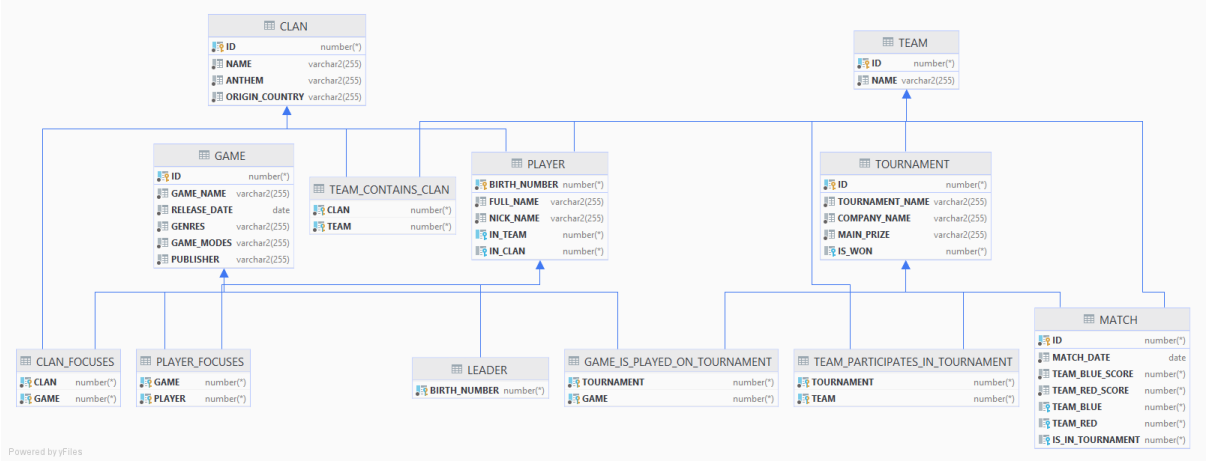
Dokumentace IDS - vytvoření relační databáze

Ladislav Dokoupil
Richard Hrmo

Úvod

Před programováním SQL databáze bylo nejdříve nutné vytvořit ER diagram na relační datový model.
V ERD byly jako první vytvořeny entity Hráč a Hra. Poté entity Klan a Vůdce, při které je využit vztah specializace. Dále byly založeny entity Tým, Turnaj a Zápas. na to dále vznikly vztahy. Vztah mezi Hráčem a Hrou je 1 ... N: 0 ... N, protože na hru se musí zaměřit alespoň 1 hráč, aby byla v databázi, ale hráč může být v databázi, aniž by se zaměřoval na hru. Stejně je to i se vztahem Klan a Hra. Vztah mezi hráčem a klanem je 1 ... N: 0 ... 1, protože hráč může být v jednom nebo v žádném klanu a klan musí mít alespoň jednoho hráče. Stejný vztah je i mezi Hráčem a Týmem. Mezi vůdcem a Klanem je vztah 1: 1, protože jeden klan musí mít jen jednoho vůdce. Klan rovněž může mít 0 ... N týmů, ale tým musí být vytvořen z hráčů alespoň jednoho klanu. Tým se může a nemusí účastnit turnaje, ale v turnaji musí být alespoň 2 týmy. Také Turnaj vyhrává právě jeden tým, ale tým může vyhrát více turnajů. Na turnaji se musí hrát alespoň jedna hra, ale hra nemusela být hrána ještě na žádném turnaji. Zápas se hraje na jednom určitém turnaji, ale turnaj obsahuje více zápasů



Vytvoření tabulek

Nejdříve se postupně vytváří všechny tabulky pomocí příkazu CREATE TABLE. Jsou zde použita také omezení jako například NOT NULL, které ovlivňují fungování databáze tak, že inserty neumožňují nechat daný sloupec prázdný.

```
15  CREATE TABLE clan
16  (
17      id             INT PRIMARY KEY,
18      name           VARCHAR(255) NOT NULL,
19      anthem         VARCHAR(255) NOT NULL,
20      origin_country VARCHAR(255) NOT NULL
21  );
```

Informace jsou vkládány do tabulek pomocí příkazu INSERT. Jsou použity dva způsoby vkládání informací. První je bez určení sloupců do nichž jsou data vkládány, který lze použít v případě, že naplňujeme všechny sloupce tabulky.

```
111  INSERT INTO clan
112  VALUES (1, 'Ninjas In Pyjamas', 'Who We Are', 'Sweden');
```

Druhý je, že se explicitně určí sloupce, do kterých se budou data vkládat.

```
133  INSERT INTO game (game_name, release_date, genres, game_modes, publisher)
134  VALUES ('Fortnite',
135          TO_DATE('2017-06-25', 'YYYY-MM-DD'),
136          'PVE, PVP',
137          'Battle Royale, Party Royale, Creative, Save the World',
138          'Epic Games');
```

Dotazy select

V prvním dotazu jsou zobrazeni všichni hráči klanu Ninjas In Pyjamas, kteří jsou seskupeni podle týmů a hráli na turnaji Katowice Tournament, ze kterého je zobrazena společnost, která organizovala turnaj. Jsou zde použity např. konstrukce join.

```
285  -- show all players of clan Ninjas In Pyjamas grouped by team that participated in Katowice Tournament and show tournaments company name
286  SELECT te.name, full_name, t.company_name
287  FROM PLAYER
288          JOIN TEAM te ON in_team = te.id
289          JOIN TEAM_PARTICIPATES_IN_TOURNAMENT tp ON in_team = tp.TEAM
290          JOIN TOURNAMENT t ON t.ID = tp.TOURNAMENT
291  WHERE in_clan = 1
292    AND tp.tournament = 1
293  GROUP BY te.id, te.name, FULL_NAME, t.company_name;
```

Ve druhém dotazu jsou zobrazeny informace z her, které jsou hrány na turnajích. Použitá zde je klauzule WHERE EXISTS.

```
295  -- show game info from games that are played on tournaments
296  SELECT game_name, release_date, genres, game_modes, publisher
297  FROM GAME
298  WHERE EXISTS
299            (SELECT GAME FROM GAME_IS_PLAYED_ON_TOURNAMENT)
300  ;
```

Ve třetím dotazu jsou zobrazeny počty hráčů v týmech od nejvyššího do nejnižšího. Je použita klauzule ORDER BY a funkce COUNT.

```
302  -- show member count of all teams in descending order
303  SELECT /*+ INDEX(in_team player_index) */ name, COUNT(birth_number)
304  FROM team t
305           LEFT JOIN player p ON t.id = p.in_team
306  GROUP BY name
307  ORDER BY COUNT(birth_number) DESC;
308  select * from table ( DBMS_XPLAN.DISPLAY );
```

Čtvrtý dotaz zobrazuje počty hráčů, kteří se zaměřují na hru, na kterou se žádný jiný hráč nezaměřuje.

```
297  -- show only players that focus on games that no other player does (with given game name)
298  SELECT p.full_name, g.game_name
299  FROM player p
300          JOIN player_focuses pf ON p.birth_number = pf.player
301          JOIN game g ON pf.game = g.id
302  WHERE pf.game NOT IN (SELECT pf1.game
303                        FROM player_focuses pf1
304                        WHERE pf1.player != pf.player
305  );
```

Poslední dotaz zobrazuje počet hráčů, kteří se zaměřují na na každou hru.

```
320  -- show number of players that focus on each game
321  SELECT DISTINCT g.game_name, COUNT(player)
322  FROM PLAYER_FOCUSES
323           RIGHT JOIN GAME g ON g.id = player_focuses.game
324  GROUP BY g.id, g.game_name
325  ORDER BY COUNT(player) DESC;
```

Triggery

Trigger CREATE_CLAN slouží k automatickému generování primárního klíče pro tabulku klan, využívá k tomu sekvenci, která se po každém použití inkrementuje a tím zaručí unikátnost primárního klíče, samotný trigger je spuštěn před každým insertem.

```
113  CREATE SEQUENCE ID_increment;
114  CREATE OR REPLACE TRIGGER create_clan
115      BEFORE INSERT ON clan
116      FOR EACH ROW
117  BEGIN
118      if :NEW.ID is null then
119          :NEW.ID := ID_increment.nextval;
120      end if;
121  END;
```

Druhý trigger byl vytvořen pro zamezení úprav názvu klanu nad tabulkou clan, kde při pokusu o aktualizaci daného sloupce vznikne vyjímka programu.

```
327  CREATE OR REPLACE TRIGGER update_clan
328      BEFORE UPDATE of name
329      ON CLAN
330      BEGIN
331          if UPDATING THEN
332              raise_application_error (-20001, 'Cannot update name');
333          end if;
334      END;
```

Procedury

První procedura vypíše informace o klanu jehož ID je zadáno. Pokud klan neexistuje tak se přejde do věty EXCEPTION a vypíše se chyba.

```
CREATE PROCEDURE print_clan_info(
    p_clan_id NUMBER
)
IS r_clan clan%ROWTYPE;
BEGIN
    SELECT  *
    INTO r_clan
    FROM clan
    WHERE clan.id = p_clan_id;

    dbms_output.put_line('Clan name: '|| r_clan.name || ', clan anthem: ' || r_clan.anthem || ', clan country: ' || r_clan.origin_country);
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('ERROR');
end;
```

Druhá procedura vypíše všechny hráče, kteří se nacházejí v zadaném klanu. Jakož i u první procedury zde zadáváme ID.

```
CREATE PROCEDURE get_players_from_clan(
    p_clan_id NUMBER
)
IS
    c_birth_number player.birth_number%type;
    c_full_name player.full_name%type;
    c_nick_name player.nick_name%type;
    CURSOR c_player IS SELECT birth_number, full_name, nick_name FROM player WHERE player.in_clan = p_clan_id;
BEGIN
    OPEN c_player;
    LOOP
        FETCH c_player into c_birth_number, c_full_name, c_nick_name;
            EXIT WHEN c_player%notfound;
            dbms_output.put_line('Birth number: ' || c_birth_number || ', Full name: ' || c_full_name || ', Nick name: ' || c_nick_name);
    end loop;
    CLOSE c_player;
end;
```

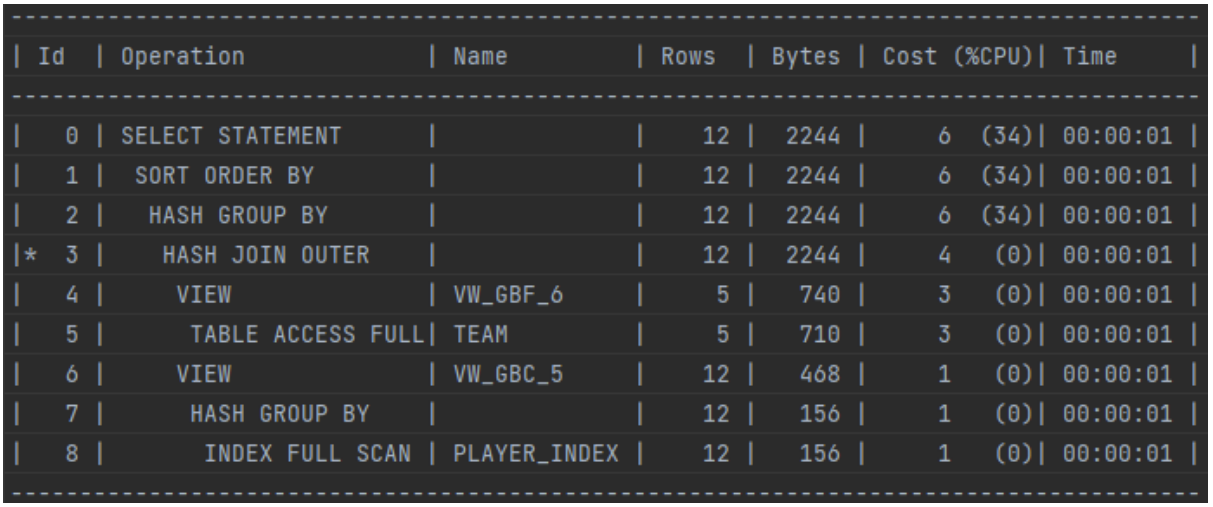
Explain plan a Index

Při nezoptimalizovaném dotazu pro výpis počtu členů každého týmu čte dotaz obě celé tabulky (TABLE ACCESS FULL), poté pro spojení tabulek a agregaci využívá efektivní hashovací funkce, která ze záznamu tabulky vytvoří číselné hashe, které následně porovnává (HASH …).

```
376  drop index player_index;
377  create INDEX player_index ON player(in_team,birth_number);
378
379
380  -- show member count of all teams in descending order
381  EXPLAIN PLAN FOR
382  SELECT /*+ INDEX(in_team player_index) */ name, COUNT(birth_number)
383  FROM team t
384           LEFT JOIN player p ON t.id = p.in_team
385  GROUP BY name
386  ORDER BY COUNT(birth_number) DESC;
387  select * from table ( DBMS_XPLAN.DISPLAY );
```

```
-------------------------------------------------------------------------------
| Id  | Operation            | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |        |    13 |  2184 |     8  (25)| 00:00:01 |
|   1 |  SORT ORDER BY       |        |    13 |  2184 |     8  (25)| 00:00:01 |
|   2 |   HASH GROUP BY      |        |    13 |  2184 |     8  (25)| 00:00:01 |
|*  3 |    HASH JOIN OUTER   |        |    13 |  2184 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| TEAM   |     5 |   710 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| PLAYER |    12 |   312 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Potenciál pro zrychlení je ve vytvoření indexu pro tabulku player, ve které se nachází mnoho sloupců, které nejsou v dotazu zapotřebí. Optimalizace je provedena pomocí vytvoření indexu pro tabulku player, kde jsou uloženy sloupce IN_TEAM a BIRTH_NUMBER, které jsou v dotazu používané. K výraznému urychlení dojde, jelikož stačí přečíst měně dat díky indexu (INDEX FULL SCAN). Pro spojení a agregaci jsou stále využity rychlé hashovací funkce.



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		12	2244	6 (34)	00:00:01
1	SORT ORDER BY		12	2244	6 (34)	00:00:01
2	HASH GROUP BY		12	2244	6 (34)	00:00:01
* 3	HASH JOIN OUTER		12	2244	4 (0)	00:00:01
4	VIEW	VW_GBF_6	5	740	3 (0)	00:00:01
5	TABLE ACCESS FULL	TEAM	5	710	3 (0)	00:00:01
6	VIEW	VW_GBC_5	12	468	1 (0)	00:00:01
7	HASH GROUP BY		12	156	1 (0)	00:00:01
8	INDEX FULL SCAN	PLAYER_INDEX	12	156	1 (0)	00:00:01

Po vytvoření indexu se povedlo snížít cenu dotazu z osmi na šest jen pomocí optimalizace z table access full na index full scan.

View

Pro možnost zobrazení počtu členů každého týmu byl vytvořen materializovaný pohled MEMBERS_COUNT_VIEW, pro který má druhý člen týmu nastaveno právo SELECT. Dále byla udělena všechna práva nad pohledem CLANS_VIEW. Ta zobrazuje sloupce ID a NAME z tabulky CLAN. Pro ukázku možných definic byla navíc vytvořena různá přístupová práva nad tabulkami PLAYER a TEAM pro druhého člena týmu.

```
397  -- view with member count of all teams in descending order
398  DROP MATERIALIZED VIEW members_counts_view;
399  CREATE MATERIALIZED VIEW members_counts_view
400              BUILD IMMEDIATE
401      REFRESH ON DEMAND
402  AS
403  SELECT name as jmeno, COUNT(birth_number) as pocet
404  FROM team t
405           LEFT JOIN player p ON t.id = p.in_team
406  GROUP BY name
407  ORDER BY COUNT(birth_number) DESC;
408
409  DROP MATERIALIZED VIEW clans_view;
410  CREATE MATERIALIZED VIEW clans_view
411              BUILD IMMEDIATE
412      REFRESH ON COMMIT
413  AS
414  SELECT id, name
415  FROM clan;
416
417  --query for xhrmor00
418  SELECT * FROM clans_view;
419  SELECT * FROM members_counts_view;
420
421  GRANT SELECT ON members_counts_view to xhrmor00;
422  GRANT ALL PRIVILEGES ON clans_view to xhrmor00;
423  --REVOKE SELECT ON members_counts_view from xhrmor00;
424
425  GRANT SELECT ON player to xhrmor00;
426  GRANT UPDATE, INSERT on team TO xhrmor00;
```

Z pohledu druhého člena týmu SELECT * zobrazí jen sloupce dostupné z daného pohledu. V případě CLANS_VIEW uvidí pouze sloupce ID a NAME. Pro dotaz nad MEMBERS_COUNT_VIEW se zobrazí sloupce JMENO a POCET.