Vysoké učení technické v Brně
Fakulta informačních technologií

Databázové systémy

Dokumentace – Finální schéma databáze Internetový obchod s pastelkami a skicáky

Petr Stehlík, Monika Šnoblová

4. května 2015

1 Úvod

1.1 Internetový obchod

Internetový obchod je webová aplikace, kterou tvoří mnoho jednotlivých stránek obsahující informace o nabízeném zboží. Zejména jeho název, popis, cenu a fotografie. Internetové obchody dnes ve velkém nahrazují kamenné obchody, protože rapidně snižují náklady na provoz obchodu a nároky na prostory.

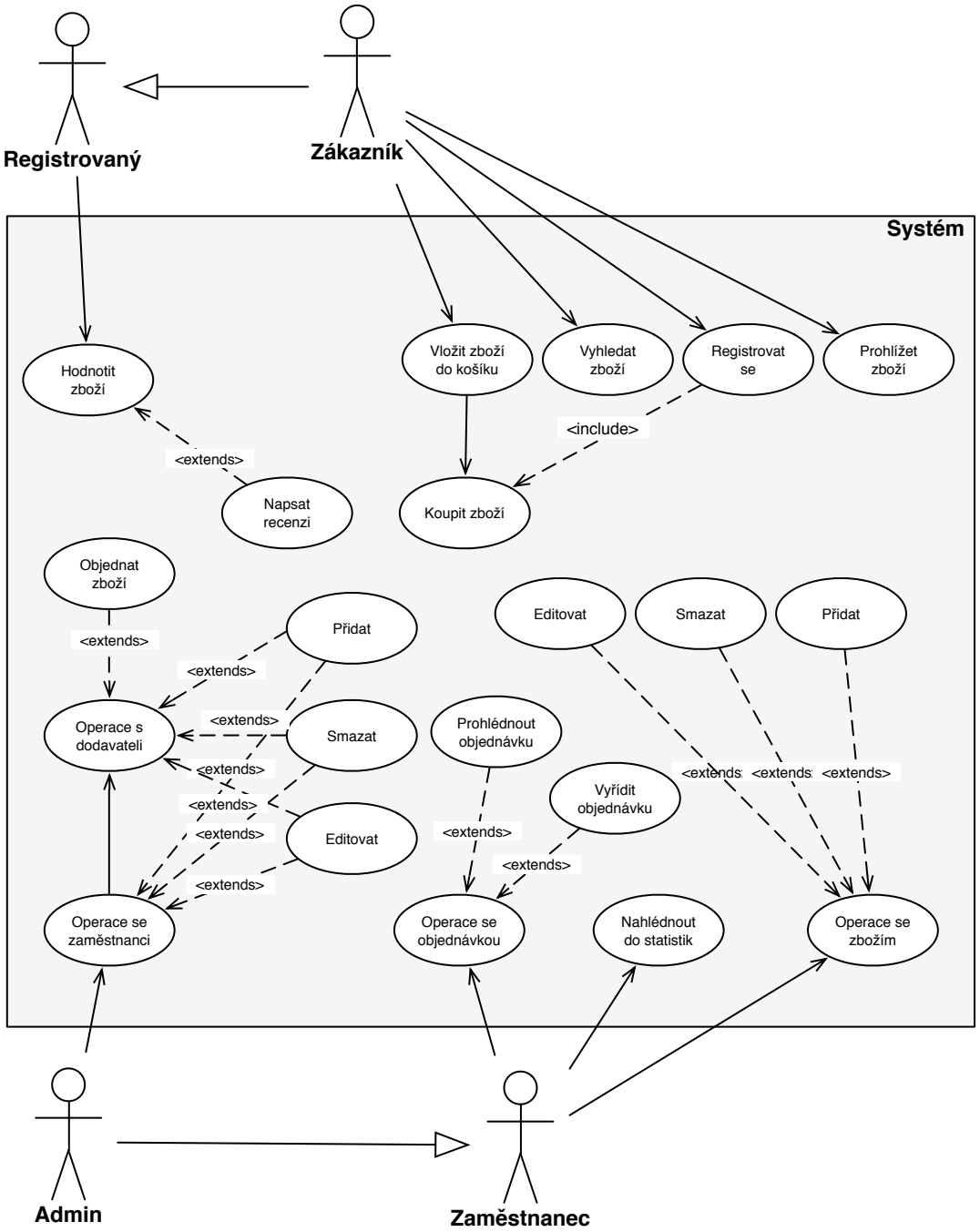
1.2 Neformální specifikace systému

Cílem je vytvoření jednoduché aplikace pro internetový obchod s pastelkami a skicáky. Návštěvníci si mohou pomocí internetového rozhraní prohlížet veškerý sortiment obchodu. Pastelky mohou lišit podle typu (obyčejné, progresso, voskovky, ...) a délky, počtu pastelek v balení, atd. Skicáky se dělí podle gramáže, velikosti, počtu papírů, apod.

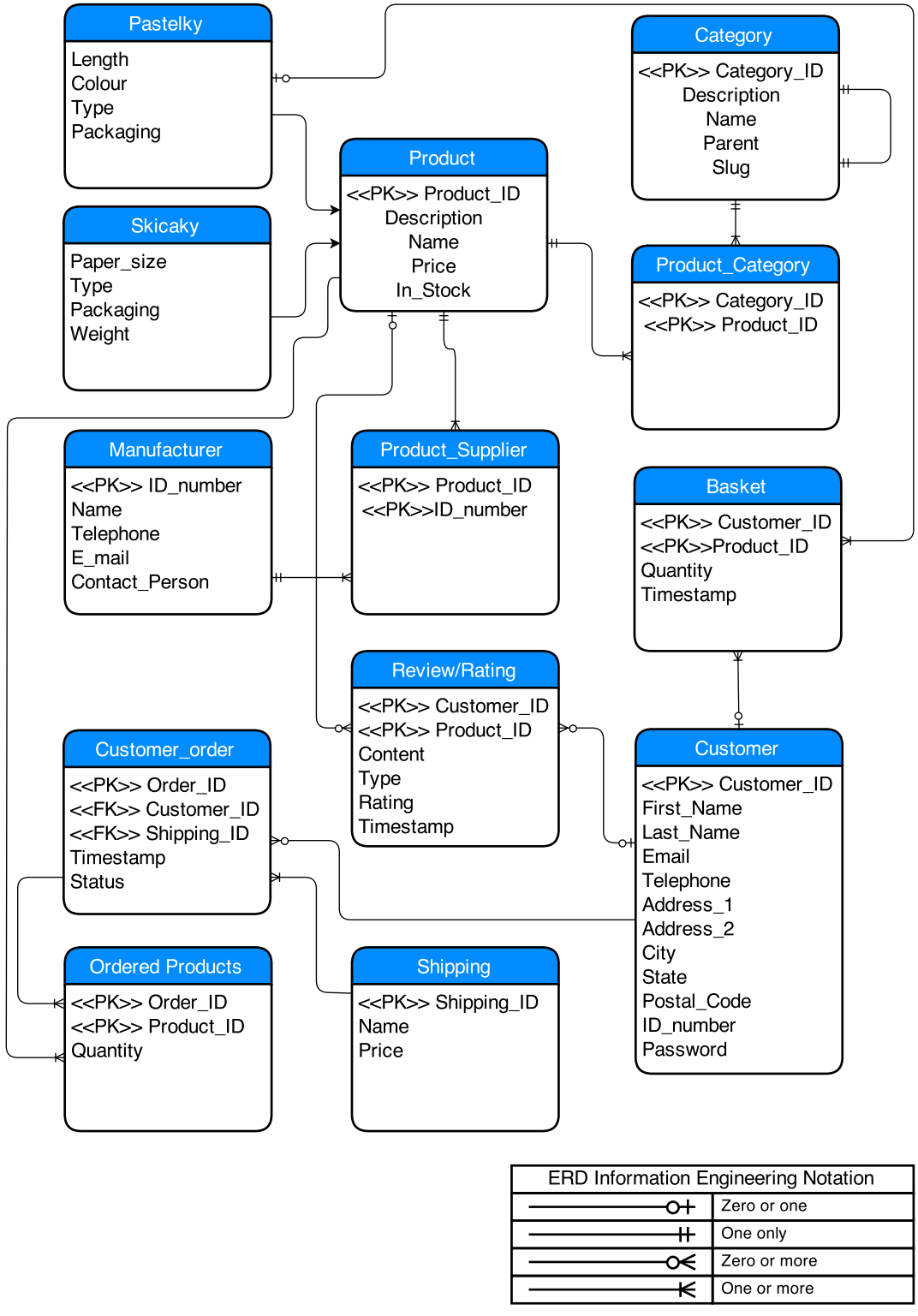
Pokud má návštěvník zájem o určitý produkt/y, může si jej vybrat (vložením do nákupního košíku). U registrovaných zákazníků, kteří jsou do systému přihlášeni, zůstává informace o vybraném zboží v košíku uložena a při opětovném přihlášení znovu načtena. Zákazník si může zboží objednat po zadání potřebných údajů (kontakt, doprava, ...). Zákazníci mohou jednotlivé zboží hodnotit a psát na něj recenze.

V systému jsou uloženy také základní údaje o dodavatelích a stavu objednávky o dodavateli pro opětovné přiobjednání dalšího zboží. Pro jeden produkt může být vice dodavatelů. Doobjednávat zboží může pouze administrátor. Zaměstnanci mohou nahlédnout do statistik hodnocení a prodejnosti zboží.

2 Model případů užití



3 ER diagram


Pastelky
Length
Colour
Type
Packaging
Category
<<PK>> Category_ID
Description
Name
Parent
Slug
Product
<<PK>> Product_ID
Description
Name
Price
In_Stock
Skicaky
Paper_size
Type
Packaging
Weight
Product_Category
<<PK>> Category_ID
<<PK>> Product_ID
Manufacturer
<<PK>> ID_number
Name
Telephone
E_mail
Contact_Person
Product_Supplier
<<PK>> Product_ID
<<PK>>ID_number
Basket
<<PK>> Customer_ID
<<PK>>Product_ID
Quantity
Timestamp
Review/Rating
<<PK>> Customer_ID
<<PK>> Product_ID
Content
Type
Rating
Timestamp
Customer_order
<<PK>> Order_ID
<<FK>> Customer_ID
<<FK>> Shipping_ID
Timestamp
Status
Customer
<<PK>> Customer_ID
First_Name
Last_Name
Email
Telephone
Address_1
Address_2
City
State
Postal_Code
ID_number
Password
Ordered Products
<<PK>> Order_ID
<<PK>> Product_ID
Quantity
Shipping
<<PK>> Shipping_ID
Name
Price
ERD Information Engineering Notation
Zero or one
One only
Zero or more
One or more

4 Generalizace a specializace

Generalizace je v tomto případě na místě v tabulce `product`, který je dvou druhů. Pastelka nebo skicák. Oba druhy sdílí určité sloupce tabulky (např. cena, ID, jméno), ale v určitých parametrech se odlišují (velikost/délka, typ/barva, atd.).

Generalizace je řešena vytvořením dvou pomocných tabulek `sketchbook` a `crayon`, které obsahují primární klíč shodný s jedním z řádků tabulky `product`. Položka v `product` ma dva sloupce: `sketchbook_id` a `crayons_id`, kde vždy má minimálně jeden z těchto sloupců NULL a druhý obsahuje cizí klíč tabulky `sketchbook` nebo `crayon`.

5 Dotazy

Bylo vytvořeno několik triggerů, které kontrolují konzistenci dat v celé databázi. Dvě optimalizace dotazů. Jedna pomocí indexu, druhá pomocí úpravy `join`. Některé výstupy byly ořezány, aby se vmístili na stránku.

5.1 Trigger `delete_order`

Tento trigger je spuštěn vždy, když je smazán záznam z tabulky `customer_order`, který je navázán na tabulku `ordered_product` a bez odstranění odpovídajících řádků nelze úspěšně daný řádek smazat (integritní omezení databáze). Trigger daný závislý záznam prvně smaže a poté se úspěšně provede dané smazání.

5.2 Trigger `add_product_id`

Pokud chybí v příkazu `insert into product ...` hodnota `product_id`, spustí se tento trigger, který daný stav ošetří a chybějící primární klíč přidá.

5.3 Trigger `check_id_number`

Trigger je řešen pomocí procedury, protože je duplicitní s `check_id_number_customer`. Oba kontrolují IČ dle zákonných podmínek pro tvorbu IČ.

5.4 Explain plan s indexem

Pro optimalizaci pomocí indexu byl zvolen select pro výběr všech objednávek mimo Brno, který obsahuje `subselect` a klauzuli `exists`. Díky indexu není zbytečně nahlíženo duplicitně do tabulky `customer`, ve které je uvedeno město zákazníka.

Výstup pro neoptimalizovaný dotaz:

```
-------------------------------------------------------------------------------
| Id  | Operation            | Name           | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                |     1 |   121 |    12   (9)|
|   1 |  HASH UNIQUE         |                |     1 |   121 |    12   (9)|
|*  2 |   FILTER             |                |       |       |            |
|*  3 |    HASH JOIN         |                |     3 |   363 |     6   (0)|
|*  4 |     TABLE ACCESS FULL| CUSTOMER       |     2 |   190 |     3   (0)|
|   5 |     TABLE ACCESS FULL| CUSTOMER_ORDER |     6 |   156 |     3   (0)|
|*  6 |    FILTER            |                |       |       |            |
|*  7 |     TABLE ACCESS FULL| CUSTOMER_ORDER |     1 |    13 |     3   (0)|
-------------------------------------------------------------------------------
```

Jak je vidět, provádí se nad tabulkami zbytečné `TABLE ACCESS FULL`. Proto jsme vytvořili index pro `customer_order(customer_id)` a dotaz je díky tomu rychlejší.

```
-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     1 |    49 |     6  (17)|
|   1 |  HASH UNIQUE                  |                 |     1 |    49 |     6  (17)|
|   2 |   NESTED LOOPS                |                 |     1 |    49 |     5   (0)|
|   3 |    NESTED LOOPS               |                 |     2 |    49 |     5   (0)|
|   4 |     NESTED LOOPS ANTI         |                 |     1 |    43 |     4   (0)|
|*  5 |      TABLE ACCESS FULL        | CUSTOMER        |     1 |    35 |     3   (0)|
|*  6 |      VIEW PUSHED PREDICATE    | VW_SQ_1         |     1 |     8 |     1   (0)|
|*  7 |       INDEX RANGE SCAN        | CUSTOMER_ID_IND |     2 |     6 |     1   (0)|
|*  8 |     INDEX RANGE SCAN          | CUSTOMER_ID_IND |     2 |       |     0   (0)|
|   9 |    TABLE ACCESS BY INDEX ROWID| CUSTOMER_ORDER  |     2 |    12 |     1   (0)|
-----------------------------------------------------------------------------------------
```

5.5 Explain plan s join

Druhá optimalizace se nabízela pro dotaz, který obsahuje `join` a klauzuli `group by`. Díky `inner join` se provádí `INDEX FULL SCAN` kvůli primárním klíčům.

```
explain plan for
       select count(customer_order.order_id) as total_orders, customer.first_name,
           customer.last_name
       from customer_order
       join customer on customer_order.customer_id = customer.customer_id
       group by customer.first_name, customer.last_name;
```

```
-------------------------------------------------------------------------------
| Id  | Operation                | Name           | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                |     5 |    95 |     7  (29)|
|   1 |  HASH GROUP BY           |                |     5 |    95 |     7  (29)|
|   2 |   MERGE JOIN             |                |     5 |    95 |     6  (17)|
|   3 |    TAB ACC BY INDEX ROWID| CUSTOMER       |     5 |    80 |     2   (0)|
|   4 |     INDEX FULL SCAN      | PK_CUSTOMER_ID |     5 |       |     1   (0)|
|*  5 |    SORT JOIN             |                |     5 |    15 |     4  (25)|
|   6 |     TABLE ACCESS FULL    | CUSTOMER_ORDER |     5 |    15 |     3   (0)|
-------------------------------------------------------------------------------
```

Stačilo nahradit `join left outer join` a dotaz je daleko méně náročnější díky vlastnostem `left outer join` .

```
explain plan for
       select count(customer_order.order_id) as total_orders, customer.first_name,
           customer.last_name
       from customer_order
       left outer join customer on customer_order.customer_id = customer.customer_id
       group by customer.first_name, customer.last_name;
```

```
---------------------------------------------------------------------
| Id |Operation           |Name          |Rows | Bytes | Cost (%CPU)|
---------------------------------------------------------------------
|   0|SELECT STATEMENT    |              |    5|    95 |     7  (15)|
|   1| HASH GROUP BY      |              |    5|    95 |     7  (15)|
|*  2|  HASH JOIN OUTER   |              |    5|    95 |     6   (0)|
|   3|   TABLE ACCESS FULL|CUSTOMER_ORDER|    5|    15 |     3   (0)|
|   4|   TABLE ACCESS FULL|CUSTOMER      |    5|    80 |     3   (0)|
---------------------------------------------------------------------
```

5.6 Materializovaný pohled

Pro pohled byl vybrán dotaz se subselectem a klauzulí `not in`.

```
create materialized view dead_soul
  nologging
  cache
  build immediate
  enable query rewrite
as
select cu.first_name, cu.last_name, cu.email
from customer cu
where cu.customer_id not in (
  select co.customer_id
  from customer_order co
  where cu.customer_id = co.customer_id
  );
```

```
-------------------------------------------------------------------------------
| Id  |Operation                     |Name          |Rows |Bytes|Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 |SELECT STATEMENT              |              |    2|   78|    6  (17)|
|   1 | MERGE JOIN ANTI              |              |    2|   78|    6  (17)|
|   2 |  TABLE ACCESS BY INDEX ROWID|CUSTOMER      |    5|  180|    2   (0)|
|   3 |   INDEX FULL SCAN            |PK_CUSTOMER_ID|    5|     |    1   (0)|
|*  4 |  SORT UNIQUE                 |              |    5|   15|    4  (25)|
|   5 |   TABLE ACCESS FULL          |CUSTOMER_ORDER|    5|   15|    3   (0)|
-------------------------------------------------------------------------------
```

Zde je vidět jak dotaz prováděn a je vidět i použití indexu, který ačkoliv byl vytvořen pro jiný dotaz, zde svou roli hraje také a dotaz je daleko rychlejší.

Po vytvoření materializovaného pohledu vidím pouze jeho použití. Tento výstup je od druhého člena týmu, kterému byl povolen přístup pro použité tabulky.

```
-----------------------------------------------------------------------------
| Id | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |           |     2 |    56 |     3   (0)|
|  1 |  MAT_VIEW REWRITE ACCESS FULL| DEAD_SOUL |     2 |    56 |     3   (0)|
-----------------------------------------------------------------------------
```

6 Závěr

Skript obsahuje i další součásti zadání. Konkrétně procedury pro kontrolu IČ a vypočítání průměrného hodnocení zadaného produktu pomocí `cursor`.

Projekt nás velmi dobře připravil na práci a tvorbu databází. Námi zvolené zadání je tím zajímavější, že je rozsáhlejšího rázu a tím pádem klade větší důraz na preciznost při tvorbě dotazů.