema City	T4xi	2007. április 16.	20:00	990
SZO01	Szolnok Plaza Cinema City	T4xi	2007. április 17.	19:00	990
SZO01	Szolnok Plaza Cinema City	300	2007. április 18.	17:00	990
SZO01	Szolnok Plaza Cinema City	300	2007. április 18.	19:00	990
SZO02	Tallin Filmszínház	A szellemlovas	2007. április 17.	19:00	990
SZO02	Tallin Filmszínház	Hannibál ébredése	2007. április 18.	20:00	890
SZO03	Tisza Art mozi	Mr. Bean nyaral	2007. április 16.	19:00	890

7. Összefoglalás

A szakdolgozatom megírása közben sikerült bővebb ismereteket szereznem a dinamikus weblapokról, és ezen belül a PL/SQL nyelven való készítésről. Megismertem kétfajta készítési módot. Számomra a PL/SQL Web Toolkit eszközökkel való készítés kicsit egyszerűbb volt. Itt a PL/SQL nyelven és a HTML-en kívül szinte elég csak a HTP csomag P eljárását ismerni, míg a PSP technikánál tudni kell a PSP direktívákat. Habár az első technika számomra egyszerűbb volt, egyik hátránya szerintem az, hogy hosszabb kódot eredményez, mint a PSP technikával elkészített weboldal. Ezen kívül a PSP-nek még vannak további előnyei is, amelyek az 5-ös fejezet legelején le vannak írva. Éppen emiatt a PSP-vel bővebben foglalkoztam. A bemutatott mintafeladatok alapján könnyen el lehet sajátítani ezeket a technikákat. Bár mindkét esetben vannak olyan dolgok, amelyeket csak elméleti szinten említettem, és a mintafeladatomban nem található példa rá (gondolok itt például PSP esetén a változó deklarációra, vagy az include direktívára stb.), de a leírás és a meglévő példák alapján úgy gondolom, hogy a dolgozatom jó támpont azok számára, akik a dinamikus weblapok készítését szeretnék megtanulni PL/SQL nyelven.

Irodalomjegyzék

[1] http://www.oracle.com/pls/db102/homepage

Books:

HTTP Server mod_plsql User's Guide

HTTP Server Administrator's Guide

Application Developer's Guide - Fundamentals: 11. és 12. fejezet:

11 Developing Applications with the PL/SQL Web Toolkit

12 Developing PL/SQL Server Pages

[2] Juhász István, Gábor András – PL/SQL programozás, Alkalmazásfejlesztés Oracle 10g-ben, Panem Könyvkiadó, 2007

[3] Dr. L. Nagy Éva – Haladó Oracle I. gyakorlati jegyzet (Feladatok10_10g)

Internetes források:

HTML: http://www.lauder.hu/~attila/tan/internet/Html4/

XML: http://www.w3c.hu/forditasok/XML_10_pontban.html

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. L. Nagy Éva számítástechnikai munkatársnak a szakdolgozatom elékszítéséhez nyújtott tanácsaiért és minden segítségért, továbbá Kollár Lajos számítástechnikai munkatársnak a technikai segítségért.