Databázové systémy 2

db.cs@vsb.cz

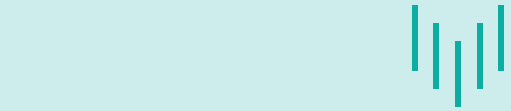
Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – Technická univerzita Ostrava

2023/2024

Úkol 1: Podmínky (1)

1. Vytvořte proceduru `PAddStudentToCourse` s parametry `p_student_login`, `p_course_code` a `p_year`. Procedura zapíše studenta s daným loginem k danému kurzu v daném roce. K zápisu však dojde pouze pokud není naplněna kapacita kurzu (atribut `Course.capacity`). V opačném případě procedura vypíše: 'Kurz je již plně obsazen'.
2. Vytvořte trigger `TInsertStudentCourse`, který před vložením záznamu do tabulky `StudentCourse` zkontroluje, zda kurz není plně obsazen. Pokud ano, vypíše se varovné hlášení: 'Kapacita kurzu byla překročena'.
3. Pomocí výjimky upravte trigger `TInsertStudentCourse` tak, aby v případě plné obsazenosti kurzu k zápisu nedošlo. Tj. deklarujte výjimku `EXCEPTION`, kterou na příslušném místě vyvolejte příkazem `RAISE`.

Úkol 1: Podmínky (2)

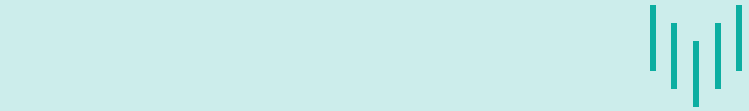


4. Vytvořte funkci `FAddStudent4` fungující obdobně jako funkce `FAddStudent3` (viz předchozí cvičení). Kromě návratových hodnot 'ok' a 'error' bude mít funkce navíc návratovou hodnotu 'full'. Funkce tuto hodnotu vrátí v případě, že kapacita daného ročníku je již naplněna. Kapacity pro jednotlivé ročníky jsou: 1. - 20, 2. - 15, 3. - 10, 4. - 10, 5. - 10.
5. Napište proceduru `PDeleteTeacher`, s parametrem `p_login` pro smazání užitele. Procedura před smazáním učitele zkontroluje, zda učitel nemá přiřazené nějaké předměty. Pokud ano, přiřadí tyto předměty jinému učiteli s nejmenším počtem vyučovaných předmětů. Poté proběhne samotné smazání učitele. Pokud žádný jiný učitel neexistuje, ke smazání učitele nedojde. Proceduru napište jako transakci.

Úkol 2: Cykly

1. Vytvořte pomocnou funkci `FLoginExists` s parametrem `p_login`, která vrátí true právě když student s daním loginem existuje.
2. Vytvořte funkci `FGetNextLogin` s parametrem `p_lname`, která bude generovat login ve formátu 'aaa000', kde 'aaa' představuje první tři písmena z příjmení a '000' sekvenční číslo zarovnané na 3 znaky (s případnými počátečními nulami). Funkce `FGetNextLogin` pomocí cyklu a pomocné funkce `FLoginExists` nalezne a vrátí neexistující login s nejnižším sekvenčním číslem.
3. Upravte funkci `FAddStudent4` (viz předchozí slide) tak, aby pro generování loginu použila `FGetNextLogin`. Ověřte generování loginu opakovaným voláním `FAddStudent4` se stejným příjmením.

Úkol 3: Kurzory

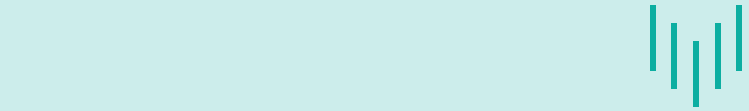


1. Vytvořte anonymní proceduru, která za pomocí kurzoru vypíše jména a příjmení všech studentů. Vyzkoušejte si jak syntaxi explicitního kurzoru (`OPEN`, `FETCH`, `CLOSE`), tak syntaxi kurzoru `FOR LOOP`.
2. Vytvořte proceduru `PAwardStudents` s parametry `p_year` a `p_amount`, která udělí stipendium 5-ti nejlepším studentům dle nasbíraných bodů ze všech předmětů v daném roce. Prvnímu ze studentů bude na účet připsána celá částka `p_amount`, každému dalšímu se pak částka sníží o polovinu.
3. Vytvořte funkci `FExportPointsCSV` s parametrem `p_year`. Funkce bude vracet textový řetězec reprezentující tabulku s body studentů v daném ročníku. Tabulka bude formátována jako CSV (comma separated values). Na každém řádku bude vždy uveden login, jméno, příjmení a celkový počet bodů. Příklad výstupu:

```
nov123,Jan,Novák,557
svo321,Petr,Svoboda,457
```

Uvažujte jen studenty, kteří v daném roce studují alespoň jeden předmět.

Domací úloha (1)

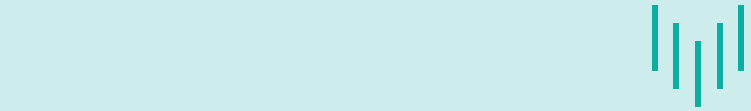


1. Vytvořte proceduru `PMoveToNextGrade`, která každému úspěšnému studentovi zvýší ročník (atribut `grade`) o 1 a zapíše mu všechny volné předměty daného ročníku. U studentů posledního ročníku bude nový ročník nastaven na -1. Úspěšný student musí za každý zapsaný předmět ve svém ročníku získat alespoň 51 bodů. Při procházení studentů a zvyšování ročníku postupujte od nejvyššího po nejnižší ročník.

   V rámci úlohy naprogramujte a využijte následující metody:

   a. Funkce `FAllSubjectsPass` s parametrem `p_login`, která zkontroluje, zda má student úspěšně ukončené všechny předměty svého ročníku. Pokud ano, funkce vrátí true, jinak false.
   b. Funkce `FNextGrade` s parametrem `p_login`, vrátí následující ročník, který se má studentovi zapsat.
   c. Procedura `PSetFreeCourses` s parametry `p_login` a `p_grade`, která studentovi zapíše všechny předměty daného ročníku, které v daném akademickém roce (atribut `year`) ještě mají volnou kapacitu. Při zjišťování obsazenosti kurzu nezapočítávejte studenty, kteří již z daného kurzu získali 51 bodů. Jako akademický rok uvažujte rok z aktuálního data.

Domací úloha (2)



1. Vytvořte funkci `FGetCreateScript` s parametrem `p_tableName`, která pro tabulku definovanou parametrem vytvoří a vrátí SQL skript pro vytvoření dané tabulky. Při sestavování skriptu uvažujte pouze strukturu tabulky bez klíčů a dalších integritních omezení. Data ignorujte.
2. Vytvořte trigger `TChangeCourseCapacity`, který před aktualizací nad tabulkou `Course` zkontroluje, zda došlo ke změně kapacity kurzu a provede následující:
   a. Pokud jde o snížení kapacity, trigger zkontroluje, zda počet studentů aktuálně studujících daný předmět (tj. mají zapsaný předmět, ale zatím nedosáhli 51 bodů) není vyšší než nová kapacita. Pokud ano, dojde k výjimce a aktualizace kapacity se neprovede.
   b. Pokud jde o zvýšení kapacity, trigger zkontroluje, zda všichni studenti ročníku, ve kterém se předmět vyučuje, mají tento předmět zapsaný. Pokud ne, studentům se daný předmět přiřadí abecedně (dle jejich příjmení) až do naplnění kapacity.