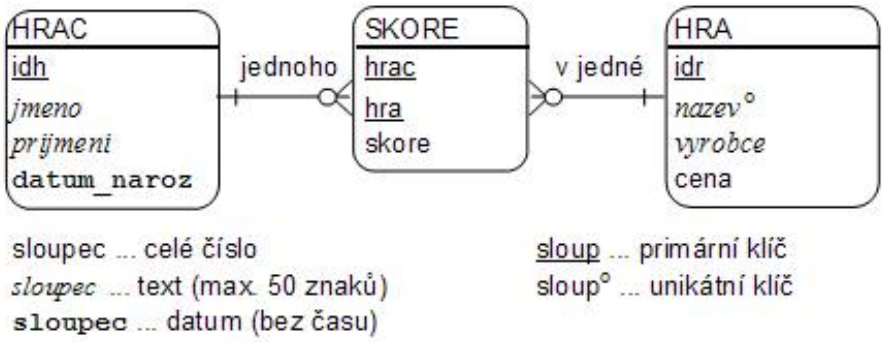
Databáze 2 – vzor testu[1] s řešením



Podle výše uvedeného ERA diagramu napište:

1. příkaz pro vytvoření domény `nazev` – povinně zadávaný text o délce max. 50 znaků;

```
CREATE DOMAIN nazev AS VARCHAR(50) NOT NULL;
```

2. příkaz pro vytvoření tabulky `HRA` (`cena` je celé číslo, `nazev` je unikátní);

```
CREATE TABLE hra (idr INTEGER NOT NULL PRIMARY KEY,
                  nazev nazev UNIQUE,
                  vyrobce nazev,
                  cena INTEGER);
```

3. příkaz, kterým se o čtvrtinu zvýší cena her od výrobce `'XYZ'`;

```
UPDATE hra SET cena=cena*1.25 WHERE vyrobce='XYZ';
```

4. příkaz pro výpis všech her, které mají alespoň jedno skóre, seřadit dle výrobce, dále dle názvu hry (vypsáno bude: `idr`, `nazev`, `vyrobce`, `cena`);

```
SELECT hra.* FROM hra JOIN skore ON idr=hra ORDER BY vyrobce, nazev;
```

5. příkaz pro vytvoření tabulky `SKORE` tak, že:

   - změny provedené v nadřazených tabulkách se promítnou i do této,
   - nebude možné smazat hru, která byla alespoň jednou hrána (= má alespoň jedno skóre),
   - při smazání hráče se smažou všechna jeho skóre;

```
CREATE TABLE skore (hrac INTEGER NOT NULL,
                    hra INTEGER NOT NULL,
                    skore INTEGER,
                    PRIMARY KEY (hrac, hra),
                    FOREIGN KEY (hra) REFERENCES hra (idr) ON UPDATE CASCADE ON DELETE NO ACTION,
                    FOREIGN KEY (hrac) REFERENCES hrac (idh) ON UPDATE CASCADE ON DELETE CASCADE);
```

6. proceduru `SKORE_PRIDEJ` se 3 vstupy (`id_hrace`, `id_hry` a `skore`), která uloží skóre zadaného hráče v zadané hře – v případě, že dvojice hráč-hra již existuje (= hráč už hru hrál) a nové skóre je vyšší než staré, uloží se pouze změna skóre. Procedura nemá žádný výstup;

```
CREATE PROCEDURE skore_pridej (id_hrace INTEGER, id_hry INTEGER, skore INTEGER)
 AS
 DECLARE VARIABLE pom INTEGER;
 BEGIN
    SELECT skore FROM skore WHERE hrac=:id_hrace AND hra=:id_hry INTO :pom;
    IF (pom IS NULL) THEN
       INSERT INTO skore VALUES (:id_hrace, :id_hry, :skore);
    ELSE
       IF (pom<skore) THEN
          UPDATE skore SET skore=:skore WHERE hrac=:id_hrace AND hra=:id_hry;
 END||
```

[1] V textu jsou červeně uvedena upřesnění původního textu.

7. příkaz pro výpis úplně všech her s jejich průměrným skóre (pokud hru zatím nikdo nehrál, bude uvedena nula). Vypsáno bude: `idr`, `nazev`, `prum_skore`. Seřaďte podle názvu hry;

```
SELECT idr, MIN(nazev) AS 'nazev', AVG(skore) AS 'prum_skore'
FROM hra LEFT JOIN skore ON idr=hra
GROUP BY idr
ORDER BY MIN(nazev);
```

8. příkaz pro výpis všech hráčů (`idh`, `jmeno`, `prijmeni`, `pocet_her`), kteří hráli alespoň pět her, seřadit od těch, co hráli nejvíc her, k nejmenším gamblerům (sestupně);

```
SELECT idh, MIN(jmeno) AS 'jmeno', MIN(prijmeni) AS 'prijmeni', COUNT(hra) AS 'pocet_her'
FROM hrac LEFT JOIN skore ON idh=hrac
GROUP BY idh HAVING COUNT(hra)>=5
ORDER BY COUNT(hra) DESC;
```

9. příkaz pro výpis těch hráčů (vždy seřadit dle příjmení), kteří

   (a) získali celkem alespoň 1000 bodů (= hodnota skóre) a hráli alespoň 5 her,

   ```
   SELECT idh, MIN(jmeno) AS 'jmeno', MIN(prijmeni) AS 'prijmeni',
          COUNT(hra) AS 'pocet_her',SUM(skore) AS 'celk_skore'
   FROM hrac JOIN skore ON idh=hrac
   GROUP BY idh HAVING SUM(skore)>=1000 AND COUNT(hra)>=5
   ORDER BY MIN(prijmeni);
   ```

   (b) skórovali alespoň v 10 hrách a v každé získali alespoň 400 bodů.

   ```
   SELECT idh, MIN(jmeno) AS 'jmeno', MIN(prijmeni) AS 'prijmeni',
          COUNT(hra) AS 'pocet_her', SUM(skore) AS 'celk_skore'
   FROM hrac JOIN skore ON idh=hrac
   WHERE skore>=400
   GROUP BY idh HAVING COUNT(hra)>=5
   ORDER BY MIN(prijmeni);
   ```

   Vypsáno bude: `idh`, `jmeno`, `prijmeni`, `pocet_her`, `celk_skore`;

10. proceduru `HRA_NOVA`, která má tyto textové vstupy: `nazev`, `vyrobce`, cena. Procedura automaticky vytvoří nové `idr` (můžete využít i generátor, avšak v tomto případě uveďte také příkaz pro jeho vytvoření!). Procedura nemá žádný výstup;

```
CREATE SEQUENCE s_hra;
CREATE PROCEDURE hra_nova (nazev VARCHAR(50), vyrobce VARCHAR(50), cena INTEGER)
AS  BEGIN
      INSERT INTO hra VALUES (gen_id(s_hra,1), :nazev, :vyrobce, :cena);
    END||
```

11. pohled `HRY_NEHRANE`, které obsahuje `idr`, `nazev`, `vyrobce`, `cena` a týká se her, které nikdo zatím nehrál;

```
CREATE VIEW hry_nehrane (idr, nazev, vyrobce, cena)
AS SELECT idr, MIN(nazev), MIN(vyrobce), MIN(cena)
   FROM hra LEFT JOIN skore ON idr=hra
   WHERE COUNT(hrac) IS NULL
   GROUP BY idr;
```

12. selektovou proceduru `SKORE_OD_DO`, jejímiž vstupy jsou `a` (= min. skóre) a `b` (= max. skóre) a výstupy jsou `idh`, `jmeno`, `prijmeni`, `idr`, `nazev`, `skore` – vypíše všechna skóre ležící v daném intervalu. Výstup je seřazen sestupně podle skóre, dále dle názvu hry.

```
CREATE PROCEDURE skore_od_do (a INTEGER, b INTEGER)
 RETURNS (idh INTEGER, jmeno VARCHAR(50), prijmeni VARCHAR(50), idr INTEGER,
          nazev VARCHAR(50), skore FLOAT)
 AS
 BEGIN
    FOR SELECT idh, jmeno, prijmeni, idr, nazev, skore
        FROM (hrac JOIN skore ON idh=hrac) JOIN hra ON hra=idr
        WHERE skore BETWEEN :a AND :b
        ORDER BY skore DESC, nazev
        INTO :idh, :jmeno, :prijmeni, :idr, :nazev, :skore
    DO SUSPEND;
 END||
```