Politechnika Gdańska
Wydział Elektroniki,
Telekomunikacji i Informatyki
Katedra Architektury Systemów
Komputerowych
Al. Gabriela Narutowicza 11/12 80-952 Gdańsk
tel. (58) 347-12-30 tel./fax (58) 347-28-63

Bezpieczeństwo Systemów Komputerowych

Instrukcja projektowa do projektu 2 – Implementacja mechanizmów kontroli dostępu do baz danych

Opracował: dr inż. Piotr Szpryngier

Gdańsk 2014

Wprowadzenie. Cel zajęć praktycznych.

Wymagania stawiane studentom

- znajomość uruchamiania programów w środowisku WINDOWS i/lub LINUX,
- podstawowa znajomość systemów zarządzania bazami danych MySQL, Postgress, MS-SQL, ewent. Informix, Oracle, DB2 (☺)
- umiejętność programowania C, C++, Java, ewent. .NET,
- umiejętność zaprojektowania i wykonania graficznego interfejsu użytkownika.

Stosowane narzędzia i technologie

- narzędzia programistyczne dostępne w laboratoriach,
- DBMS dostępne w laboratorium oraz w Internecie,
- przeglądarka WWW.

Materiały wprowadzające i pomocnicze

- dostępne materiały wykładowe:
http://www.eti.pg.gda.pl/katedry/kask/pracownicy/Piotr.Szpryngier/
- podręczniki dostępne w czytelni.

Cel zajęć

- jest utrwalenie przez studenta wiedzy związanej z zagadnieniami tworzenia aplikacji baz danych – począwszy od pomysłu bazy danych, poprzez jej model do implementacji w SZBD w postaci kodu SQL (grupa DDL) aż do implementacji interfejsu oraz typowych manipulacji na danych,
- zaprojektowanie i implementacja w tak stworzonej bazie danych wybranego modelu autoryzacji i kontroli dostępu.

Przebieg zajęć.

1. Analiza wymagań postawionych przez prowadzącego zajęcia,
2. Projekt struktury bazy danych,
3. Projekt interfejsu użytkownika,
4. Wybór techniki zarządzania użytkownikami i uprawnieniami,
5. Implementacja aplikacji i testy,
6. Podczas zaliczania przedłożenie raportu końcowego.
7. Zasady zaliczania.

Ad. 1
Celem wyznaczonym studentowi jest zaprojektowanie, implementacja i uruchomienie aplikacji służącej do zarządzania uprawnieniami użytkowników i kontroli dostępu wg wybranego modelu DBMS i sposobu autoryzacji.

Wymagania

- technologia wykonania – dowolna, ale ograniczona możliwościami dostępnego DBMS;
- Baza danych powinna być prosta – kilka, najwyżej kilkanaście relacji, np. wypożyczalnia płyt DVD, gabinet lekarski, ogłoszenia, członkowie stowarzyszeń, itp.
- powinna być zaimplementowana pełna funkcjonalność modelu kontroli dostępu z uwzględnieniem podstawowych operacji SQL na obiektach (tablicach).
- Dostęp do bazy danych powinien być zasadniczo zrealizowany przez przeglądarkę i tunel SSL. Dopuszczalne są także dedykowane aplikacje klienckie.
- W przypadku modelu DAC z delegacją uprawnień należy kontrolować (eliminować) obecność cykli w grafie delegacji uprawnień, a także eliminować możliwość nadawania tego samego uprawnienia do tego samego obiektu przez więcej jak jednego dawcę. Ponadto dla uprawnienia przejmij należy przyjąć, że posiadane dotychczas uprawnienia przejmującego zostają skasowane i zastąpione uprawnieniami przejętymi od dawcy, natomiast dawca jest pozbawiany absolutnie wszystkich posiadanych uprawnień - nawet gdy przekazuje tylko jedno przejmującemu.
- W przypadku modelu MAC (Jajodia-Sadhu) należy pamiętać o zasadach kontroli przepływu danych (zasada: no-read-up, no-write-down).
- W przypadku modelu RBAC należy uwzględnić statyczną separację ról podczas pracy, tzn. w czasie trwania sesji można pełnić tylko jedną rolę (posiadając ich wiele), pomimo otwarcia np. wielu połączeń z bazą danych i z użyciem różnych przeglądarek.

Ad. 2

Projektowanie aplikacji relacyjnej bazy danych wygodnie jest podzielić na cztery podstawowe etapy:

1. Zdefiniowanie problemu – w tym miejscu należy zebrać możliwe jak najwięcej informacji na temat bazy danych, którą chcemy zaprojektować. Jeśli to jest wymagane, należy zgromadzić wymagania na warunki pracy aplikacji oraz założenia jej towarzyszące. W ramach projektu każda grupa studentów będzie miała odrębny problem do analizy i opracowania.
2. Zamodelowanie rozwiązania problemu przy użyciu diagramów związków encji. Wynikiem działania powinien być diagram uwzględniający wszystkie relacje w bazie danych oraz powiązania pomiędzy nimi. Do tego celu należy wykorzystać narzędzia graficzne do zobrazowania wyników pracy analitycznej. Na tym etapie należy podjąć równolegle prace nad postacią interfejsu użytkownika.
3. Stworzenie diagramu relacyjnego, wygenerowanie kodu zakładającego poszczególne tabele, wprowadzenie danych inicjalnych (zapełnienie bazy danych) w dość dużej ilości, ocena spójności. Rejestracja użytkowników z różnymi uprawnieniami.
4. Ostatnim etapem tworzenia aplikacji bazy danych jest zaprojektowanie i skonstruowanie ewentualnych (niekoniecznie) zapytań biznesowych i manipulacji na danych, a następnie przeprowadzenie testów aplikacji z uwzględnieniem kontroli poprawności z punktu widzenia posiadanych przez użytkownika uprawnień.

Przykładowa baza danych:

Hurtownia (bez określenia branży). Zakupy dokonuje się u dostawcy – też firmy. Jeden zakup może się składać z wielu pozycji, każda z nich ma swoją cenę zakupu i ilość lub liczbę sztuk danego asortymentu. Podobnie wygląda sprzedaż – klientem hurtowni jest firma. Opis towaru (cechy organoleptyczne, jednostki miary, typ opakowania, kod SWW, stawka VAT, itp.) są podane w odrębnej encji. Schemat został znormalizowany. Na pozycji magazynowej znajdują się odpowiednio w relacji 1:1 towary zakupione w danej pozycji zakupu. Dlaczego? Dlaczego nie można umieścić w jednej encji zakupów i sprzedaży, klientów i dostawców?

Kod SQL utworzenia przykładowej bazy danych:

```
create database hurtownia_nazwa;

use hurtownia_nazwa;

CREATE TABLE dostawcy (
  nip_dostawcy VARCHAR(10) NOT NULL,
  nazwa VARCHAR(30) NOT NULL,
  telefon VARCHAR(9) NOT NULL,
  ulica VARCHAR(30) NOT NULL,
  nr_domu VARCHAR(10) NOT NULL,
  nr_lokalu INT NOT NULL,
  kod_pocztowy VARCHAR(6) NOT NULL,
  miejscowosc VARCHAR(30) NOT NULL,
  PRIMARY KEY(nip_dostawcy)
```

```
);

CREATE TABLE zakup (
  id_zakupu auto_increment INT NOT NULL,
  data DATE NOT NULL,
  nip_dostawcy VARCHAR(10) NOT NULL,
  nr_faktury VARCHAR(50),
  foreign key (nip_dostawcy) references dostawcy(nip_dostawcy)
  PRIMARY KEY (id_zakupu)
);

CREATE TABLE poz_zakupu (
  id_poz_zakupu INT auto_increment NOT NULL,
  id_zakupu INT NOT NULL,
  cena_zakupu DOUBLE NOT NULL,
  ilosc INT NOT NULL,
  PRIMARY KEY(id_poz_zakupu),
  foreign key (id_zakupu) references zakup(id_zakupu)
);

CREATE TABLE opis_towaru (
  id_opisu INT auto_increment NOT NULL,
  kod_sww INT NOT NULL,
  nr_ident INT NOT NULL,
  jedn_miary VARCHAR(20) NOT NULL,
  kategoria VARCHAR(30) NOT NULL,
  nazwa_producenta VARCHAR(30),
  nazwa_towaru VARCHAR(30) NOT NULL,
  opis_slowny TEXT NOT NULL,
  stawka_VAT DOUBLE
  PRIMARY KEY(id_opisu)
);

CREATE TABLE magazyn (
  id_poz_mag INT auto_increment NOT NULL,
  id_opisu INT NOT NULL,
  cena_sprzedazy DOUBLE NOT NULL,
  lokalizacja VARCHAR(50) NOT NULL,
  id_poz_zakupu INT NOT NULL,
  ile INTEGER UNSIGNED NULL,
  PRIMARY KEY(id_poz_mag),
  foreign key (id_opisu) references opis_towaru(id_opisu),
  foreign key (id_poz_zakupu) references poz_zakupu(id_poz_zakupu)
);

CREATE TABLE klienci (
  nip_klienta VARCHAR(10) NOT NULL,
  nazwa VARCHAR(50) NOT NULL,
  ulica VARCHAR(50) NOT NULL,
  nr_domu VARCHAR(10) NOT NULL,
```

```
  nr_lokalu INT NOT NULL,
  kod_pocztowy VARCHAR(6) NOT NULL,
  miasto VARCHAR(50) NOT NULL,
  telefon VARCHAR(9) NOT NULL,
  rabat INT NOT NULL,
  PRIMARY KEY(nip_klienta)
);

CREATE TABLE sprzedaz (
  nr_faktury char(30) NOT NULL,
  nip_klienta VARCHAR(10) NOT NULL,
  data DATE NOT NULL,
  PRIMARY KEY(nr_faktury),
  foreign key (nip_klienta) references klienci(nip_klienta)
);

CREATE TABLE lista_poz_sprzedazy (
  id_poz_sprzedazy INT auto_increment NOT NULL,
  nr_faktury char(30) NOT NULL,
  id_poz_mag INT NOT NULL,
  ile INT NOT NULL,
  rabat INT NOT NULL,
  PRIMARY KEY(id_poz_sprzedazy),
  foreign key (nr_faktury) references sprzedaz(nr_faktury),
  foreign key (id_poz_mag) references magazyn(id_poz_mag)
);
```

Powyższy schemat można adoptować na usługi (kupujemy pracę wykonawcy usługi, sprzedajemy usługi i materiały służące do jej realizacji.

Ad. 3

Studenci powinni zaprojektować interfejs użytkownika, uwzględniając wymagania podane w p. 1. Technika wykonania – dowolna.

Ad. 4

Wybór metody kontroli dostępu jest narzucony warunkami projektu. Grupa projektowa powinna zaimplementować pełną funkcjonalność tego modelu.

Ad. 5

Implementacja aplikacji powinna być zrealizowana w środowisku najbardziej przyjaznym dla studentów. Prowadzący zajęcia nie narzuca wyboru narzędzi do realizacji projektu, aczkolwiek należy tu uwzględnić sposób implementacji dostępnej wersji DBMS. Ponadto grupa studentów powinna przeprowadzić testy aplikacji przed jej przekazaniem do oceny.

Ad. 6

Krótki raport powinien zawierać informacje o temacie zadania, kształcie interfejsu użytkownika, strukturze bazy danych oraz wynikach testów z badania skuteczności kontroli dostępu.

Zasady oceniania.

Zadanie zostanie ocenione wg następujących zasad:
- dokumentacja – do 5 pkt.
- funkcjonalność interfejsu użytkownika – do 10 pkt.
- poprawna realizacja tematu w terminie – do 25 pkt.

I termin zaliczenia projektu – 19-23 maja 2014 w godzinach i salach zgodnych z planem zajęć dla każdej grupy oraz w godzinach konsultacji.
.
Za każdy tydzień opóźnienia od tej daty (19-23.05.2014) **końcowa ocena** będzie pomniejszana o 5pkt.
Ostateczny termin zaliczeń – 16 czerwca 2014. Po tym terminie projekty nie będą oceniane z wyjątkiem usprawiedliwionych sytuacji szczególnych (zwolnienia lekarskie, przypadki losowe).