Vypnout / zapnout zobrazení komentaru

X36DBS - Databázové systémy, Zima 2006, lichý týden

Zbyněk Lstibůrek (lstiburz1) Čtvrtek 7.30

lstibuz1@fel.cvut.cz

Prohlašuji, že jsem svou semestrální práci vypracoval samostatně. Všechny zdroje, ze kterých jsem čerpal, jsou uvedeny v části Odkazy.

Pěvecký sbor (ukázková semestrálka a komentáře)

Popis

Pěvecký sbor je skupina lidí, která se víceméně pravidelně schází na zkouškách, kde nacvičuje hudební skladby, jež pak provádí na koncertech v různých místech. Databázový systém slouží k evidenci členské základny, zapůjčených not a koncertních šatů, informací o konání a programu zkoušek a koncertů a také podrobných údajů o skladbách v repertoáru – pro zjednodušení práce při tisku koncertních programů.

K O M E N T Á Ř E

Tohle je komentář k 1. odstavci popisu semestrální práce.
Zachovává formátování.

Zobrazení komentářů si můžete VYPNOUT
- v menu XML Mind Editoru View -> Hide comments
- při html zobrazení tlačítkem "Vypnout/zaptnout zobrazeni komentaru" přímo na html stránce.

Komentář můžete používat k vlastním poznámkám k částem práce,
dále jej může využít váš cvičící k tomu, aby připojil svoje hodnocení
k jednotlivým částem práce.

Komentovat lze samostatně:
jednotlivé odstavce textů v částech Popis, Datové schéma, Skripty, Závěr, Odkazy
obrázek datového schématu
popis SQL příkazu
formulaci v relační algebře
jednotlivé SQL příkazy
tabulku pokrytí kategorií SQL příkazů.

Další informace o semestrální práci, struktuře, požadavcích a termínech jsou uvedeny
formou komentářů přímo v tomto dokumentu.

Zpěváci se dělí podle své zpěvní polohy (typicky soprán, alt, tenor, bas) do hlasových skupin, z nichž každá má určeného svého vedoucího. Skupiny mohou mít např. samostatné (dělené) zkoušky vedené právě vedoucím hlasu.

S T R U K T U R A S E M E S T R Á L N Í P R Á C E

Semestrálka má tyto části:
- hlavička (údaje o vás a o čase cvičení, prohlášení o autorství, název práce)
- popis - stručná charakteristika vašeho projektu (několik odstavců, kde popíšete "realitu", kterou budete ve vaší semestrálce databázově modelovat
- datové schéma (skládá se z obrázku (nejlépe formát png) a diskuse
obrázek je vytvořen nějakým nástrojem pro konceptuální datové modelování (doporučený je ER Modelář, ale po dohodě se cvičícím můžete použít jiný).
Diskuse k datovému schématu zahrnuje
výčet integritních omezení, které nebylo možné zahrnout do obrázku (jsou-li)
diskuse ke každé smyčce, která se vyskytne ve vašem schématu

- SQL příkazy a příkazy v relační algebře - vypracujete:
alespoň 25 SQL příkazů nad vaší databází (tak aby pokryly kategorie v níže uvedené tabulce)
alespoň 10 z nich (jen dotazy) formulujete také v relační algebře
- tabulka pokrytí požadovaných kategorií vašimi SQL příkazy
- skripty se zdrojjovými texty vašeho řešení:
- zdrojový soubor pro datové schéma
- SQL skripty pro vytvoření vaší databáze (ER modelář ho pro vás vygeneruje)
- SQL skript pro vložení testovacích dat do vaší databáze
- SQL skript s odladěnými příkazy
- log soubor (nasnímaný výstup), kterým doložíte, že vaše SQL příkazy jsou funkční a vrací správná data
- závěrečnou diskusi
- odkazy na použité zdroje

Všechny části semestrální práce jsou povinné.

Protože noty jsou velmi drahé (předpokládá se neporušování autorského práva kopírováním:-), vede sbor přesnou evidenci o tom, kdo má u sebe který výtisk. Noty mohou mít buď podobu partů (vždy jen s notami určitého hlasu), nebo partitury (ta obsahuje všechny hlasy). V případě partů to znamená, že různí zpěváci mohou zpívat stejnou skladbu z jiných not. Dále existují zpěvníky nebo sborníky, které obsahují větší počet skladeb, někdy od různých autorů.

TECHNOLOGIE A NÁSTROJE

Semestrální práci odevzdáte ve formě XML dokumentu, který nahrajete do vašeho projektového adresáře na serveru service.felk.cvut.cz. Do tohoto adresáře umístíte také všechny zdrojové souboru a obrázek datového schématu

UMÍSTĚNÍ PRÁCE:
http://service.felk.cvut.cz/courses/X36DBS/prj/<vase_uzivatelske_jmeno> nebo
http://service.felk.cvut.cz/courses/Y36DBS/prj/<vase_uzivatelske_jmeno>
technologie přístupu do této lokace je WebDAV, návody a nástroje naleznete na:
http://service.felk.cvut.cz/pubguide/

STRUKTURA DOKUMENTU:
Odevzdaná práce (XML dokument) nechť je validní dle specifikace sprojekt. Na adrese:
http://service.felk.cvut.cz/courses/X36DBS/xml/sprojekt/
najdete specifikace ve formátu DTD, XML Schema, Relax NG (sprojekt.dtd, sprojekt.xsd, sprojekt.png).
Pro editaci práce tedy lze použít libovolný XML editor.
Kompletní podpora pro formátování semestrálky (šablony a ukázkové semestrálky)
pro kompletní offline práci je:
http://service.felk.cvut.cz/courses/X36DBS/xml/sprojekt.zip

TVORBA DOKUMENTU (semestrálky)
Pokud použijete jiný editor než XML Mind (viz níže), doporučuji stáhnout si šablonu pro semestrálku z DBS, tu otevřít a vyplnit vlastním textem vaší semestrální práce.
http://service.felk.cvut.cz/courses/X36DBS/xml/sprojekt/templates/dbs_semestral_work_template.xml
Pokud toto neuděláte, pak je třeba, aby váš nově vytvořený dokument odkazoval na formátovací šablonu pro zobrazení v html podobě. Za specifikaci kódování v XML souboru:
<?xml version="1.0" encoding="UTF-8"?>
je třeba umístit tuto instrukci pro provedení:
<?xml-stylesheet type="text/xsl" href="http://service.felk.cvut.cz/courses/Y36DBS/xml/sprojekt_html.xsl"?>

EDITOR XML MIND - DOPORUČUJI PRO TVORBU SEMESTRÁLKY
Home Edition tohoto nástroje je volně ke stažení pro platformy linux i windows:
http://www.xmlmind.com/xmleditor/download.shtml
Až si ho stáhnete a nainstalujete a spustíte, doinstalujte si podporu dokumentu sprojekt (tvorba semestrálky):

Instalace balíku "sprojekt" (šablony pro semestrálku z DBS) v XML Mind
Options -> Preferences... -> Install add-ons
do okna "Download add-ons from this servers"

přidat adresu:
http://service.felk.cvut.cz/courses/X36DBS/xml/sprojekt.xxe_addon
Options -> Install Add-ons...
Načtou se informace o balicích, tam najdete balik "DBS Semestralni prace"

Po dokončení instalace je třeba XML Mind restartovat.
Po dalším spuštění se vám přes volbu File -> New... v nabídce existujících formátů objeví nabídka:
DBS Semestral Work Documentation
a vy si můžete vybrat, zda si otevřete šablonu kam doplníte vlastní text semestrálky
(A Template For Semestral Work With Comments) nebo vzorovou semestrálku s komentáři
(Sample of Semestral Work), což přesně tento dokument.

Každá skladba může mít jednoho skladatele (nemusí, může jít třeba o lidovou píseň, ani anonymních skladeb není málo) a – zejména v případě lidových písní – i autora úpravy. Rozumí se, že úpravu skladeb provádějí hudební skladatelé, takže každý skladatel v databázi nemusí být nutně autorem nějaké skladby. Větší skladby se mohou dělit na několik částí, jejich názvy bývá zvykem uvádět v koncertním programu.

TERMÍNY ODEVZDÁNÍ - KONTROLNÍ BODY

Váš cvičící s vámi na začátku semestru domluví přesné termíny 2 kontrolních bodů (a písemky.)
To znamená deadline pro odevzdání části vaší semestrálky.
Nedodrženi deadline bude potrestáno strženim 5 bodů z počtu, který můžete za semestrálku získat (20).

1. Kontrolní bod
- *doporučovaný termín (garantem předmětu) 3. - 4. cvičení u počítačů*
- *kontroluje se:*
  - *hlavička semestrálky*
  - *popis*
  - *datové schéma (včetně diskuse smyček)*

2. kontrolní bod
- *doporučovaný termín (garantem předmětu) zápočtový týden*
- *kontrolují se zbylé části dokumentace, tedy:*
  - *SQL příkazy*
  - *tabulka pokrytí kategorií SQL příkazů*
  - *skripty se zdrojovými texty (včetně LOG souboru s naskenovanými výsledky SQL příkazů)*

Pokud jde o sborové zkoušky, bylo uvedeno, že některé mohou být dělené, tj. některých zkoušek se nemusí účastnit všichni. U každé zkoušky je ale dán seznam těch, kteří na ní mají být, a na základě toho se eviduje docházka. Pro účast na koncertech platí stejný princip. U koncertů se sleduje také (hrubá) divácká účast a výše vybraného vstupného – kvůli různým statistikám.

ZOBRAZENÍ SEMESTRÁLKY NA WWW

Pokud dodržíte výše uvedený postup, pak vám stačí do vašeho adresáře
na webis umístit (v závorce jsou uvedené soubory z tohoto příkladu):
xml soubor se semestrálkou (test.xml)
obrázek datového schématu (pevecky_sbor.png)
SQL skripty (create.sql, data.sql, query.sql, output.log)

Jak to bude vypadat ve výsledku se můžete podívat zde (cože je :
http://service.felk.cvut.cz/courses/X36DBS/xml/test/test.xml

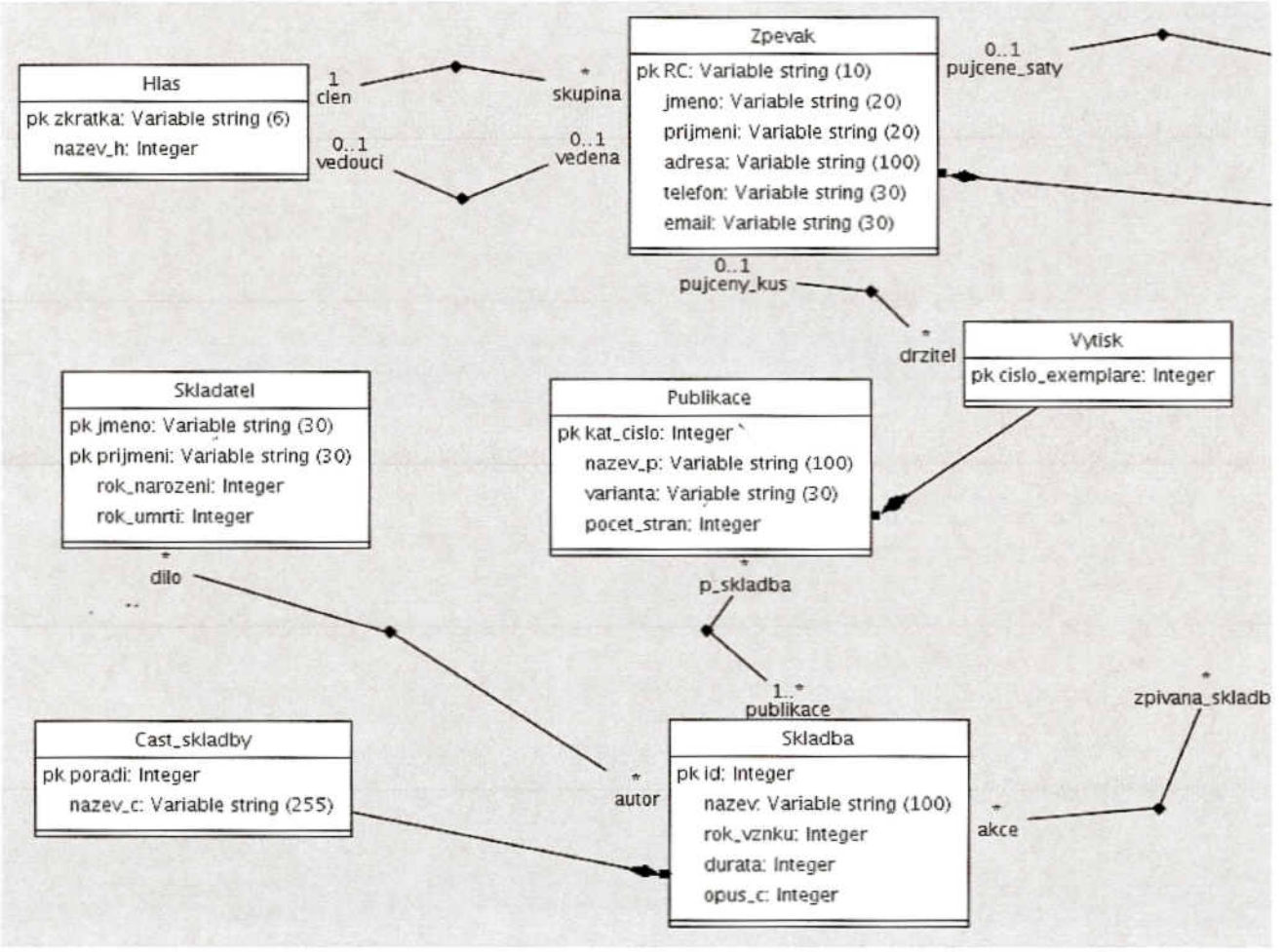
Motivace pro použití této technologie:
1. *zamezení různým nedorozuměním v tom, co má obsahovat vaše semestrálka*
2. *věřím tomu, že až zvládnete práci s nějakým XML editorem, bude to pro vás výrazné zjednodušení (nemusíte se vůbec zabývat grafickou stránkou semestrálky) ,*
3. *vyrobením semestrálky v XML nám poskytnete vhodná data pro experimenty, které provádíme v rámci projektu CellStore - implemetnace XML-nativního databázového stroje*
   http://cellstore.felk.cvut.cz

Na tomto projektu se podílí rovněž studenti (mimo jiné byl zdrojem několika velmi pěkných a již úspěšně obhájených bakalářských a diplomových prací).

Zobrazení semestrálky v XML na WWW bylo úspěšně otestováno na prohlížečích FireFox 2 a Internet Explorer 7. Nefunguje například v prohlížeči Konquerror.

Máte-li nějaké (pádné ;-)) důvody k tomu, že nemůžete semestrálku odevzdat formou XML, řešte to, prosím, individuálně se svými cvičícími.

datové schéma



Datové schéma vaší semestrálky by mělo být zhruba takto rozsáhlé. Je žádoucí, abyste si při návrhu vyzkoušeli pokud možno všechny konstrukty ER modelu:

silná entita
slabá entita (potažmo vztahová entita)
vztahy s různými kardinalitami (1-1, 1-N, M-N) a parcialitami (povinná, nepovinná účast entity ve vztahu),
ISA hierarchie (pozor, tady je to výrazně jiné, než jste zvyklí z OOP)

Element imagedata, který reprezentuje obrázek, má atribut fileref, kam musíte napsat jméno obrázku.

Někteří z vás časem postoupí do magisterského studijního programu a zapíší si předmět "Jazyk SQL".
Tam budete mít také vypracovat semestrální práci. V zadání vám bude uloženo, abyste navázali na výsledek této sem
Nezapomeňte si její dokumentaci pečlivě uložit. Ušetříte tím potom hodně práce.
I tento argument mluví pro to, abyste své schéma navrhli dostatečně rozsáhlé. I výše zmíněné smyčky se vám budou i

Moje schéma obsahuje smyčku Zpěvák – Příslušnost – Hlas – Vedení – Zpěvák. Toto je ovšem v pořádku, protože mezi entitami HLAS a ZPEVAK existují dva nezávislé vztahy. Vztahy "je členem" a "je vedoucím" nejsou vzájemně nijak ovlivňovány. Vedoucím hlasové skupiny bude v drtivé většině případů některý z jeho členů, ale může jím být i někdo "zvenku", kdo je ale také registrován jako zpěvák. Jiné řešení kdy, bych např. zavedl atribut "je_vedouci" u vztahu Příslušnost, by pokrývalo požadavek, že vedoucím skupiny typu HLAS může být pouze příslušník skupiny, ale nebylo by možné datově zajistit pravidlo, že skupina může mít jednoho vedoucího. Možným řešením této problematiky by bylo i přidání atributu "vedoucí" k typu Hlas, který by mohl nabývat jakékoliv (i prázdné) hodnoty nepodléhající žádnému omezení (žádné kontrole).

Tohle je komentář k 1. odstavci diskuse datového schématu.
Komentář může mít více řádků.
Zachovává formátování.

ZPEVAK-UCAST-UDALOST-CAST_PROGRAMU_SKADBA-ZARAZENI-PUBLIKACE-VYTISK-ZAPUJCKA_NOT. Interpretace této smyčky z hlediska entity zpěvák je taková, že zpěvák může mít půjčeny noty na skladbu, která se nehraje na žádné zkoušce nebo koncertě, kterých by se zúčastnil. V "praktickém" životě je taková situace běžná (řekněme, že si noty oné skladby vypůjčil z jiného důvodu).

Dotazy

1. Seznam koncertů, na nichž bylo více než 100 diváků

Komentář k popisu dotazu.

```
Koncert(divaku > 100)
```

Komentář k relační algebře.

```
SELECT *
FROM Koncert
WHERE divaku > 100;
```

Komentář k SQL dotazu.

2. Seznam zpěváků, jejichž vedoucím je Alena Havlíková.

```
A := {Hlas * Vedeni * Zpevak(jmeno='Alena' and prijmeni='Havlíková')}[zkratka]
Vysledek := Zpevak <hlas = zkratka] A
```

```
SELECT Zpevak.*
FROM Zpevak JOIN (
        SELECT zkratka FROM Vedeni NATURAL JOIN Zpevak
        WHERE jmeno='Alena' AND prijmeni='Havlíková'
        ) ON hlas=zkratka;
```

3. Stejné zadání jako předchozí, nyní s poddotazem v klauzuli WHERE.

```
SELECT *
FROM Zpevak
WHERE hlas = (
    SELECT zkratka FROM Vedeni NATURAL JOIN Zpevak
    WHERE jmeno='Alena' AND prijmeni='Havlíková'
    );
```

4. Názvy všech skladeb A. Dvořáka v repertoáru.

```
{Skladatel(jmeno='Antonín' and prijmeni='Dvořák') *
   Autorstvi * Skladba}[nazev]
```

```
SELECT nazev
```

```
FROM Skladba NATURAL JOIN Autorstvi NATURAL JOIN Skladatel
WHERE jmeno='Antonín' AND prijmeni='Dvořák';
```

5. Seznam zpěváků, kteří nemají zapůjčeny šaty.

```
Zpevak \ {Zpevak <* Zapujcka_satu}
```

```
SELECT * FROM Zpevak
MINUS
SELECT Zpevak.*
    FROM Zpevak JOIN Zapujcka_satu
         ON Zpevak.RC = Zapujcka_satu.RC;
```

6. Stejné zadání jako předchozí, řešení pomocí poddotazu.

```
SELECT *
FROM Zpevak
WHERE RC NOT IN (SELECT RC FROM Zapujcka_satu);
```

7. Seznam dosud neprovedených skladeb (nebyly na programu žádného koncertu), včetně autorů.

```
A := Skladba[id, nazev] \ {Skladba <* {Cast_programu * Koncert}}[id, nazev]
Vysledek := {A *_L Autorstvi}[jmeno, prijmeni, nazev]
```

```
SELECT jmeno, prijmeni, nazev
FROM (
        SELECT id, nazev FROM Skladba
        MINUS
        SELECT id, nazev FROM Skladba
        NATURAL JOIN Cast_programu NATURAL JOIN Koncert
       ) LEFT JOIN Autorstvi USING (id)
         LEFT JOIN Skladatel USING(jmeno, prijmeni);
```

8. Seznam vícedílných skladeb.

```
Skladba <* Cast_skladby
```

```
SELECT jmeno, prijmeni, nazev
FROM (SELECT * FROM Skladba WHERE id IN (SELECT id FROM Cast_skladby)
      )LEFT JOIN Autorstvi USING (id)
       LEFT JOIN Skladatel USING(jmeno, prijmeni);
```

9. Program koncertu 18. 1. 2005.

```
{Skladba * Cast_programu * Koncert(cas='18.01.2005')}[id, nazev]
```

```
ALTER Session SET NLS_DATE_FORMAT='DD.MM.YYYY';
SELECT jmeno, prijmeni, nazev
FROM Skladatel
     JOIN Autorstvi using (jmeno, prijmeni)
     JOIN Skladba using (id)
     JOIN Cast_programu using (id)
     JOIN Koncert using (misto, cas)
WHERE TRUNC(cas)=TRUNC(TO_DATE('18.01.2005'));
```

Schválně si můžete vyzkoušet, že, když dotaz změníte takto:

```
SELECT jmeno, prijmeni, nazev
FROM Skladatel
   NATURAL JOIN Autorstvi
   NATURAL JOIN Skladba
   NATURAL JOIN Cast_programu
   NATURAL JOIN Koncert
WHERE TRUNC(cas)=TRUNC(TO_DATE('18.01.2005'));
```

Dostanete ve výsledku místo 7 řádků (správná odpověď) řádků 3528.

Proč je tomu tak?

*Celá záležitost se pokazí projekcí, pokud v klauzuli SELECT necháte *, je výsledek stále ještě správný.*

Pokud bychom tedy trvali na použití NATURAL JOIN, lze dotaz formulovat takto:

```
Select jmeno, prijmeni, nazev
from
 (select *
 from  skladatel
    natural join autorstvi
    natural join skladba
    natural join cast_programu
    natural join koncert
 WHERE TRUNC(cas)=TRUNC(TO_DATE('18.01.2005'))
 );
```

Oracle má zjevně v tomto kontextu problém, ale pouze tehdy, když pomocí NATURAL JOIN spojujeme více než 3 tabulky. Například dotaz 7 funguje správně.
Ať to budeme nazývat "bug" nebo "feature", je třeba s tímto fenoménem počítat!!!

10. Seznam zpěváků se 100% docházkou.

```
Zpevak <* {Ucast[RC] \ Ucast(pritomen=0)[RC]}
```

```
SELECT Zpevak.*
FROM Zpevak JOIN (
         SELECT DISTINCT RC FROM Ucast
         MINUS
         SELECT DISTINCT RC FROM Ucast WHERE pritomen=0
          ) Vzorni ON Zpevak.RC=Vzorni.RC;
```

11. Stejný dotaz jako předchozí, tentokrát řešený poddotazem.

```
SELECT *
FROM Zpevak
WHERE RC NOT IN (
     SELECT RC FROM Ucast WHERE pritomen=0
      );
```

12. Opět stejný dotaz řešený vztaženým poddotazem.

```
SELECT *
FROM Zpevak
WHERE NOT EXISTS (
        SELECT * FROM Ucast WHERE pritomen=0 AND Ucast.RC=Zpevak.RC
         );
```

13. Seznam zpěváků, kteří nemají noty na Čtyři písně o Marii.

```
Zpevak \ {Zpevak <* {Zapujcka_not * Publikace * Zarazeni
  * Skladba(nazev='Čtyři písně o Marii')}}
```

```
SELECT jmeno, prijmeni FROM Zpevak
 MINUS
SELECT jmeno, prijmeni FROM Zpevak
     JOIN Zapujcka_not using (RC)
     JOIN Publikace using (kat_cislo)
     JOIN Zarazeni using (kat_cislo)
     JOIN Skladba using (id)
WHERE nazev='Čtyři písně o Marii';
```

Proč ne NATURAL JOIN? Viz diskuse u dotazu 9.

14. Skladatelé, kteří jsou autorem nějaké skladby a zároveň upravovatelem nějaké skladby.

```
{Skladatel <* Autorstvi(je_uprava=0)} ∩ {Skladatel <* Autorstvi(je_uprava<>0)}
```

```
SELECT jmeno, prijmeni FROM Skladatel JOIN Autorstvi using (jmeno, prijmeni) WHERE je
 INTERSECT
```

```
SELECT jmeno, prijmeni FROM Skladatel JOIN Autorstvi using (jmeno, prijmeni) WHERE je
```

15. Seznam publikací, od kterých je v inventáři méně než pět exemplářů. První realizaci můžeme zjednodušit tak, vybereme ty publikace, kde existují jen exempláře s pořadovými čísly < 5.

```
Publikace \ {Publikace * Vytisk}(cislo_exemplare >=5)[Publikace.*]
```

```
SELECT *
FROM Publikace A
WHERE NOT EXISTS (
      SELECT * FROM Vytisk
      WHERE A.kat_cislo=Vytisk.kat_cislo AND cislo_exemplare >= 5);
```

16. Stejný dotaz s použitím agregace, navíc bez výše použitého zjednodušení.

```
SELECT *
FROM Publikace
WHERE 5 > (SELECT COUNT(*) FROM Vytisk
            WHERE Publikace.kat_cislo=Vytisk.kat_cislo);
```

17. Seznam publikací, které obsahují více než jednu skladbu.

```
Publikace <*
  {Zarazeni[kat_cislo=k_c AND id < id_d]Zarazeni<kat_cislo -> k_c, id -> id_d>}
```

```
SELECT *
FROM Publikace NATURAL JOIN (
      SELECT DISTINCT A.kat_cislo
      FROM Zarazeni A JOIN Zarazeni B
           ON A.kat_cislo=B.kat_cislo AND A.id < B.id
);
```

18. Stejný dotaz přehledněji s agregací.

```
SELECT *
FROM Publikace
WHERE 1 < (
            SELECT COUNT(*) FROM Zarazeni
            WHERE Zarazeni.kat_cislo=Publikace.kat_cislo
      );
```

19. Seznam zpěváků, kteří nikdy nezpívali žádnou skladbu Jana Hanuše.

```
  -- Všechny události, kde se zpívala nějaká skladba Jana Hanuše:
  A := {Skladatel(jmeno='Jan' and prijmeni='Hanuš')
      * Autorstvi * Skladba * Cast_programu * Udalost}[misto, cas]
  -- Seznam zpěváků, kteří byli na některé z těchto událostí:
  B := {A * Ucast(pritomen<>0)}[RC]
Vysl := Zpevak \ {Zpevak * B}
```

```
SELECT jmeno, prijmeni
FROM Zpevak
WHERE RC NOT IN (
      SELECT RC FROM Ucast
      WHERE (misto, cas) IN (
         SELECT misto, cas
         FROM Skladatel
              JOIN Autorstvi using (jmeno, prijmeni)
              JOIN Skladba using (id)
              JOIN Cast_programu using (id)
              JOIN Udalost using (misto, cas)
         WHERE jmeno='Jan' AND prijmeni LIKE 'Hanuš'
         )
      AND pritomen <> 0
      );
```

Proč ne NATURAL JOIN? Viz diskuse u dotazu 9.

20. Seznam všech koncertů a generálek.

```
{Koncert[cas, misto] x {('Koncert')}}
 U {Zkouska(je_generalka)[cas, misto] x {('Generálka')}}
```

```
SELECT cas, misto, 'Koncert' AS druh FROM Koncert
 UNION
SELECT cas, misto, 'Generálka' AS druh FROM Zkouska WHERE je_generalka<>0
ORDER BY cas;
```

21. Seznam koncertů, na nichž byly provedeny všechny skladby B. Martinů na repertoáru.

```
  -- ID všech skladeb B. Martinů:
 SkladbyBM := {Skladatel(jmeno='Bohuslav' and prijmeni='Martinů')
               * Autorstvi * Skladba}[id]
  -- Všechny kombinace koncert-skladba B. Martinů:
  VseKomb := SkladbyBM x Koncert[misto, cas]
  -- Všechny neuskutečněné kombinace:
  Nerealne := VseKomb \ Cast_programu[id, misto, cas]
  -- Koncerty, na nichž chyběla nějaká skladba B. Martinů:
  Neuplne := Koncert <* Nerealne

Vysl := Koncert \ Neuplne
```

```
SELECT misto, cas FROM Koncert
 MINUS
SELECT misto, cas
FROM (SELECT id FROM Skladatel
           JOIN Autorstvi using (jmeno, prijmeni)
           JOIN Skladba using (id)
      WHERE jmeno='Bohuslav' AND prijmeni='Martinů'
      ) CROSS JOIN Koncert
WHERE (id, misto, cas) NOT IN (SELECT id, misto, cas
                               FROM Cast_programu);
```

22. Který zpěvák má největší z půjčených šatů?

```
SELECT jmeno, prijmeni, velikost
FROM Zpevak
     JOIN Zapujcka_satu using (RC)
     JOIN Saty using (inv_cislo)
WHERE velikost=(SELECT MAX(velikost) FROM Saty NATURAL JOIN Zapujcka_satu);
```

Proč ne NATURAL JOIN? Viz diskuse u dotazu 9.

23. Přehled účasti na zkouškách (v procentech).

```
SELECT jmeno, prijmeni,
   CAST((SELECT COUNT(*) FROM Ucast NATURAL JOIN Zkouska
         WHERE Zpevak.RC=Ucast.RC AND pritomen<>0)
         AS REAL)
        /(SELECT COUNT(*) FROM Ucast NATURAL JOIN Zkouska
          WHERE Zpevak.RC=Ucast.RC
        )*100 || ' %' AS dochazka
FROM Zpevak ORDER BY prijmeni;
```

24. Seznam hlasových skupin: název, počet členů, jméno vedoucího.

```
SELECT  nazev_h,
  poc_clenu,
  jmeno || ' ' || prijmeni AS vedouci
FROM (
      SELECT zkratka, nazev_h, COUNT(RC) AS poc_clenu
      FROM Hlas LEFT JOIN Zpevak ON Hlas.zkratka=Zpevak.hlas
      GROUP BY zkratka, nazev_h
     )
     LEFT JOIN Vedeni USING(zkratka) LEFT JOIN Zpevak USING(RC);
```

25. Seznam míst, na kterých se koncertovalo vícekrát a vždy bylo přítomno více než 50 diváků.

```
SELECT misto, count(misto) AS pocet_koncertu, AVG(divaku) AS prum_divaku
FROM Koncert
WHERE divaku > 50
GROUP BY misto
HAVING count(misto) > 1;
```

26. Skladatel nejvíce zastoupený na koncertech co do počtu skladeb.

```
--Využívám pohled rating_skladatelu:
SELECT jmeno, prijmeni, pocet
FROM Rating_skladatelu
WHERE pocet = (SELECT MAX(pocet) FROM Rating_skladatelu);
```

27. Pro divácky nejúspěšnější koncert určit rozdíl ceny vstupného od průměru.

```
 SELECT (
        SELECT vstupne FROM Koncert
            WHERE divaku = (
                             SELECT max(divaku) FROM Koncert
                            )
       ) - (
        SELECT AVG(vstupne) FROM Koncert
       )
AS odchylka FROM Dual;
```

28. Skladatelé, kteří se mohli během života setkat s Bohuslavem Martinů.

```
SELECT * FROM skladatel
 WHERE rok_narozeni < (SELECT rok_umrti FROM skladatel
                       WHERE jmeno='Bohuslav' AND prijmeni='Martinů')
       AND rok_umrti > (SELECT rok_narozeni FROM skladatel
                        WHERE jmeno='Bohuslav' AND prijmeni='Martinů');
```

29. Seznam skladatelů, jejichž skladby byly na programu koncertů s nadprůměrnou návštěvou.

```
SELECT DISTINCT jmeno, prijmeni
FROM skladatel
   JOIN autorstvi using (jmeno, prijmeni)
   JOIN skladba using (id)
   JOIN cast_programu using (id)
   JOIN koncert using (misto, cas)
WHERE divaku > (SELECT AVG(divaku) FROM koncert);
```

Proč ne NATURAL JOIN? Viz diskuse u dotazu 9.

30. Koncert, na němž bylo uvedeno nejvíce různých skladeb.

```
SELECT *
FROM (
   SELECT misto, cas, COUNT(*) AS pocet
   FROM cast_programu NATURAL JOIN koncert
   GROUP BY misto, cas
      ) WHERE pocet = (
          SELECT MAX(pocet) FROM (
               SELECT COUNT(*) AS pocet
               FROM cast_programu NATURAL JOIN koncert
               GROUP BY misto, cas
              )
     );
```

31. Přehled ošacení - všichni zpěváci a jejich šaty, zároveň všechny šaty a jejich držitelé.

```
SELECT
    COALESCE(prijmeni, 'VOLNÉ') AS drzitel,
    inv_cislo, velikost
FROM Zpevak LEFT JOIN Zapujcka_satu USING(RC)
     FULL JOIN Saty USING(inv_cislo);
```

32. Skladatelé a počet uvedení jejich skladeb

```
CREATE OR REPLACE VIEW rating_skladatelu AS
SELECT jmeno, prijmeni, count(*) AS pocet
FROM Skladatel
      JOIN Autorstvi using (jmeno, prijmeni)
      JOIN Skladba using (id)
      JOIN Cast_programu using (id)
      JOIN Koncert using (misto, cas)
GROUP BY jmeno, prijmeni;

select * from rating_skladatelu;
```

Všimněte si syntaxe - CREATE OR REPLACE, co to dělá je zřejmé - pokud pohled již existuje v datovém slovníku, je nahrazen novou definicí.
Pokud by pohled existoval a vy zadali pouze CREATE VIEW, dostanete chybovou hlášku: "ORA-00955: name is already used by an existing object"

Proč JOIN USING namísto NATURAL JOIN? Viz poznámka u dotazu 9.

33. **Vytvoříme redundantní tabulku ucasti_zpevaku, kde budou předspočítané hodnoty. Tuto tabulku naplníme daty. Atribut koef vyjadřuje koeficient (ponechme stranou, co to znamená).**

```
drop table ucasti_zpevaku;

create table ucasti_zpevaku (
  rc number(10),
  jmeno varchar2(30),
  prijmeni varchar2(30),
  zkousek number(4),
  koncertu number(4),
  koef number(1));


insert into ucasti_zpevaku
  SELECT rc, jmeno, prijmeni,
     (SELECT COUNT(*) FROM Ucast NATURAL JOIN Zkouska
           WHERE Zpevak.RC=Ucast.RC AND pritomen<>0)
            as zkousek,
     (SELECT COUNT(*) FROM Ucast NATURAL JOIN Koncert
            WHERE Zpevak.RC=Ucast.RC AND pritomen<>0)
            as koncertu,
      0 as koef
  FROM Zpevak ORDER BY prijmeni;

commit;

select * from ucasti_zpevaku;
```

Příkaz DROP TABLE na začátku této sekvence je zde opět proto, že předpokládáme vícenásobné spouštění skriptu dotazy.sql (viz skripty). Pokud bude odvozena tabulka ucasti_zpevaku již existovat, opět bychom dostali hlášku:
"ORA-00955: name is already used by an existing object"
Nicméně příkaz CREATE TABLE nemá variantu CREATE OR REPLACE!!!

U testování DML příkazů (INSERT, UPDATE, DELETE) nezapomeňte transakci ukončit příkazem TCL (COMMIT - potvrzení změn nebo ROLLBACK - odvolání změn).

Když to neuděláte, tak vám následné spuštění skriptu, případně DML nebo DDL příkaz nad stejnou tabulkou zůstane viset.

Z hlediska transakčního zpracování je to správné řešení - zdroje v té tabulce jsou dosud blokovány jinou transakcí!!!

Pokud se do této situace dostanete - může vám pomoci ukončení SQL klienta, ze kterého jste nepotvrzený (tedy nenásledovaný příkazem COMMIT či ROLLBACK) DML příkaz poslali.

Ale také to může systému chvíli trvat než se vzpamatuje, takže je lepší s tím počítat.

DDL příkazy (CREATE, ALTER, DROP) jsou "samo-potvrzovací" - před jejich provedením se automaticky provede COMMIT a po jejich provedení rovněž.

Výše uvedené 3 příkazy můžeme v SQL spojit do jednoho:

```
drop table ucasti_zpevaku;

create table UCASTI_ZPEVAKU as
  SELECT rc, jmeno, prijmeni,
     (SELECT COUNT(*) FROM Ucast NATURAL JOIN Zkouska
           WHERE Zpevak.RC=Ucast.RC AND pritomen<>0)
            as zkousek,
     (SELECT COUNT(*) FROM Ucast NATURAL JOIN Koncert
            WHERE Zpevak.RC=Ucast.RC AND pritomen<>0)
            as koncertu,
      0 as koef
   FROM Zpevak ORDER BY prijmeni;

select * from ucasti_zpevaku;
```

34. Zpěvákům, kteří zpívají alt nastavíme v relaci ucasti_zpevaku hodnotu atributu koef tak, že vyjadřuje počet jimi zapůjčených výtisků not.

```
update ucasti_zpevaku u
set koef =
    (select count(*)
     from zapujcka_not z
     where u.rc = z.rc)
where u.rc in (select rc from zpevak where hlas = 'A');

commit;
```

35. Z tabulky ucasti_zpevaku vymažeme ty zpěváky, kteří zpívají basy.

```
delete from ucasti_zpevaku
where rc in (select rc from zpevak where hlas = 'B');

commit;
```

Tabulka pokrytí kategorií SQL příkazů

Kategorie	Pokryta příklazy číslo:	Charekteristika kategorie příkazu
A	1, 4, 9	jednoduché dotazy (SELECT ... FROM ... WHERE)
B	1, 3, 8, 9, 14, 15, 17, 20, 25, 28, 29	Vyber všechny, pro něž platí, že...
C	5, 7, 13	Vyber všechny, pro něž NEplatí, že...
D	10, 21	Vyber ty, pro něž všechny...
E	19	Vyber ty, pro něž žádné...
F1	2, 5, 17	spojení - JOIN ON
F2	4, 7, 9, 13, 19, 29, 31	spojení - NATURAL JOIN | JOIN USING
F3	21	spojení - CROSS JOIN
F4	7, 8, 24, 31	polospojení (vnější) - LEFT | RIGHT OUTER JOIN
F5	31	plné (vnější) spojení - FULL (OUTER) JOIN
G1	3, 6, 8, 11, 15, 16, 18, 19, 21, 22, 28, 29, 30	vnořený dotaz v klauzuli WHERE
G2	2, 7, 8, 10, 17, 21, 24, 30	vnořený dotaz v klauzuli FROM

G3	23, 27	vnořený dotaz v klauzuli SELECT
G4	12, 15, 16, 17	vztažený vnořený dotaz (EXISTS | NOT EXISTS)
H1	20	množinové sjednocení - UNION
H2	5, 7, 10, 13, 21	množinový rozdíl - MINUS (v Oracle)
H3	14	množinový průnik - INTERSECT
I1	16, 18, 22, 23, 24, 25, 26, 27, 29, 30	agregační funkce (count | sum | min | max| avg)
I2	24, 25, 30l	agregační funkce nad seskupenými řádky - GROUP BY (HAVING)
J	10+11+12	stejný dotaz ve třech různých formulacích SQL
K	25	všechny klauzule - SELECT FROM WHERE GROUP BY HAVING ORDER BY
L	32	pohled VIEW
M	26	dotaz nad pohledem
L	33	INSERT .... SELECT příkaz
M	34	UPDATE s vnořeným SELECT příkazem
N	35	DELETE s vnořeným SELECT příkazem

V semestrálce se požaduje alespoň 10 dotazů v relační algebře a alespoň 25 SQL příkazů. SQL příkazů samozřejmě můžete mít více (jako v této semestrálce), je ale podstatné, abyste jimi pokryli všechny kategorie ve výše uvedené tabulce. Samozřejmě tedy jeden dotaz může pokrýt více kategorií.

Skripty

Zdroj pro ER-modelář - pevecky_sbor.xml

Skript pro vytvoření databáze - create.sql - vytvořený ER Modelářem, nicméně zde můžete udělat svoje úpravy. Samozřejmě je vhodné je komentovat.

I v této sekci můžete komentovat jednotlivé odstavce.

Když vyrobíte uvnitř odstavce (para) entitu link, nezapomeňte, že je třeba u ní zadat též hodnotu atributu url.

Skript pro vložení dat do databáze - data.sql

Vaše testovací data by měla být natolik rozsáhlá a tak navržená, abyste na nich mohli dobře ukázat, že vámi navržené SQL příkazy fungují správně. Insert příkazy můžete psát přímo v nějakém textovém editoru, ale můžete také své tabulky naplnit pomocí formulářů v nástroji SQL Developer, který umožňuje exportovat data z tabulek ve formě INSERT příkazů.

Skript s SQL dotazy, který je možné přímo spustit - dotazy.sql

Pokud máte k dispozici nějaký xslt procesor, pak si výše uvedený skript můžete snadno vyrobit ze zdrojového textu semestrálky pomocí formátovací šablony http://service.felk.cvut.cz/courses/X36DBS/xml/make_sql_queries_script.xsl

Pokud nemáte k dispozici XSLT procesor, ale používáte Firefox (nebo Mozillu - tedy Sea Monkey, nebo jak se to vlastně teď jmenuje), pak lze použít tento postup:

1. Stáhněte si html dokument:
http://service.felk.cvut.cz/courses/X36DBS/xml/zobraz_dotazy.html
a umístěte ho do svého projektového adresáře na service.felk.cvut.cz.

2. Zdrojový text vaší semestrálky (umístěn tamtéž) nechť se jmenuje semestralka.xml
(jinak je třeba náležitě editovat soubor zobraz_dotazy.html, viz komentáře uvnitř)

3. Když nyní (ve FireFoxu nebo Mozille) necháte zobrazit soubor zobraz_dotazy.html (umístěný ve vašem projektovém adresáři), tak javascript vložený v této stránce provede XSLT transformaci, jejímž výsledkem bude stránka, ve které je výstupní skript dotazy.sql. Je uzavřený v elementu <pre> takže zachovává formátování. Stačí jej tedy přes schránku vložit do textového souboru dotazy.sql

Výsledek celého postupu aplikovaný na ukázkový soubor test.xml) je zde:
http://service.felk.cvut.cz/courses/X36DBS/xml/test/zobraz_dotazy_test.html

Jednotlivé SQL příkazy budete zřejmě ladit v nástroji SQLDeveloper, protože je to nejpohodlnější.
Vřele však doporučuji použít řádkového klienta sqlplus (pro generování výstupu - viz dotazy.html) do vaší semestrálky.
Umožňuje totiž formátovat výstup přímo do html. Je to zařízeno pomocí příkazu
SET MARKUP HTML a několika dalších příkazů SET, které nastavují prostředí SQL*Plus klienta:

```
set pagesize 1000
set echo on
set markup html on spool on head "
<title>X36DBS - Čtvrtek 7.30 - Zbyněk Lstibůrek - Výstup SQL příkazů </title>
<style type ='text/css'><!--body {background: #ffffc6} --></style>"
body "<h2>X36DBS - Čtvrtek 7.30 - Zbyněk Lstibůrek - Výstup SQL příkazů </h2>"
spool dotazy.html

..... pak jsou samotné dotazy, s jejich slovním zadáním (komentář je uvozen pomocí --)

set markup html off
spool off
```

!!! A ještě pozor - v tomto komentáři jsem kvůli formátování rozdělil příkaz set markup thml on
na 4 řádky. Nicméně v souboru dotazy.sql, tohle neprojde !!!!

Spuštění souboru dotazy.sql v prostředí SQL*Plus klienta (na solarisech) vypadá takto
(předpokládám, že v aktuálním adresáři máte skript dotazy.sql, po skončení bude výstup v dotazy.html):

```
-bash-3.00$ sqlplus

SQL*Plus: Release 10.2.0.2.0 - Production on Mon Sep 24 11:26:50 2007

Copyright (c) 1982, 2005, Oracle.  All Rights Reserved.

Enter user-name: <vaze_uživatelské_jméno>
Enter password: <vaše_heslo>

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.2.0 - Production
With the Partitioning and Data Mining options

SQL> @dotazy.sql
..... Teď se provádí dotazy, výstup jde na obrazovku a do souboru dotazy.html

SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.2.0 - Production
With the Partitioning and Data Mining options
-bash-3.00$
```

*(Jak vidno, podstatným příkazem v SQL*Plus klientovi je "@jmeno*

Výstup předchozího skriptu - dotazy.html.

Vyrobil jsem ho výše uvedeným postupem - tedy s použitím šablony, která ze zdrojového textu vaší semestrálky vytáhne pouze SQL příkazy (a jejich slovní specifikaci, ze které udělá komentář) a obalí to řídícími příkazy pro vytvoření HTML stránky.
Vycházím z toho, že příkazy budete ladit jednotlivě (zřejmě v nástroji SQL Developer, kde to je velmi pohodlné). Pro celkovou kontrolu je však mnohem lepší mít všechno pěkně v jednom dokumentu.

Závěr

Úspěšně jsem vytvořil svůj první databázový projekt a osvojil jsem si základní metody databázového návrhu. Vzniklé databázi lze jistě vytknout řadu nedostatků. Ty pramení zejména z toho, že v době vytváření ER schématu jsem neměl jasnou představu o některých implementačních problémech souvisejících např. se spojováním tabulek. Kdybych navrhoval databázi znovu, asi bych nevolil za klíč řetězcové atributy, tím méně jejich dvojice. Také bych lépe rozložil atributy v ISA hierarchii Událost-Koncert-Zkouška, aby se např. místo konání nekopírovalo do všech instancí (to souvisí i s předchozí poznámkou o volbě klíčů). Protože jsem si tyto problémy začal naplno uvědomovat teprve v okamžiku, kdy jsem měl již hotová zkušební data a zabýval se dotazy, rozhodl jsem se schéma již neměnit a vyhnout se tak zavlečení nějakých dalších chyb.

Stejně tak lze komentovat závěr.

Odkazy

[1] Pokorný, Jaroslav, Halaška, Ivan, Databázové systémy, Praha 1998

A bude-li to třeba, můžete komentovat také odkazy.

[2] Šimůnek, Milan, SQL, kompletní kapesní průvodce, Praha 1997