FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Hledání min

KIV/DB2 – semestrální práce

Plzeň, 30.5.2018

David Pivovar

pivovar@students.zcu.cz

Obsah

Zadání

Cílem této práce je navrhnout a vytvořit relační databázi pro hraní známé počítačové hry *Hledání min*. Protože bude řešena pouze databázová vrstva aplikace, snažte se co nejvíce programových rutin uložit do databáze a také zajistěte jejich automatickou aktivaci při nastalé události.

Herní oblast má zpravidla tvar obdélníku, ve kterém se nachází několik min. Velikost oblasti a počet min v ní definuje obtížnost hry. Hráč si může vybrat jednu ze tří předdefinovaných obtížností nebo si může definovat obtížnost vlastní. Úkolem hráče je odkrýt všechna pole oblasti, která nejsou zaminována. Hráči se bude od začátku hry měřit čas, aby bylo možné dosažené výsledky porovnávat. Po odkrytí libovolného pole (hráčem nebo databází) může nastat jedna z těchto událostí:

- Hráč šlápl na minu. Hra končí neúspěchem a výsledek se zaznamená do databáze.
- Bylo odkryto poslední pole, na kterém není mina. Hra končí úspěchem, protože zbylá neodkrytá pole obsahují miny. Také v tomto případě se výsledek uloží do databáze.
- Bylo odkryto pole, které je volné a nesousedí s žádným zaminovaným polem. V tomto případě databáze automaticky odkryje všechna sousední pole – ty mají společný min. jeden vrchol.

Pro snazší hraní si může hráč označovat ta pole, o kterých si myslí, že jsou zaminovaná. K tomuto rozhodnutí mu pomohou čísla již odkrytých polí, která určují, s kolika zaminovanými poli toto pole sousedí. Takto označené pole nelze odkrýt, ale toto označení lze kdykoliv zrušit.

V relační databázi budou evidována data v těchto tabulkách:

OBTIZNOST - Každá hra musí mít definovanou obtížnost. Buď bude vybrána předdefinovaná, nebo si hráč definuje obtížnost vlastní. Hodnoty parametrů vlastní obtížnosti se ukládají do tabulky OBLAST. Podle originální hry jsou předdefinované obtížnosti nastaveny takto:

- Začátečník: 9 řádků x 9 sloupců, 10 min
- Pokročilý: 16 řádků x 16 sloupců, 40 min
- Expert: 16 řádků x 30 sloupců, 99 min

OMEZENI - Každá vlastní obtížnost musí splňovat jistá omezení. Např. počet řádků či sloupců nesmí být menší než 9 a větší než 100. Také je vhodné pohlídat, aby počet rozmístěných min v zaminované oblasti nebyl příliš velký, např. nepřekročil 40 procent její velikosti.

OBLAST - Každá zaminovaná oblast je vytvořená podle vlastní obtížnosti a obsahuje její hodnoty. Hráč má za úkol oblast od min vyčistit.

POLE - Elementární část zaminované oblasti definovaná svými souřadnicemi, která může nést minu nebo informaci, s kolika zaminovanými poli sousedí.

MINA - Hráčem označovaná pole, o kterých si myslí, že jsou zaminovaná.

TAH - Hráčem odkrývaná pole v zaminované oblasti. Ke každému tahu se bude automaticky ukládat časová značka, kdy byl tah vykonán.

STAV - Číselník obsahující, v jakých stavech se hra může vyskytovat. Stavy mohou být tyto: rozehraná, úspěšně ukončená, neúspěšně ukončená.

HRA - Průběžně aktualizované informace o probíhající hře. Obsahuje časové značky prvního a naposledy provedeného tahu, počet označených min a stav hry.

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

CHYBNE_MINY - Seznam polí v zaminované oblasti, které byly chybně označené jako zaminované. Nabízí data pro všechny oblasti.

VITEZOVE - Výsledková tabulka her, které byly úspěšně dokončené. Měla by ukazovat parametry obtížnosti (rozměry oblasti a počet min) dané hry a také dobu hraní hry (rozdíl časových značek posledního a prvního tahu).

PORAZENI - Výsledková tabulka her, které byly neúspěšně dokončené. Měla by navíc (oproti pohledu VITEZOVE) ukazovat, kolik min bylo správně odhaleno.

OBLAST_TISK - Zobrazení celé zaminované oblasti včetně odkrytých polí a (hráčem) označených min. Každý řádek oblasti bude zobrazen voláním funkce RADEK_OBLASTI.

V databázi budou uloženy a používány tyto procedury:

ZAMINUJ_OBLAST - Položení min na (náhodná) místa v definované oblasti. Počet zaminovaných polí je uloženo v tabulce OBLAST. Do tabulky POLE ukládá hodnotu -1.

SPOCITEJ_OBLAST - Pro každé nezaminované pole v oblasti spočítá, s kolika zaminovanými poli sousedí. Do tabulky POLE ukládá hodnoty z intervalu 0 až 8.

ODKRYJ_POLE - Rekurzivní procedura, která pro právě odkryté pole, které nesousedí s žádným zaminovaným polem, odkryje všechna jeho neodkrytá sousední pole.

OZNAC_MINY - Po úspěšném dohrání hry budou dosud neodkrytá pole označená jako zaminovaná, tj. vloží odpovídající záznamy do tabulky MINA.

V databázi budou uloženy a používány tyto funkce:

SPATNY_PARAMETR - Oznámí, zda hodnota parametru vlastní obtížnosti porušila definovaná omezení.

RADEK_OBLASTI - Vrátí řetězec znaků ukazující aktuální podobu daného řádku zaminované oblasti.

ODKRYTA_MINA - Oznámí, že právě odkryté pole skrývá minu, což znamená neúspěšný konec hry.

MNOHO_MIN - Nelze označit více zaminovaných polí, než kolik min je v oblasti.

VYHRA - Počet neodkrytých polí se rovná počtu min, které se v oblasti nachází. Pokud ano, hra končí úspěchem.

O automatické činnosti v databázi se postarají triggery hlavně o:

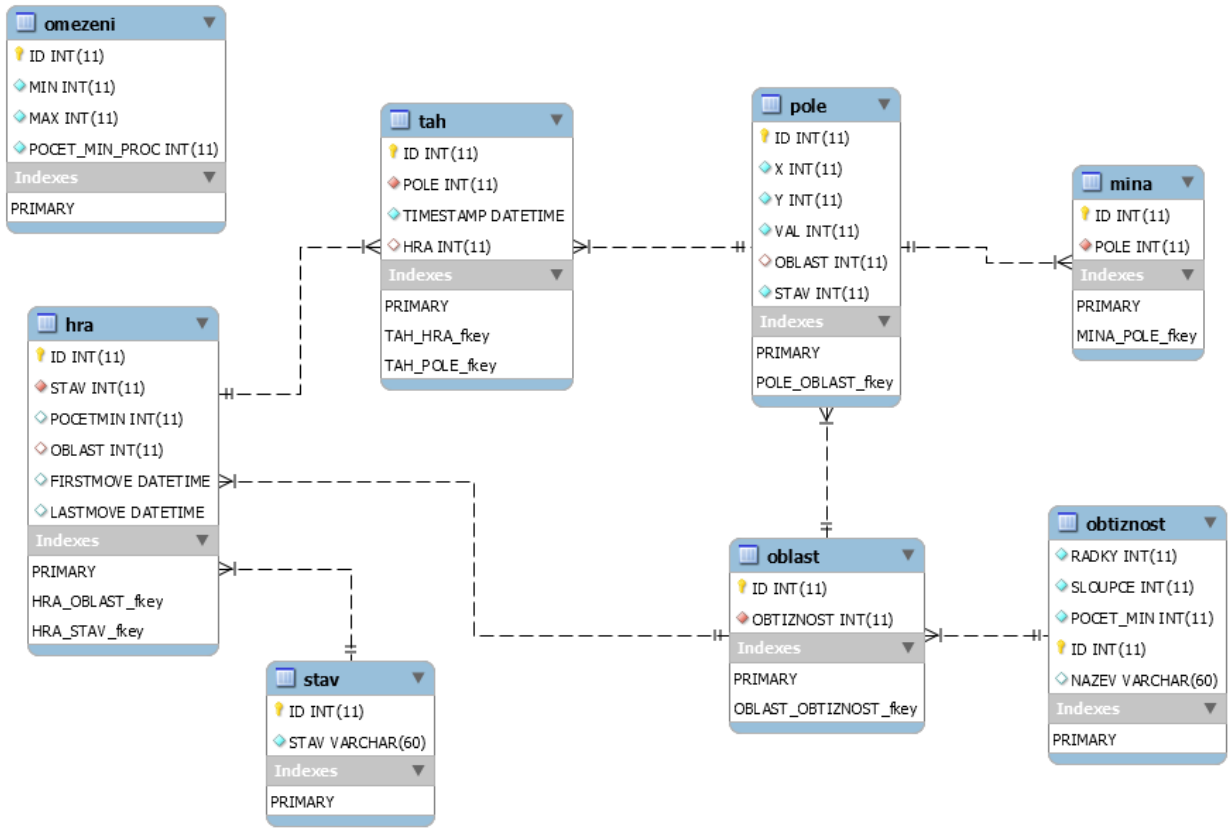
- hlídání hodnot parametrů vlastní obtížnosti (volání funkce SPATNY_PARAMETR).
- kopírování hodnot parametrů obtížnosti pro aktuální oblast, pokud byla zvolena základní obtížnost.
- zaminování nastavené oblasti (volání procedury ZAMINUJ_OBLAST) a očíslování polí této oblasti (volání procedury SPOCITEJ_OBLAST).

- volání automatických kontrol (volání funkcí ODKRYTA_MINA a VYHRA) a případně akcí (volání procedury ODKRYJ_POLE) při odkrytí pole.
- zabránění odkrytí již odkrytého pole.
- hlídání počtu polí označených jako zaminované (volání funkce MNOHO_MIN).
- zabránění označení pole jako zaminované, které je již takto označeno nebo je odkryté.
- průběžnou aktualizaci hry po každém jejím tahu.
- označení min, pokud hra skončila úspěšně (volání procedury OZNAC_MINY).
- zabránění odkrytí pole, pokud hra (neúspěšně) skončila.
- zabránění odkrytí pole, které je hráčem označené jako zaminované.

Konfigurace, spuštění a průběh hry

1) Jednorázová konfigurace databáze:
    a. Naplnění tabulky OBTIZNOST daty reprezentující 3 základní obtížnosti hry.
    b. Naplnění tabulky OMEZENI daty definující omezení pro vlastní obtížnost hry.
    c. Naplnění tabulky STAV daty odpovídající různým stavům hry.
2) Hra je zahájena vložením nového záznamu do tabulky OBLAST. Automaticky se spustí plnění daty tabulky POLE, které reprezentují podobu definované zaminované oblasti. Nakonec se do tabulky HRA automaticky vloží nový záznam.
3) Dále má hráč na výběr jednu z možností:
    a. Zobrazit si aktuální podobu zaminované oblasti prostřednictvím pohledu OBLAST_TISK.
    b. Odkrýt libovolné pole vložením záznamu do tabulky TAH. Dále dochází k aktualizaci příslušného záznamu v tabulce HRA.
    c. Označit libovolného pole jako mina vložením záznamu do tabulky MINA.
    d. Zrušit označení zaminovaného pole smazáním odpovídajícího záznamu v tabulce MINA.
4) Pokud hra pokračuje, pokračuj bodem 3, jinak bodem 5 (úspěch) nebo 6 (neúspěch).
5) Hra skončila úspěchem. Je vhodné si zobrazit:
    a. Výsledkovou listinu vítězů voláním pohledu VITEZOVE.
    b. Zobrazení odminované oblasti voláním pohledu OBLAST_TISK.
6) Hra skončila neúspěchem. Je vhodné si zobrazit:
    a. Výsledkovou listinu poražených voláním pohledu PORAZENI.
    b. Seznam chybně označených min voláním pohledu CHYBNE_MINY.
7) 7. Novou hru zahájíme bodem 2.

Datová analýza



Funkční analýza

Databázový pohled VITEZOVE

Zobrazí tabulku her, které byly úspěšně dokončeny. Zobrazuje parametry obtížnosti hry a dobu hraní.

```
-- View: public."VITEZOVE"

-- DROP VIEW public."VITEZOVE";

CREATE OR REPLACE VIEW public."VITEZOVE" AS
 SELECT "HRA"."ID",
    "STAV"."STAV",
    "OBTIZNOST"."NAZEV",
    "OBTIZNOST"."RADKY",
    "OBTIZNOST"."SLOUPCE",
    "OBTIZNOST"."POCET_MIN",
    "HRA"."LASTMOVE" - "HRA"."FIRSTMOVE"
   FROM "HRA"
     JOIN "OBLAST" ON "HRA"."OBLAST" = "OBLAST"."ID"
     JOIN "STAV" ON "HRA"."STAV" = "STAV"."ID"
     JOIN "OBTIZNOST" ON "OBLAST"."OBTIZNOST" = "OBTIZNOST"."ID"
  WHERE "STAV"."ID" = 2;

ALTER TABLE public."VITEZOVE"
    OWNER TO postgres;
```

Databázový pohled PORAZENI

Zobrazí tabulku her, které byly úspěšně dokončeny. Zobrazuje parametry obtížnosti hry, dobu hraní a počet správně odhalených min.

```
-- View: public."PORAZENI"

-- DROP VIEW public."PORAZENI";

CREATE OR REPLACE VIEW public."PORAZENI" AS
 SELECT "HRA"."ID",
    "STAV"."STAV",
    "OBTIZNOST"."NAZEV",
    "OBTIZNOST"."RADKY",
    "OBTIZNOST"."SLOUPCE",
    "OBTIZNOST"."POCET_MIN",
    "HRA"."LASTMOVE" - "HRA"."FIRSTMOVE",
    count("MINA"."ID") AS count
   FROM "HRA"
     JOIN "OBLAST" ON "HRA"."OBLAST" = "OBLAST"."ID"
     JOIN "STAV" ON "HRA"."STAV" = "STAV"."ID"
     JOIN "OBTIZNOST" ON "OBLAST"."OBTIZNOST" = "OBTIZNOST"."ID"
     JOIN "POLE" ON "POLE"."OBLAST" = "HRA"."OBLAST"
     JOIN "MINA" ON "MINA"."POLE" = "POLE"."ID"
  WHERE "STAV"."ID" = 3 AND "POLE"."VAL" = '-1'::integer
  GROUP BY "HRA"."ID", "STAV"."STAV", "OBTIZNOST"."NAZEV", "OBTIZNOST"."RADKY",
"OBTIZNOST"."SLOUPCE", "OBTIZNOST"."POCET_MIN", ("HRA"."LASTMOVE" - "HRA"."FIRSTMOVE");

ALTER TABLE public."PORAZENI"
    OWNER TO postgres;
```

Funkce VYHRA

Pokud počet neodkrytých polí se rovná počtu min, vrací funkce 1 (výhra)

```
-- FUNCTION: public."VYHRA"(integer)
-- DROP FUNCTION public."VYHRA"(integer);

CREATE OR REPLACE FUNCTION public.VYHRA(
        oblast integer)
    RETURNS integer
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
DECLARE
count     integer;
pocet_min         integer;

BEGIN
  SELECT count(*) INTO count FROM "POLE" WHERE "POLE"."OBLAST" = oblast AND ("POLE"."STAV" = 0 OR
"POLE"."STAV" = -1 OR "POLE"."STAV" = null);
  SELECT "OBTIZNOST"."POCET_MIN" INTO pocet_min FROM "OBTIZNOST"
  INNER JOIN "OBLAST" ON "OBLAST"."OBTIZNOST" = "OBTIZNOST"."ID"
  WHERE "OBLAST"."ID" = oblast ;

  IF count = pocet_min THEN
    RETURN 1;
        UPDATE "HRA" SET "STAV"=2;
  ELSE
        RETURN 0;
  END IF;
END;

$BODY$;

ALTER FUNCTION public."VYHRA"(integer)
    OWNER TO postgres;
```

Procedura ZAMINUJ_OBLAST

Náhodně rozmístí miny v oblasti.

```
-- FUNCTION: public."ZAMINUJ_OBLAST"(integer)
-- DROP FUNCTION public."ZAMINUJ_OBLAST"(integer);

CREATE OR REPLACE FUNCTION public.ZAMINUJ_OBLAST(
        oblast integer)
   RETURNS void
   LANGUAGE 'plpgsql'
   COST 100
   VOLATILE
AS $BODY$
DECLARE
pocet_min integer;
x_count integer;
y_count integer;
x integer;
y integer;
mina integer;

BEGIN

SELECT "OBTIZNOST"."POCET_MIN", "OBTIZNOST"."SLOUPCE", "OBTIZNOST"."RADKY"  INTO pocet_min,
x_count, y_count FROM "OBLAST" INNER JOIN "OBTIZNOST" ON "OBTIZNOST"."ID" = "OBLAST"."OBTIZNOST"
WHERE "OBLAST"."ID" = oblast;

FOR i IN 1..pocet_min LOOP
   LOOP
    SELECT random_between(1, x_count), random_between(1, y_count) INTO x, y;
    SELECT COUNT(*) INTO mina FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x AND "Y" = y;
    EXIT WHEN mina = 0;
   END LOOP;

   INSERT INTO "POLE" ("OBLAST", "X", "Y", "VAL", "STAV") VALUES (oblast, x, y, -1, 0);
 END LOOP;
END;
$BODY$;

ALTER FUNCTION public."ZAMINUJ_OBLAST"(integer)
   OWNER TO postgres;
```

Procedura SPOCITEJ_OBLAST

Dopočte pro každé pole počet sousedících min.

```
-- FUNCTION: public."SPOCITEJ_OBLAST"(integer)

-- DROP FUNCTION public."SPOCITEJ_OBLAST"(integer);

CREATE OR REPLACE FUNCTION public.SPOCITEJ_OBLAST(
        oblast integer)
    RETURNS void
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
DECLARE
pocet_min integer;
x_count integer;
y_count integer;
x integer;
y integer;
val integer;
mine_count integer;

BEGIN

SELECT "OBTIZNOST"."POCET_MIN", "OBTIZNOST"."SLOUPCE", "OBTIZNOST"."RADKY"  INTO pocet_min,
x_count, y_count FROM "OBLAST" INNER JOIN "OBTIZNOST" ON "OBTIZNOST"."ID" = "OBLAST"."OBTIZNOST"
WHERE "OBLAST"."ID" = oblast;
```

```
FOR x IN 1..x_count LOOP
  FOR y IN 1..y_count LOOP

        SELECT count(*) INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x AND "Y" = y;
    CONTINUE WHEN val > 0;

        mine_count := 0;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x-1 AND "Y" = y-
1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x-1 AND "Y" = y;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x-1 AND "Y" =
y+1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x AND "Y" = y-1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x AND "Y" = y+1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x+1 AND "Y" = y-
1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x+1 AND "Y" = y;
        IF val = -1 THEN mine_count := mine_count +1; END IF;

        SELECT "VAL" INTO val FROM "POLE" WHERE "OBLAST" = oblast AND "X" = x+1 AND "Y" =
y+1;
        IF val = -1 THEN mine_count := mine_count +1; END IF;
```

```
INSERT INTO "POLE" ("OBLAST", "X", "Y", "VAL", "STAV") VALUES (oblast, x, y, mine_count, 0);

  END LOOP;
END LOOP;

END;
$BODY$;

ALTER FUNCTION public."SPOCITEJ_OBLAST"(integer)
    OWNER TO postgres;
```

Trigger TGR_NEW_GAME

Tento trigger se spouští po vložení záznamu do tabulky OBLAST, čímž se spoští nová hra. Trigger zavolá procedury ZAMINUJ_OBLAST a SPOCITEJ_OBLAST, čímž inicializuje hrací plochu.

```
-- Trigger: TGR_NEW_GAME
-- DROP TRIGGER "TGR_NEW_GAME" ON public."OBLAST";

CREATE TRIGGER "TGR_NEW_GAME"
    BEFORE INSERT
    ON public."OBLAST"
    FOR EACH ROW
    EXECUTE PROCEDURE public."TG_NEW_GAME"();

-- FUNCTION: public."TG_NEW_GAME"()
-- DROP FUNCTION public."TG_NEW_GAME"();

CREATE FUNCTION public."TG_NEW_GAME"()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
PERFORM ZAMINUJ_OBLAST(NEW."ID");
PERFORM SPOCITEJ_OBLAST(NEW."ID");

INSERT INTO public."HRA"(
            "STAV", "OBLAST")
            VALUES (0, NEW."ID");
END;

$BODY$;

ALTER FUNCTION public."TG_NEW_GAME"()
    OWNER TO postgres;
```

Trigger TGR_OBTIZNOST

Tento trigger se spoustí při vložení záznamu do tabulky OBTIZNOST a hlídá správné parametry nové obtížnosti.

```
-- Trigger: TGR_OBTIZNOST
-- DROP TRIGGER "TGR_OBTIZNOST" ON public."OBTIZNOST";

CREATE TRIGGER "TGR_OBTIZNOST"
    BEFORE INSERT
    ON public."OBTIZNOST"
    FOR EACH ROW
    EXECUTE PROCEDURE public."TG_OBTIZNOST"();

-- FUNCTION: public."TG_OBTIZNOST"()
-- DROP FUNCTION public."TG_OBTIZNOST"();

CREATE FUNCTION public."TG_OBTIZNOST"()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
res integer;

BEGIN

SELECT SPATNY_PARAMETR(NEW."RADKY", NEW."SLOUPCE", NEW."POCET_MIN") INTO res;
IF res = 1 THEN
    RAISE EXCEPTION 'Chybny parametr';
  END IF;

  RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public."TG_OBTIZNOST"()
    OWNER TO postgres;
```

Scénář

Hra se zahájí vložením nového záznamu do tabulky OBLAST. Tím se aktivuje trigger, který voláním procedury ZAMINUJ_OBLAST oblast zaminuje a volaním procedury SPOCITEJ_OBLAST dopočte ostatním polím počet sousedících min.

Pole se odkrývá vložením záznamu do tabulky TAH kde STAV=1. Spuštěný trigger testuje, zda hra stále běží, pole již nebylo odkryto nebo označeno, či hráč nevyhrál (funkce VYHRA). Pokud nastala výhra, upraví se příslušný záznam v tabulce HRA a volá se procedura OZNAC_MINY, která označí zbývající miny. Pokud hráč odkryl minu hra končí, upraví se příslušný záznam v tabulce HRA a volá se procedura OZNAC_MINY, která označí zbývající miny.

Zaminové pole se označuje vložením záznamu do tabulky TAH kde STAV=-1. Trigger hlídá jestli již nebylo označeno příliš mnoho min voláním funkce MNOHO_MIN. Odznačení miny se provede vložením záznamu do tabulky TAH kde STAV=0.

View OBLAST_TISK vypíše celé herní pole.

View VITEZOVE zobrazí tabulku všech úspěšně dokončených her.

View PORAZENI zobrazí tabulku všech neúspěšně dokončených her.

Závěr

Hlavní funkční část aplikace jsem splnil, hra je hratelná dle platných pravidel. Hra je spustitelná a dohratelná a zároveň automaticky hlídá nastane-li výhra, prohra, nebo hráč provede neplatný tah.

V první verzi bylo řešeno odkrývání polí přes tabulku POLE. To nebylo optimální protože protože se pak cyklicky volalo odkrývání sousedících polí funkcí ODKRYJ_POLE spouštěné aktivovaným triggerem. Jelikož tato funkce byla řešena rekurzivně, aplikaci došla memory heap. V aktuální verzi je odkrývání polí řešeno přes tabulku TAH, na které je i trigger spouštějící odkrývání sousedních polí.

Ač se dle mého názoru databázové systémy nehodí k vytváření aplikační logiky, mohl jsem si při vývoji hry vyzkoušet hlubší práci s databázovými systémy, hlavně s funkcemi, procedurami a triggery.

Na závěr musím zhodnotit nástroj pgAdmin. Dle mého názoru je to nejhorší IDE, se kterým jsem se kdy setkal. Vyloženě hází vývojáři klacky pod nohy. Ve spoustě případů lze obtížně provádět potřebné úpravy, takže je ve výsledku jednodušší vytvořit danou strukturu odznova. V mnoha to dokonce nejde vůbec a člověk je nucen začít znovu. Mohu konstatovat, že díky tomu mi semestrální práce zabrala třikrát tolik času, než by měla.