

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Databázové systémy Dokumentácia k projektu – 28. Letisko

3. května 2021

Jakub Zaukolec, Slavomír Svorada

1 Úvod

Úlohou projektu bolo vytvoriť návrh a implementovať relačnú databázu na nami zvolenú tému.

2 Zadanie

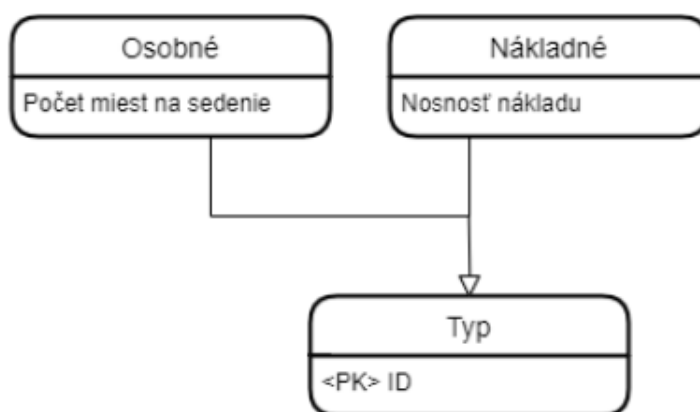
Rozhodli sme sa zvoliť si tému z predmetu IUS, kde sme vytvárali use case diagram a ER diagram, tento projekt sme mali už vtedy dobre ohodnotený, akurát sme riešili drobné úpravy.

Zadanie:

Navrhňte informačný systém pro letiště, který bude schopen evidovat lety. Každému cestujícímu je na základě letenky vydána palubní vstupenka na určitý let a místo v letadle. Různá letadla (i stejného typu) mají různý počet a rozmístění míst. Místo může být v letadle u okýnka, u uličky, či uprostřed a může být v různé třídě (turistická, business, první, ...). U každého letadla je evidován výrobce, typ, datum výroby, datum poslední revize, počet členů posádky. U letu je evidován čas odletu, očekávaná doba letu, terminál a číslo gatu. Každý gate má přiřazen typy letadel, které z něj mohou odlétat. Předpokládejte, že se jedná o větší letiště s více terminály, na každém terminálu je více než jeden gate.

3 1. část projektu

Ako som spomínal vyššie, nebolo potrebné robiť v ER diagrame nejaké výrazné zmeny. Okrem úpravy pár atribútov v tabuľkách a vzťahov sme museli dorobiť generalizáciu. Rozhodli sme sa generalizovať tabuľku Typ. Keďže typ lietadla môže byť ako osobné tak aj nákladné, tak nám to prišlo ako vhodné využitie.



Obrázek 1: Generalizácia

4 2. časť projektu

Ďalšou úlohou bolo vytvoriť tabuľky v jazyku SQL a naplniť ich ukázkovými hodnotami. Pri tomto bode sme vychádzali z predošlého ER diagramu, kde sme vytvárali databázy v tabuľke podľa neho. Náš skript obsahuje nasledovné tabuľky:

- cestujuci
- letenka
- lietadlo
- sedadlo
- let
- gate
- terminal
- clen_posadky
- lietadlo_posadka
- typ
- typ_vyrobca
- vyrobca

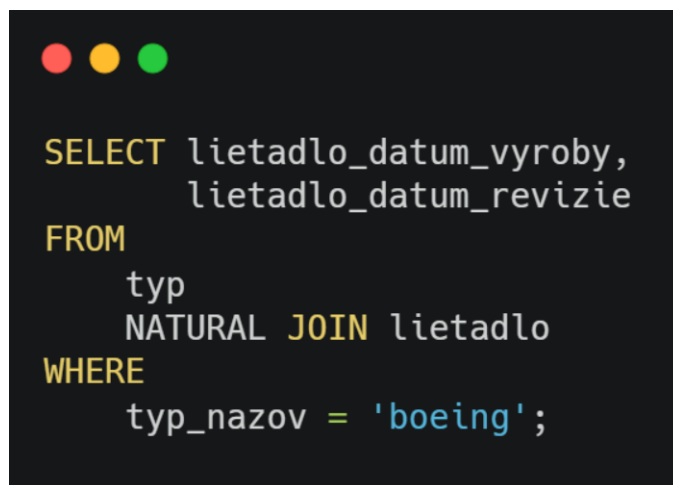
Generalizáciu z predchádzajúceho kroku, sme riešili spôsobom, že sme všetko vložili do jednej tabuľky. Následne po vytvorení tabuliek sme pomocou príkazu `INSERT` vložili do tabuliek dáta.

5 3. časť projektu

Úlohou tretej časti projektu bolo vytvoriť dotazy `SELECT`, ktoré museli spĺňať určité podmienky.

5.1 Príklad select dotazu využívajúceho spojenie dvoch tabuliek

Tento dotaz vyberie všetky lietadla z databázy typu boeing a vypíše ich dátum výroby a dátum revízie.

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal displays an SQL query in a light-colored font. The query is:

```
SELECT lietadlo_datum_vyroby,  
       lietadlo_datum_revizie  
FROM   typ  
       NATURAL JOIN lietadlo  
WHERE  typ_nazov = 'boeing';
```

Obrázek 2: Select 1

5.2 Príklad select dotazu využívajúceho spojenie troch tabuliek

Dotaz vyberie všetkých cestujúcich medzi všetkými letmi, ktorých batožina presiahla 20kg.

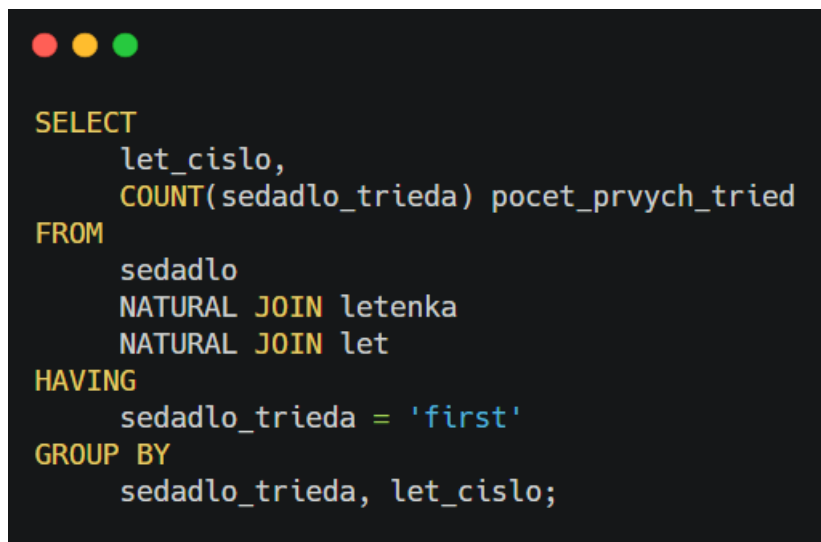


```
SELECT let_cislo,
       cestujuci_meno,
       cestujuci_priezvisko
FROM   cestujuci
       NATURAL JOIN letenka
       NATURAL JOIN let
WHERE  letenka_batozina > 20;
```

Obrázek 3: Select 2

5.3 Príklad select dotazu využívajúceho GROUP BY a agregáciu funkciu

Dotaz vypíše číslo letu a počet ľudí, ktorí sedia v prvej triede.

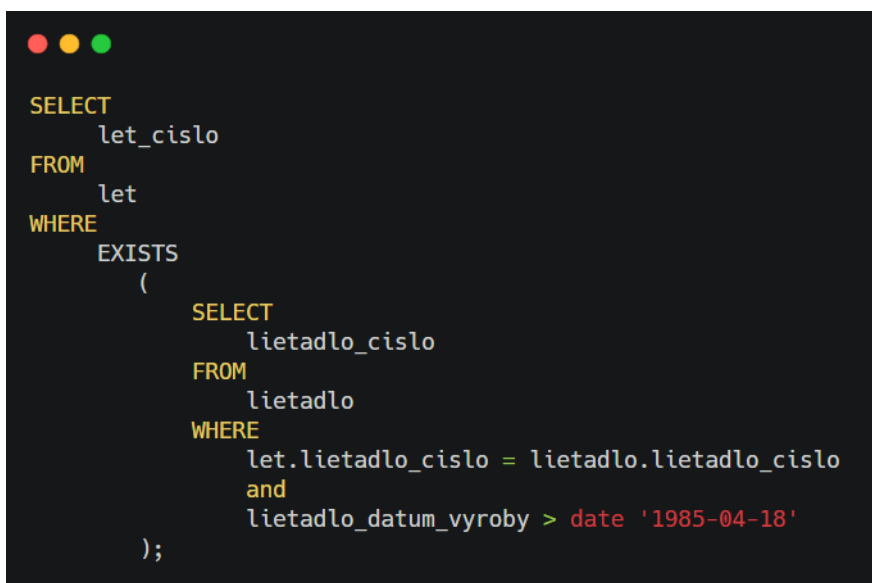


```
SELECT
    let_cislo,
    COUNT(sedadlo_trieda) pocet_prvych_tried
FROM
    sedadlo
    NATURAL JOIN letenka
    NATURAL JOIN let
HAVING
    sedadlo_trieda = 'first'
GROUP BY
    sedadlo_trieda, let_cislo;
```

Obrázek 4: Select 3

5.4 Príklad select dotazu využívajúceho predikát EXISTS

Dotaz vráti čísla letov, ktoré lietajú lietadlom nie starším ako 36 rokov.

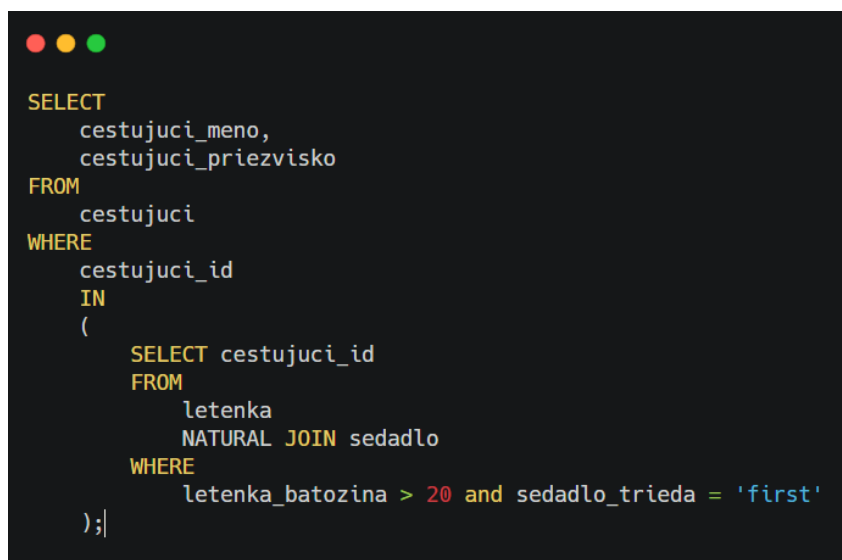
A screenshot of a SQL query in a dark-themed editor. The query uses the EXISTS predicate to find flight numbers from a table named 'let' where the aircraft is not older than 36 years. The subquery selects aircraft numbers from the 'lietadlo' table where the production date is after 1985-04-18.

```
SELECT
    let_cislo
FROM
    let
WHERE
    EXISTS
    (
        SELECT
            lietadlo_cislo
        FROM
            lietadlo
        WHERE
            let.lietadlo_cislo = lietadlo.lietadlo_cislo
            and
            lietadlo_datum_vyroby > date '1985-04-18'
    );
```

Obrázek 5: Select 4

5.5 Príklad select dotazu využívajúceho predikát IN s vnoreným selectom

Dotaz vráti mená cestujúcich, ktorí sedia v prvej triede a majú batožinu ťažšiu ako 20 kg.

A screenshot of a SQL query in a dark-themed editor. The query selects passenger names from the 'cestujuci' table where their ID is in a list of passengers who have a first-class ticket and a baggage weight greater than 20 kg. The subquery joins 'letenka' and 'sedadlo' tables.

```
SELECT
    cestujuci_meno,
    cestujuci_priezvisko
FROM
    cestujuci
WHERE
    cestujuci_id
    IN
    (
        SELECT cestujuci_id
        FROM
            letenka
            NATURAL JOIN sedadlo
        WHERE
            letenka_batozina > 20 and sedadlo_trieda = 'first'
    );
```

Obrázek 6: Select 4

6 4. časť projektu

Poslednou úlohou bolo vytvoriť dva netriviálne databázové trigger, dve netriviálne procedury, použitie `EXPLAIN PLAN` s využitím indexu aj bez, pridelenie práv druhému členovi tímu a vytvorenie materializovaného pohľadu.

6.1 Databázové trigger

Začali sme implementáciou databázových triggerov. Prvý trigger slúži pre automatickú generáciu a inkrementáciu primárneho kľúča v tabuľke `cestujuci`. Tento trigger je realizovaný pomocou sekvencie.

Druhý vytvorený trigger slúži na kontrolu dát, ktoré sú buď pridávané alebo updatované v tabuľke `letenka`. Konkrétne kontroluje hmotnosť batožiny v závislosti na triede v akej cestujúci sedí. Pokiaľ cestujúci cestuje prvou triedou, tak nesmie presiahnuť limit 30 kg, pre cestujúceho v business triede je tento limit 25 kg a pre cestujúceho v economy triede to je 20 kg. Tento trigger je realizovaný ako `BEFORE INSERT OR UPDATE OF letenka_batozina ON letenka`.

6.2 Procedúry

Vytvorili sme 2 procedury s názvami `potrebna_revizia` a `pocet_pasazierov`.

Procedúra `potrebna_revizia` vypíše všetky lietadla, ktoré potrebujú prejsť revíziou. Lietadlo potrebuje prejsť revíziou vtedy ak od poslednej revízie uplynulo viac ako 5 rokov. Pri tejto procedúre sú vytvárané premenné, ktorých typ je odvodený z typu hodnôt v tabuľkách. Taktiež je tu použitý cursor `curs_lietadlo`, ktorý zaisťuje prechod cez všetky tabuľky lietadiel.

Procedúra `pocet_pasazierov` vypíše počet všetkých lietadiel v databáze, vypíše aj počet všetkých osobných lietadiel v databáze a vypočíta percentuálny podiel osobných lietadiel. Taktiež ako aj vyššie sa tu nachádzajú premenné, ktorých typ je odvodený z typov z tabuliek a taktiež sa tu využíva cursor. Je tu navyše ošetrenie výnimky `ZERO_DIVIDE` oproti predchádzajúcej procedúre, keďže tu dochádza k deleniu pri počítaní percentuálneho podielu zastúpenia osobných lietadiel voči všetkým lietadlám.

6.3 EXPLAIN PLAN

Pomocou príkazu `EXPLAIN PLAN` si vieme pozrieť postupnosť operácií pre daný príkaz. My sme `EXPLAIN PLAN` používali na príkaz `SELECT`. Konkrétne sa jednalo o `SELECT`, ktorý vypíše všetky lety v databázi a ku každému letu vypíše počet obsadených prvých tried, economy tried a business tried.

```

SELECT let_cislo,
       COUNT(CASE sedadlo_trieda WHEN 'first' THEN 1 ELSE NULL END) as first,
       COUNT(CASE sedadlo_trieda WHEN 'business' THEN 1 ELSE NULL END) as business,
       COUNT(CASE sedadlo_trieda WHEN 'economy' THEN 1 ELSE NULL END) as economy
FROM
  letenka
  natural join sedadlo
GROUP BY let_cislo;

```

Obrázek 7: EXPLAIN PLAN SELECT

6.3.1 EXPLAIN PLAN bez indexu

Po vykonaní EXPLAIN PLANu vyzeral výstup následovne:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		8	368	7 (15)	00:00:01
1	HASH GROUP BY		8	368	7 (15)	00:00:01
* 2	HASH JOIN		8	368	6 (0)	00:00:01
3	TABLE ACCESS FULL	LETENKA	8	168	3 (0)	00:00:01
4	TABLE ACCESS FULL	SEDADLO	9	225	3 (0)	00:00:01

Obrázek 8: EXPLAIN PLAN bez indexu

Môžeme si všimnúť, že najskôr boli vykonaná operácia SELECT, následne operácia HASH GROUP BY, ktorá zgrupuje položky podľa daného hashovacieho kľúča. Ďalej bola vykonaná operácia JOIN a nakoniec sa pristúpilo k tabuľkám bez použitia indexov TABLE ACCESS FULL.

6.3.2 EXPLAIN PLAN s indexom

Index môže byť vhodné použiť v prípade častého vyhľadávania v určitej tabuľke alebo taktiež keď má naša databáza veľa hodnôt. My v našom SELECTE často krát pristupujeme k tabuľkám letenka a sedadlo. Z toho dôvodu sme sa rozhodli využiť index pre položku sedadlo_trieda z tabuľky sedadlo.

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		8	368	5 (20)	00:00:01
1	HASH GROUP BY		8	368	5 (20)	00:00:01
* 2	HASH JOIN		8	368	4 (0)	00:00:01
3	TABLE ACCESS FULL	LETENKA	8	168	3 (0)	00:00:01
4	INDEX FULL SCAN	EXPLAN_IND	9	225	1 (0)	00:00:01

Obrázek 9: EXPLAIN PLAN bez indexu

Pôvodne sme si mysleli, že budeme vidieť poriadne zlepšenie. Vzhľadom na to, že naše tabuľky obsahujú veľmi málo dát, tak k tomuto nedošlo. Môžeme si taktiež všimnúť, že namiesto posledného pôvodného TABLE FULL ACCESS máme teraz operáciu, ktorá skenuje indexy INDEX FULL SCAN, čo sa odzrkadlilo na Cost a %CPU, prístup na disk máme s indexom lepši ale čo sa týka zaťaženia CPU tak tam sme sa zhoršili o 5 %.

6.4 Prístupové práva

Pomocou príkazu GRANT ALL ON boli druhému členovi tímu pridelené prístupové práva ku všetkým tabuľkám. Takisto mu boli pridelené práva aj na vykonávanie jednotlivých procedúr pomocou príkazu GRANT EXECUTE ON. Po vytvorení materializovaného pohľadu, mu boli pridelené práva aj na tento pohľad.

6.5 Materializovaný pohľad

Vytvorili sme materializovaný pohľad vypis_osob do ktorého bol priradený jednoduchý SELECT, ktorý len vypíše všetky dáta z tabuľky cestujuci. Následne po vytvorení pohľadu je pre demonštráciu zavolaný výpis tohoto pohľadu pomocou SELECTu, následne sa do tabuľky cestujuci pridá nový cestujúci pomocou príkazu INSERT a opäť je zavolaný výpis pohľadu aby sme videli, že aj keď sme do tabuľky cestujuci pridali nový záznam, tak pohľad sa nám nezmenil a zostal stále rovnaký. Pomocou materializovaného pohľadu ukazujeme klientovi len dáta, ktoré chceme ukázať a taktiež slúži pre uloženie často využívaného pohľadu na disk za účelom rýchlejšieho prístupu.

6.6 Záver

Čo sa týka projektu ako takého tak bol pre nás veľmi prínosným, naučil nás pracovať s databázou, s čím sme pred tým obaja nemali žiadne skúsenosti.