Vysoké učení technické v Brně
Fakulta informačních technologií

Databázové systémy

Dokumentace projektu

Galaktické impérium

Aleš Jakšík
xjaksi01

Václav Doleček
xdolec03

Obsah

Zadání 3
Entity Relationship Diagram 3
Grafický model databáze 4
Vytvoření základních objektů 4
Vložení položek 5
Čištění 5
Dotazy SELECT 5
GROUP BY 5
EXIST 5
IN 6
Vytváření pokročilých objektů schématu databáze 6
TRIGGERY 6
PROCEDURE 6
EXPLAIN PLAN 6
MATERIALIZED VIEW 7
Přístupové práva 7

Zadání

Zadání pro vytvoření databáze Galaktické impérium:

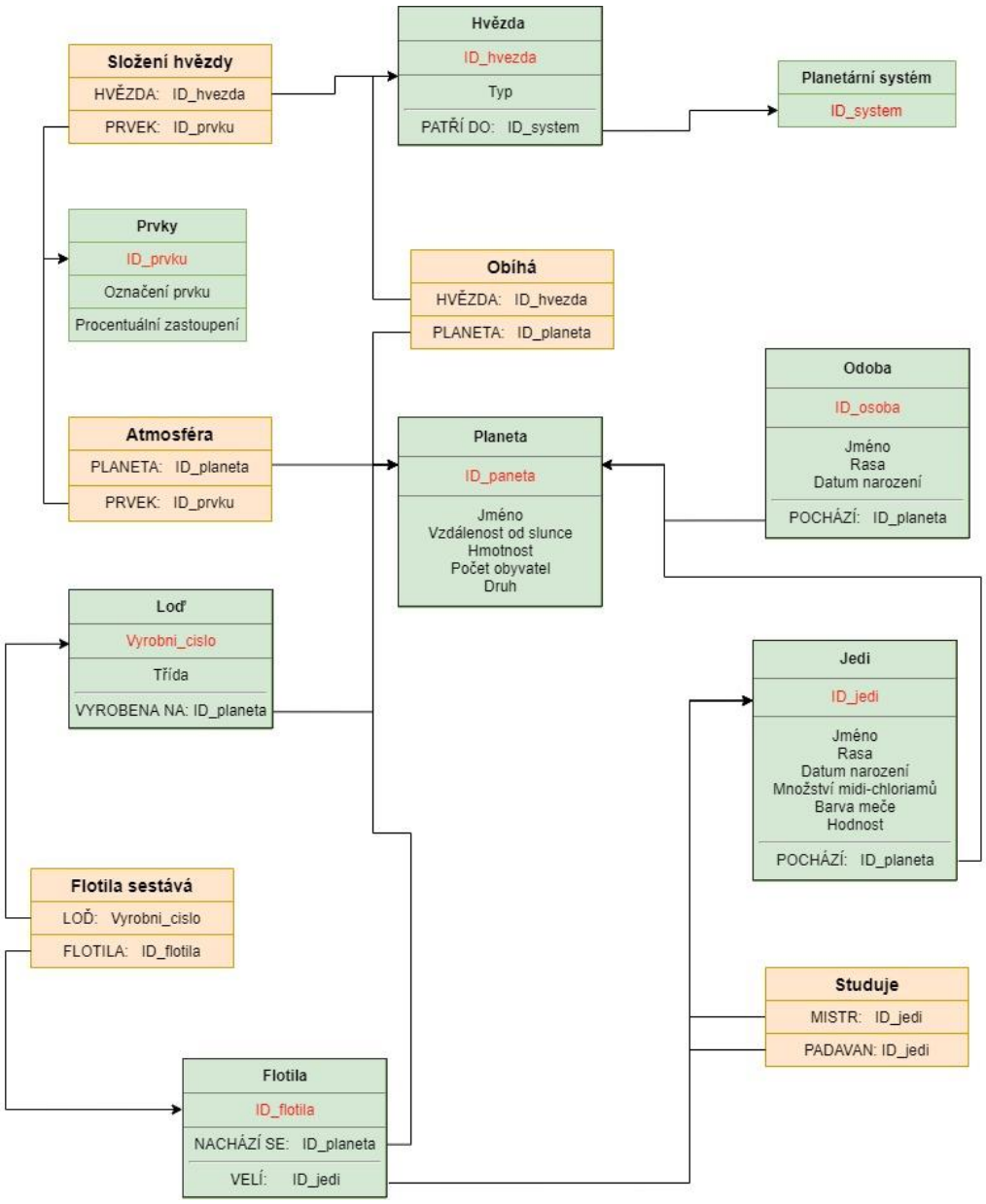
Galaktické impérium je ve fázi svého největšího rozmachu. Kontinuálně obsazuje nové planetární systémy a papírová evidence je již nedostačující. Imperátor Palpatine Vás proto pověřil vytvořením nového elektronického informačního systému pro potřebnou evidenci. Systém musí umět evidovat objevené planetární systémy v galaxii (uvažujte, že Impérium zatím expanzi do jiných galaxií neplánuje). Planetární systémy se nějak jmenují, mají identifikační číslto a mohou mít jednu či více hvězd (jako má např. planetární systém planety Tatooine), tyto mohou být různého typu (červený trpaslík, modrý obr, atp.) a mít různé složení co se týče procentuálního zastoupení různých prvků. Planetární systémy dále mohou obsahovat planety; u každé planety je potřeba vědět jak se jmenuje, její identifikační číslo, v jaké vzdálenosti od slunce/sluncí obíhá (uvažujte, že oběžná dráha planety tvoří kružnici se středem v těžišti hvězdy/hvězd), její hmotnost, počet obyvatel, typ (plynný obr, pouštní planeta, ledová planeta, ...) a složení atmosféry (procentuální zastoupení různých prvků v atmosféře). Impérium dále vlastní řadu flotil vesmírných lodí, které se většinou nachází na orbitu některé planety (nemusí být však na žádné, třeba pokud se právě přesouvají). Každá flotila sestává z několika vesmírných lodí, které mohou být různého typu a mohou být vyrobeny na různých planetách. Ve flotile se může vyskytovat i několik rytířů Jedi, přičemž každý z nich je na nějaké lodi a jeden je velitelem celé flotily (neexistuje flotila bez velitele). Rytíři Jedi mají jméno, rasu, jisté množství midi-chlorianů, domovskou planetu a světelný meč určité barvy. Je třeba si u nich též pamatovat, který rytíř byl padawanem jiného rytíře (ale ne všichni rytíři Jedi mají nutně v systému svého mistra). Imperátor chce mít přes systém možnost přímo zadávat příkazy svým flotilám: přesun mezi planetami, přeskupení, změna velitele, apod. Velitelé flotil pak mohou v systému měnit informace o svých lodích (například stav poškození po opravě, atp.). Systém musí umět rytířům Jedi zasílat automaticky blahopřání k narozeninám---uvažujte, že rytíři Jedi mají narozeniny jen jednou ročně. Pokud selžete, Imperátor Palpatine se s Vámi setká osobně.

Entity Relationship Diagram

Podle zadání jsme vytvořili ERD, který byl nápomocen později při vytváření.

Grafický model databáze

ER Diagram jsme si převedli do této podoby, aby byly zřejmé jednotlivé tabulky a jejich vztahy.



Vytvoření základních objektů

Vytvořili jsme základní tabulky podle předchozího obrázku. Jména tabulek i jednotlivé atributy nesou totožná jména (malá písmena, bez diakritiky), pro pozdější jednodušší práci.
Primární klíče, které nesou v názvu písmena 'ID' jsou generovány automaticky při vytvoření.
Oranžové tabulky zastupující vztahy entit, přejímají primární klíče tabulek, které spojují.
Jediný primární klíč, který není generován automaticky je Vyrobni_cislo v tabulce Loď. Tento klíč je zadáván ručně a má unikátní tvar xxxx-xxxx-xxxx-xxxx, kde x je libovolné číslo či písmeno, tato skutečnost je kontrolována regulárním výrazem, stejně tak jako data narození u Osoba a Jedi.

Vložení položek

Pomocí funkce INSERT byla do tabulek vloženy testovací data.

Čištění

Pro opětovné otestování (naplnění tabulek) musí být stará data smazána, proto na začátku programu je využita funkce DROP TABLE pro každou tabulku.

Dotazy SELECT

Pro jednoduché spojení dvou tabulek jsme vybrali tabulky Osoba a Planeta, výsledek tohoto nám dá seznam obyčejných osob, které pochází z planety 'Khe-atu-miean'.

```
SELECT o.id_osoba, o.jmeno, o.datum_narozeni
FROM osoba o, planeta p
WHERE o.id_planeta = p.id_planeta AND p.jmeno = 'Khe-atu-miean';
```

Druhé spojení dvou tabulek využívá stejného principu, přitom vypíše hvězdy, jež jsou součástí planetárního systému 'Stopařova zhouba'.

Spojení tří tabulek zjišťuje, které planety obíhají kolem jednotlivých hvězd.

```
SELECT h.id_hvezda, p.id_planeta, p.jmeno
FROM hvezda h, planeta p, obiha o
WHERE h.id_hvezda = o.id_hvezda AND o.id_planeta = p.id_planeta;
```

GROUP BY

Prví skupina vypíše planety a počet midi-chloriamu Jediho, který jich má nejvíce a pochází z dané planety, k tomu je využita agregační funkce MAX().

```
SELECT p.id_planeta, p.jmeno, MAX(j.mnozstvi_midichloriamu)
FROM jedi j, planeta p
GROUP BY p.id_planeta, p.jmeno;
```

Druhá skupina opět vypíše planety, ale tentokrát k nim spočítá, kolik na nich bylo vyrobeno lodí, pomocí funkce COUNT(). Zde musí být navíc využit příkaz HAVING jako podmínka přiřazení.

```
SELECT p.id_planeta, p.jmeno, COUNT(l.vyrobni_cislo)
FROM planeta p, lod l
GROUP BY p.id_planeta, l.id_planeta, p.jmeno
HAVING l.id_planeta = p.id_planeta;
```

EXIST

Pomocí této funkce jsme zobrazily pouze ty Jedi, kteří velí nějaké flotile.

```
SELECT j.id_jedi, j.jmeno
FROM jedi j
WHERE EXISTS
  (
    SELECT f.id_flotila
    FROM flotila f
    WHERE f.id_jedi = j.id_jedi
  );
```

IN

Tato funkce vypíše ID Jedi, který je mistr a zároveň má midi-chloriamů více než 400. Ve funkci IN je použita jednoduchá podmínka x > 400.

```
SELECT s.mistr
FROM studuje s
WHERE s.mistr
IN  (
    SELECT j.id_jedi
    FROM jedi j
    WHERE j.mnozstvi_midichloriamu > 400
  );
```

Vytváření pokročilých objektů schématu databáze

TRIGGERY

Prví TRIGGER vytváří ID planety, ID je vygenerováno při vkládání planety do databáze. TRIGGER využívá SEQUENCE a pomocí NEXVAL je získáno ID následující planety.

Druhý TRIGGER kontroluje formát data narození u Osoby. Tentokrát je nejen spuštěn při vkládání Osoby, ale i při aktualizaci této informace. V TRIGGERU jsou použity podmínky, které při nesplněné vyhodí chybu pomocí RAISE_APPLICATION_ERROR(), s číslem chyby a odpovídající chybovou hláškou. Protože je datum vkládáno ve formátu dd.mm.rrrr a jako datový typ VARCHAR, muselo se využít funkce SUBSTR(), pro vyříznutí konkrétní části řetězce a ta se následně porovnala s validními daty přes operátor IN.

Samotné provedení TRIGGERŮ je implementováno po SELECT části.

PROCEDURE

Skript obsahuje dvě procedury:

První 'vypis_planet_tvoru_lodi_pocet' vypíše počet lodí, osob, planet, poměr normálních osob a Jedi, průměrný počet lodí ve flotile. Tyto informace jsou zjištěny pomocí funkce COUNT, nebo základních matematických operací. Kvůli přítomnosti dělení je použit i EXCEPTION pro ošetření možnosti dělení nulou (ZERO_DIVIDE). Zjištěné informace jsou tisknuty na výstup pomocí DBMS_OUTPUT.putline().

Druhá procedura 'vypis_flotil_na_planete' vypočítá kolik flotil se nachází na zadané planetě. Jméno planety je zadáno jako argument při volání této procedury. Kontrola probíhá v cyklu a pro případ, že planeta není nalezena v databázi je přítomen EXCEPTION jako NO_DATA_FOUND, který vypíše náležící hlášku. Všechny výpisy jsou prováděny totožným způsobem jako v předchozí proceduře.

EXPLAIN PLAN

Pro tuto část jsme vtvořili další skupinu (GROUP BY), která obsahuje všechny flotily, které obsahují Loď třídy Křižník. Před tímto byl napsán výraz EXPLAIN PLAN FOR, díky čemuž jsme se po zavolání

mohly podívat na podrobný výpis.

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                |    18 |  3978 |    10  (10)| 00:00:01 |
|*  1 |  FILTER                  |                |       |       |            |          |
|   2 |   HASH GROUP BY          |                |    18 |  3978 |    10  (10)| 00:00:01 |
|   3 |    MERGE JOIN CARTESIAN  |                |    18 |  3978 |     9   (0)| 00:00:01 |
|   4 |     MERGE JOIN CARTESIAN |                |     6 |   858 |     6   (0)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN| SYS_C001594264 |     2 |    26 |     2   (0)| 00:00:01 |
|   6 |      BUFFER SORT         |                |     3 |   390 |     4   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL  | LOD            |     3 |   390 |     2   (0)| 00:00:01 |
|   8 |     BUFFER SORT          |                |     3 |   234 |     8  (13)| 00:00:01 |
|   9 |      INDEX FAST FULL SCAN| SYS_C001594270 |     3 |   234 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Poté jsme vytvořily index pomocí CREATE INDEX pro výrobní číslo a třídu lodě. Proces s vytvářením skupiny s EXPLAIN PLAN a následné volání nám zobrazilo výpis, tentokrát trochu jiný.

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                |    18 |  3978 |     8  (13)| 00:00:01 |
|*  1 |  FILTER                  |                |       |       |            |          |
|   2 |   HASH GROUP BY          |                |    18 |  3978 |     8  (13)| 00:00:01 |
|   3 |    MERGE JOIN CARTESIAN  |                |    18 |  3978 |     7   (0)| 00:00:01 |
|   4 |     MERGE JOIN CARTESIAN |                |     6 |   546 |     4   (0)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN| SYS_C001594264 |     2 |    26 |     2   (0)| 00:00:01 |
|   6 |      BUFFER SORT         |                |     3 |   234 |     2   (0)| 00:00:01 |
|   7 |       INDEX FAST FULL SCAN| SYS_C001594270 |     3 |   234 |     1   (0)| 00:00:01 |
|   8 |     BUFFER SORT          |                |     3 |   390 |     7  (15)| 00:00:01 |
|   9 |      INDEX FAST FULL SCAN| INDX_CISLO     |     3 |   390 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Tentokrát byla tabulka naskenována pomocí indexu, díky čemuž byla celá operace méně náročná na CPU.

MATERIALIZED VIEW

Tento pohled se dívá na systémy a počítá kolik se v každém nachází hvězd. Je proto vytvořena skupina. Proces je proveden, poté se provede aktualizace dat v tabulce a opět se podíváme na výpis skupiny. Z cílového výpisu vidíme, že data v MATERIALIZET VIEW nebyla aktualizována. Tento proces nám totiž lokálně uloží tabulku, která je později zobrazena. Díky tomu je akce sice rychlejší, ale informace nejsou stále aktuální.

Přístupové práva

V poslední části skriptu jsou vytvořena přístupová práva pro druhého člena v týmu pro každou vytvořenou základní tabulku, provedení procedur a materializovaný pohled.