Projekt IDS 2020/2021 - dokumentace

Dokumentace k SQL projektu (soubor `xbuchn00_xjirgl01.sql`) do předmětu IDS 2020/2021

Autoři: Tereza Buchníčková (`xbuchn00`), Karel Jirgl (`xjirgl01`)

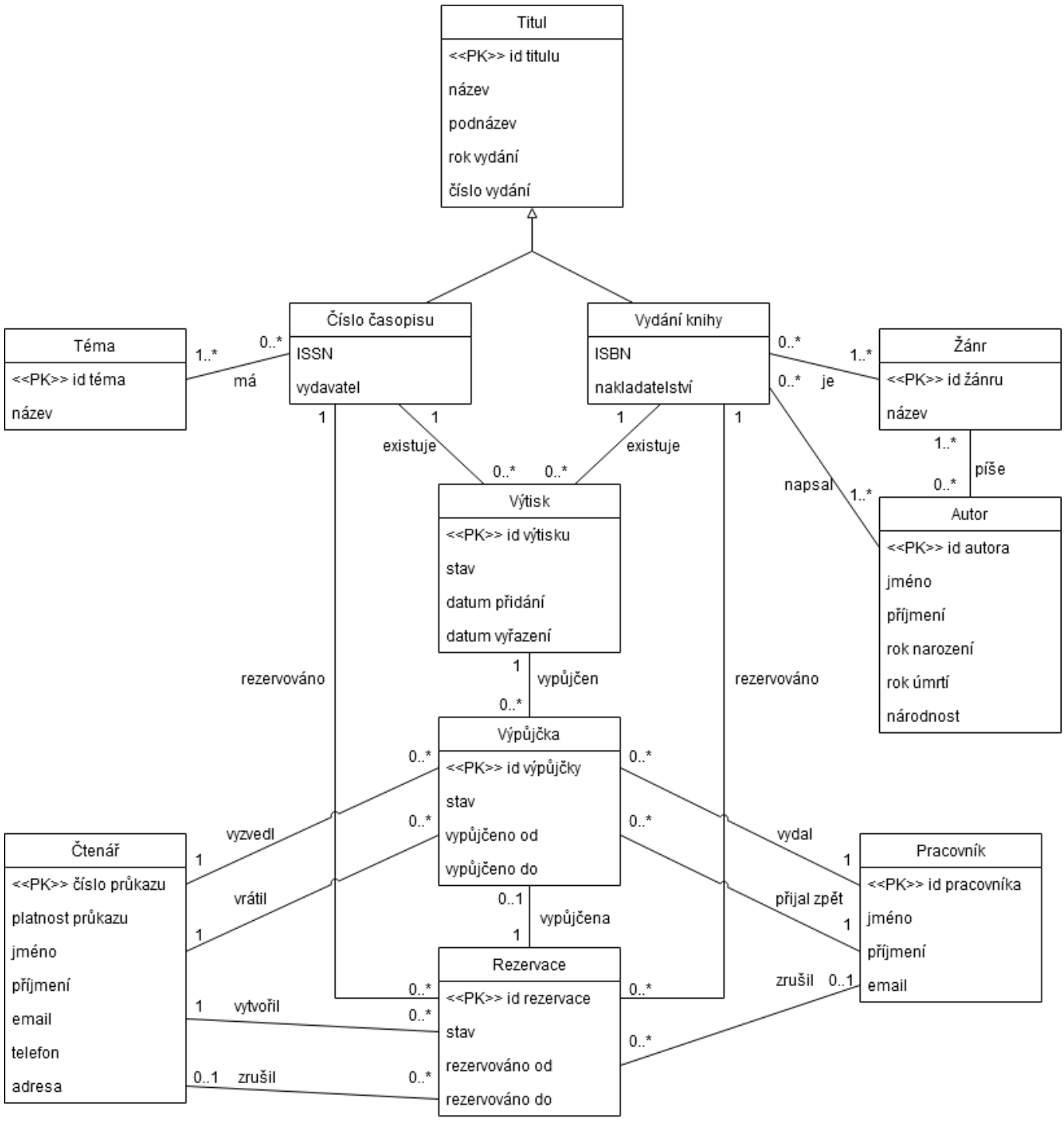
Projekt č. 21 - Knihovna1

zadání

Vytvořte jednoduchý IS knihovny, který by poskytoval informace o titulech knih a časopisů, o registrovaných čtenářích, vypůjčených exemplářích apod. a umožňoval také provádět rezervace žádaných titulů. Z hlediska přístupu k datům uvažujte dvě skupiny uživatelů: pracovníky knihovny a čtenáře.

1. část - ER diagram databáze

V průběhu implementace dalších částí projektu, jsme narazili na nedokonalosti prvního návrhu ER diagramu, a tak došlo k jeho mírné úpravě.



Popis ER diagramu

V knihovně máme dva druhy **titulů**, **knihy** a **časopisy**. Protože se jedná o podskupiny s rozdílnými atributy, rozhodli jsme se použít generalizaci/specializaci. Typ entity kniha je dále navázán na **autora** a **žánr**. Žánr uchováváme jako typ entity, protože několik knih může mít stejný žánr a je tedy vhodnější, aby se nejednalo jen o atribut.

Dále jsme vytvořili typ entity **vydání knihy** a poté samotný **výtisk**, který označuje jeden konkrétní kus knihy. U časopisů jsme vytvořili typ entity **téma**, který popisuje, o jaký druh časopisu se jedná (např.: sportovní). Časopis je navázán na **číslo časopisu**, a to je navázáno na **výtisk**, který je společný pro knihu i časopis, protože se u samotného výtisku jejich atributy neliší.

Dále jsme navrhli typ entity **rezervace** a **výpůjčka**, kde každá instance rezervace nebo výpůjčky se vztahuje pouze na jeden výtisk. Každému čtenáři by mělo být umožněno zarezervovat si konkrétní vydání knihy nebo číslo časopisu. Rezervaci jsme proto spojili s těmito typy entit, protože se vydání mezi sebou mohou lišit. K vytvoření výpůjčky výtisku dojde, pokud si čtenář rezervovaný výtisk vyzvedne.

Dále v našem systému ukládáme informace o **čtenářích** a **pracovnících**. Pracovníky máme v našem IS jen z důvodu, aby bylo možné dohledat, jaký pracovník vydal nebo přijal zapůjčený titul a pro tyto účely o pracovnících uchováváme jen základní informace. **Čtenář** si může výtisk vyzvednout a vrátit. **Pracovník** vydává nebo přijímá výtisky zpět a také může čtenáři rezervace rušit.

2. část - vytvoření tabulek databáze

Základní kámen naší databáze je tabulka `Titul`. Její generalizaci/specializaci jsme rozhodli řešit sloučením tabulek `Číslo časopisu` a `Vydání knihy` do jedné tabulky. Byl přidán atribut `typ`, který rozlišuje titul na **knihu** nebo **časopis**. Atributy `vydavatel` a `nakladatelství` byly sloučeny do jednoho, ale atributy `ISBN` a `ISSN` zůstaly oba, protože některé tituly mohou mít přidělena obě identifikační čísla. Tabulky `Číslo časopisu` a `Téma`, ale i tabulky `Vydání knihy`, `Žánr` a `Autor` jsou propojeny **vazebními tabulkami**.

Tabulka `Výtisk` obsahuje atribut `id_titulu`, který spojuje konkrétní výtisk s daným titulem. Další atribut `stav` pak zachycuje dostupnost výtisku a může nabývat hodnot: `'skladem'`, `'rezervován'`, `'vypůjčen'` nebo `'vyřazen'`.

Tabulka `Rezervace` obsahuje také atribut `stav` a ten může nabývat hodnot: `'platná'`, `'zrušena'` nebo `'ukončena'`. Pokud dojde k vyzvednutí rezervace a je vytvořena výpůjčka, označí se stav rezervace jako `'ukončena'`. V případě zrušení rezervace je stav nastaven na `'zrušena'`. To zda rezervaci zrušil čtenář nebo pracovník knihovny, je možné zjistit pomocí atributu `id_pracovnika_zrusil`. Pokud není tento atribut roven hodnotě `NULL`, došlo ke zrušení rezervace pracovníkem knihovny. V opačném případě rezervaci zrušil čtenář, jehož identifikační číslo je uloženo v atributu `id_ctenare`. Rezervace je spojena s titulem pomocí atributu `id_titulu` a pokud dojde k vyzvednutí rezervace je nastaven i atribut `id_vypujcky`, který rezervaci spojuje s její výpůjčkou.

Záznamy v tabulce `Výpůjčka` jsou propojeny s danou rezervací pomocí atributu `id_rezervace` a s konkrétním výtiskem, který je vypůjčen, atributem `id_vytisku`. Atribut `id_ctenare` říká, kdo si danou výpůjčku/výtisk vypůjčil. Atributy `id_pracovnika_vydal`/`id_pracovnika_prijal` ukládají identifikační číslo pracovníka, který výpůjčku vydal/přijal. Atribut `stav` může nabývat hodnot `'vypůjčeno'` nebo `'vráceno'`.

Tabulky `Čtenář` a `Pracovník` obsahují záznamy o osobách, které se účastní procesu rezervování/vypůjčování knih a časopisů.

3. část - SELECT dotazy

Bylo vytvořeno několik `SELECT` dotazů, které odpovídají zadání a mají simulovat reálné situace při použití tohoto konkrétního schématu databáze. Jednotlivé dotazy jsou popsány komentáři přímo ve zdrojovém kódu.

4. část - vytváření pokročilých objektů databáze

TRIGGERY

První trigger `Trigger_id_pracovnika` byl vytvořen podle zadání a slouží k vytváření nového identifikačního čísla pracovníka. Čísla jsou z posloupnosti začínající od čísla 1 a zvětšující se o 1 s každým nově přidaným pracovníkem do databáze `Pracovnik`.

Další tři triggery se starají o to, aby při změně stavu záznamu z tabulky `Výtisku`, `Rezervace` nebo `Výpůjčka` nebyla narušena integrita databázových dat (např.: vytvoření rezervace neexistujícího titulu apod.).

FUNKCE a PROCEDURY

Funkce `zobraz_statistiky_nevracenych_vytisku()` získá a zobrazí data o nevrácených výpůjčkách. Pro procházení všech výpůjček, které nejsou vráceny, ale čas na jejich vrácení již vypršel, a výpočet požadovaných dat je použit `kurzor`. Také je ošetřena výjimka `ZERO_DIVIDE`, která nastane, pokud je tabulka `Čtenář` prázdná.

Procedura `vytvoreni_rezervace_knihy()` vytvoří novou rezervaci knihy. Nejprve dojde k vytvoření záznamu v tabulce `Rezervace` a následně se upraví záznam v tabulce `Výtisk`, který odpovídá rezervovanému titulu, na stav `rezervován`. Pokud jsou zadány nevalidní údaje, dojde k obnovení stavů všech tabulek před spuštěním funkce a je vrácena chyba.

INDEXY a EXPLAIN PLAN

Pro vyzkoušení práce s indexy a optimalizaci jsme si vytvořili tento dotaz:

```
SELECT

Titul.NAZEV, COUNT(*) as pocet

FROM Titul_autor

JOIN Titul

ON Titul.id_titulu = Titul_autor.id_titulu

JOIN Autor

ON Autor.id_autora = Titul_autor.id_autora

WHERE jmeno='Karel' AND prijmeni='Čapek'

GROUP BY Titul.NAZEV;
```

Jedná se o složitější dotaz obsahující spojení více tabulek, agregační funkci, klauzuli a `GROUP BY`.

Pomocí příkazu `EXPLAIN PLAN` jsme si nechali vypsat plán provedení tohoto dotazu.

```
Explained.
```

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
Plan hash value: 384973338

--------------------------------------------------------------------------------
-------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT      |                |     2 |   206 |     7  (15)|
00:00:01 |
|   1 |  HASH GROUP BY        |                |     2 |   206 |     7  (15)|
00:00:01 |
|   2 |   NESTED LOOPS        |                |     2 |   206 |     6   (0)|
00:00:01 |
|   3 |    MERGE JOIN CARTESIAN|               |    11 |   847 |     6   (0)|
00:00:01 |
|*  4 |     TABLE ACCESS FULL  | AUTOR          |     1 |    37 |     3   (0)|
00:00:01 |
|   5 |     BUFFER SORT        |                |    11 |   440 |     3   (0)|
00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
|   6 |      TABLE ACCESS FULL | TITUL          |    11 |   440 |     3   (0)|
00:00:01 |
|*  7 |    INDEX UNIQUE SCAN   | PK_TITUL_AUTOR |     1 |    26 |     0   (0)|
00:00:01 |
--------------------------------------------------------------------------------
-------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter("AUTOR"."JMENO"='Karel' AND "AUTOR"."PRIJMENI"='Čapek')
   7 - access("TITUL"."ID_TITULU"="TITUL_AUTOR"."ID_TITULU" AND
              "AUTOR"."ID_AUTORA"="TITUL_AUTOR"."ID_AUTORA")


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

25 rows selected.
```

V plánu se nacházeli dvě operace `TABLE ACCESS FULL`, které nejsou příliš efektivní, jedná se totiž o postupné procházení celé tabulky, bez použití indexu. Pro optimalizaci jsme se tedy rozhodli použít indexy, aby nedocházelo k této operaci.

```
CREATE INDEX jmeno_titulu
ON titul (nazev);
CREATE INDEX i_autor
ON autor (prijmeni, jmeno);
```

Indexy jsme vytvořili pro sloupce, které jsou použity pro vyhledávání, abychom dosáhli jeho urychlení.

Po požití těchto indexů jsme použili znovu příkaz EXPLAIN PLAN a získali tak nový plán pro provedení dotazu.

```
Explained.


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------------------------
Plan hash value: 95025610

--------------------------------------------------------------------------------
-------------------------
| Id  | Operation                           | Name             | Rows  | Bytes
| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
-------------------------
|   0 | SELECT STATEMENT                    |                  |     3 |   138
|     6  (17)| 00:00:01 |
|   1 |  HASH GROUP BY                      |                  |     3 |   138
|     6  (17)| 00:00:01 |
|*  2 |   HASH JOIN                         |                  |     3 |   138
|     5   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                     |                  |     3 |    81
|     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID BATCHED| AUTOR            |     1 |    21
|     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN               | I_AUTOR          |     1 |
|     1   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------------------------
|*  6 |     INDEX FULL SCAN                 | PK_TITUL_AUTOR   |     3 |    18
|     1   (0)| 00:00:01 |
|   7 |    VIEW                             | VW_GBF_17        |    11 |   209
|     2   (0)| 00:00:01 |
|   8 |     VIEW                            | index$_join$_002 |    11 |   176
|     2   (0)| 00:00:01 |
|*  9 |      HASH JOIN                      |                  |       |
```

```
|            |          |
|  10 |       INDEX FAST FULL SCAN             | JMENO_TITULU     |    11 |   176
|     1   (0)| 00:00:01 |
|  11 |       INDEX FAST FULL SCAN             | PK_TITUL         |    11 |   176
|     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
-------------------------

Predicate Information (identified by operation id):
---------------------------------------------------


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------
-------------------------
   2 - access("ITEM_1"="TITUL_AUTOR"."ID_TITULU")
   5 - access("AUTOR"."PRIJMENI"='Čapek' AND "AUTOR"."JMENO"='Karel')
   6 - access("AUTOR"."ID_AUTORA"="TITUL_AUTOR"."ID_AUTORA")
       filter("AUTOR"."ID_AUTORA"="TITUL_AUTOR"."ID_AUTORA")
   9 - access(ROWID=ROWID)

Note
-----
   - this is an adaptive plan

31 rows selected.
```

Tento plán je optimálnější, protože zde už nedochází k postupnému procházení všech řádků tabulek, ale vzhledem k malému množství dat v tabulkách, nemůžeme příliš porovnat například časovou náročnost provedeného příkazu.

Přístupová práva a materializovaný pohled

Dále jsme vytvořili materializovaný pohled. Jeho vytvoření výrazně urychluje volání dotazu, protože je jeho výsledek již uložen. Pro pohled jsou nastavená práva pro druhého člena týmu.