VYSOKÉ UČENI TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Databázové systémy

2016/2017

Dokumentácia

Přihlašování do předmětů

1.máj 2017

Michal Vaško (xvasko14)

Martin Vaško (xvasko12)

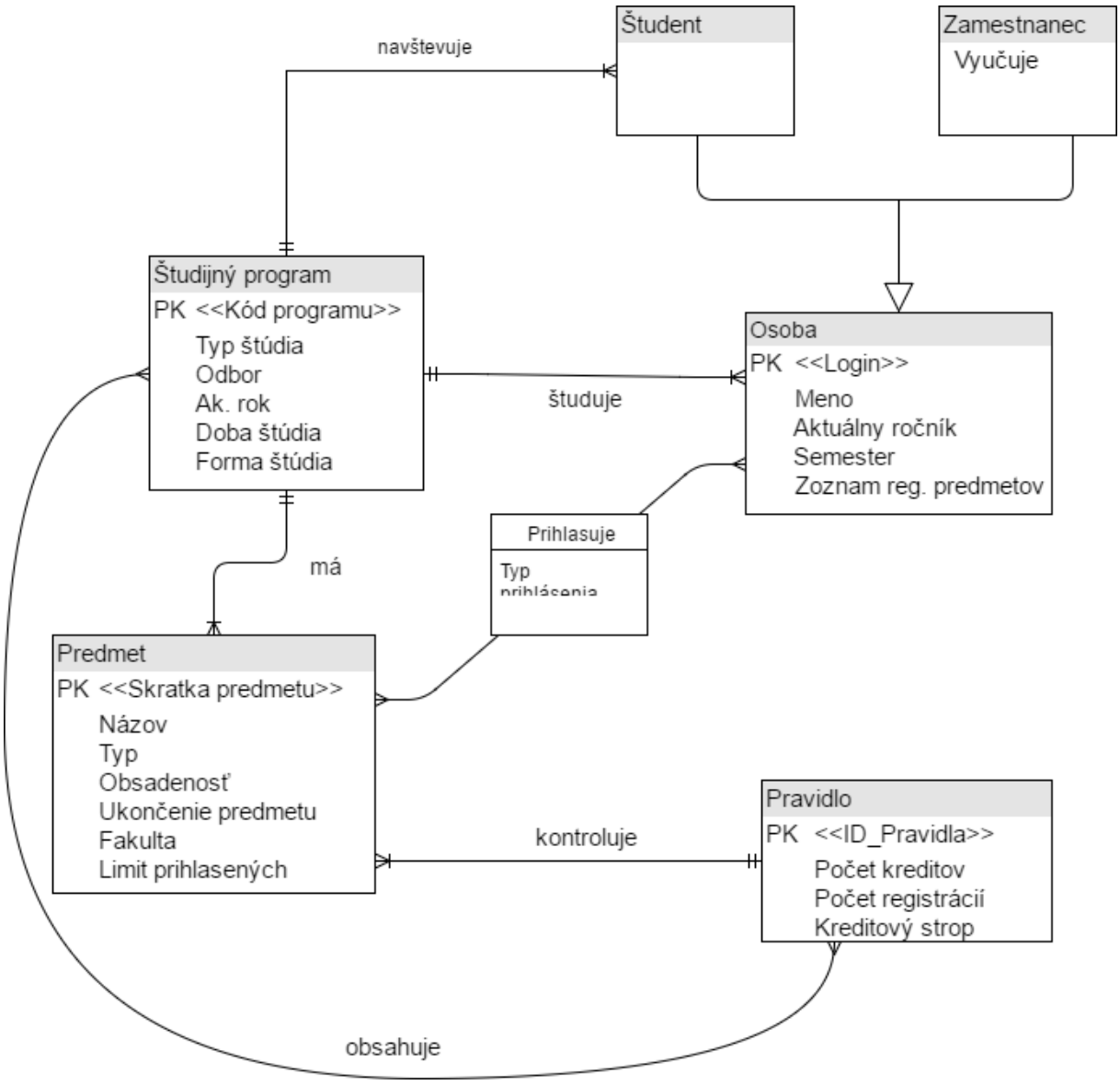
Zadanie projektu

Přihlašování do předmětů

Navrhněte modul informačního systému fakulty, který bude umožňovat zápis studentů do předmětů prostřednictvím webu. Systém bude spravovat informace o nabízených studijních programech a předmětech různých typů (povinné, volitelné apod.) v nich. Škola používá kreditový systém a každý program má určitá pravidla pro počty kreditů z jednotlivých typů předmětů, které student musí získat. Systém musí zajistit při zápisu kontrolu těchto pravidel ale musí být schopen reagovat pružně na případnou změnu pravidel. Při analýze požadavků využijte zkušeností z programu, který studujete.

Dátový model – ERD

Finálny a upravený dátový model nášho zadania.

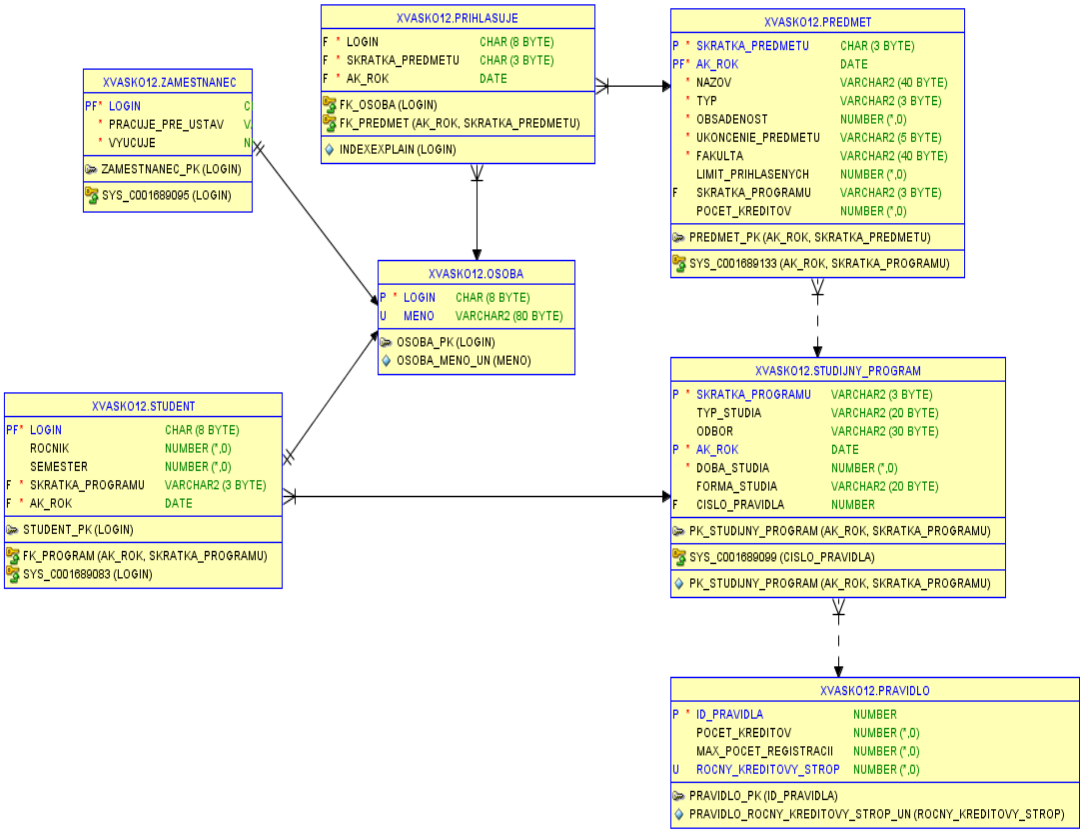


Generalizácia

Generalizáciu sme využili pri entite Osoba. Keďže nejaká osoba môže byt aj študent ktorý študuje na danej škole ale aj zamestnanec ktorý tam vyučuje. Pri transformácii tejto generalizácie sme vytvorili ďalšie dve tabuľky a to študent a zamestnanec.

Schéma relačnej databázy

Po dátovom modeli tu máme vygenerovanú našu schému relačnej databázy.



Implementácia

Skript nám najskôr vytvorí základne objekty a tabuľky ktoré následne naplní ukážkovými hodnotami a taktiež nastaví primárne a cudzie kľúče. Skript potom obsahuje časti z predošlých pod úloh projektu ako *SELECT* príkazy a taktiež úlohy tohto zadania a to triggery, procedúry, materializovaný pohľad, *EXPLAIN PLAN* a pridelenie práv druhému členovi tímu.

Triggery

Vytvorili sme štyri triggery z toho prvý sme implementovali podľa napevno stanoveného zadania. Jedna sa o automatickú inkrementáciu primárneho kľúča umiestneného v tabuľke *PRAVIDLO*. Konkrétne tento stanovený trigger sme realizovali pomocou sekvencie ktoré si pamätá posledné číslo posledného pridaného primárneho kľúča. Ďalší trigger nám zisťuje ci sme prekročili limit pre registrovanie do predmetu konkrétne v tabuľke *PREDMET*. Cele to je založené na tom že porovnávame limit prihlásených žiakov s obsadenosťou a na základe toho vieme určiť či to prekročilo danú hodnotu alebo sa tam dá ešte registrovať. Tretí trigger funguje na princípe, že ak sa prihlásime k nejakému predmetu tak sa zvýši jeho obsadenosť. Špeciálne sme pri tomto triggeru aj použili procedúru ktorú si voláme. Použitá je tabuľka *PRIHLASUJE* a taktiež ma príznak *AFTER INSERT ON*, čo znamená že sa spúšťa po vložení dát do určitej tabuľky. Posledný trigger kontroluje login a to v zmysle, že zadáme náš login a on zisťuje či je správny, napr. či ma potrebný počet znakov a pod. Tu sa využíva tabuľka *OSOBA* a *BEFORE INSERT ON* ktorý sme využívali aj v prvých dvoch a ten nám hovorí že sa daný trigger spúšťa pred vložením dát do určitej tabuľky.

Procedúry

Vytvorili sme dve netriviálne procedúry a jednu triviálnu už vyššie spomenutú ktorú sme využili pri triggery. Podľa zadania sme v každej procedúre použili kurzor a taktiež bola použitá premenná s dátovým typom odkazujúcim sa na riadok či typ stĺpca tabuľky. V prvej procedúre zisťujeme percentuálny podiel študentov s počtom kreditov. Funguje to na princípe že sa urči nejaké číslo a my zistíme koľko percent študentov ma viac kreditov od toho čísla. Ako parameter vložíme *pocetKreditov*. Druhá procedúra nám zisťuje našich spolužiakov. Ako parameter vložíme meno žiaka a na základe jeho štúdia, ročníka a teda študijného programu vypíše jeho spolužiakov.

Materializovaný pohlaď

Podľa zadania sme mali implementovať materializovaný pohlaď patriaci druhému členovi tímu, ktorí používa tabuľky prvého člena tímu. Ďalej bolo potrebne vytvoriť materializované logy ktoré v sebe obsahujú zmeny hlavnej tabuľky, aby sa mohlo pri zmenách tabuľky používať rýchlu obnovu po potvrdení zmien, namiesto kompletnej obnovy, kedy by sa muselo cely dotaz materializovaného pohľadu spúšťať odznova čo by samozrejme trvalo dlhšie. Následne sme si vytvorili materializovaný pohlaď a vyskúšali jeho funkčnosť a možnosti.

Logging - zaznamenáva operácie s pohľadom

Cache - optimalizácia čítania z pohľadu

Build immediate - naplní pohlaď ihneď po jeho vytvorení

Refresh fast on commit - aktualizácia pohľadov podľa logov po potvrdení zmien v tabulkach

Enable query rewrite – bude používaný optimalizatorom

Explain plan a vytvorenie indexu

Explain plan sme mali predviesť na jednoduchom dotaze. Použili sme *SELECT* dotaz a spustili explain plan ale bez použitia indexu. Potom sme si zadefinovali index a spustili znova. Následne sme dostali výstupy ktoré môžete vidieť tu.

ID	Operation	Name	Cost(%CPU)
0	SELECT STATEMENT		7(15)
1	HASH GROUP BY		7(15)
2	NESTED LOOPS		6(0)
*3	HASH JOIN		6(0)
4	TABLE ACCES FULL	PREDMET	3(0)
5	TABLE ACCES FULL	PRIHLASUJE	3(0)
*6	INDEX UNIQUE SCAN	SYS_C001672513	0(0)

ID	Operation	Name	Cost(%CPU)
0	SELECT STATEMENT		6(17)
1	HASH GROUP BY		6(17)
2	NESTED LOOPS		5(0)
*3	HASH JOIN		5(0)
4	TABLE ACCES FULL	PREDMET	3(0)
5	TABLE ACCES BY INDEX ROWID BATCHED	PRIHLASUJE	2(0)
6	INDEX UNIQUE SCAN	INDEXEXPLAIN	1(0)
*7	INDEX UNIQUE SCAN	SYS_C001672513	0(0)

Index ktorý sme si definovali, databáza najprv nepoužila keďže pri takom malom počte riadkov sa to nevyplatí. Avšak my sme ju donútili použiť ten index čo malo následok na *COST* ktorý sa zmenši ale naopak zvýšilo sa vyťaženie CPU.

Následne si rozoberieme fungovanie dotazu. *SELECT STATEMENT* hovorí o tom že bol uskutočnení dotaz select, *HASH GROUP BY* tu zoraďujeme podľa hash kľúča, potom *NESTED LOOPS*, ktoré hovoria že tabuľky sa spájajú naivne, prechádzaním po jednej tabuľke a pre každú položku z prvej tabuľky sa prejdú všetky riadky z druhej tabuľky. Ďalej nasleduje *HASH JOIN* teda párovanie záznamov spojovaných tabuliek cez hash kľúče spojenia. *TABLE ACCES FULL* prechádzanie tabuľkou bez použitia indexu. Nakoniec sa použil *INDEX UNIQUE SCAN*, ktorý sa dostáva k tabuľkám cez B-strom a vypadne nám jedinečný riadok podľa primárneho kľúča. Všetko to bolo bez použitia indexu. Ak sme išli s použitím indexom tak zmena nastala pri použití *TABLE ACCES BY INDES ROWID BATCHED* v tabuľke *PRIHLASUJE* na piatom ID kedy sa pristupuje do tabuľky cez konkrétny riadok. Potom sa vykoná *INDEX FULL SCAN* s indexom *INDEXEXPLAIN*, čiže sme použili index na výpis, bez toho aby sme sa museli pozerať do tabuľky.

Prístupové práva

Tu sme simulovali pridelenie práv. Konkrétne sme tuto možnosť dali členovi tímu xvasko12.Dostal pravá ako študent k tabuľkám, ktoré sú potrebne pre správu študenta a zároveň ma prístup k materializovanému pohľadu.

Zhodnotenie

Cely skript sme vytvorili v nastrojí SQL Developer , v prostredí Oracle na školskom serveri Oracle 12c. Všetky potrebne informácie k tvorbe projektu sme brali z prednášok, opory a hlavne z cvičení ktoré nám dosť pomohli. Niektoré dodatočne informácie sme si našli aj na internete ako na Tutorialspoint a pod.