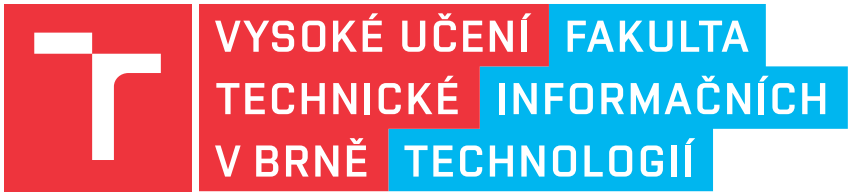


IDS - Databázové systémy

Dokumentace k projektu

53 – Velká éra pirátů

3. května 2021

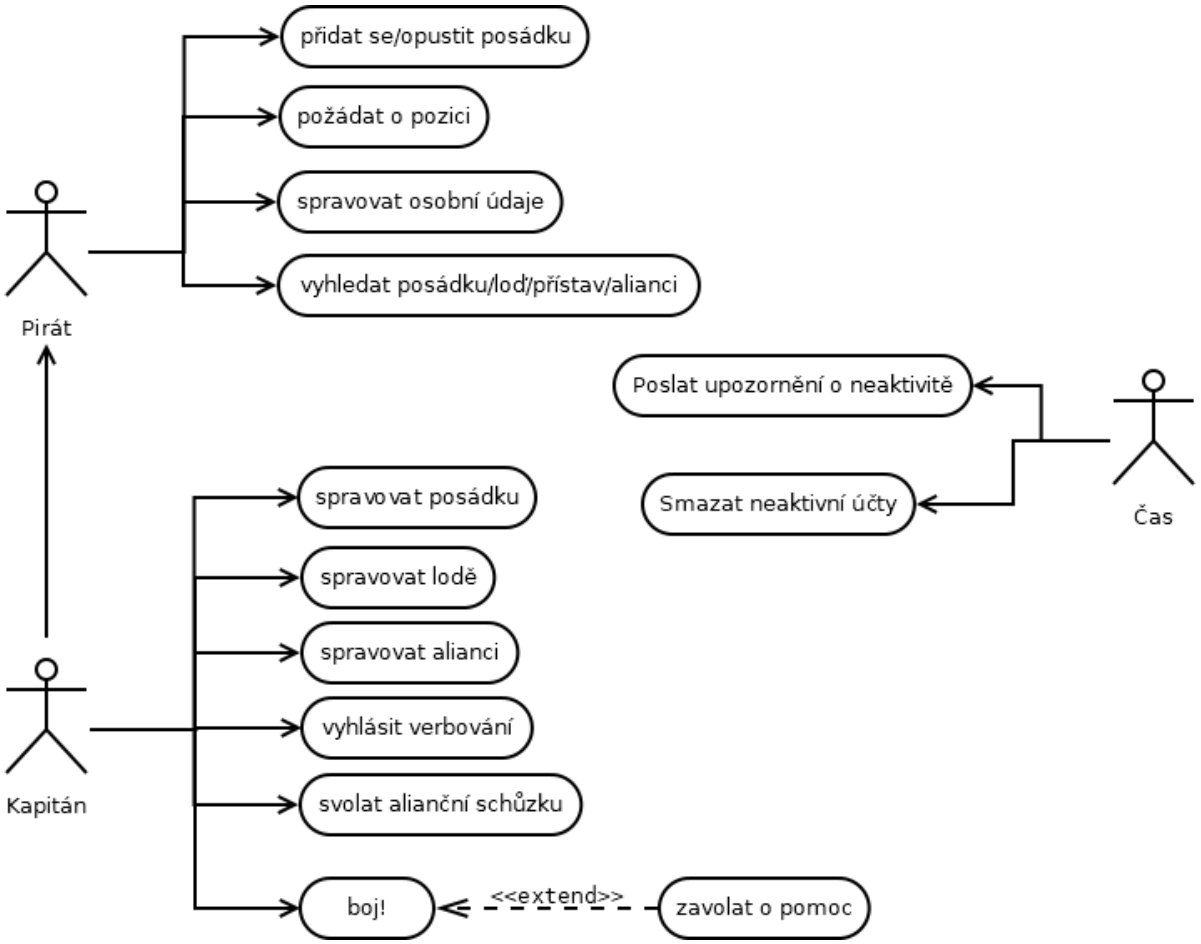
Adrián Bobola xbobol00
Václav Sysel xsysel09

Obsah

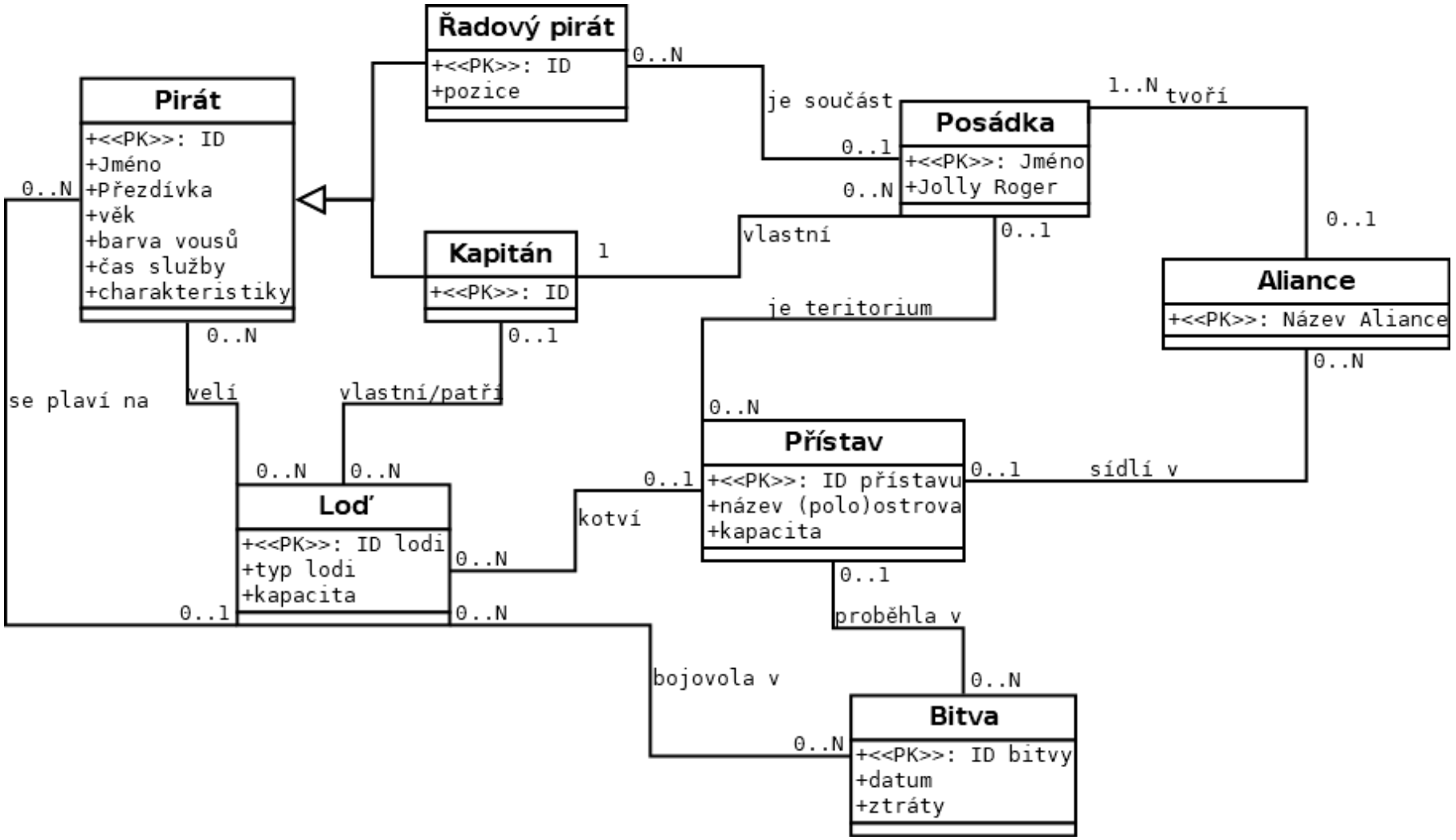
Zadání projektu:

53. Velká éra pirátů (IUS)

Piráti dopili rum a chtěji vytvořit informační systém, který by zefektivnil jejich rabování. Pirátské posádky mají svá unikátní jména, své Jolly Rogery (tzn. vlajky) a sestávají (pochopitelně) z bandy pirátů. Jednotliví piráti, mimo svého jméno (nicméně existuje řada bezejmenných pirátů) a přezdívky (např. Černovous), jsou charakterizováni svou pozicí v posádce (navigator, kuchař, doktor, kormidelník, ...), věkem, barvou vousů, časem stráveným v posádce a seznamem charakteristik (chybějící oko, papoušek na rameni, dřevěná noha, ...). Každá posádka má svého (hlavního) kapitána a vlastní buď jednu loď, nebo celé flotily sestávajících se z více lodí. Každá loď i flotila má pak svého divizního kapitána (může to být i kapitán celé posádky), a je charakterizována typem (karavela, brigantina, ...) a přístavem ve kterém kotví, přičemž kapacita každé lodi pro posádku je omezená. Piráti se můžou plavit maximálně na jedné lodi. U přístavů uchováváme informace o tom, zda se jedná o teritorium specifické pirátské posádky, nebo o neutrální uzemí a název (polo)ostrova, kde se nachází a kapacitu ukotvených lodí. Pirátské posádky dále vytváří vzájemné aliance. Tyto aliance mají dohodnutý jeden přístav (může se jednat o teritorium jedné z posádek), ve kterém probíhají alianční schůzky. Jednotlivé posádky i celé aliance pak mezi sebou mohou bojovat. U těchto bitev evidujeme, zda probíhaly v přístavu (a případně ve kterém) nebo na volném moři a dále počet ztrát (stačí kvantitativně) a konkrétní lodě které se v bitvě zapojily. Systém navíc umožňuje kapitánům posádek vyzvat alianční posádky o pomoc (při chystané bitvě; předpokládejte formu jednoduchého broadcastu). Pro jednoduchost předpokládejte, že POUZE kapitán posádky může manipulovat v IS s údaji o svých posádkách a lodích (tedy divizní a flotilní kapitáni mají v systému stejná práva jako řádoví piráti a jejich speciální privilegia a povinnosti se projevují pouze v reálných událostech, jako jsou bitvy a plavby).



Obrázek 1: Model případů užití (Use Case Diagram)



Obrázek 2: Datový model (ERD)

Popis diagramu

Entita posádka má ako svoj primárny kľúč meno, keďže je unikátne pre danú entitu. Jednotlivé posádky môžu tvoriť maximálne jednu alianciu - z čoho vyplýva kardinalita 0..1, pričom alianciu tvorí jedna až N-posádok. Aliancia zároveň môže sídliť v max jednom prístave. Posádka môže mať svoje teritóriá v prístave (nula až n-prístavov).

Ďalšími entitami sú řádový pirát a kapitán, ktoré vznikli generalizáciou entity pirát, z ktorej preberajú jednotlivé vlastnosti piráta. Entita řadový pirát obsahuje ešte navyše parameter pozíciu piráta. Řadový pirát môže byť súčasťou žiadnej alebo jednej posádky, ktorú vlastní práve jeden kapitán. Kapitán zároveň môže vlastniť žiadnu alebo N-lodí. Z dôvodu uchovania informácii o kapacite jednotlivých lodí tu máme aj vzťah "plaví sa na". Pirát sa môže plaviť na jednej alebo žiadnej lodi a na každej z lodí sa môže plaviť viacero pirátov (prípadne žiadny).

Pozícia divízneho kapitána je v diagrame zaznačená pomocou vzťahu "velí". Pirát nemusí veliť žiadnej lodi a lodi nemusí veliť žiadny pirát. Tento vzťah je v generalizácii, pretože aj kapitán môže byť divíznym kapitánom svojich lodí.

Ďalšou entitou je loď, kde primárny kľúč je unikátne ID lode. Táto entita obsahuje vlastností lode ako typ a kapacitu. Každá loď kotví môže kotviť v max jednom prístave a mohla bojovať v žiadnej až v n-"bitvách", ktoré prebehli v max jednom prístave.

1 Popis implementácie

1.1 Mazanie existujúcich tabuliek

Po spustení programu sa najskôr vykoná sekvencia príkazov `DROP`, vďaka ktorej dôjde k odstráneniu možných existujúcich tabuliek, procedúr a materializovaných pohľadov. Po úspešnom vykonaní príkazov `DROP` môžeme vytvárať nové tabuľky v databáze.

1.2 Vytváranie nových tabuliek

Jednotlivé tabuľky následne vytvárame pomocou príkazu `CREATE TABLE Meno_tabuľky`, kde ďalej deklarujeme požadované názvy a parametre jednotlivých stĺpcov. Každá tabuľka musí obsahovať svoj unikátny primárny kľúč, ktorý je vo väčšine tabuliek generovaný automaticky. V niektorých prípadoch sú aplikované regulárne výrazy, ktoré kontrolujú napríklad dĺžku zadaného reťazca.

1.3 Vkládanie hodnôt

Požadované hodnoty ukladáme do tabuliek pomocou príkazu `INSERT INTO názov_tabuľky`. Údaje prípadne môžeme modifikovať pomocou príkazu `UPDATE názov_tabuľky`.

1.4 Spájanie tabuliek

V tretej časti projektu používame príkazy `SELECT` na spojenie dvoch, prípadne troch tabuliek.
Príkaz `SELECT` nám slúži na "výber" dát z požadovanej databázy. Následne príkazom `FROM` špecifikujeme, ktoré tabuľky požadujeme "vybrať". Príkaz `GROUP BY` nám umožní zoskupiť riadky s rovnakými hodnotami ktoré následne použitím `JOIN` skombinujeme z dvoch, prípadne viacerých tabuliek do jednej.

V niektorých prípadoch využívame aj predikáty `EXISTS` na testovanie existencie záznamu v poddotaze prípadne príkaz `IN` pre možnosť zadávania viacero hodnôt vo `WHERE` klauzule.

1.5 Použitie "triggrov"

Vo štvrtej časti projektu využívame dva `triggre`.
Prvý z nich generuje a následne priradzuje primárny kľúč pri vytvorení nového piráta.
Druhý `trigger` porovná kapacitu lode s kapacitou prístavu, v ktorom daná loď kotví. V prípade, ak je kapacita danej lode väčšia ako samotná kapacita prístavu, nastaví sa parameter `kotví v` na hodnotu `null` a uskutoční sa volanie vytvorenej výnimky tzv.`EXCEPTION` ktorá následne vypíše túto skutočnosť užívateľovi pomocou využitia `DBMS_OUTPUT.PUT_LINE`.

1.6 Použitie procedúr

Vytvorili sme dve procedúry. Procedúra `lode_s_vetsou_kapacitou_nez` vypisuje lode, ktoré majú väčšiu alebo rovnakú kapacitu ako zadaná veľkosť. Na výpise je taktiež vidieť jej percentuálne zastúpenie, v ktorom sme implementovali `výnimku - EXCEPTION` pri delení nulou. Procedúra vytvorí kurzor a skúma jednotlivé lode v databáze, ukladá si štatistiky o počte lodí a lode vyhovujúce podmienke vypíše pomocou `DBMS_OUTPUT.PUT_LINE`. Táto procedúra pracuje s výnimkou `zero_divide`, ktorá značí, že v databáze nie sú žiadne lode k prehľadaniu. V tomto prípade sa užívateľovi táto skutočnosť oznámi.

Druhá procedúra `rozirene_informace_lodi` vypisuje informácie o lodiach, kto ich vlastní, kto na nej velí a kto sa na nej plaví - pretože tieto informácie sú v databáze ťažko vyhľadateľné. Procedúra má kurzor, ktorým prechádza všetky dostupné lode. Pri prechádzaní lodí si vytvára ďalší kurzor, ktorým prechádza tabuľku velenia lodi a tabuľku pirátov. Pri nájdenom velení, majiteľovi či pirátoch plaviacich sa na lodi vypisuje ich ID a prezývku. V opačnom prípade vypíše, že na lodi nikto nie je / nikto jej nevelí. Procedúra je zároveň ošetrená pre prípad, ak by tabuľka lodí prípadne nejaká z iných tabuliek bola prázdna.

1.7 EXPLAIN PLAN

EXPLAIN PLAN začína použitím operácie `SELECT`, ktorá zaisťuje dotaz `STATEMENT`. Ďalej pokračuje operáciou "filter", ktorá odstraňuje riadky nesplňujúce podmienku ("počet bitev≫ 0). Hash `GROUP BY` uloží jednotlivé operácie do tabuľky rozptýlených indexov.

V prípade nevyužitia indexu sa pomocou `HASH JOIN` využíva tabuľka rozptýlených indexov a pripája sa k záznamom druhej tabuľky. K tomu je potrebné dvojnásobné vykonanie operácie `TABLE ACCESS FULL`, ktorá spracuje všetky riadky a stĺpce danej tabuľky.

V prípade využitia indexu sa dvakrát použije `NESTED LOOP`. Prvý `NESTED LOOP` opäť využije `TABLE ACESS FULL` pre spracúvanie celej tabuľky lode. Pri druhej tabuľke sa využijú už vytvorené indexy a použije sa operácia `INDEX RANGE SCAN`. Druhý `NESTED LOOP` využije záznamy s použitím operácie `TABLE ACCESS BY INDEX ROWID`. Vďaka tomu sa už nevykonáva dvojité prehľadávanie, ale získa sa priamo prislúchajúci riadok. Týmto dôjde k výraznému urýchleniu dotazu.

Plán dotazov bez použitia indexu:

PLAN_TABLE_OUTPUT

```
Plan hash value: 915779580

-----------------------------------------------------------------------------------------
| Id  | Operation             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                     |     1 |    68 |     8  (25)| 00:00:01 |
|*  1 |  FILTER               |                     |       |       |            |          |
|   2 |   SORT GROUP BY       |                     |     1 |    68 |     8  (25)| 00:00:01 |
|   3 |    VIEW               | VM_NWVW_1           |     1 |    68 |     7  (15)| 00:00:01 |
|   4 |     HASH GROUP BY     |                     |     1 |    81 |     7  (15)| 00:00:01 |
|*  5 |      HASH JOIN        |                     |     1 |    81 |     6   (0)| 00:00:01 |
|*  6 |       TABLE ACCESS FULL| LOD                |     1 |    55 |     3   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| LOD_BOJOVALA_V_BITVA |     4 |   104 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Plán dotazu s použitím indexu:

```
Plan hash value: 66414639

-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                     |     1 |    68 |     6  (34)| 00:00:01 |
|*  1 |  FILTER                       |                     |       |       |            |          |
|   2 |   SORT GROUP BY               |                     |     1 |    68 |     6  (34)| 00:00:01 |
|   3 |    VIEW                       | VM_NWVW_1           |     1 |    68 |     5  (20)| 00:00:01 |
|   4 |     HASH GROUP BY             |                     |     1 |    81 |     5  (20)| 00:00:01 |
|   5 |      NESTED LOOPS             |                     |     1 |    81 |     4   (0)| 00:00:01 |
|   6 |       NESTED LOOPS            |                     |     1 |    81 |     4   (0)| 00:00:01 |
|*  7 |        TABLE ACCESS FULL      | LOD                 |     1 |    55 |     3   (0)| 00:00:01 |
|*  8 |        INDEX RANGE SCAN       | BITVA_ID_LOD_INDEX  |     1 |       |     0   (0)| 00:00:01 |
|   9 |       TABLE ACCESS BY INDEX ROWID| LOD_BOJOVALA_V_BITVA |     1 |    26 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

1.8 Pohľad MATERIALIZED VIEW

Pre demonštráciu sme vytvorili pohľad `barva_vousu_piratu`. Následne sme do tabuľky pridali ďalšieho piráta. Pri zobrazení pohľadu `barva_vousu_piratu` nie je zobrazený novo-pridaný pirát. Pohľad zostáva nezmenený. Toto ukazuje, že materializovaný pohľad má skopírované dáta vo vlastnej tabuľke. Pri zaslaní príkazu `COMMIT` sa pohľad aktualizuje a nové dáta sú v pohľade prístupné.