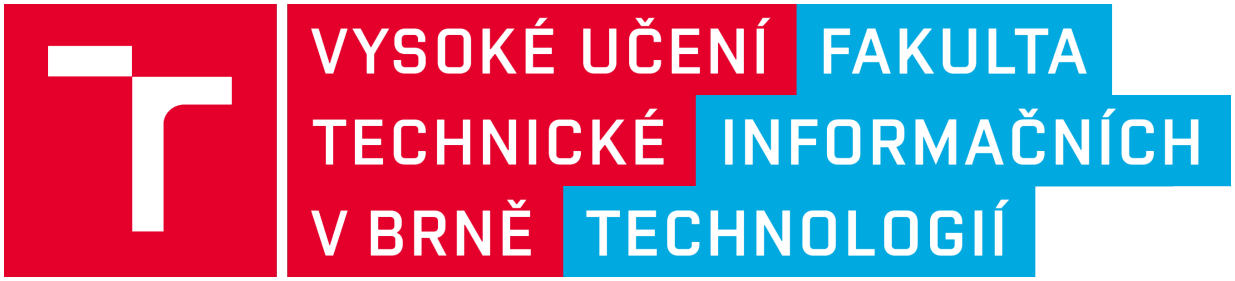


IDS - Databázové systémy

Návrh databáze

Zkoušky

Adrián Hučko(xhucko02)
Martin Bublavý(xbubla02)

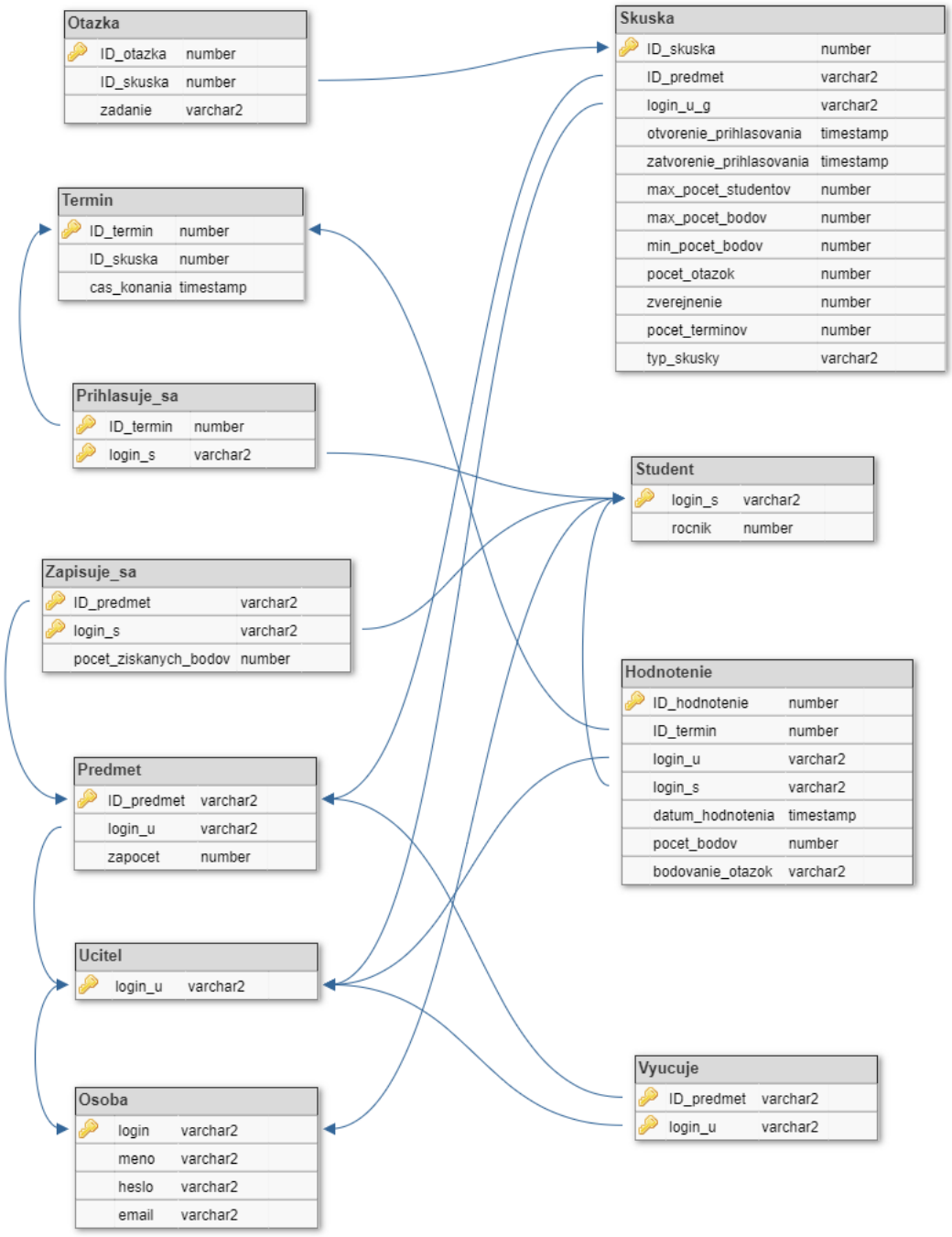
25. dubna 2022

1 Zadanie

Zkoušky

Navrhněte modul fakultního informačního systému, který bude umožňovat studentům se přihlašovat na zkoušky a vyučujícím tyto zkoušky hodnotit. Zkouška může být buď semestrální nebo půl semestrální, přičemž semestrální zkouška má vždy alespoň 3 termíny. Studenti se mohou na zkoušky přihlašovat pouze pokud je přihlašování otevřeno, u semestrální zkoušky musí navíc systém zajistit, že se student nesmí přihlásit na více než 3 termíny a také, že na další termín se může přihlásit až po proběhnutí termínu, na který je aktuálně přihlášen. Pro každou zkoušku je potřeba uchovávat základní informace o ní (datum a čas, kdy se koná, maximální počet studentů, jenž se mohou přihlásit, maximální počet bodů ze zkoušky a minimální počet bodů potřebný pro úspěšné složení zkoušky, počet otázek, ...). Body ze zkoušky mohou vkládat pouze učitelé, jenž jsou k předmětu, pod který zkouška spadá, přiřazeni. Pokud má zkouška nastaveno více otázek, musí hodnocení obsahovat body pro každou z těchto otázek. Hodnocení také může obsahovat komentář. Kromě přihlašování musí systém poskytnout studentům i možnost se odhlásit (pokud je přihlašování stále otevřeno). Studenti také mohou zjišťovat body získané ze zkoušky spolu s informacemi kdo a kdy tuto zkoušku hodnotil.

2 Finálna schéma databázy



Obrázek 1: Schéma databázy

3 Generalizácia/špecializácia

Generalizácia sa u nás využíva vo vzťahu medzi entitnou množinou Osoba a entitnými množinami `Študent` a `Učiteľ`. Pre tieto dve entity sme sa taktiež rozhodli vytvoriť tabuľky z dôvodu rôzneho obsahu. Všetky tieto tabuľky zdielajú spoločný primárny kľúč `login`, pričom v tabuľkách `Učiteľ` a `Študent` sa zároveň jedná o cudzí kľúč ,ktorý odkazuje naspäť do tabuľky osoba.

4 Implementácia

4.1 Triggery

Trigger na kontrolu zadaného loginu ,ktorý kontroluje či login zodpovedá formátu a správnosť poradového čísla. Trigger vytvára testovací login ,ktorý nakoniec porovnaný so zadaným. Vytváranie loginu prebieha v krokoch:

1. Vytvorenie textovej časti
   - ak je prezvisko dlhšie alebo rovné ako 5 písmen tak sa textová časť loginu vytvorí z priezviska
   - ak je menšie ako 5 písmen tak sa zoberie celé priezvisko a k nemu sa pripojí chýbajúci počet písmen z mena
2. Vytvorenie poradového čísla
   - ak je počet ľudí s rovnakým loginom menší ako 10 tak je 1. znak 0 a 2. znak je
     $$number\_of\_people + 1$$
   - ak je počet ľudí s rovnakým loginom väčší ako 9 a menší ako 100 tak je poradové číslo
     $$number\_of\_people + 1$$
   - ak je počet ľudí s rovnakým loginom väčší ako 99 a menší ako 360
     - tak 1. znak je
       $$number\_of\_people \;//\; 26$$
     - tak 2. znak je
       $$(number\_of\_people \;\%\; 26) + 97$$
   - ak je počet ľudí s rovnakým loginom väčší ako 359 a menší ako 1036
     - tak 1. znak je
       $$(number\_of\_people \;//\; 26) + 97$$
     - tak 2. znak je
       $$(number\_of\_people \;\%\; 26) + 97$$
   - ak je počet ľudí s rovnakým loginom väčší ako 1035 tak by pridanie novej osoby spôsobilo prekročenie limitu tak sa vyhodí chyba a osoba sa nepridá.

Výsledky výpočtov ,v ktorých sa pripočítava 97 sa prevádzajú na hodnotu v ASCII tabuľke pomocou funkcie `CHR`. Poradové čísla sú v rozmedzí 00 až ZZ. Delenie a modulo 26 je z dôvodu dĺžky abecedy.

Trigger na kontrolu emailu kontroluje správny formát zadaného emailu pomocou funkcie `REGEXP_LIKE`,ktorá porovnáva zadaný email podľa vytvoreného regulárneho výrazu.

Trigger na kontrolu či je skúška zadaná garantom vyberie z tabuľky `predmet` všetky záznamy ,ktoré obsahujú spojenie daného učiteľa s daným predmetom ak takýto záznam nenájde tak vypíše error a skúška sa nevytvorí.

4.2 Procedúry

V skripte sa nachádzajú implementácie dvoch procedúr `pocet_pisomiek` a `student_info`, ktoré pre svoju činnosť využívajú kurzor pre prípadnú prácu s viacerými riadkami a premenné s dátovým typom odkazujúcim sa na riadok tabuľky.

Procedúra pocet_pisomiek : Po prijatí argumentu `ID_skuska` načíta do kurzoru všetky loginy učiteľov ,ktorí hodnotili termíny skúšky s daným ID. Ďalej pokračuje iterovaním cez všetky loginy a ak príde na taký ,ktorý ešte nebol spracovaný tak vypíše informácie o učiteľovi s daným loginom a zvýši počítadlo učiteľov. Taktiež pre každý záznam zvyšuje počítadlo písomiek. Po spracovaní všetkých záznamov vypíše celkový počet písomiek a učiteľov. Ak žiadne záznamy nenájde tak končí s chybovou správou.

Príklad výstupu :

```
Meno ucitela: Augustus Moody
Login ucitela: xmoody00
Email ucitela: erik3791@sgisfg.com

Meno ucitela: Harry Potter
Login ucitela: xpotte00
Email ucitela: irenka933@debb.com

Meno ucitela: Palo Sek
Login ucitela: xsekpa00
Email ucitela: Palo@debb.com

Celkovy pocet pisomiek: 9
Pocet ucitelov: 3
```

Procedúra student_info : Po prijatí argumentu `login_student` načíta do kurzoru záznamy všetkých predmetov ,ktoré má zapísané. Ďalej načíta do premennej `osoba_info` záznam s informáciami o ňom a do premennej `rocnik` načíta ročník ,v ktorom práve je a tieto informácie vypíše následne iteruje cez všetky záznamy kurzoru a pre každý vypíše ID predmetu a aktuálny počet bodov.

Príklad výstupu :

```
Meno studenta: Jozef Discord'
Login studenta: xdisco00
Email studenta: igor1006@enhytut.com
Rocnik: 4

-------IFJ-------
Ziskane body: 35

-------IOS-------
Ziskane body: 33
```

4.3 Materializovaný pohľad

Počas vytvárania materializovaného pohľadu sme danému pohľadu nastavili hneď na začiatku niekoľko vlastností.

- Vlastnosť `CACHE` - postupne optimalizuje čítanie z pohľadu `BUILD IMMEDIATE` ktorá naplní pohľad ihneď po jeho vytvorení
- Vlastnosť `ENABLE QUERY REWRITE` - materializovaný pohľad bude použitý pri optimalizácii dotazu, na ktorom bol založený

Po vytvorení pohľadu sme si najprv nechali vypísať obsah, následne sme zmenili dáta v tabuľke, z ktorej bol tento pohľad vytvorený, a opäť sme vypísali jeho obsah. Tým, že jednotlivé výpisy sa navzájom líšili, sme si potvrdili funkčnosť nášho pohľadu.

4.4 EXPLAIN PLAN

Pomocou príkazu `EXPLAIN PLAN` sme demonštrovali optimalizáciu pri spracovaní dotazu zavedením indexu do tabuľky. Ako 1. sme pustili dotaz na tabuľkou bez indexu a následne s indexom ,kde vidíme pokles ceny. Ešte väčšie urýchlenie by sa dalo dosiahnuť pomocou druhého indexu na stĺpec `login_u` v tabuľke učiteľ kvôli tomu ,že sa jedná o cudzí kľúč.

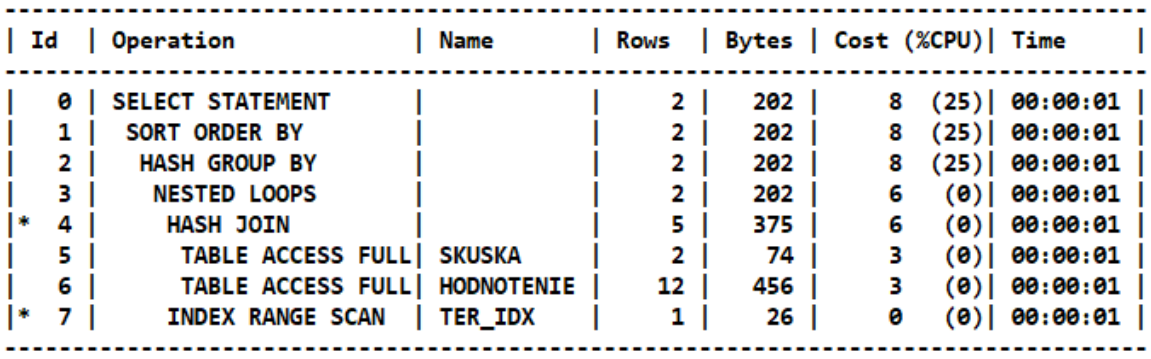
Indexovanie sme zaviedli na stĺpce `ID_skuska` a `ID_termin` z tabuľky `termín` kvôli tomu ,že sa tabuľky cez tieto stĺpce spájajú.

```
-------------------------------------------------------------------------------
| Id  | Operation            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |            |     2 |   202 |    11  (19)| 00:00:01 |
|   1 |  SORT ORDER BY       |            |     2 |   202 |    11  (19)| 00:00:01 |
|   2 |   HASH GROUP BY      |            |     2 |   202 |    11  (19)| 00:00:01 |
|*  3 |    HASH JOIN         |            |     2 |   202 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN        |            |     4 |   252 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| SKUSKA     |     2 |    74 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| TERMIN     |     4 |   104 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | HODNOTENIE |    12 |   456 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Obrázek 2: Bez indexu

4.4.1 Postup vykonávania dotazu bez indexu

1. **TABLE ACCESS FULL** - prečíta celú tabuľku `SKUSKA`
2. **TABLE ACCESS FULL** - prečíta celú tabuľku `TERMIN`
3. **HASH JOIN** - spojí záznamy tabuliek `SKUSKA` a `TERMIN` cez hashovací kľúč spojenia
4. **TABLE ACCESS FULL** - prečíta celú tabuľku `HODNOTENIE`
5. **HASH JOIN** - spojí záznamy tabuliek už spojených tabuliek `SKUSKA` a `TERMIN` a tabuľky `HODNOTENIE` cez hashovací kľúč spojenia
6. **HASH GROUP BY** - zoskupí záznamy tabuľky na základe hashovacieho algoritmu
7. **SORT ORDER BY** - zoradí záznamy tabuľky



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		2	202	8 (25)	00:00:01
1	SORT ORDER BY		2	202	8 (25)	00:00:01
2	HASH GROUP BY		2	202	8 (25)	00:00:01
3	NESTED LOOPS		2	202	6 (0)	00:00:01
* 4	HASH JOIN		5	375	6 (0)	00:00:01
5	TABLE ACCESS FULL	SKUSKA	2	74	3 (0)	00:00:01
6	TABLE ACCESS FULL	HODNOTENIE	12	456	3 (0)	00:00:01
* 7	INDEX RANGE SCAN	TER_IDX	1	26	0 (0)	00:00:01

Obrázek 3: S indexom

4.4.2 Postup vykonávania dotazu s indexom

1. **TABLE ACCESS FULL** - prečíta celú tabuľku `SKUSKA`
2. **TABLE ACCESS FULL** - prečíta celú tabuľku `HODNOTENIE`
3. **HASH JOIN** - spojí záznamy tabuliek `SKUSKA` a `HODNOTENIE` cez hashovací kľúč spojenia
4. **INDEX RANGE SCAN** - pristúpy k priľahlým záznamom indexu a potom použije hodnoty `ROWID` na získanie riadkov tabuľky
5. **NESTED LOOPS** - porovnáva riadky z jednej tabuľky s riadkami z druhej tabuľky
6. **HASH GROUP BY** - zoskupí záznamy tabuľky na základe hashovacieho algoritmu
7. **SORT ORDER BY** - zoradí záznamy tabuľky

5 Prístupové práva

Druhý člen tímu dostal prístupové práva na všetky tabuľky pomocou príkazu `GRANT ALL ON` a takisto boli udelené práva na spúšťanie predpripravených procedúr pomocou príkazu `GRANT EXECUTE ON`.