Databázové systémy

Projekt - Dokumentace

autoři: Findra Michal (xfindr00)
Tverdokhlib Vladyslav (xtverd01)

Brno 2021

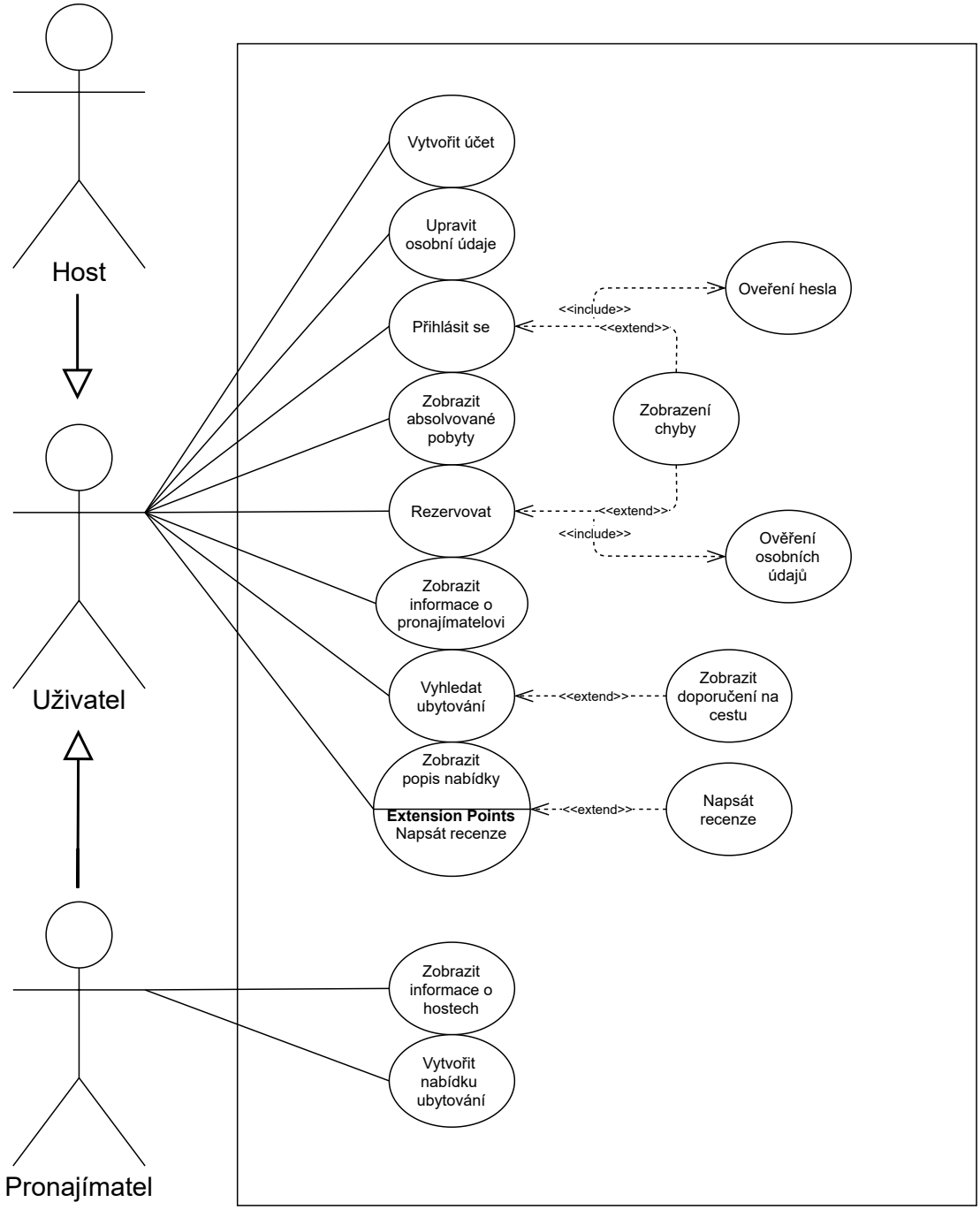
1 Zadání

Nabídka ubytování

Navrhněte IS OffHause (Offer flat or hause), který by poskytoval přehled o nabízeném ubytování, recenze hostů, údaje o pronajímatelích, o hostech, jejich pobytech, platby za pokoje, atd. Hosté mohou provádět rezervace pokojů (tím pádem musí zadat své osobní údaje). Host si může v rámci jedné rezervace objednat více pokojů, třeba i na jiné datum. Stačí, když majitel nemovitosti bude mít informace pouze o jednom hostu, který zaštiťuje celý pobyt (o ostatních účastnících pobytu nemusí být dostupné žádné informace). Pobyt je vytvořen na základě rezervace, k uskutečnění pobytu však nemusí dojít. Jednotlivé domy či byty mají jisté vybavení (parkování, výtah, kuchyň, topení, apod.), pravidla, popis ubytování, informace o přístupu pro hosty a bezpečnostní prvky. Host si může rezervovat buď jednotlivé pokoje nebo celý byt/dům. Cena ubytování se navíc může lišit podle období, po které se host ubytuje (týdenní, měsíční sleva). IS bude dále schopen evidovat skutečně absolvované pobyty - tj. která ubytování host od kdy do kdy využíval. U každého pobytu je evidována platba za daný pobyt. Aby se hosté mohli správně rozhodnout, jaké ubytování si zvolí, můžou psát na ubytování recenze a musí znát i pronajímatele ubytování - základní informace o něm, zda se s ním domluví, jak vypadá a jak rychle odpovídá na dotazy. Host si vybírá ubytování především podle destinace, do které chce jet, a tak mu OffHause nabízí nejen tuto možnost, ale také potřebné informace o tom, co může v daném městě dělat, jak se tam může přepravovat a seznam aktivit a míst, které může navštívit.

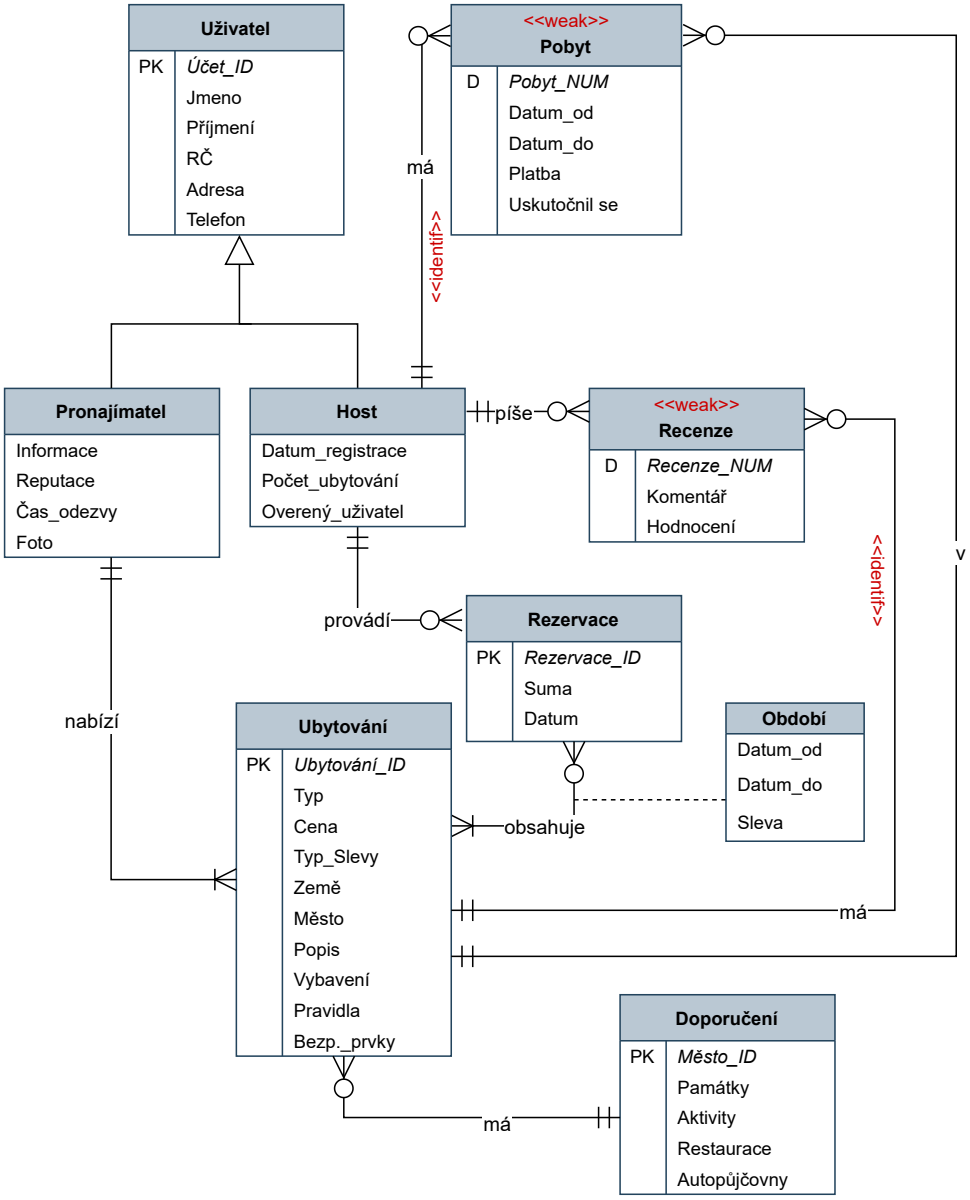
1.1 Use Case

OffHause



1.2 ER Diagram

OffHause



2 Popis

Jednoduchý informační systém ubytovaní hostů.

Systém uchovává informace o uživatelích a pobytech. Dá se zadat recenze od hosta, a každé ubytování má aj informace o svém okolí.

Uživatel: Každý uživatel má příslušné údaje pro registraci. Uživatelé se dělí na *pronajímatele* a *hosty*.
-Generalizace/specializace *uživatel*/*pronajímatel* & *host* je konjunktivní a totální.
Každý uživatel si může vytvořit účet, upravit udaje, zobrazit absolvované pobyty, vyhledat pobyt, rezervovat pobyt, zobrazit informace o pronajímateli, zobrazit popis nabídky.
Při přihlašování v IS probíhá ověření hesla. Neplatné údaje vedou k zobrazení chyby.
Při rezervaci probíhá kontrola na ověření osobních údajů.
V prostoru zobrazení popisu nabídky má možnost uživatel napsat recenzi. Při vyhledávání ubytování si může uživatel zobrazit odpovídající doporučení.

Pronajímatel: Dědí údaje od *uživatele*. Reputace závisí na hodnocení z recenzi hostů.
-Pronajímatel musí nabízet alespoň jedno *ubytování* v IS.
Každý pronajímatel si může zobrazit informace o jeho hostech a vytvořit nabídku ubytování.

Host: Dědí údaje od *uživatele*. Má možnost získat status ověřeného uživatele na základe stanovených podmínek. Ověřený host má hodnotnější recenze.
-Host má možnost provádět *rezervace* a psát *recenze*.
-V DB se zaznamenají taky *pobyty* hostu bez ohledu či ubytovaní proběhlo nebo ne.

Ubytování: Entitní množina nabídek ubytování od *pronajímatele*. Typ ubytování je rozmanitý - od pokoje po vily. Typ slevy se může lišit podle *období* ubytování.
-Na základě města se ubytování přiřazuje jedno *doporučení* na cestu od IS.
-Jedno ubytovaní může mít libovolný počet *recenzí*.
-Jedno ubytování může mít libovolný počet *rezervací*, tudíž i *pobytů*.

Pobyt: Slabý typ entity, který má složeny PK z ID *hostu* a číslování. Pobyt je uskutečněn, když se host ubytoval. Atribut platba informuje, jestli byl pobyt zaplacen a jakým způsobem.
-Jednotlivé pobyty jsou vždy přiřazené jednomu *hostu*.

Doporučení: K *ubytovaní* se váže i doporučení o blízkých aktivitách, památkách atd. které mužů ubytovaný hosté navštívit.

Rezervace: Každá rezervace má jednoznačné ID. Suma je vypočtena s ohledem na slevu.
-V rámci jedné rezervace může být rezervováno i více *ubytování*.

Období: Atributy vztahu *Rezervace-Ubytování*. Sleva se určuje například podle období.

Recenze: Slabý typ entity, který má složeny PK z ID *hostu* a číslování. Hodnocení je číselná hodnota ubytování, která ovlivňuje také reputaci pronajímatele. Ověřený host může zanechat aj komentář jak byl s ubytováním spokojen.

3 Návrh databáze

Ku každé entitní množině byla vytvořena příslušná tabulka. Pro implementaci konjunktní-disjunktní a totální generalizace/specializace jsme vytvořili tři tabulky.

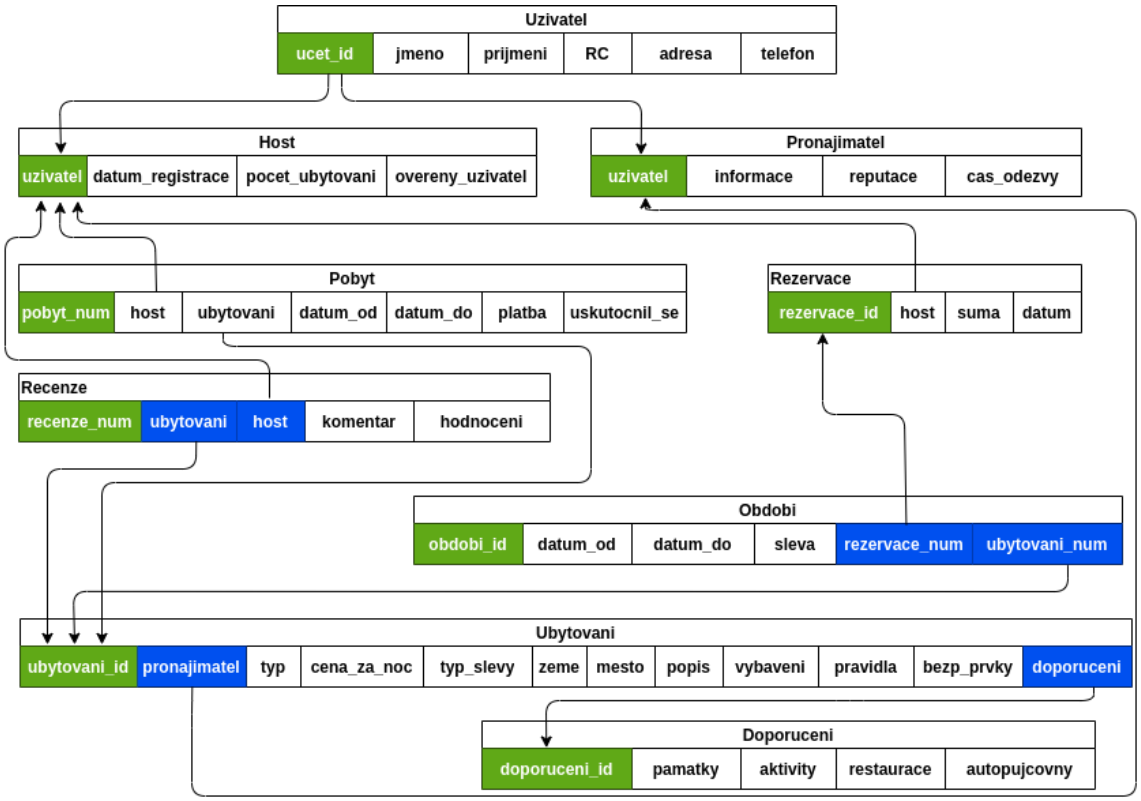
- Uživatel - rodič. Má PK, který může být společným pro Hosta a Pronajímatele. Obsahuje atributy, které se týkají hosta a pronajímatele.

  Entita z Uživatel není samostatná a nemůže být vytvořena bez vztahu k alespoň jednomu z potomků

- Host - potomek. Má FK, jehož hodnota odkazuje především na PK Uživatele pomocí REFERENCES. Obsahuje pak individuální atributy pro hosta

- Pronajímatel - potomek. Má FK, jehož hodnota především odkazuje na PK Uživatele pomocí REFERENCE. Obsahuje pak individuální atributy pro pronajímatele.

Definičním oborem FK potomků jsou hodnoty PK rodiče.
V obrázku 1 jsou v relační databázi znázorněny vztahy mezi tabulkami. Primární klíč v tabulkách je zvýrazněn zelenou a cizí klíč je zvýrazněn modrou barvou.



Obrázek 1: Relační databáze

4 Inicializace

Na začátku vykonávaní se stále zkusí vymazat tabulky, triggery a procedury pokud existují a vytvoří se nové tabulky s novýma hodnotami, nové triggery a nové procedury aby se předšilo práci s neplatnými nebo špatnými daty.

5 Triggery

Implementované jsou dva typy triggerů.

5.1 Trigger uzivatel_gen_PK

První trigger automaticky generuje hodnoty PK uživatele. Používá CREATE SEQUENCE, který vytvoří sekvenci. Každé volání NEXTVAL vrátí další hodnotu.

5.2 Trigger recenze_komentar_integrity

Druhy trigger před vložením komentářů v recenzi ověří, či je host ověřeny uživatel, jestliže není, tak se komentář nastaví na hodnotu NULL.

6 Procedury

6.1 nejdrazsi_pobyt_hostu

První procedura pracuje na spojení tabulek `rezervace, období, pobyt, host`. Jediný argument je id hosta. Procedura kontroluje:

- jestli je uživatel host
- jestli má host pobyty které se uskutečnili

Procedura pomocí kurzoru

```
CURSOR kurz IS SELECT distinct r.host, u.cena_za_noc, o.datum_od,
o.datum_do, o.sleva, p.uskutocnil_se, p.pobyt_num
```

projede v cykle všechna dostupná data a uloží je do `kurzrow`, který je následne spracován a vyhodnocen.

Příklad úspěšného vykonání procedury:

```
exec  nejdrazsi_pobyt_hostu(5);
```

dá na výstup: Nejdrazsi pobyt hostu id 5 je id 1 (47.25).

6.2 discount_application

Druhá procedura aplikuje zadánu svolenu slevu ku rezervaci a aktualizuje potřebná data v tabulkách. Procedura dokáže zpracovat slevu typu int aj slevu typu float. Prvý argument je id rezervace, a druhý procentuální sleva.

Procedura kontroluje:

- jestli jsou platná zadána procenta
- jestli existuje daná rezervace

Procedura používa dva kurzory:

```
CURSOR kurz IS SELECT suma from rezervace where rezervace_id = rez_id;
CURSOR kurz2 IS SELECT sleva from obdobi where rezervace_num = rez_id;
```

pomocí kterých získá data z potřebných tabulek.

Následně procedura vypočte slevu a aktualizuje potřebné hodnoty v tabulce `rezervace, obdobi`.

Příklad úspěšného vykonání druhé procedury:

```
exec discount_application(2, 50);
```

dá použitím příkazu

```
select distinct r.rezervace_id, r.suma, o.sleva from rezervace r
LEFT JOIN obdobi o on r.rezervace_id = o.rezervace_num
where r.rezervace_id = 2;
```

na výstup tabulku v které je v prvním sloupci je id rezervace(2), v druhým suma slevy v eurech (40*0,5) a v třetím aktualizovaná sleva v eurech(40*0,5+10 (původní sleva)).

rezervace_id	suma	sleva
2	20	30

7 Explain plan a Index

Funkce EXPLAIN PLAN byla použita na dotaz, který zjisti jaka byla nejdražší rezervace uživatele Jan Novák.

Neoptimalizovaný běh dotazu byl vykonán následovně:

- SELECT STATEMENT – příkaz SELECT
- SORT GROUP BY NOSORT – přeskočení seřazení
- HASH JOIN – spojení dvou tabulek
- TABLE ACCESS FULL – přístup ku všem řádkům tabulek

V optimalizovaném běhu programu byl použit INDEX na tabulce uživatel podle jména a příjmení, který měl za výsledek optimálnější běh programu a namísto TABLE ACCESS FULL byl použit TABLE ACCESS BY INDEX ROWID. Optimalizaci jde vidět v sloupci `Cost (%CPU)`.

Tabulka s výstupem příkazu EXPLAIN PLAN pro neoptimalizovaný dotaz:

```
-------------------------------------------------------------------------------
| Id | Operation           | Name      | Rows | Bytes | Cost (%CPU)| Time |
-------------------------------------------------------------------------------
| 0  | SELECT STATEMENT    |           | 1    | 297   | 6 (0)  | 00:00:01 |
| 1  | SORT GROUP BY NOSORT|           | 1    | 297   | 6 (0)  | 00:00:01 |
|* 2 | HASH JOIN           |           | 1    | 297   | 6 (0)  | 00:00:01 |
|* 3 | TABLE ACCESS FULL   | UZIVATEL  | 1    | 271   | 3 (0)  | 00:00:01 |
| 4  | TABLE ACCESS FULL   | REZERVACE | 4    | 104   | 3 (0)  | 00:00:01 |
-------------------------------------------------------------------------------
```

Tabulka s výstupem příkazu EXPLAIN PLAN pro optimalizovaný dotaz s použitím INDEX:

```
-------------------------------------------------------------------------------
| Id | Operation           | Name      | Rows | Bytes | Cost (%CPU)| Time |
-------------------------------------------------------------------------------
| 0  |SELECT STATEMENT     |           | 1        | 297 | 3 (0)| 00:00:01 |
| 1  |SORT GROUP BY NOSORT |           | 1        | 297 | 3 (0)| 00:00:01 |
| 2  |NESTED LOOPS SEMI    |           | 1        | 297 | 3 (0)| 00:00:01 |
| 3  |TABLE ACCESS FULL    | REZERVACE | 4        | 104 | 3 (0)| 00:00:01 |
|* 4 |TABLE ACCESS BY INDEX ROWID| UZIVATEL | 1   | 271 | 0 (0)| 00:00:01 |
|* 5 |INDEX RANGE SCAN     | CUSTOM_INDEX | 1     |     | 0 (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

8 Přístupová práva

Přístupová práva druhému členu týmu jsou pro tabulky přiražena použitím příkazu:

```
GRANT ALL ON XXX TO xtverd01;
```

kde namísto XXX je dosazen název tabulky.

Přístupová práva druhému členu týmu jsou pro procedury jsou přiražena použitím příkazu:

```
GRANT EXECUTE ON YYY TO xtverd01;
```

kde namísto YYY je dosazen název procedury.

9 Materialzed view

Materializovaný pohled je vytvořený na dotaz, která ubytovaní jsou ponoukaná na území Slovenska. Pohled by byl použitelný například když by mala aplikace technickou podporu pro jednotlivé krajiny. Při tvořeni pohledu jsme používali následovné možnosti pohledu:

- CACHE – optimalizace čtení z pohledu
- BUILD IMMEDIATE – naplnění pohledu ihned po vytvořeni
- REFRESH ON COMMIT – obnovení pohledu po commitu

Následně jsou práva na dotaz udělená druhému členovi týmu následovne:

```
GRANT ALL ON zeme_sk TO xtverd01;
```

V skripte jsou uvedené aj dotazy využívající daný pohled. Je otestovaná aj funkčnost po aktualizaci dat.

10 Závěr a poděkování

Skript byl testován v aplikaci Oracle SQL devloper[1] a ve Visual Studio Code[2] za pomoci rozšíreni Oracle Developer Tools for VS Code[3].

V průběhu semestru jsme se obeznámili s obojími vývojovými prostředími. Při vytvářeni projektu jsme jsi taktéž prakticky skoušeli látku vysvětlenou na přednáškách a demo cvičeních. Jsme si jist, že získané znalosti se nám hodí aj v praxi.

[1] Dostupný tady: `https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html`

[2] Dostupný tady: `https://code.visualstudio.com`

[3] Dostupný tady: `https://marketplace.visualstudio.com/items?itemName=Oracle.oracledevtools`