FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská univerzita v Plzni

Databázové systémy 2

KIV/DB2

Dokumentace semestrální práce

Vojtěch Daniš́ík
A19N0028P
danisik@students.zcu.cz

1. března 2020

Obsah

1 Zadání

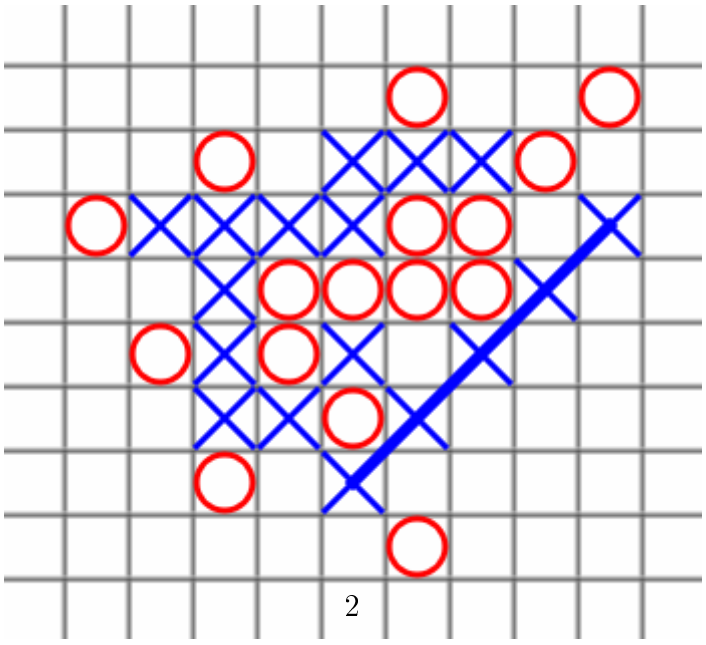
Tradiční piškvorky

Cílem této práce je navrhnout a vytvořit relační databázi pro hraní známé strategické deskové hry *Piškvorky*. Ze hry bude řešena pouze databázová vrstva aplikace, proto se snažte co nejvíce programových rutin uložit do databáze a také zajistěte jejich automatické spouštění při nastalé události.

Tradiční piškvorky jsou hra pro dva hráče, která se hraje na čtverečkovaném papíře. Ve hře se hráči střídají po jednom tahu, ve kterém každý hráč umístí na volné místo na papíře svoji značku. Obvykle hráči používají symboly kolečko (`O`) a křížek (`X`). Vítězí ten hráč, kterému se podaří sestavit nepřerušovanou řadu alespoň pěti svých značek v libovolném směru (vodorovně, svisle, uhlopříčně). Pokud není možné umístit novou značku, hra končí remízou.

Hra probíhá na papíře, který bude mít definovanou velikost. Nejmenší rozměr papíru, na kterém bude možné hrát, může mít rozměry 5x5 čtverečků, největší rozměr papíru nesmí překročit velikost 20x20 čtverečků. Také bude možné definovat na kolik značek se bude hrát. Velikost vítězné řady by měla být z intervalu 5 až 15. Je třeba si dát pozor na skutečnost, aby se vítězná řada vešla na definovaný papír.

Také se bude měřit herní čas. Hra začíná umístěním první značky začínajícího hráče. V ten samý okamžik se začíná měřit herní čas druhého hráče a to až do doby, kdy umístí svoji značku. Pak se začíná měřit čas začínajícího hráče. Jakmile začínající hráč umístí svoji druhou značku, pozastaví se měření jeho herního času a opětovně se spustí měření času druhého hráče. A tak pořád dokola, až hra skončí výhrou jednoho z hráčů nebo remízou.



Zdroj: `https://cs.wikipedia.org/wiki/Piškvorky`

V relační databázi budou evidována data v těchto tabulkách:

`OMEZENI` Tato tabulka slouží jako parametry programu. Obsahuje informace, jak veliký či malý může být čtverečkovaný papír, na kterém se bude hrát a také jak dlouhá či krátká může být minimálně řada symbolů vítězného hráče.

`STAV` Číselník ukazující, v jakém stavu se může nacházet hra. Stavy mohou být tyto: *rozehraná, vítězství začínajícího hráče, prohra začínajícího hráče* nebo *remíza*.

`HRAC` Každý hráč, který bude chtít hrát, musí být zaregistrován. Výhodou bude možnost sledovat jeho statistiky hraní, tj. počet vítězství, proher či remíz, a zda začínal nebo hrál jako druhý.

`HRA` Každá hra se hraje na novém čistém papíře o dané velikosti na požadovaný počet vítězných symbolů. Hru hrají dva různí hráči, kde jeden z nich umisťuje kolečka, ten druhý křížky a jeden z nich celou hru začíná. Tabulka bude také obsahovat, v jaké stavu je hra a také herní časy obou hráčů.

`TAH` Umístění své značky hráčem, který je na řadě v rozehrané hře. Ke každému tahu se bude automaticky ukládat časová značka, která dočasně nebo trvale zastavuje měření času právě hrajícímu hráči.

Z uložených dat v databázi vytvořte databázové pohledy, které nabídnou tato data:

`PAPIR` Zobrazení čtverečkovaného papíru obsahující všechny dosud provedené tahy právě probíhající hry. Každý řádek papíru bude zobrazen voláním funkce `RADEK_PAPIRU`.

`VYHRY_ZACINAJICI` Hry, ve kterých zvítězil začínající hráč. Obsahuje parametry hry (rozměry papíru, požadovaná velikost vítězné řady), jména hráčů, kdo začínal (a zvítězil), kdo používal jaké značky, jak dlouho celá hra trvala v sekundách, kolik bylo zahráno tahů.

`PROHRY_ZACINAJICI` Hry, ve kterých prohrál začínající hráč. Obsahuje parametry hry (rozměry papíru, požadovaná velikost vítězné řady), jména hráčů, kdo začínal (a prohrál), kdo používal jaké značky, jak dlouho celá hra trvala v sekundách, kolik bylo zahráno tahů.

`REMIZY` Hry, které dospěly do remízy. Obsahuje parametry hry (rozměry papíru, požadovaná velikost vítězné řady), jména hráčů, kdo začínal, kdo používal jaké značky, jak dlouho celá hra trvala v sekundách.

V databázových pohledech `VYHRY_ZACINAJICI`, `PROHRY_ZACINAJICI` a `REMIZY` získejte dobu hraní hry jako součet herních dob obou hráčů.

V databázi budou uloženy a používány tyto funkce (s parametry):

`SPATNY_PARAMETR` Podle návratové hodnoty funkce poznáme, že je:

- `0` - vše v pořádku.
- `1` - příliš malý počet řádků na papíru (menší než 5).
- `2` - příliš velký počet řádků na papíru (větší než 20).
- `3` - příliš malý počet sloupců na papíru (menší než 5).
- `4` - příliš velký počet sloupců na papíru (větší než 20).
- `5` - příliš malý počet znaků ve vítězné řadě (menší než 5).
- `6` - příliš velký počet znaků ve vítězné řadě (větší než 15).
- `7` - vítězná řada delší, než šířka papíru.
- `8` - vítězná řada delší, než výška papíru.

`RADEK_PAPIRU` Vrátí řetězec, který odpovídá konkrétnímu řádku papíru dané hry. Pro výpis zvolte v řetězci tyto symboly:

- `X` - značka křížek jednoho hráče dané hry.
- `O` - značka kolečko druhého hráče hry.
- `mezera` - symbol označující volné políčko na papíře.

`HERNI_CAS` Vrátí číslo určující, kolik sekund hrál daný hráč danou hru, tj. sečte rozdíly časových značek, kdy hrál daný hráč a kdy hrál před ním druhý hráč.

`REMIZA` Vrátí hodnotu `TRUE`, pokud daná hra dospěla do remízového stavu, tj. není možné udělat další tah. Jinak vrací hodnotu `FALSE`.

`VYHRA` Vrátí hodnotu `TRUE`, pokud právě hrající hráč v dané hře vyhrál, tj. svým posledním tahem docílil požadované minimální délky vítězné řady svých značek. Jinak vrací hodnotu `FALSE`.

V databázi budou uloženy a jako těla triggerů používány tyto procedury (s parametry):

`ZABRAN_HRE` Zabrání vytvoření nové hry, pokud je nastaven špatně libovolný parametr hry.

`ZABRAN_TAHU` Zabrání tahu, který není možno udělat.

`KONEC_HRY` Spočítání herních časů hráčů právě dokončené hry.

`STATISTIKY` Aktualizace statistických údajů hráčů, kteří dohráli danou hru.

O automatické činnosti v databázi se postarají triggery o:

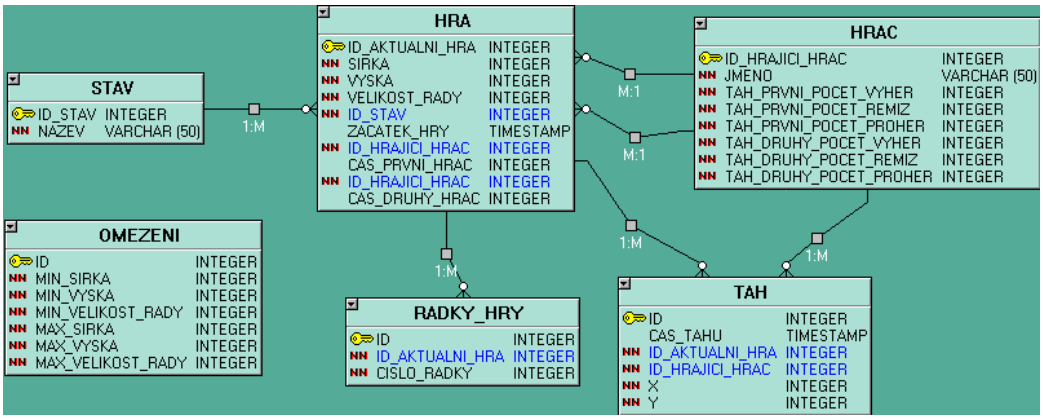
- hlídání parametrů nové hry (volání funkce `SPATNY_PARAMETR`),
- hlídání hráčů, aby se ve hře pravidelně střídali po jednom tahu,
- hlídání hráče, aby nepokládal svoji značku na již obsazené místo,
- hlídání hráče, aby nemohl realizovat tah ve hře, ve které nehraje,
- hlídání hráče, aby nemohl realizovat tah ve hře, která již skončila,
- hlídání a aktualizace stavu hry, tj. zda nedospěla do remízy nebo vítězství jednoho z hráčů (volání funkcí `REMIZA` a `VYHRA`),
- spočítání herní doby hráčů, kteří dohráli aktuální hru (volání funkce `HERNI_CAS`),
- aktualizace statistických údajů hráčů, kteří dohráli aktuální hru.

Konfigurace, spuštění a průběh hry

1. Jednorázová konfigurace databáze:
    a. Naplnění tabulky `OMEZENI` hraničními hodnotami parametrů nové hry.
    b. Naplnění tabulky `STAV` daty odpovídající různým stavům hry.
2. Registrace hráčů je provedena vkládáním nových záznamů do tabulky `HRAC`.
3. Hra je zahájena úspěšným vložením nového záznamu do tabulky `HRA`. Neúspěšné vložení znamená, že některé parametry hry jsou mimo hraniční hodnoty definované v tabulce `OMEZENI`.
4. Že hra běží, ověříme zobrazením papíru, tj. vypsáním řádků prostřednictvím databázového pohledu `PAPIR` pro aktuálně rozehranou hru.
5. Každý tah ve hře je proveden vložením nového záznamu do tabulky `TAH`. Pokud ke vložení nedošlo, byla aktivována některá z výše uvedených kontrol.
6. Po úspěšném tahu je vhodné si vždy zobrazit aktuální situaci na papíře prostřednictvím databázového pohledu `PAPIR`.
7. Pokud hra neskončila, pokračuj bodem 5, jinak bodem 8 (výhra) nebo bodem 9 (remíza).
8. Hra skončila vítězstvím jednoho z hráčů. To je vhodné ověřit voláním pohledu `VYHRY_ZACINAJICI` nebo `PROHRY_ZACINAJICI` a také se podívat na aktualizované statistiky hrajících hráčů v tabulce `HRAC`. Jdi na bod 10.
9. Hra skončila remízou. To je vhodné ověřit voláním pohledu `REMIZY` a také se podívat na aktualizované statistiky hrajících hráčů v tabulce `HRAC`.
10. Novou hru zahájíme bodem 3, s případnou registrací nového hráče bodem 2.

2 Datová analýza

Datový model semestrální práce obsahuje celkem 6 tabulek. Všechny tabulky mají definovaný primární klíč s názvem ID. Integrita databáze je zajištěna. Diagram datového modelu je zobrazen na obrázku 1.



Obrázek 1: ER diagram datového modelu.

3 Funkční analýza

3.1 Vybrané pohledy

3.1.1 Pohled PAPIR

Pohled **PAPIR** zobrazí všechny řádky na papíru i se všemi značkami v dané hře. Začínající hráč má vždy přiřazenou značku "**O**"a druhý hráč má přiřazenou značku "**X**". Řádky jsou od sebe odděleny značkou "|". Funkční SQL kód viz kód 1.

```
CREATE OR REPLACE VIEW PAPIR(ID_HRA) AS
SELECT
        ID_HRA,
        CISLO_RADKY,
        RADEK_PAPIRU(ID_HRA, CISLO_RADKY)
FROM RADKY_HRY;
```

Listing 1: SQL kód pohledu **PAPIR**.

3.1.2 Pohled REMIZY

Pohled **REMIZY** zobrazí všechny ukončené hry, které skončili remízou. Každý řádek obsahuje:

- ID hry
- šířka papíru
- výška papíru
- velikost vítězné řady
- celková doba hraní hry
- jméno prvního hráče
- jméno druhého hráče

Při zobrazování se používá tabulka *HRA*, ke které se napojí tabulka *HRAC*. Funkční SQL kód viz kód 2.

```
CREATE OR REPLACE VIEW REMIZY AS
SELECT
        HRA.ID AS ID_HRY,
        HRA.SIRKA,
        HRA.VYSKA,
        HRA.VELIKOST_RADY,
        CAS_PRVNI_HRAC + CAS_DRUHY_HRAC
        AS DOBA_HRANI_HRY,
        H1.JMENO AS ZACINAJICI_HRAC,
        H2.JMENO AS DRUHY_HRAC
        FROM HRA
                LEFT JOIN HRAC AS H1
                ON H1.ID = HRA.ID_PRVNI_HRAC
                LEFT JOIN HRAC AS H2
                ON H2.ID = HRA.ID_DRUHY_HRAC
        WHERE REMIZA(HRA.ID);
```

Listing 2: SQL kód pohledu **REMIZY**.

3.2 Vybraná funkce

3.2.1 Funkce REMIZA

Funkce **REMIZA** kontroluje stav hry. Pokud je stav hry roven 4, je vráceno *TRUE*. V ostatních případech je vrácená hodnota *FALSE*. Funkční SQL kód viz kód 3.

```
CREATE OR REPLACE FUNCTION REMIZA(ID_HRA INTEGER)
RETURNS BOOLEAN AS $$
        DECLARE
                I_HRA HRA;
        BEGIN
                SELECT * INTO I_HRA FROM HRA
                WHERE ID = ID_HRA;
                IF I_HRA.ID_STAV = 4 THEN
                        RETURN TRUE;
                END IF;

                RETURN FALSE;
        END;
        $$ LANGUAGE plpgsql;
```

Listing 3: SQL kód funkce **REMIZA**.

3.3 Vybraná procedura

3.3.1 Procedura KONEC_HRY

Procedura **KONEC_HRY** kontroluje stav hry. Procedura v případě, že stav hry je roven 1, tak vypočítá herní dobu obou hráčů a následně tyto hodnoty vloží do tabulky **HRA**. Funkční SQL kód viz kód 4.

```
CREATE OR REPLACE PROCEDURE KONEC_HRY (ID_HRA INTEGER)
AS $$
        DECLARE
                I_HRA HRA;
        BEGIN
                SELECT * INTO I_HRA FROM HRA WHERE
                ID = ID_HRA;

                IF I_HRA.ID_STAV = 1 THEN
                        RETURN;
                END IF;
                UPDATE HRA SET CAS_PRVNI_HRAC =
                HERNI_CAS(I_HRA.ID, I_HRA.ID_PRVNI_HRAC)
                , CAS_DRUHY_HRAC =
                HERNI_CAS(I_HRA.ID, I_HRA.ID_DRUHY_HRAC)
                WHERE ID = ID_HRA;
        END;
        $$ LANGUAGE plpgsql;
```

Listing 4: SQL kód procedury **KONEC_HRY**.

3.4 Vybraný trigger

3.4.1 Trigger HRA_DOHRANA

Trigger **HRA_DOHRANA** se spustí vždy po úpravě stavu hry. Trigger poté volá funkce **VYHRA** a 3.2.1. Pokud alespoň jedna z těchto funkcí vrátí hodnotu *TRUE*, tak se provede ukončení hry voláním procedur 3.3.1 a **STATISTIKY**. Funkční SQL kód viz kód 5.

```
CREATE OR REPLACE FUNCTION VYTVOR_TRIGGER_HRA_DOHRANA()
RETURNS TRIGGER AS $VYTVOR_TRIGGER_HRA_DOHRANA$
        DECLARE
        BEGIN
                IF VYHRA(NEW.ID) = TRUE
                OR REMIZA(NEW.ID) = TRUE THEN
                        CALL KONEC_HRY(NEW.ID);
                        CALL STATISTIKY(NEW.ID);
                END IF;

                RETURN NEW;
        END;
        $VYTVOR_TRIGGER_HRA_DOHRANA$ LANGUAGE plpgsql;

CREATE TRIGGER HRA_DOHRANA AFTER UPDATE OF ID_STAV
ON HRA FOR EACH ROW EXECUTE PROCEDURE
VYTVOR_TRIGGER_HRA_DOHRANA();
```

Listing 5: SQL kód triggeru **HRA_DOHRANA**.

4 Testovací scénář

Pro zahájení hry je potřeba vložit záznam do tabulky **HRA**. Příklad vytvoření nové hry viz Listing 6.

```
INSERT INTO HRA(SIRKA, VYSKA, VELIKOST_RADY,
ID_PRVNI_HRAC, ID_DRUHY_HRAC) VALUES (20, 20, 5, 1, 2);
```

Listing 6: Příklad vložení záznamu do tabulky **HRA**.

Jakmile se hra vytvoří, tak může začínající hráč provést svůj první tah (pokud se bude snažit vložit tah druhý hráč ještě předtím, než vloží tah začínající hráč, tak mu to trigger **ZKONTROLUJ_TAH** zakáže a vyhodí vyjímku). Příklad provedení tahu viz Listing. 7.

```
INSERT INTO TAH(ID_AKTUALNI_HRA, ID_HRAJICI_HRAC, X, Y)
VALUES (2, 1, 1, 1);
```

Listing 7: Příklad vložení záznamu do tabulky **TAH**.

Pro zobrazení celého papíru se zavolá pohled **PAPIR** pomocí příkazu *SELECT* se zadaným ID hry, pro kterou chceme zobrazit papír. Pro zobrazení

her, ve kterých vyhrál nebo prohrál začínající hráč, se zavolají pohledy **VYHRY_ZACINAJICI, PROHRY_ZACINAJICI** pomocí příkazu *SELECT*. Pro zobrazení her, které skončili remízou, se zavolá pohled **REMIZY** pomocí příkazu *SELECT*.

5 Závěr

Semestrální práce splňuje zadanou funkcionalitu. Po vložení nového záznamu do tabulky **TAH** jsou triggery schopny rozpoznat, zda daný hráč vytvořil řadu po sobě jdoucích značek o dané velikosti a následně na to zareagovat ukončením hry a aktualizováním statistik. Pro potřeby výpisu daného řádku hry bylo potřeba vytvořit navíc tabulku **RADKY_HRY**, kde každý záznam reprezentuje pár ID hry - Číslo řádky. V semestrální práci byly implementovány všechny požadované funkce, procedury, pohledy a tabulky, které jsou používány triggery. Navíc bylo vytvořeno 5 testovacích scénářů, pomocí kterých byla otestována funkčnost kontroly stavu hry.