VYSOKE UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Databázové systémy

Dokumentace k projektu

Zadání 26: Truhlářství

Marie Kuželová (xkuzel08)
Jaroslav Kvasnička (xkvasn14)

2020/2021

Obsah

1 Úvod

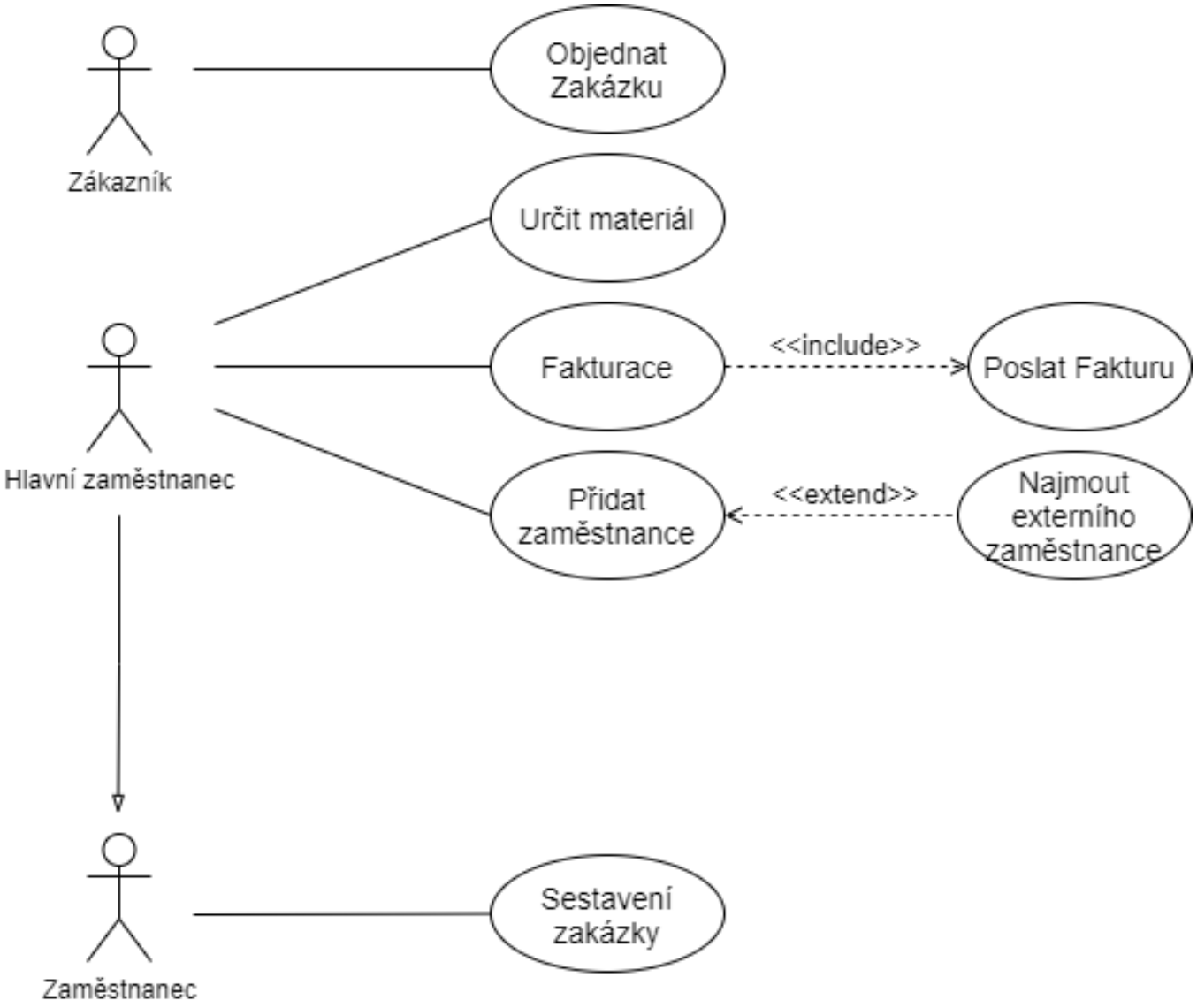
Tato dokumentace slouží k přiblížení tvorby našeho projektu, popsání jednotlivých částí kódu a k celkovému zhodnocení. Kapitoly jsou rozděleny a seřazeny stejně jako projekty ve wisu. První kapitola se věnuje modelu případů užití, datovému modelu a jejich postupu vytváření jednotlivých entit a vztahů. V následujících kapitolách je popsána tvorba tabulek, dotazů SELECT, EXPLAIN PLAN a další. V dokumentaci se nachází i snímky popsaných kódů pro větší přehlednost.

2 Datový model (ERD), model případů užití

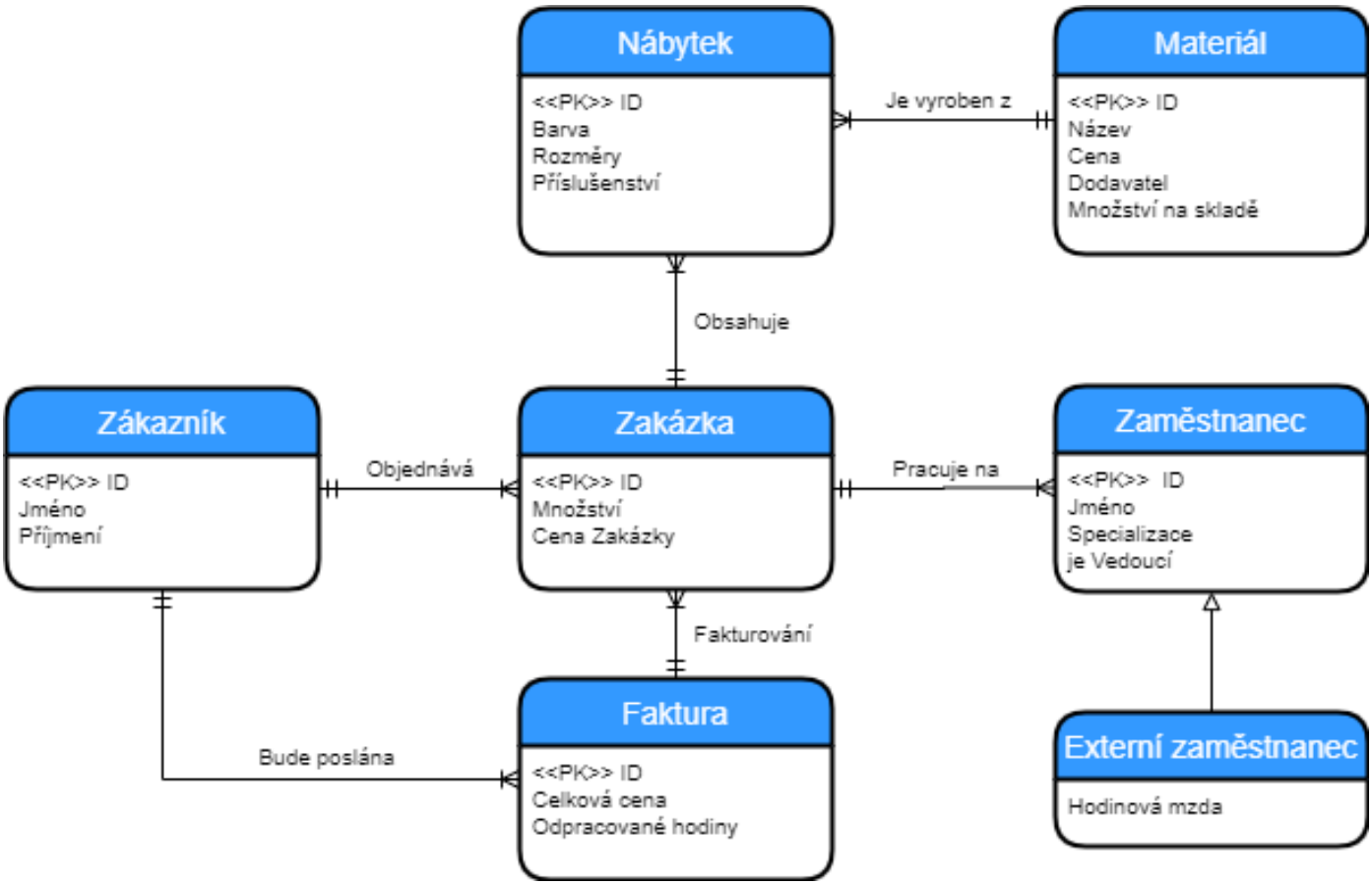
2.1 Zadání

Navrhněte informační systém malého truhlářství, které přijímá zakázky od zákazníků. Každá zakázka má specifické požadavky. Každý kus nábytku má určen materiál, barvu, rozměry, příslušenství, atd. Systém uchovává informace o využitém materiálu na zakázce - materiál, množství, cena, atd. Firma odebírá materiál od více dodavatelů, přičemž každý může z dodavatelů může mít jiné ceny. Každá zakázka má zodpovědného zaměstnance, který řídí celou zakázku, přiděluje další zaměstnance na zakázku, určuje jaký materiál se použije apod. Na zakázce může pracovat více zaměstnanců; každý zaměstnanec má svoji specializaci. Firma může na zakázku najmout i externího zaměstnance, který má opět svoji specializaci a hodinovou mzdu. Jednotlivé položky na zakázce jsou fakturovány (použitý materiál, odpracované hodiny zaměstnanců, ...), z těchto položek je pak vyhotovena celková faktura zakázky.

2.2 Model případů užití



2.3 Datový model



2.4 Dokumentace

Každý kus nábytku má určen materiál, barvu, rozměry, příslušenství, atd. Firma odebírá materiál od více dodavatelů, přičemž každý může z dodavatelů může mít jiné ceny.

- Vytvořili jsme entitu *Nábytek* kde jejími atributy jsou: materiál, barva, rozměry, příslušenství.
- Obdobně jsme postupovali u vytvoření entity *Materiál*. Rozhodli jsme se dodavatele zahrnout mezi atributy této entity, neboť nám přišlo zbytečné modelovat tuto informaci o materiálu jako samostatnou entitu především když o dodavateli není potřeba vědět další informace.

Každá zakázka má zodpovědného zaměstnance, který řídí celou zakázku, přiděluje další zaměstnance na zakázku, určuje jaký materiál se použije apod. Na zakázce může pracovat více zaměstnanců; každý zaměstnanec má svoji specializaci. Firma může na zakázku najmout i externího zaměstnance, který má opět svoji specializaci a hodinovou mzdu.

- Tuto část jsme vyřešili vytvořením entity *Zaměstnanec* s atributy jméno, je Vedoucí a specializace. Atribut je Vedoucí představuje hodnotu typu bool, která bude určovat jestli zaměstnanec je Hlavním zaměstnancem, nebo jiným pracovníkem.
- Jelikož entita externího zaměstnance sdílela atributy s entitou zaměstnance, rozhodli jsme se tento problém minimalizovat pomocí generalizace.

Jednotlivé položky na zakázce jsou fakturovány (použitý materiál, odpracované hodiny zaměstnanců, ...), z těchto položek je pak vyhotovena celková faktura zakázky.

- Atributy entity *Faktura* jsme zredukovali na celkovou cenu a odpracované hodiny, neboť materiál a množství je již zahrnuto v entitě *Zakázka*, ze které vychází.

2.5 Vztahy mezi entitami

Zákazník si může objednat více zakázek a také mu může být zasláno více faktur. Faktura se vystavuje zákazníkovi ve chvíli, když už jsou jeho zakázky splněny, může se ke každé zakázce poslat faktura jednotlivě, nebo hromadně. Na zakázce může pracovat několik zaměstnanců s tím, že mezi nimi bude právě jeden, který bude mít status hlavního zaměstnance. Zakázka je sestavena z jednoho nebo několika kusů nábytku, podle objednávky, který je vyroben z jednoho druhu/typu materiálu od jednoho dodavatele.

3 SQL skript pro vytvoření základních objektů schématu

Jako jednu z prvních věcí jsme se v této části věnovali tvorbě tabulek. Při tom jsme se řídili námi vytvořeným ER diagramem, který jste mohli vidět v první kapitole. Proto jsme si vytvořili celkem devět tabulek. Kde prvních pět tabulek představuje jednotlivé entity diagramu a zbytek jsou tabulky pro spojení jednotlivých tabulek (= entit v diagramu). Dalším krokem při tvorbě tohoto programu bylo vložení dat do jednotlivých tabulek.

4 SQL skript s několika dotazy SELECT

V této části projektu jsme vytvořili celkem 8 dotazů SELECT. Nyní se zaměříme na většinu z těchto příkazů a přestavíme Vám je blíže.

4.1 JOIN

Pro spojení tabulek jsme použili dotaz NATURAL JOIN, který spojuje sloupce se stejným názvem. Tento druh spojení pracuje podobně jako INNER JOIN.

Aby daný uživatel naší aplikace se mohl rychle zorientovat mezi vedoucími pracovník vytvořili jsme tento příkaz, který vyhledá nejen hlavní pracovníky ale taktéž ID dané zakázky.

```
SELECT DISTINCT Person.PersonName, ORDER_WORKER.ORDER_ID
FROM Person NATURAL JOIN Order_Worker
WHERE Person_ID = Worker_ID AND ISMAINWORKER = 1;
```

4.2 klauzule GROUP BY

Tento příkaz se hodí například pro průzkum nejvíce používaného materiálu. Uživatel zjistí, kolik materiálu bylo použito na výrobu nábytku a také jeho cenu. Podle toho se může pak dál řídit při objednávce budoucího materiálu. Může například zvážit zakoupení nejpoužívanějšího materiálu u jiného dodavatele.

```
SELECT MATERIAL.MATERIALNAME, MATERIAL.COST, FURNITURE.COMPONENTS
FROM MATERIAL NATURAL JOIN FURNITURE, MATERIAL_FURNITURE
WHERE MATERIAL.MATERIAL_ID = MATERIAL_FURNITURE.MATERIAL_ID AND FURNITURE.FURNITURE_ID = MATERIAL_FURNITURE.FURNITURE_ID
GROUP BY MATERIAL.MATERIALNAME, MATERIAL.COST, FURNITURE.COMPONENTS
ORDER BY MATERIAL.COST DESC;
```

Výstupem tohoto příkazu je počet jednotlivých barev v zakázkách. Uživateli poslouží jakožto statistika nejpoužívanějších barev. Díky tomu může zvážit jaké další barvy zařadit do objednávky.

```
SELECT Furniture.COLOR, COUNT(Furniture.color)
FROM Furniture NATURAL JOIN MATERIAL_FURNITURE
GROUP BY FURNITURE.COLOR;
```

4.3 predikát EXISTS

Výpis jmen zákazníku, kteří si u teto společnosti už vystavili objednávku.

```
SELECT PERSON.PERSONNAME
FROM PERSON
WHERE EXISTS(
    SELECT ORDERCOST
    FROM THEORDER
    WHERE OrderCost > 0
)
AND EXISTS(
    SELECT *
    FROM BILL_ORDER_CUSTOMER
    WHERE ORDER_ID = ORDER_ID AND PERSON_ID = CUSTOMER_ID
);
```

4.4 predikát IN

Příkaz vypíše tabulku zaměstnanců v této firmě, který slouží pro rychlou orientaci mezi zaměstnanci.

```
SELECT *
FROM PERSON
WHERE PERSON_ID IN (
    SELECT WORKER_ID
    FROM ORDER_WORKER
    WHERE WORKER_ID is not null
);
```

5 Poslední část projektu

5.1 TRIGGER

Funkce triggerů spočívá v tom, že reagují na nějakou událost, aby automaticky provedli nějaký kód. Pomocí triggerů lze více automatizovat funkci databáze. Vytvořili jsme sekvenci pro automatické přiřazování ID pro tabulku BILL. Jelikož již v tabulce existují dva prvky, museli jsme posunout inicializaci hodnot v sekvenci. První trigger používá tuto sekvenci. Pokud se vloží nějaké hodnoty do tabulky BILL, tak se automaticky přiřadí novému řádku informací nová hodnota pomocí sekvence.

Druhý a třetí trigger reaguje na vkládání nových hodnot do tabulky MATERIAL_. Tato relační tabulka obsahuje své ID, a ID Materiálů a ID Nábytků. Pokud se do ní vloží nový řádek, s hodnotami, které neexistují v mateřských tabulkách, tak se provede kód v triggerech a vloží do MATERIAL popř FURNITURE nový řádek s nulovými hodnotami. To proto, aby relační tabulka nemohla být navázána na prázdné hodnoty v tabulkách, a mohlo se vkládat nejen v pořadí MATERIAL, FURNITURE, MATERIAL_, ale i naopak. Nejprve si triggery zjistí existenci hodnot v tabulkách. Pokud hodnoty v tabulkách neexistují, tak se provede INSERT v části exception, který vloží to MATERIAL popř FURNITURE nové hodnoty.

5.1.1 DEMONSTRACE TRIGGERŮ

vloží se do MATERIAL_nový řádek s hodnotami ID-15, MATERIAL_-15, FURNITURE_-150. Po vložení, by se měly zobrazit hodnoty v tabulce MATERIAL_a také prázdné řádky s univerzálními hodnotami s těmito ID's v tabulkách MATERIAL a FURNITURE.

5.2 PROCEDURE

Je databázové schéma, které při zavolání provede příslušný kód.

Před vytvořením procedur jsme museli nastavit SERVEROUTPUT na ON, jelikož nám nefungovalo vypisování na výstup. první procedura používá kurzor. Kurzor je ukazatel na návratové hodnoty nějakého SELECT příkazu. Tím je možné je podržet, dokud nepřijde čas na jejich zpracování. Kurzor používáme k vyselektování vybraného pracovníka. Pokud pracovník pracuje na 2 zakázkách, měly by se zobrazit dva řádky s daty. Poté v těle procedury postupně načítáme jednotlivé řádky a počítáme je a nakonec se vypíše jméno pracovníka a na kolik zakázkách pracuje.

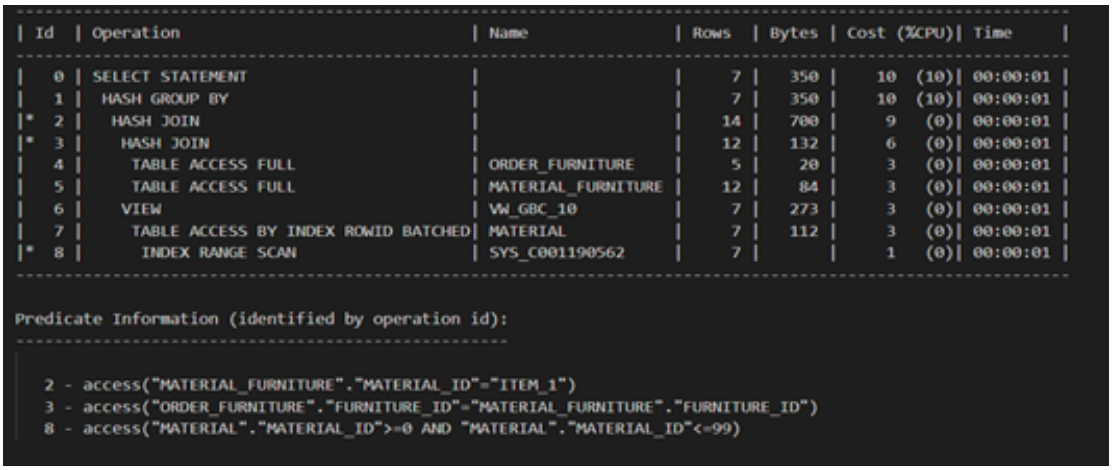
Druhá procedura počítá průměrnou cenu ve všech zakázkách a průměrný čas strávený na zakázkách. V první části procedury se inicializují všechny pomocné hodnoty použité později pro počítání. Implementace FOR EACH lze zastoupit použitím kurzoru a funkcí LOOP. Postupně se načte a posléze i vypočte průměrná hodnota všech nalezených zakázek. k hledání se použil kurzor. Pokud by se žádná zakázka nenalezla, tak se toto zachytí výjimkou.

5.2.1 DEMONSTRACE PROCEDUR

Při hledání "Otto Balgoz"v proceduře have_orders by se mělo zobrazit, že tento pracovník pracuje na 2 zakázkách. Při hledání K̇lára Podřízkováv̈ proceduře have_orders by se mělo zobrazit, že tato pracovnice pracuje na 1 zakázce. Average_all_bills je procedura bez vstupních hodnot a zobrazí průměry cen a časů ze všech zakázek.

5.3 EXPLAIN PLAN

Pro provedení požadovaného databázového dotazu jsme vytvořili příkaz, který slouží pro vyhledání zakoupeného materiálu jeho množství, počtu a celkové ceny. V případě provedení dotazu EXPLAIN PLAN dostaneme jakožto výstup následující tabulku (plan table).

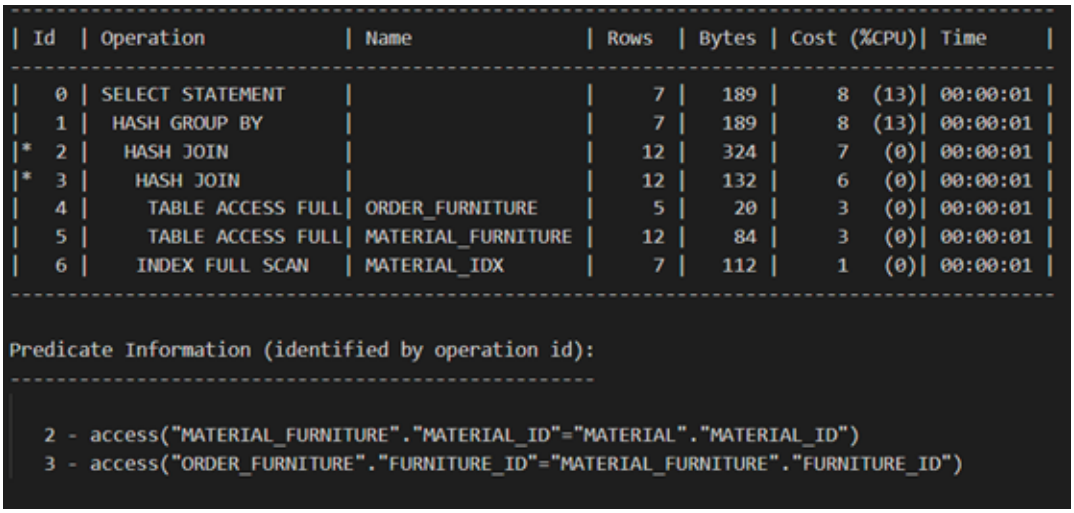


```
| Id  | Operation                           | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT                    |                   |     7 |   350 |    10  (10)| 00:00:01 |
|   1 |  HASH GROUP BY                      |                   |     7 |   350 |    10  (10)| 00:00:01 |
|*  2 |   HASH JOIN                         |                   |    14 |   700 |     9   (0)| 00:00:01 |
|*  3 |    HASH JOIN                        |                   |    12 |   132 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL               | ORDER_FURNITURE   |     5 |    20 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL               | MATERIAL_FURNITURE|    12 |    84 |     3   (0)| 00:00:01 |
|   6 |    VIEW                             | VW_GBC_10         |     7 |   273 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS BY INDEX ROWID BATCHED| MATERIAL       |     7 |   112 |     3   (0)| 00:00:01 |
|*  8 |      INDEX RANGE SCAN               | SYS_C001190562    |     7 |       |     1   (0)| 00:00:01 |

Predicate Information (identified by operation id):

   2 - access("MATERIAL_FURNITURE"."MATERIAL_ID"="ITEM_1")
   3 - access("ORDER_FURNITURE"."FURNITURE_ID"="MATERIAL_FURNITURE"."FURNITURE_ID")
   8 - access("MATERIAL"."MATERIAL_ID">=0 AND "MATERIAL"."MATERIAL_ID"<=99)
```

Jak můžeme vidět nejdříve se vykoná příkaz/dotaz SELECT za ním následuje GROUP BY a JOIN. Po vykonání těchto úkonů se přistupuje k jednotlivým tabulkám, které jsou součástí dotazu JOIN.

5.4 INDEX

Pro efektivní využití indexu je důležité se zaměřit na takové sloupce vybrané tabulky, které se přímo používají v dotazu SELECT a WHERE. Když nyní po vytvoření daného indexu znovu spustíme tentýž EXPLAIN PLAN dotaz výstupem bude tato tabulka.



```
| Id  | Operation           | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT    |                   |     7 |   189 |     8  (13)| 00:00:01 |
|   1 |  HASH GROUP BY      |                   |     7 |   189 |     8  (13)| 00:00:01 |
|*  2 |   HASH JOIN         |                   |    12 |   324 |     7   (0)| 00:00:01 |
|*  3 |    HASH JOIN        |                   |    12 |   132 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| ORDER_FURNITURE  |     5 |    20 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| MATERIAL_FURNITURE|    12 |    84 |     3   (0)| 00:00:01 |
|   6 |    INDEX FULL SCAN  | MATERIAL_IDX      |     7 |   112 |     1   (0)| 00:00:01 |

Predicate Information (identified by operation id):

   2 - access("MATERIAL_FURNITURE"."MATERIAL_ID"="MATERIAL"."MATERIAL_ID")
   3 - access("ORDER_FURNITURE"."FURNITURE_ID"="MATERIAL_FURNITURE"."FURNITURE_ID")
```

Jak můžeme vidět tato tabulka se nejvíce liší od předchozí v sloupci COST. (Taktéž se liší v počtu řádků, což znamená menší počet zásahů při vyhledávání v tabulce) Důvodem je, že INDEX vyhledává všechny řádky, které odpovídají sloupcům poté prochází podmnožinu tabulky. Proto se využití indexu projeví na výkonu/náročnosti a čase, neboť na rozdíl od procházení celé tabulky to zabere o dost méně času.
Dotaz jsme urychlili už tím, že jsme si na začátku pomocí příkazu EXPLAIN PLAN vyzkoušely další dva druhy spojení tří tabulek, a to konkrétně INNER a FULL OUTER JOIN. Největší časový úsek zabere spojení FULL OUTER JOIN jak lze vidět na této tabulce. Důvod je ten, že toto spojení vrací všechny záznamy z tabulek.

5.5 MATERIALIZED VIEW

Je databázový objekt obsahující výsledky nějaké query. Pracuje se s ním stejně jako s VIEW.

Náš MATERIALIZED VIEW stable_person_view vyselektuje hodnoty pro oderder s ID-10000 z tabulek patřících jednomu z týmu. Tento M.VIEW je nastaven na REFRESH ON COMMIT, takže by se tento VIEW měl aktualizovat pouze při commitech. Jelikož školní db commituje při každém spuštění jakékoli query, neměli jsme možnost si názorně otestovat tuto funkci. To by šlo vyřešit, kdyby administrátor zrušil autocommit v databázi. Museli jsme proto zvolit jiný způsob dokázání jeho správné funkčnosti.

5.5.1 DEMONSTRACE MATERIALIZED VIEW

Zobrazíme si hodnoty z MATERIALIZED VIEW stable_person_view a pokusíme se do něj vložit nový řádek dat. Toto nebude fungovat, protože jedna z vlastností MATERIALIZED VIEW je, že do něj nelze přímo vkládat hodnoty. Po dalším zobrazení zjistíme, že se MATERIALIZED VIEW nezměnil.

5.6 GRANTY

Udělení práv druhému členu týmu jsme udělali pomocí GRANT ALL ON -název tabulky/view- TO -xlogin00-.

6 Závěr

Dokumentaci jsme zvolili jako obsáhlejší popis celého projektu z toho důvodu, že se nekonají obhajoby. Tak jsme vám chtěli alespoň tímto způsobem přiblížit náš projekt.