VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projekt 1. část - Datový model (ERD), model případů užití

Internetový obchod

Databázové systémy

12. března 2023 Mark Barzali(xbarza00), Dinara Garipova(xgarip00)

Obsah

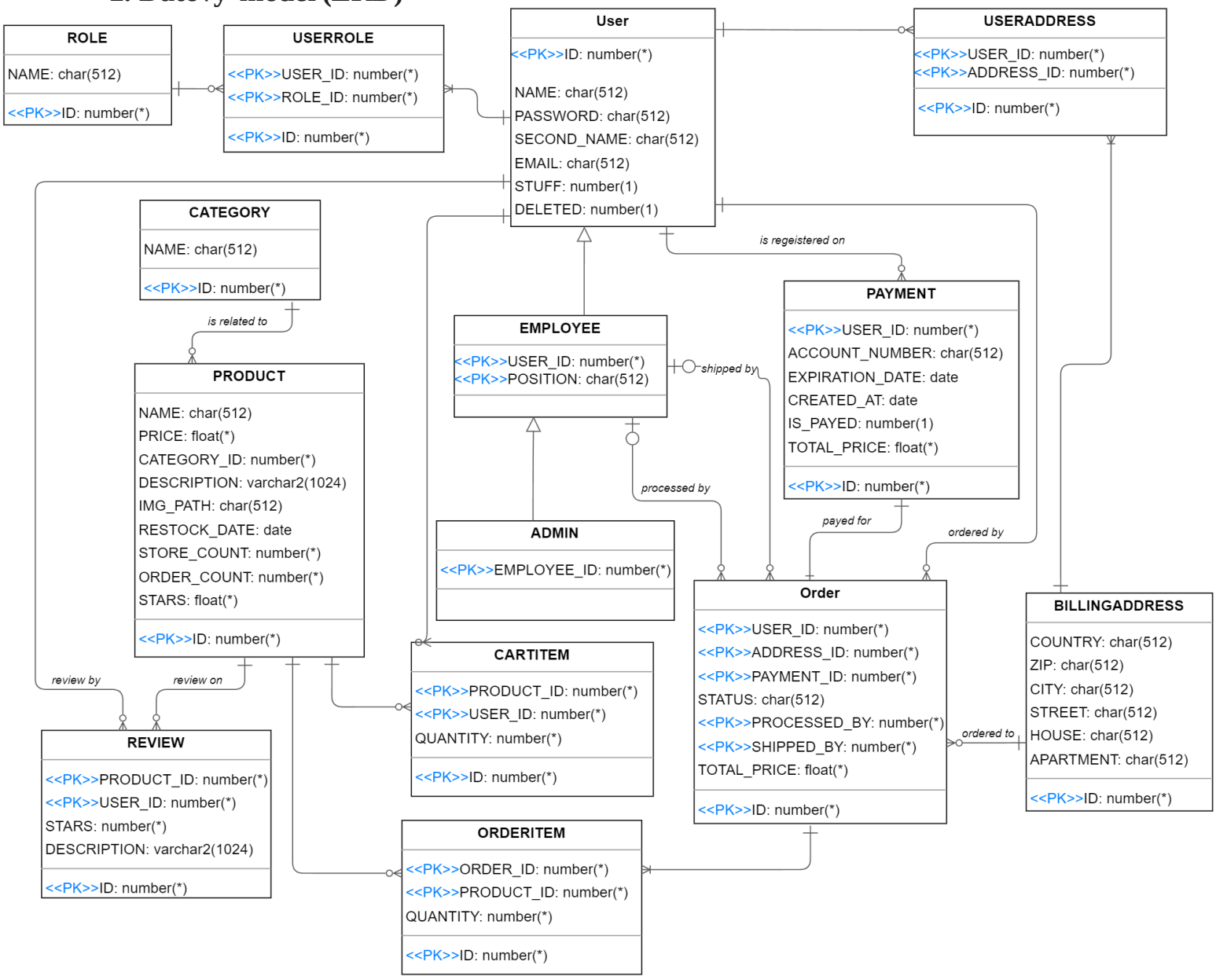
1. Zadání

Název projektu: Internetový obchod

Zadání:

Cílem je vytvoření jednoduché aplikace pro internetový obchod s určitým druhem zboží např. knihkupectví. Návštěvníci WWW stránek mají možnost prohlížet si veškerý sortiment obchodu, který je členěn do kategorií, ať už daný produkt je skladem či nikoliv. Pokud má návštěvník zájem o určitý produkt, může si jej vybrat (vložení do nákupního košíku). Vybrané zboží si může objednat po zadání potřebných údajů (kontakt, doprava, ...). Zboží si může objednat pouze registrovaný uživatel, pokud uživatel nakupuje poprvé, musí se zaregistrovat a získá přihlašovací jméno a heslo.
Tyto údaje může použít k modifikaci osobních informací. Po zaplacení zákazníkem za zboží je považována obchodní transakce za vyřízenou a zaměstnanec obchodu vyexpeduje zboží podle objednávky. Vedení obchodu má informace o celkových tržbách, oblíbenosti zboží, jeho kapacitě, o objednávkách, kdo ji vyřizoval atd.

2. Datový model (ERD)



ER diagram ukazuje jednotlivé entity pro internetový obchod. Každá entita ma svůj unikátní klíč - ID. Centrální entita je “User” od kterého dědí entita “Employee”, od kterého už dědí “Admin”. Každý usermá jednou nebo několik rolí. Pro nákup user povinen zadat svoji adresu pro zúčtování. “User” muže zadat i víc adres. Nákup je reprezentován entitou “Order”. Každý nákup musí být splácen, a každé placení v systému registrováno entitou “Payment”. Každý nákup musí být vyřízen a odeslán zaměstnancem obchodu. Každý user má svůj košík pro nákupy, ve kterém uložené jednotlivé produkty(M:M “Product”-“CartItem”-“User”), každý produkt má svoji jednou kategorii. Na každý produkt uživatel rozhraní muže napsat review.

3. Model případů užití

Diagram obsahuje čtyři aktéry:

Admin má právo vidět informace o všech tržbách, oblíbenosti zboží, jeho kapacitě,o objednávkách a kdo ji vyřizoval. Zaměstnanec má právo spravovat objednávky: zpracovávat je, vyzvedávat a odesílatna zadanou adresu.
Registrovaný uživatel "Registered User" má možnost modifikovat osobní informace a provést objednávku (objednat produkty). Případ použití "Order products" je zobecnění, které zahrnuje případy použití "Entering billingadress" a "Entering contact".
Případ použití "Entering billingadress" byl tak důležitý, že běžel současně se šesti závislými případy použití, což představuje vazba
<<include>>.
Každý uživatel „User“ má možnost prohlédnout si celý sortiment obchodu a vložit do košíku produkt, který se mu líbí, a také se v případě potřeby přihlásit a zaregistrovat.
Pro případy použití "Modify personal information", "Register" a "Login to the system" je však povinné zahrnout (<<include>>) zadání přihlašovacího jména a hesla.

4. Triggery

V tomto případě dva spouštěče kontrolují roli uživatele, který provádí změny v tabulce "Order" a vkládá nové záznamy do tabulky "review".

Popis triggerů:

Trigger „check_processed_by_employee“:
Tento spouštěč se spustí po aktualizaci sloupce "processed_by" v tabulce "Order". Kontroluje, že objednávky může zpracovávat pouze uživatel s rolí „Employee“. Pokud uživatel, který aktualizoval sloupec „processed_by“, nemá roli „Employee“, aktivační událost selže a aktualizace bude zamítnuta.

Trigger "check_review_user_role":
Tato aktivační událost se spustí před vložením nového záznamu do tabulky "review". Kontroluje, že recenze může vytvářet pouze uživatel s rolí „User“. Pokud uživatel, který se pokouší vložit nový záznam, nemá roli "User", spouštěč selže a vložení bude odmítnuto.

5. Procedury

Procedury určené pro práci s daty o objednávkách v databázi.

PRINT_ORDERS:
Procedura PRINT_ORDERS vytiskne informace o všech objednávkách v databázi, včetně jména zákazníka, celkové hodnoty objednávky a stavu objednávky. Používá kurzor k iteraci přes tabulku Order a proměnnou typu Order%ROWTYPE k uložení aktuálního řádku.

Proměnné
order_cursor - kurzor používaný k načítání řádků z tabulky Order.
order_row je proměnná typu Order%ROWTYPE, která se používá k uložení aktuálního řádku z tabulky Order.
user_name je proměnná typu "User".name%TYPE, která se používá k uložení jména uživatele, který provedl objednávku.
cena - proměnná typu Payment.total_price%TYPE, slouží k uložení celkových nákladů na objednávku.

Výsledek provedení
Procedura zobrazuje informace o každé objednávce na obrazovce pomocí funkce DBMS_OUTPUT.PUT_LINE.

UPDATE_ORGER_STATUS
Procedura update_order_status slouží k aktualizaci stavu objednávky v tabulce Objednávka na základě daného ID objednávky a nového stavu. Pomocí kurzoru načte konkrétní objednávku a zpracuje výjimku, pokud objednávka s daným ID neexistuje.

Možnosti

p_order_id je celočíselný parametr obsahující ID objednávky, jejíž stav je třeba aktualizovat. p_new_status je řetězcový parametr obsahující stav nové objednávky.

Výsledek provedení
Postup aktualizuje stav objednávky v tabulce Objednávka a v případě úspěšné operace zobrazí na obrazovce zprávu "Stav objednávky byl úspěšně aktualizován". Pokud objednávka s daným ID neexistuje, procedura zobrazí chybové hlášení "Chyba: Objednávka s ID {id} neexistuje". Pokud dojde k jakékoli jiné chybě, procedura vytiskne chybovou zprávu obsahující informace o chybě SQL.

6. Explain plan

SQL dotaz, který agreguje (sčítá) náklady na všechny objednávky obsahující pohovky. Dotaz používá dvě tabulky v databázi, "Order" a "OrderItem", a jeden vnořený poddotaz k nalezení ID produktu pro pohovky.

Před dotazem na databázi je zavolán příkaz "EXPLAIN PLAN", aby se získal plán provádění dotazu. Plán ukazuje, jak databáze provede dotaz a jaké indexy (pokud nějaké) budou použity.

Poté se v tabulce "Objednávka" explicitně vytvoří index, který obsahuje sloupce "id" a "celková_cena" použité v dotazu. Tento index pomáhá optimalizovat dotaz a urychlit jeho provádění.

Poté se v dotazu znovu zavolá "EXPLAIN PLAN", aby se zajistilo, že vytvořený index bude použit při provádění dotazu.

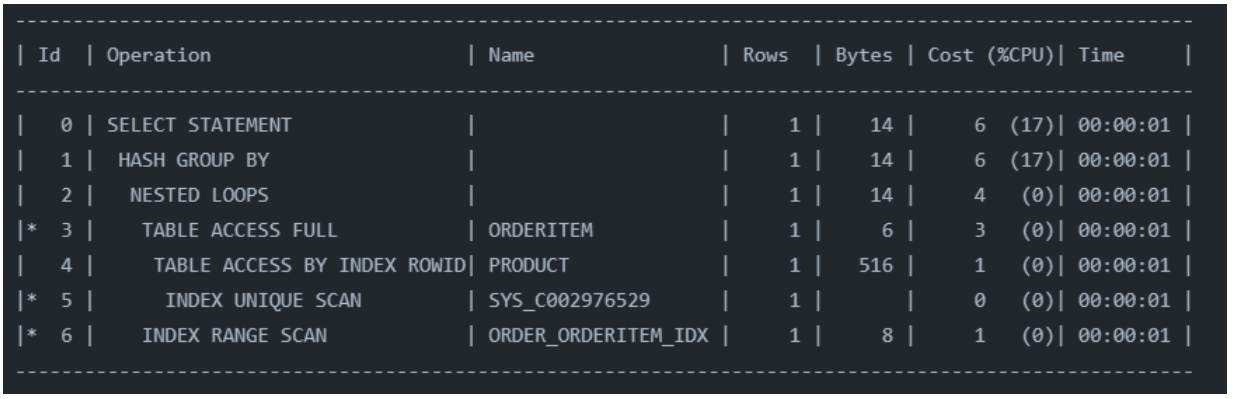
Na konci kódu je samotný SQL dotaz, který vybere součet nákladů všech zakázek obsahujících pohovky. Připojí tabulku "OrderItem" k tabulce "Order", aby přiřadila každou objednávku k jejím položkám objednávky, a poté filtruje pouze ty řádky, které obsahují pohovky. V důsledku toho jsou objednávky seskupeny podle jejich ID a vypočítá se součet jejich nákladů.

Explain plan byl proveden dvakrát nad stejným dotazem. První tabulka ukazuje provedení bez využití indexu, druhá s jeho použitím.

Before:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	14	6 (17)	00:00:01
1	HASH GROUP BY		1	14	6 (17)	00:00:01
2	NESTED LOOPS		1	14	4 (0)	00:00:01
3	NESTED LOOPS		1	14	4 (0)	00:00:01
* 4	TABLE ACCESS FULL	ORDERITEM	1	6	3 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	PRODUCT	1	516	1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	SYS_C002976529	1		0 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	Order_pk	1		0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	Order	1	8	1 (0)	00:00:01

After:



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	14	6 (17)	00:00:01
1	HASH GROUP BY		1	14	6 (17)	00:00:01
2	NESTED LOOPS		1	14	4 (0)	00:00:01
* 3	TABLE ACCESS FULL	ORDERITEM	1	6	3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCT	1	516	1 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	SYS_C002976529	1		0 (0)	00:00:01
* 6	INDEX RANGE SCAN	ORDER_ORDERITEM_IDX	1	8	1 (0)	00:00:01

7. Udělení práv

Uživatel xgarip00 bude mít plný přístup k těmto tabulkám a bude moci provádět datové operace, jako je zobrazení, vkládání, aktualizace a mazání. Tyto příkazy mohou správci databází použít ke kontrole přístupu k datům v databázi a zajištění bezpečnosti dat.

8. Materializovaný pohled

CREATE MATERIALIZED VIEW xgarip00_employees_names – Vytvoří nový materializovaný pohled s názvem „xgarip00_employees_names“.

BUILD IMMEDIATE – Způsobí, že se zhmotněný pohled vytvoří ihned po vytvoření. To znamená, že bude okamžitě naplněn daty a bude k dispozici k použití.

REFRESH COMPLETE ON COMMIT – materializovaný pohled je kompletně aktualizován při každé transakci COMMIT. To znamená, že kdykoli se databáze změní, materializovaný pohled bude aktualizován.

AS SELECT U.name, E.user_id – Určuje, jaká data budou uložena v materializovaném pohledu. V tomto případě načítáme uživatelská jména a ID uživatelů z tabulek „User“ a „Employee“.

SELECT * FROM xgarip00_employees_names – Vybere všechny řádky z materializovaného pohledu „xgarip00_employees_names“, aby se ověřilo, že je správně vytvořen a naplněn daty.

9. Komplexní dotaz SELECT

SQL dotaz, který používá operátor WITH a operátor CASE k výpočtu celkové ceny každé objednávky, kterou koupil náš zaměstnanec.
V první části kódu se vytváří dočasná tabulka s názvem "order_total", která vypočítává celkovou cenu každé objednávky spojením tabulek "Order", "OrderItem" a "Product". Poté se tato dočasná tabulka spojí s tabulkami "User", "Employee" a "Order" a pomocí operátoru CASE se zjišťuje, zda je objednávka zrušena nebo ne. Pokud je objednávka zrušena, vrací se hodnota -1, jinak se vrací celková cena objednávky z tabulky "order_total".
Tento dotaz tedy vrací jména zaměstnanců, jejich pozice a celkovou cenu objednávek, které provedli. Pokud byla objednávka zrušena, pak bude hodnota ve sloupci "total_price" rovna -1.