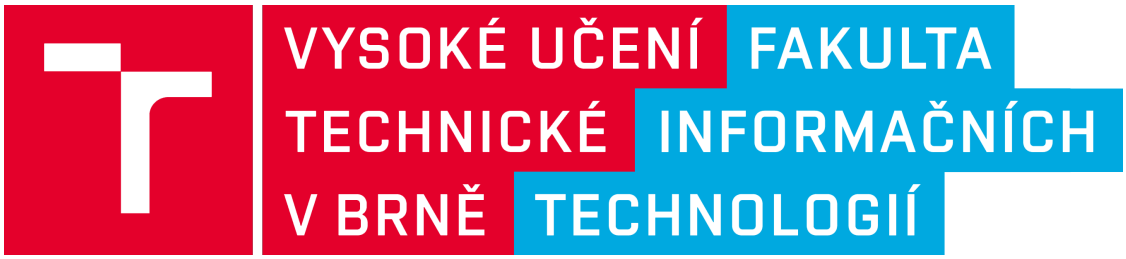


Projekt 5. část

Datový model (ERD) a model případů užití

Zadání č. 66 - Spolujízda

David Chocholatý (xchoch09)
Martin Baláž (xbalaz15)

Brno, 1. května 2022

Obsah

1 Zadání č. 66 - Spolujízda

1.1 Zadání

J&E (Join and enjoy) je nová společnost, která se rozhodla pomoct všem, kteří rádi cestují, ale nemají s kým. Informační systém J&E by tak měl poskytovat přehled o nabízených spolujízdách, nabídkách na společné výlety a zkušenosti uživatelů. Každý uživatel může jak nabízet, tak se i nějaké spolujízdy účastnit. Systém umožnuje uživateli vkládat plánované jízdy, včetně informací o čase výjezdu, autě, kterým pojede, nástupním a výstupním místě, ceně, případné možné zajížďce, časové flexibilitě a možnostech zavazadla. Uživatel samozřejmě může vlastnit i více aut. K snadnému výběru toho správného řidiče poslouží uživatelům následující informace - základní informace o řidiči, e-mailová adresa, telefonní číslo, pár vět o něm, profilová fotka, hodnocení od ostatních uživatelů včetně počtu hvězdiček, zda má rád hudbu, zda mu nevadí kouření nebo zvířata, jestli si s ním dobře popovídáte a jak moc je zkušený. Zkušenost řidiče se odvozuje od počtu jízd, které nabídl. Sám řidič může hodnotit, jak byl spokojený se spolucestujícími a zda byli dochvilní a přátelští. Aby řidič maximálně využil poptávky, může cestovat na různých úsecích cesty s jinými lidmi. Jednotlivé zastávky, na kterých se můžou ostatní uživatelé připojit, jsou rovněž uvedeny v systému. 30 minut před odjezdem obdrží každý přihlášený uživatel spolujízdy SMS zprávu jako upomínku, aby na jízdu nezmeškal. IS J&E navíc nabízí možnost společných výletů. Pokud rádi cestujete, máte spoustu nápadů a inspirace, pak můžete naplánovat výlet včetně popisu plánovaného programu, možnosti ubytování, předpokládaných nákladů na výdaje mimo jízdné, možných aktivit, míst, které hodláte navštívit, náročnost a předpokládané nároky na vybavení (stan, pohorky, surf, .). Z výletu pak jednotliví účastníci můžou vytvořit vlog nebo sepsat článek, aby k podobné cestě inspirovali i ostatní. Vlog stejně jako článek bude mít obsah (popis) a oprávnění (veřejný, sdílený mezi účastníky nebo soukromý).

1.2 Rozšíření

Řešení zadání pro předmět IDS rozšiřuje původní zadání z předmětu IUS v ohledu, kdy *"Uživatel samozřejmě může vlastnit i více aut."*. Realizace návrhu pouze pro případ, kdy uživatel tzv. *"vlastní"* auto by znamenalo omezení pro zachycení situace, kdy například více členů rodiny a zároveň uživatelů systému J&E by nemohlo řídit stejné rodinné auto, jelikož majitelem auta může být pouze jedna osoba. Z toho důvodu jsme vztah *"vlastní"* nahradili vztahem *"řídí"*, kdy ze vztahu *1:N* vznikne vztah *M:N*. Naše řešení tedy zachycuje jak původní situaci, kdy uživatel může řídit více aut, tak zároveň zachycuje i druhý výše zmíněný případ, tedy že jedno auto může řídit více uživatelů.

1.3 Popis

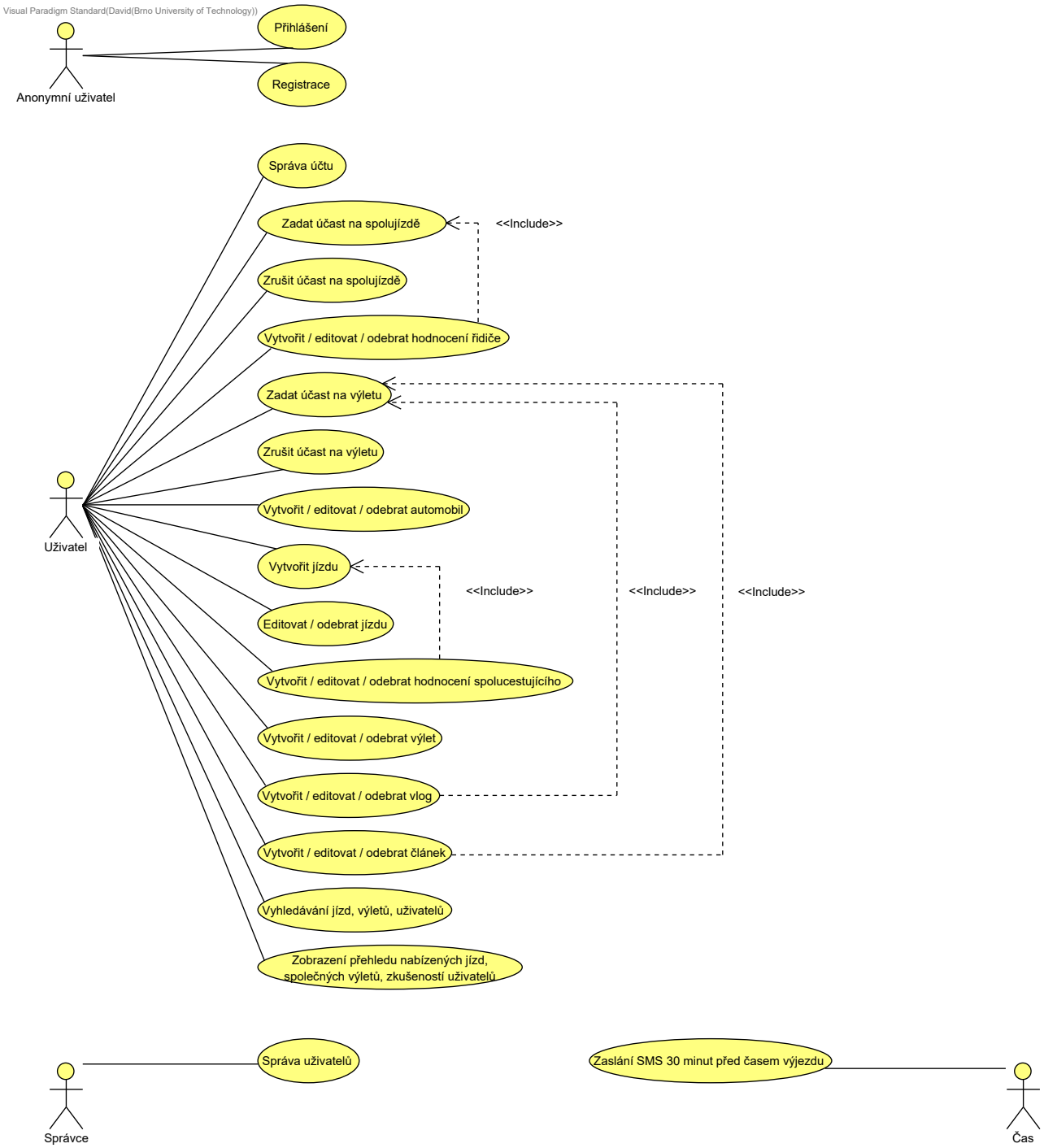
1.3.1 Model případů užití

Pro používání systému uživatelem je nutné se zaregistrovat. Anonymní uživatel má tedy možnost vytvořit zmíněnou registraci nebo se přihlásit. Přihlášený uživatel může pracovat se systémem dle uvedeného zadání. Dále si může také zobrazovat přehledy nabízených jízd, společných výletů a zkušeností uživatelů.

1.3.2 Datový model

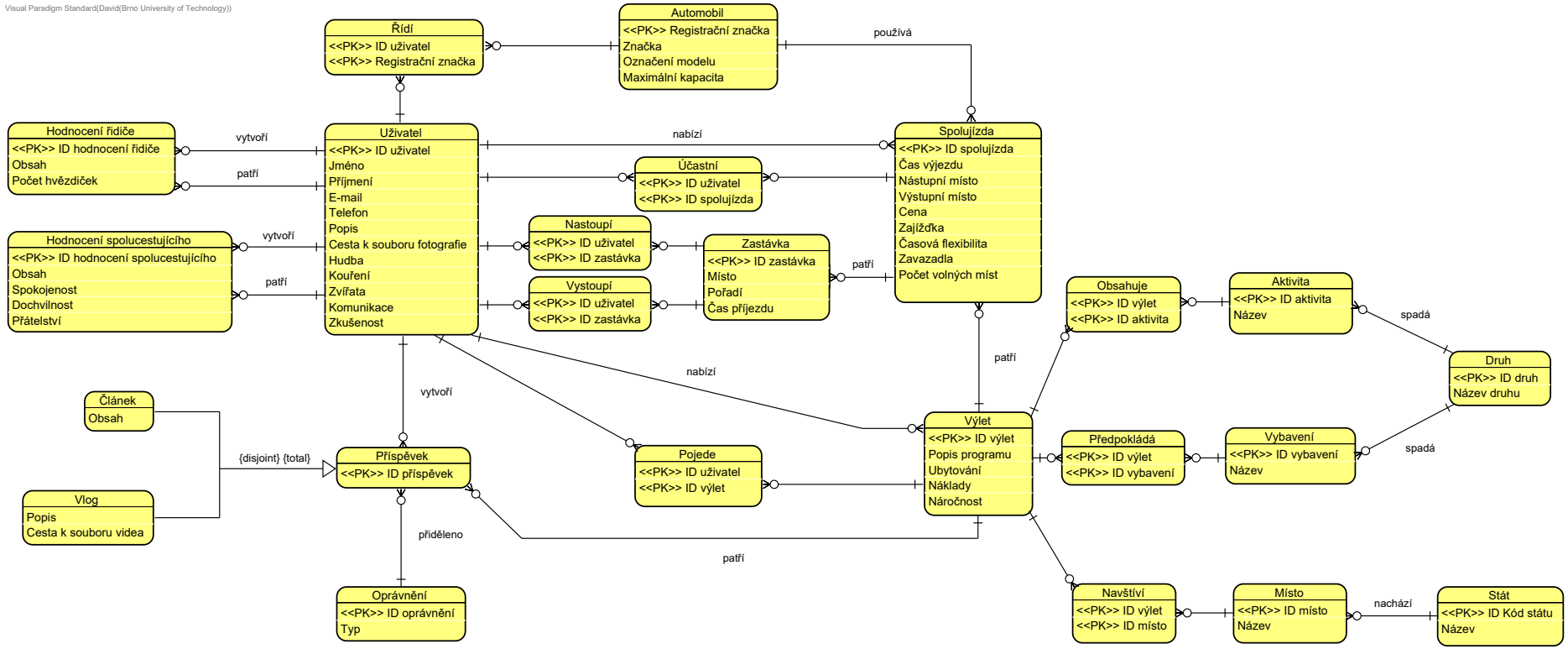
Pro popis entit a vztahů datového modelu lze přímo odkazovat na uvedené zadání, které zachycují. Názvy entit a atributů se shodují s názvy uvedenými v zadání. Jediný rozdíl lze uvést ve zmíněném rozšíření, kdy uživatel auto "nevlastní", ale "řídí". Touto změnou, jak již bylo řečeno, bylo docíleno možnosti, kdy více uživatelů může řídit stejné auto. Dále v entitě *Automobil* je přidán atribut *Maximální kapacita*, což lze při implementaci celého systému použít pro výchozí nabídku počtu volných míst pro uživatele vytvářející novou spolujízdu. Dané rozšíření usnadní práci se systémem. Poslední přidaný atribut se nachází v entitě *Zastávka*. Jedná se o atribut *Čas příjezdu*, kdy uživatel nabízející spolujízdu může uvést předpokládaný čas, kdy na zastávku dorazí. Díky danému atributu při reálném nasazení systému bude velice ulehčena domluva obou stran spolujízdy.

2 Model případů užití



Obrázek 1: Model případů užití

3 Datový model



Obrázek 2: Datový model (ERD)

4 Implementace

Skript nejdříve maže všechny vytvořené tabulky. Poté vytváří tabulky nové dle navrženého datového modelu. Dále implementuje triggery a naplní vytvořené tabulky ukázkovými daty. Následuje vytvoření procedur včetně jejich spuštění, zobrazení prováděcích plánů pomocí `EXPLAIN PLAN`, vytvoření indexu a zobrazení prováděcích plánů po optimalizaci. Nakonec skript přiděluje práva druhému členovi týmu a vytváří materializovaný pohled.

4.1 Triggery

1. První trigger zajišťuje kontrolu, že spolujízdy se nezúčastní více osob, než je maximální kapacita automobilu. Tento trigger se spouští před vložením záznamu do tabulky `ucastni`. Maximální kapacita automobilu je v systému uložena v tabulce `automobil` a je zadávána bez započítání místa pro řidiče. Jedná se tedy o maximální počet pasážerů. Tento trigger využívá agregační funkce `COUNT(*)` a obsahuje dva dotazy pomocí příkazu `SELECT`. První dotaz zjišťuje aktuální obsazení míst na spolujízdě a druhý maximální kapacitu automobilu, který pojede na dané spolujízdě. Trigger nedovolí překročit maximální kapacitu automobilu.

2. Druhý trigger kontroluje, že uživatel nenastoupí na více zastávkách v rámci jedné spolujízdy. Tento trigger se spouští před vložením záznamu do tabulky `nastoupi`. Využívá opět agregační funkce `COUNT(*)` a dvou dotazů pomocí `SELECT` příkazu. První dotaz zjišťuje identifikátor spolujízdy pro danou zastávku. Druhý zjišťuje aktuální počet záznamů s daným uživatelem a danou spolujízdou. Trigger nedovolí přidat záznam, pokud již existuje v tabulkách záznam takový, že uživatel již nastupuje na některé ze zastávek pro danou spolujízdu.

3. Poslední trigger implementuje podobnou vlastnost jako trigger předchozí, liší se ovšem pouze v tom, že kontroluje, že uživatel ***nevystoupí*** na více zastávkách v rámci jedné spolujízdy.

4.2 Procedury

Skript implementuje dvě netriviální procedury, a to `prumerny_pocet_jizd_na_uzivatele` a `pocet_uzivatelu_jedoucich_na_vylet`.

1. První procedura vypíše průměrný počet absolvovaných jízd na uživatele systému. Procedura pracuje s celkovým počtem uživatelů a celkovým počtem jízd uvedených v systému. Procedura pracuje s ošetřením výjimky pro dělení nulou (`ZERO_DIVIDE`). Dělení nulou může nastat v případě, že v systému není zaveden žádný uživatel. Průměrný počet jízd na uživatele se vypočítá pomocí následujícího vztahu:

$$prumerny\_pocet\_na\_uzivatele = pocet\_spolujizd / pocet\_uzivatelu$$

   Tato procedura byla vytvořena za účelem vedení statistik o systému.

2. Druhá procedura vypíše počet uživatelů, kteří pojedou na daný výlet. Tato procedura pracuje s argumentem `arg_id_vylet` určující identifikátor výletu, pro který se bude hledat počet uživatelů. Dále procedura využívá proměnné s datovým typem odkazujícím se na sloupec tabulky. Jedná se konkrétně o proměnnou `id_vylet_tmp` a sloupec `id_vylet` z tabulky `pojede`. Při implementaci byl také využit kurzor, a to konkrétně kurzor `kurzor_vylet`. Do toho se nahrávají identifikátory jednotlivých výletů z tabulky `pojede`. Pokud se daný identifikátor výletu shoduje s hledaným identifikátorem, je inkrementována hodnota čítače `cilovi_uzivatele` Postupně se iteruje přes všechny záznamy z tabulky `pojede`. Poté proměnná `cilovi_uzivatele` obsahuje počet uživatelů, kteří pojedou na výlet s identifikátorem zadaným argumentem procedury.

4.3 Přidělení přístupových práv

Druhému členovi týmu byla přidělena práva na všechny tabulky pomocí příkazu `GRANT ALL ON`. Dále byla udělena také práva pro spouštění implementovaných procedur (`GRANT EXECUTE ON`) a nakonec i právo pro práci s materializovaným pohledem po jeho vytvoření.

4.4 Materializovaný pohled

Před vytvořením samotného materializovaného pohledu byly při implementaci vytvořeny logy pro uchování změn hlavní tabulky. Dále byly při implementaci materializovaného pohledu využity následující vlastnosti:

- CACHE

  Za účelem postupné optimalizace čtení z pohledu.

- BUILD IMMEDIATE

  Naplnění pohledu ihned po jeho vytvoření.

- REFRESH FAST ON COMMIT

  Aktualizace materializovaných pohledů po `COMMIT`.

- ENABLE QUERY REWRITE

  Použití materializovaného pohledu pro optimalizaci dotazu, pro který je vytvořený.

Implementovaný materializovaný pohled je vytvořený pro příkaz, který vypisuje všechna auta v systému, která mohou pojmout více jak tři pasážery. Pro předvedení dotazu je nejprve vypsán jeho obsah, poté jsou přidány nové záznamy do systému, které slouží pro testovaní funkčnosti pohledu a opět je vypsán obsah pohledu.

4.5 EXPLAIN PLAN

Při implementaci se nejprve vypisují prováděcí plány dotazu pomocí příkazu `EXPLAIN PLAN`. Tento příkaz pracuje s dotazem, který slouží pro zjištění informace o tom, kolik spolujízd nabízí nebo nabízeli jednotliví uživatelé. Jedná se o příkaz `SELECT` pracující se spojením dvou tabulek a s agregační funkcí `COUNT()`. Dále obsahuje klauzule `GROUP BY`, `HAVING` a `ORDER BY`. Prováděcí plány byly vytvořeny na poskytnutém školním databázovém serveru Oracle. Příkaz `EXPLAIN PLAN` vypisuje následující výstup:

```
-------------------------------------------------------------------------------------
| Id  | Operation            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |            |     4 |   632 |     8  (25)| 00:00:01 |
|   1 |  SORT ORDER BY       |            |     4 |   632 |     8  (25)| 00:00:01 |
|*  2 |   FILTER             |            |       |       |            |          |
|   3 |    HASH GROUP BY     |            |     4 |   632 |     8  (25)| 00:00:01 |
|*  4 |     HASH JOIN        |            |     4 |   632 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| SPOLUJIZDA |     4 |    52 |     3   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------
|   6 |      TABLE ACCESS FULL| UZIVATEL   |     6 |   870 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Zpracování daného dotazu bez použití indexu je následující:

- SORT ORDER BY

  Seřazení záznamů na základě zadaného kritéria.

- FILTER

  Vyfiltrování záznamů dle klauzule `HAVING`.

- HASH GROUP BY

  Seskupení tabulky na základě hashovacího algoritmu.

- HASH JOIN

  Propojení záznamů tabulek přes hashovací klíč spojení.

- TABLE ACCESS FULL - SPOLUJIZDA

  Přečtení celé tabulky `spolujizda`.

- TABLE ACCESS FULL - UZIVATEL

  Přečtení celé tabulky `uzivatel`.

4.6 Vytvoření indexu

Za účelem optimalizace byl vytvořen index pro sloupec `id_uzivatel` v tabulce `spolujizda`. Tento sloupec byl vybrán pro vytvoření indexu z důvodu, že se jedná o cizí klíč tabulky. Výpis prováděcích plánů pomocí příkazu `EXPLAIN PLAN` s použitím vytvořeného indexu je následující:

```
-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                      |     4 |   632 |     5  (40)| 00:00:01 |
|   1 |  SORT ORDER BY        |                      |     4 |   632 |     5  (40)| 00:00:01 |
|*  2 |   FILTER              |                      |       |       |            |          |
|   3 |    HASH GROUP BY      |                      |     4 |   632 |     5  (40)| 00:00:01 |
|   4 |     NESTED LOOPS      |                      |     4 |   632 |     3   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| UZIVATEL             |     6 |   870 |     3   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------
|*  6 |       INDEX RANGE SCAN | SPOLUJIZDA_ID_UZIVATEL |     1 |    13 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Z důvodu použití indexu došlo ke snížení ceny operací. Ovšem při použití vytvořeného indexu došlo k navýšení procesorového času. Zpracování daného dotazu bez a s indexem je následující:

- SORT ORDER BY

  Seřazení záznamů na základě zadaného kritéria.

- FILTER

  Vyfiltrování záznamů dle klauzule `HAVING`.

- HASH GROUP BY

  Seskupení tabulky na základě hashovacího algoritmu.

- NESTED LOOPS

  Ve vnořených cyklech se každý záznam první tabulky porovná s každým záznamem druhé tabulky.

- TABLE ACCESS FULL - UZIVATEL

  Přečtení celé tabulky `uzivatel`.

- INDEX RANGE SCAN - SPOLUJIZDA_ID_UZIVATEL

  Během skenování rozsahu indexu je přistupováno k sousedním položkám indexu a poté jsou použity hodnoty `ROWID` v indexu k načtení řádků tabulky. Skenování rozsahu indexu je použito z důvodu, že daný SQL příkaz obsahuje omezující klauzuli, která vyžaduje sekvenční rozsah hodnot, které jsou indexy pro tabulku.

Při použití/nepoužití indexu se provedení daného dotazu liší v ohledu, že bez použití indexu jsou záznamy tabulek propojovaný přes hashovací klíč spojení (`HASH JOIN`) a jsou celé načteny obě tabulky, které dotaz používá (`SPOLUJIZDA`, `UZIVATEL`). Narozdíl od popsaného zpracování příkazu bez indexu, varianta dotazu s použitím indexu nepropojuje záznamy tabulek přes hashovací klíč spojení (`HASH JOIN`), ovšem porovnává každý záznam první tabulky s každým záznamem tabulky druhé (`NESTED LOOPS` spojení). Poté je přečtena celá pouze tabulka `UZIVATEL` a je použito skenování rozsahu s využitím indexu (`INDEX RANGE SCAN`).