Hallgatói Útmutató

Beadandó anyag követelményei

A labor során egy SQL szkriptet kell készíteni, amely a kiadott sablonnak felel meg. Ebben egyértelműen látszik, hogy melyik utasítás melyik feladat megoldása. **Ha a beadmány nem felel meg a sablonnak, a beadott megoldás értéktelen**. Külön jegyzőkönyvet nem kérünk. Ha a feladat szövege nem egyértelmű (minden feladatszöveg pontosan specifikált, de ha mégsem világos valami), **még beadás előtt**, e-mailben konzultáljunk a laborvezetőnkkel, aki a bizonytalanságot feloldja.

A táblákat létrehozó és próbaadatokkal feltöltő szkriptek az információs honlapról elérhetők, a feladatsor egy ennek letöltését kérő 0. feladattal indul. A letöltés menete: böngészőablakba az állomány betöltése, majd „copy-paste" művelettel hozzuk létre a megfelelő állományt. Az állományt mentsük el, majd futtassuk! Használjuk az F5 gombot SQL Developer-ben vagy a START operátort SQLcl-ben! Ha valaki a „Mentés másként…" menüponttal próbálkozik, ott helyenként előfordul, hogy a sorvége jelek nem jönnek át.

A beadandó anyag formátuma:

`<neptun>-2-<csoportkod>.zip` ezen belül:

`<neptun>-2-<csoportkod>` könyvtár, ezen belül:

`<neptun>-2-<csoportkod>.sql` a feladatokat megvalósító SQL szkript

`<neptun>-2-<csoportkod>.txt` a fenti szkript futtatásával nyert kimenet

Az SQL szkript formai és funkcionális megkötései:

- A szkript UTF-8 kódolású legyen!
- A szkriptnek a kiadott sablonnak kell megfelelnie, a feladatokat a feladatlapon szereplő sorrendben kell tartalmaznia.
- A szkript a futása elején NE hívja meg az inicializáló szkriptet.
- Ha az inicializáló és megoldásszkriptet többször lefuttatjuk ebben a sorendben egymás után, ugyanazt az eredményt kell kapnunk. Ennek érdekében a szkriptet úgy szervezzük, hogy az a saját objektumok törlésével kezdődjön (DROP TABLE..., stb.). Az inicializáló szkript törli a hivatalos objektumokat, a sajátokat viszont magunknak explicite kell törölni.
- A szkript nem használhat procedurális PL/SQL elemeket. Ami az Oracle 12c SQL referenciájában szerepel (http://docs.oracle.com/database/121/SQLRF/toc.htm, linkelve a tárgyhonlapról is),. Ez a referencia néhol átküld a PL/SQL referenciába, amit ott olvashattok, az már procedurális is lehet. (Tehát pl. a create procedure utasításról szól az SQL referencia is, de rögtön át is küld a PL/SQL referenciába, mivel eljárásokat PL/SQL-ben definiálunk)
- A szkript SQL utasításaiban TILOS a sémanevek használata az objektumnevek előtt, eredményoszlopokat átnevezni pedig csak akkor szabad, ha a feladat szövegéből ez

egyértelműen következik (pl. meg van adva az eredményoszlop neve, és az alapértelmezetten – átnevezés nélkül – más néven jelenne meg).

- A szkriptnek végeredményben egy XML formátumú megoldást kell kiírnia a kimenetre, ehhez a kapott sablonhoz a következő pontokon szükséges hozzányúlni:
  - A `prompt <tasks>` sor előtt kerüljön sor a saját objektumok eldobására.
  - Minden feladat megoldásának a `prompt <tasks>` és a `prompt </tasks>` sorok közé kell kerülnie a feladatlap szerinti sorrendben.
  - Minden feladat megoldását a lenti két sornak kell bevezetnie. Az n attribútum a feladat feladatlapon szereplő sorszáma, jelen esetben 1.1, ez feladatonként értelemszerűen változtatandó; minden feladatnál a **feladatlapon adott** sorszámmal kell egyeznie.

    ```
    prompt <task n="1.1">
    prompt <![CDATA[
    ```
  - Ezután következhet maga az SQL utasítás.
  - Az SQL utasítás után jön a következő két sor (ez tehát változatlan):

    ```
    prompt ]]>
    prompt </task>
    ```
  - Meg nem oldott feladatok bevezető és lezáró `prompt` üzenetei ne maradjanak üresen a szkriptben, azokat inkább távolítsuk el!
  - Az adatmanipulációs feladatok előtt a szkriptnek ki kell adnia a `set feedback on`, utána pedig a `set feedback off` utasítást.
- A végleges, összeállt szkriptnek SQLcl-ben futnia kell megfelelően, tehát ezt is ellenőrizzük a saját szkript előtt **kézzel** futtatva a környezetbeállító szkriptet. Az SQLcl letölthető a tárgy és az Oracle honlapjáról. Az SQLcl indítása a következőképpen történik:

  `sql /nolog` (a `/nolog` kapcsoló miatt itt nem kér connection string-et)

  Ezután a következőképpen kapcsolódhatunk az adatbázisszerverhez:

  `CONNECT felhasznalonev@//rapid.eik.bme.hu:1521/szglab`

  Hasznos tudni, hogy a lekérdezéseredmények szöveges fájlba való mentése a `SPOOL <fájlnév>` paranccsal kapcsolható be SQLcl-ben.
- Ellenőrizzük SQLcl-ben a fenti módon, hogy a beadandó szkriptünk valóban jólformált XML formátumú kimenetet produkál-e `<tasks>` gyökérelemmel és ezen belüli `<task>` elemekkel, amelyeken belül a szkript adott feladathoz tartozó kimenete `<![CDATA[kimenet]]>` módon jelenik meg!
- Magának az SQL szkriptnek is jólformált XML-nek kell lennie a `<tasks>` és `prompt </tasks>` elemei között.
- A **jólformáltság ellenőrzése** a következő módon történik az XML dokumentumok esetében:

- Vágjuk ki az SQL szkriptünk a fent említett, `<tasks>` és `prompt </tasks>` közötti szakaszát, helyezzük egy xml kiterjesztésű állományba, amit nyissunk meg egy böngészőben (Firefox, Chrome vagy IE)! Ha szépen megjelenik hierarchikusan **fa formátumban** az eredmény, akkor jólformált. Ha hibát kapunk és/vagy nem fa formátumban jelenik meg a fájl tartalma, akkor nem jólformált a szkriptünk.
- A szkriptünk lefuttatásával nyert kimenetet hasonlóképpen nyissuk meg egy böngészőben, és ellenőrizzük a jólformáltságát!

- **Ha a szkript nem felel meg a fenti követelményeknek, a beadott feladat értékelhetetlen.**

Ügyeljünk továbbá arra, hogy a megoldásunk elkerülje a hatékonyságot befolyásoló illetve koncepcionális hibákat. Ilyenek például:

- beágyazott SELECT használata HAVING feltétel helyett
- beágyazott SELECT használata illesztés helyett
- halmazműveletek indokolatlan használata
- külső illesztés mellőzése
- NULL érték vizsgálata nem IS NULL-lal

Végül egy megjegyzés: Több helyen kérjük, hogy az oszlopnevek felsorolása nélkül oldódjék meg az adott feladat. Ez nem azt jelenti, hogy a kimeneten ne jelenjenek meg oszlopnevek, csupán azt, hogy ezek a lekérdezésben magában ne legyenek felsorolva.

Néhány megjegyzés az SQL-lel kapcsolatban:

- Minden lekérdezés SELECT – FROM – WHERE – GROUP BY – HAVING – ORDER BY sorrendben íródik, már ahol vannak ezek az elemek, kivéve(!) a hierarchikus lekérdezéseket.
- Relációs algebrát illetően a (SELECT – FROM – WHERE) kis különbséggel a projekció – Descartes-szorzat – szelekció sorozatnak felel meg, ahol is a kulcsszók után álló nevek a műveletek operandusai. A különbség annyi, hogy míg a relációalgebra műveletei halmazműveletek (azaz nincsenek ismétlődések az eredményhalmazban), addig SQL esetén SELECT helyett SELECT DISTINCT utasítást kell alkalmaznunk az ismétlődések szűréséhez.
- HAVING és WHERE között ég és föld a különbség: HAVING mindig(!) aggregátumra vonatkozik, azaz SUM, AVG, COUNT stb. elemre vonatkozó kényszert határoz meg.
- PRIMARY KEY = UNIQUE + NOT NULL, de míg a baloldaliból csak 1 lehet minden sémában, addig a jobboldaliból „tetszőlegesen” sok.
- A UNIQUE paraméter megadása nem zárja ki a NULL értéket.

- NULL = NULL, NULL <> NULL egyaránt hamis (pontosabban UNKNOWN értékű), ennek megfelelően – mivel a lekérdezés csak azokat a rekordokat adja vissza, amelyekre a lekérdezés igaz – az UNKNOWN elemek nem kerülnek az eredményhalmazba.
- A UNION alapesetben DISTINCT elemek halmazát adja, és ez az egyetlen SQL utasítás, amelynek ez az alapértelmezése. Ha ismétlést is szeretnénk látni, akkor UNION ALL direktívát kell alkalmazni.
- A SELECT után álló valamennyi nem aggregátumot a GROUP BY paraméterében meg kell adni – egyébként a lekérdezés nem fordul le.
- A CHAR és VARCHAR2 között nagy különbség van: az előző fix méretű, az utóbbi változó méretű sztring. Jelentősége a mintaillesztésnél szembeötlő, hiszen pl. CHAR(5) esetében az L betűre végződő hárombetűs szavak valójában nem L betűre, hanem két SPACE karakterre végződnek.
- TO_DATE és TO_CHAR függvények részletes leírása az Oracle SQL kézikönyvben olvasható: http://docs.oracle.com/database/121/SQLRF/functions.htm#SQLRF006.
- Az AS kulcsszó kiírása nem kötelező a SELECT listában. A FROM listában a táblák átnevezésekor pedig nem szabad használni.
- FROM után is állhat beágyazott lekérdezés (a WHERE előtt, zárójelben).
- Idézőjelek használata esetén az adatbázisban tárolt táblák és attribútumnevek jellemzően (de nem mindig) case-sensitive módon kerülnek értékelésre. Tábla létrehozásánál, beszúrásnál kerüljük a használatát, de egyébként is elhagyhatóak.
- Nem kell a táblákhoz megadni a sémaneveket (pl. a felhasználó nevét), ahogy egyértelmű esetekben a táblanevet sem kell kiírni az attribútumértékek mellé.
- A külső illesztés (outer join) „(+)” jelét mindig oda kell írni, ahol NULL érték állhat.
- Külső illesztésnél a „(+)” jelet ki kell tenni a lekérdezésben szereplő valamennyi olyan attribútum után, amely a NULL értékekkel kiegészített táblához tartozik. Kivételt képez ez alól a többszörös külső illesztés. Magyarázatként álljon itt egy példa:

```
select a.id, b.id, c.id
  from a, b, c
 where
    -- join
    a.id = b.a_id (+)
    and b.c_id = c.id (+)
    -- szűrések, itt már kell a (+) b és c összes
    -- előfordulásához
    and b.szin (+) in ('piros', 'zöld')
;
```

- A hierarchikus lekérdezésben a PRIOR kulcsszó mindig azon attribútum előtt áll, amelyhez tartozó egyedre az attribútum értékével másik egyed hivatkozik (akit a következő iterációban írunk ki, PRIOR tehát az elsőként kiírt egyedre mutat vissza). A PRIOR attribútumhoz köt,

azaz nem állhat operátor vagy függvény a hatáskörében, csak és kizárólag attribútumnév követheti.

- Hasznos lehet az NVL2 függvény, amely paraméterezése a következő: NVL2(attribútumnév, output_nem_null_érték_esetén, output_null_érték_esetén)
- Szinte biztosan használandó valamely feladatnál az NVL és DECODE függvény is, ezeknek is érdemes utánanézni.