Rozdział

Elastyczny system kontroli uprawnień użytkowników bazy danych

Mirosław Makaroś
merlin.com.pl Sp. z. o.o., Miroslaw.Makaros@sen-sys.pl

Krzysztof Stencel
Instytut Informatyki Uniwersytetu Warszawskiego, stencel@mimuw.edu.pl

Streszczenie

W niniejszej pracy przedstawiono mechanizm kontroli uprawnień użytkowników relacyjnej bazy danych, który umożliwia określanie praw dostępu z dokładnością do pojedynczych wartości przechowywanych w tabelach. Przy realizacji tego systemu wykorzystano między innymi wyzwalacze INSTEAD na perspektywach. System stanowi ciekawe zastosowanie tego rodzaju wyzwalaczy.

Opisywany system kontroli uprawnień powstał w ramach Uniwersyteckiego Systemu Obsługi Studiów. Jest jednak od niego niezależny i może być zastosowany dla dowolnego schematu bazy danych.

1. Wstęp

Użytkownicy bazy danych nie mogą cieszyć się pełnią praw dostępu do danych w bazie danych. Dzieje się tak z wielu względów; są nimi m.in. ochrona poufności danych oraz ochrona danych przed świadomym i nieświadomym uszkodzeniem. Sercem systemu informacyjnego jest baza danych. Inżynierowie oprogramowania muszą więc rozwiązywać problemy związane z kontrolą praw dostępu użytkowników bazy danych. Nie inaczej było w wypadku Uniwersyteckiego Systemu Obsługi Studiów (USOS) [MINC2002, KORU2000]. USOS to ogólnouniwersytecki system do obsługi spraw studiów zbudowany przez pracowników i studentów Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. USOS obejmuje sprawy całej uczelni, dlatego niezbędny jest system kontroli uprawnień, choćby po to, żeby poszczególne wydziały nie wykonywały sprzecznych operacji na tych samych danych.

Niestety na początku USOS był małym systemem jedynie dla Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego i nie projektowano go z myślą o konieczności kontroli praw dostępu użytkowników. Odpowiednie udogodnienia trzeba było wprowadzać do już istniejącego działającego oprogramowania. Opracowanie tych udogodnień musiało więc odbywać się przy uwzględnieniu pewnych założeń, które umożliwiły minimalizację niezbędnych zmian w istniejącym kodzie.

Od tamtej chwili zbudowano już jeden zintegrowany interfejs dla wszystkich użytkowników. Ów interfejs napisany w Oracle*Forms odwoływał się do konkretnych nazw obiektów bazy danych. Uniknięcie konieczności poważnych zmian wymagało takiego opracowania systemu kontroli uprawnień, żeby istniejący interfejs użytkownika mógł odwoływać się do tych samych nazw obiektów. Osiągnięto to poprzez synonimy i perspektywy. Oznaczało to również, że system ten musiał być zbudowany za pomocą mechanizmów baz danych, a nie oprogramowania klienckiego. Będzie o tym jeszcze mowa w dalszej części pracy. Oczywiście system kontroli uprawnień należało zbudować w oparciu o mechanizmy SZBD Oracle, na którym zbudowano bazę danych USOS.

Zaprogramowanie systemu kontroli uprawnień przy takich założeniach nie było banalnym zadaniem. W rezultacie otrzymano bardzo elastyczne oprogramowanie umożliwiające definiowanie uprawnień z dokładnością do jednego wiersza i jednej kolumny. Przy budowie tego systemu intensywnie korzystano z możliwości, jakie daje nowoczesna relacyjna baza danych, tzn. z aktualizowalnych perspektyw z wyzwalaczami INSTEAD. Powstałe rozwiązanie jest bardzo interesującym zastosowaniem tego mechanizmu. Należy też zwrócić uwagę, że nie jest ono uzależnione od USOS i może być zastosowane z dowolnym schematem bazy danych.

Rozwiązanie przedstawione w tej pracy jest napisane dla SZBD Oracle. Niestety standard SQL [ISO1992, DADA2000] nie obejmuje zagadnień takich jak wyzwalacze, więc nie można było opracować analogicznego przenośnego rozwiązania, które działałoby z dowolną bazą danych zgodną ze standardem SQL. Omawiany system kontroli uprawnień można jednak przy pewnej dozie wysiłku dostosować do każdego SZBD, który ma mechanizm perspektyw i wyzwalaczy INSTEAD OF.

Chociaż opisywany system powstał w wyniku niewłaściwego postępowania tzn. wstępnego pominięcia zagadnienia kontroli uprawnień, jednak wyniki tych prac pozwalają także na realizację scenariusza pozytywnego — programista aplikacyjny tworzy aplikację dla użytkownika, który może wszystko. Nie dba wcale o uprawnienia a już w żadnym przypadku nie uwzględnia ich w kodzie. Może tak postępować ponieważ wie, że przezroczysty system kontroli uprawnień „dostosuje" kod do rzeczywistych uprawnień danego użytkownika. Jest to o tyle istotne, że kontrola uprawnień dotyczy każdego elementu każdego komercyjnego oprogramowania aplikacyjnego. Powstaje w ten sposób wzorzec projektowy do powszechnego użycia.

Standard SQL [ISO1992, DADA2000] obejmuje pewne mechanizmy do kontroli uprawnień użytkowników (role), jednak są one niewystarczające. Za pomocą poleceń GRANT można nadać prawa do całej tabeli albo perspektywy lub poszczególnych kolumn, ale nie wierszy. Można oczywiście powiedzieć, że rozwiązaniem są perspektywy (każdemu użytkownikowi dajemy perspektywę z danymi, do których ma on prawo) i na tym skończyć rozważania. USOS ma jednak setkę użytkowników i około dwie setki

tabel, co daje 20.000 takich perspektyw. Zarządzanie takim zasobem perspektyw nie jest łatwe. Opisywany w niniejszej pracy system kontroli uprawnień użytkowników umożliwia wygodne tworzenie tych perspektyw i administrowanie nimi.

Szczegółowe informacje o opisywanym rozwiązaniu można znaleźć w [MAKA2002], a sam kod jest dostępny w dystrybucji systemy USOS w katalogu *Role* [USOS2002]. Pod tymże adresem URL można też znaleźć informacje o licencji.

2. Możliwe rozwiązania

Zakładamy, że użytkownicy otrzymują uprawnienia za pośrednictwem *ról* SQL. Rola to w istocie grupa uprawnień, które można łącznie nadać użytkownikowi. Uprawnienia użytkownika, któremu nadano kilka ról, są sumą uprawnień przypisanych tym rolom.

Najpierw należy utworzyć rolę z odpowiednimi uprawnieniami, a następnie nadać ją poszczególnym użytkownikom. To umożliwia wielokrotne wykorzystanie tej samej definicji. Uprawnienia użytkowników są definiowane poprzez warunki wierszowe (wyrażenia logiczne ograniczające zestaw wierszy) i listy udostępnionych kolumn.

Istnieje kilka możliwych sposobów implementacji. Ze względu na sposób kontroli uprawnień można je podzielić na dwie grupy:

- uprawnienia są sprawdzane przez formularze (raporty), które wyświetlają tylko te informacje, do których dany użytkownik ma uprawnienia; narzędzia te sprawdzają poziom uprawnień przy wykonywaniu operacji UPDATE, INSERT i DELETE;
- uprawnienia są sprawdzane po stronie bazy danych.

W dalszej części przedstawiono kilka możliwych sposobów implementacji, które spełniają podane wymagania.

2.1 Kontrola w formularzach

Pomysł ten należy do grupy pierwszej, a więc uprawnienia sprawdzane są tu tylko i wyłącznie poprzez formularze. Formularze podczas uruchomienia, w zależności od roli, w jakiej dany użytkownik pracuje, uwzględniają:

- uprawnienia kolumnowe – blokują odpowiednie pola formularza przed modyfikacją, nie wyświetlają pól, których użytkownik nie ma prawa odczytać,
- uprawnienia wierszowe – dołączają odpowiednią klauzulę WHERE do bloków danych w formularzach (prawo wykonywania SELECT); wszystkie operacje modyfikacji są przysłonięte wyzwalaczami, które po sprawdzeniu, czy użytkownik ma do nich prawo, wykonują je lub też wyświetlają komunikat o braku uprawnień.

Rozwiązanie to ma niestety kilka poważnych wad:

- Istnieje możliwość obejścia zabezpieczenia przy użyciu innego interfejsu niż właściwe formularze.

- Powstaje problem z blokami formularzy, które są oparte na zapytaniach, a nie na tabeli. Złożenie klauzuli WHERE odpowiedniej dla bloku opartego na zapytaniu na podstawie warunków zdefiniowanych dla poszczególnych tabel może okazać się niemożliwe.
- Powstaje podejrzenie, że takie rozwiązanie wymagałoby dużych zmian w każdym z formularzy. Ze względu na dużą różnorodność i stopień komplikacji wielu formularzy, może okazać się, że nie uda się napisać na tyle ogólnych mechanizmów, które kontrolowałyby uprawnienia we wszystkich formularzach bez specjalnego ich przystosowywania.

2.2 Tabele towarzyszące

Dla każdej tabeli definiujemy perspektywę, która zależy od wyniku wywołania funkcji USER (jej wynikiem jest identyfikator bieżącego użytkownika) i dzięki temu poszczególnym użytkownikom wyświetla tylko te dane, do których użytkownik ten ma uprawnienia. Operacje DELETE, UPDATE i INSERT są przysłonięte wyzwalaczami INSTEAD, które po sprawdzeniu uprawnień (w zależności od wyniku funkcji USER) wykonują właściwą operację. W rozwiązaniu tym zakłada się, że każdej tabeli odpowiada *tabela towarzysząca* określająca, czy użytkownik ma określone prawo do każdego wiersza z właściwej tabeli. Tabela ta jest uzupełniana w momencie ustalania uprawnień dla danej roli. Następnie dla każdej tabeli tworzy się perspektywę, która jest złączeniem tabeli głównej i tabeli towarzyszącej z odpowiednim warunkiem selekcji z wywołaniem funkcji USER. Użytkownik dostaje uprawnienia (SELECT, INSERT, UPDATE, DELETE) do tej perspektywy i odpowiedni synonim (żeby interfejs użytkownika mógł korzystać z tych samych nazw tabel dla każdego użytkownika).

Do wad tego rozwiązania należy:

- Podejrzenie o niewystarczającą wydajność (tabele towarzyszące mogą mieć dużo wierszy, co spowolni wykonywanie wszelkich operacji).
- Brak sposobu, aby danemu użytkownikowi nie dawać prawa SELECT do wybranej kolumny– to mógłby robić tylko formularz, np. ukrywając dany element bloku, jednak z innego interfejsu użytkownik zobaczy wszystkie kolumny. Definiując perspektywę zależną od USER możemy tylko kontrolować wiersze, a nie kolumny.
- Przy każdej modyfikacji właściwej tabeli należałoby przetwarzać również pomocniczą tabelę.
- Powyższa wada uwidacznia się jeszcze bardziej, kiedy w warunku wierszowym dla tabeli występuje podzapytanie. Wtedy przy zmianie zawartości tabel z tego podzapytania należałoby również przeliczać zawartość tabel towarzyszących. Potrzebne są dodatkowe struktury, aby przechowywać informację o tym które tabele towarzyszące należy aktualizować po modyfikacji zawarto¶ci określonych tabel.
- Przy modyfikacji dowolnej tabeli nie wiadomo, w których warunkach wierszowych dla tabeli występuje ona w podzapytaniu, więc nie wiadomo, które pomocnicze tabele należy przeliczyć.

Zaletami tego rozwiązania są:

- Wymaga ono minimalnych zmian w formularzach.
- Cały system uprawnień zdefiniowano w bazie, co uniezależnia go od interfejsu, tzn. niezależnie od tego czy użytkownik używa właściwych formularzy, czy np. SQL*PLUS, nie uda mu się wykonać operacji, do których nie ma uprawnień.
- Warunki wierszowe jakie ograniczają uprawnienia użytkownika, są uwzględnione podczas inicjowania tabel towarzyszących. Użytkownik odwołując się do tabeli korzysta z tych tabel, co może okazać się szybsze niż metoda, w której dopiero w momencie odwołania użytkownika oblicza się poszczególne warunki.

2.3 Perspektywy

Każda rola ma swoje perspektywy, które przesłaniają wszystkie tabele systemu. Dla każdej roli i każdej tabeli definiuje się odpowiednią perspektywę uwzględniającą wszystkie uprawnienia danej roli do danej tabeli. Perspektywa taka posiada klauzulę WHERE złożoną z warunków wierszowych określonych dla tabeli i jej kolumn. Uprawnienia DELETE, UPDATE i INSERT są sprawdzane przez wyzwalacze INSTEAD na tych perspektywach. Wyzwalacze te po sprawdzeniu uprawnień wykonają bądź nie odpowiednie operacje.

Każdy użytkownik ma uprawnienia SELECT, INSERT, UPDATE, DELETE do perspektyw, które są w rolach, do których należy. Każdy użytkownik ma swoje synonimy, które odnoszą się do odpowiednich perspektyw, a nazwy synonimów są takie same jak nazwy tabel, na jakich oparte są perspektywy.

Zalety:

- Cały system uprawnień jest zdefiniowany w bazie danych, co uniezależnia go od interfejsu, z którego korzysta użytkownik.
- Łatwo uwzględnić brak prawa odczytu wybranej kolumny (w przeciwieństwie do p. 2.2) – perspektywa zawsze wstawia NULL w takiej kolumnie.
- Rozwiązanie wymaga minimalnych zmian w formularzach.

Wady:

- Przy bezpośrednim zastosowaniu tego pomysłu powstanie dużo perspektyw.

2.4 Rozwiązanie mieszane

To połączenie p. 2.2 i 2.3. Do niektórych tabel stosujemy podejście z p. 2.2, a do niektórych z p. 2.3. Na przykład w przypadku tabel, w których mamy bardzo skomplikowane warunki wierszowe lepsze może okazać się rozwiązanie z p. 2.2.

Zalety:

- Optymalny, jeśli chodzi o szybkość wykonywania (dla operacji SELECT pomysł z p. 2.2 może być szybszy, gdy dla tabeli są określone złożone warunki wierszowe).

Wady:

- W znacznym stopniu komplikuje cały mechanizm, w porównaniu z rozwiązaniami opartymi tylko na p. 2.2 i 2.3, powoduje też, że cały system nie jest jednorodny.

2.5 Analiza

Podstawową wadą pomysłu z p. 2.1 i całej grupy pomysłów, w których za kontrolę uprawnień odpowiedzialne są formularze jest to, że zabezpieczenia można obejść przy pomocy innego interfejsu, tak więc pomysł ten należy odrzucić.

Wada pomysłu z p. 2.2 polegająca na konieczności przeliczania tabel pomocniczych przy każdej zmianie w zasadniczej tabeli dyskwalifikuje ten pomysł w porównaniu z pomysłem z p. 2.3. Mamy tu dodatkowy narzut podczas operacji edycyjnych na tabelach, natomiast operacja czytania z tabel może okazać się szybsza tylko w niektórych przypadkach.

Biorąc pod uwagę powyższe porównanie, a także wady i zalety poszczególnych pomysłów, wydaje się, że rozwiązanie z p. 2.3 jest najodpowiedniejsze. Przede wszystkim rozwiązanie to pozwoli zrealizować wymaganie, aby system kontroli uprawnień działał niezależnie od interfejsu.

3. Realizacja

3.1 Zasadnicze właściwości

Głównym zadaniem budowanego systemu jest kontrola uprawnień użytkowników przy wykonywaniu operacji związanych z tabelami i perspektywami *Właściciela tabel* (konto w bazie danych, na którym zainstalowano obiekty, do których dostęp ma być kontrolowany). Ponadto system jest również odpowiedzialny za zarządzanie uprawnieniami użytkowników w dostępie do sekwencji (tzn. generatorów numerycznych identyfikatorów) oraz pakietów *Właściciela tabel*, a także uprawnieniami systemowymi.

SZBD Oracle ma wbudowane mechanizmy pozwalające na zarządzanie uprawnieniami użytkowników w systemie. Mechanizmy, jakie udostępnia SZBD Oracle, wykorzystano w budowanym systemie kontroli uprawnień. Za ich pomocą nie można jednak:

- nadawać uprawnienia SELECT do poszczególnych kolumn tabeli, tzn. takiego, które powoduje, że wykonanie operacji SELECT przekaże wartości tylko w kolumnach, do których użytkownik został uprawniony, w pozostałych kolumnach użytkownik otrzyma wartość NULL,
- nadawać uprawnień z dokładnością do wierszy. Dotyczy to wszystkich operacji: INSERT, UPDATE, DELTE, SELECT. Poszczególne wiersze, do których dana rola ma mieć uprawnienia, specyfikuje się poprzez warunki wierszowe.

W opisywanym systemie kontroli uprawnień poszczególne obiekty służące do kontroli uprawnień (perspektywy i wyzwalacze INSTEAD) są tworzone na specjalnym koncie *Administratora ról*, który ma uprawnienia do wszystkich obiektów należących do *Właściciela tabel* z prawem ich dalszego przekazywania. Na kontach użytkowników instaluje się jedynie synonimy (głównie do perspektyw *Administratora ról*).

Dla *Administratora ról* opracowano graficzny interfejs użytkownika, za którego pomocą definiuje on uprawnienia poszczególnych ról i nadaje te role użytkownikom. Wszystkie niezbędne obiekty (perspektywy, wyzwalacze i synonimy) są wówczas automatycznie tworzone na koncie *Administratora ról* i na kontach użytkowników.

Użytkownikowi można nadać więcej niż jedną rolę, jednak w jednej chwili może on korzystać z praw tylko jednej roli. Tę rolę nazywamy *domyślną*. Synonimy na koncie użytkownika wskazują właśnie jego rolę domyślną.

3.2 Kontrola uprawnień do tabel i perspektyw

Kontrola uprawnień poszczególnych użytkowników w dostępie do tabel i perspektyw *Właściciela tabel* jest podstawowym zadaniem budowanego systemu. Ponieważ konieczne jest nadawanie uprawnień z dokładnością do poszczególnych wierszy, a także określania uprawnienia SELECT do wybranych kolumn tabel lub perspektyw, nie można wykorzystać mechanizmów Oracle w bezpośredni sposób. Założeniem projektu jest zdefiniowanie uprawnień po stronie bazy danych. Użytkownik odwołuje się do wybranej tabeli, ale w wyniku dostanie tylko wiersze i kolumny, do których ma uprawnienia (w przypadku operacji SELECT), lub też operacja, jaką wykonuje powiedzie się, gdy ma uprawnienia do wszystkich wierszy, których operacja ta dotyczy (w przypadku operacji INSERT, UPDATE, DELETE).

Podstawową ideą kontroli uprawnień przy dostępie do tabel i perspektyw *Właściciela tabel* jest pomysł odwoływania się do nich poprzez dodatkowe, pośrednie perspektywy. Te perspektywy są instalowane na koncie *Administratora ról*. Za kontrolę uprawnień jest odpowiedzialna odpowiednia konstrukcja perspektywy, konstrukcja wyzwalaczy INSTEAD utworzonych na perspektywie oraz odpowiednie przekazanie roli uprawnień do perspektywy, a także odpowiednie zdefiniowanie użytkownikowi synonimów. Perspektywy takie są tworzone dla każdej roli i tabeli (jeżeli rola ta ma zdefiniowane jakiekolwiek uprawnienia do tej tabeli). Użytkownik ma synonimy do wszystkich perspektyw swej roli, dzięki czemu odwołując się do nazwy tabeli *Właściciela tabel* odwołuje się przez odpowiednią perspektywę.

Jeżeli danej roli nadano uprawnienie SELECT do całej tabeli, to jest tworzona perspektywa wyświetlająca wszystkie kolumny i roli nadaje się uprawnienie SELECT do tej perspektywy.

Jeżeli danej roli nadano uprawnienie SELECT tylko do niektórych, wybranych kolumn tabeli, to jest utworzona perspektywa wyświetlająca tylko te wybrane kolumny, roli nadaje się uprawnienie SELECT do tej perspektywy (nie można nadać roli uprawnienia SELECT tylko do niektórych kolumn, dlatego za kontrolę uprawnień odpowiedzialna jest w tym przypadku konstrukcja perspektywy).

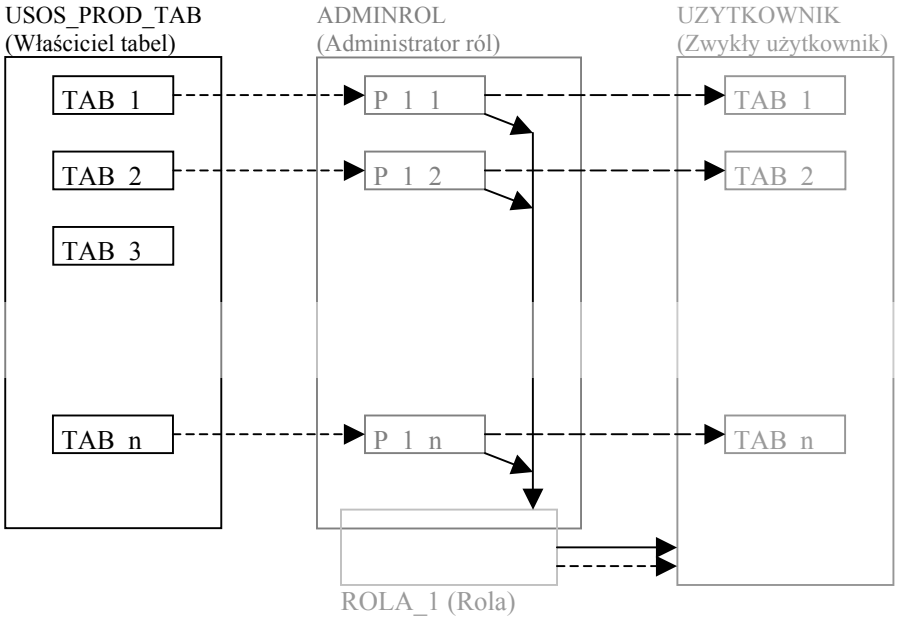
Jeżeli danej roli nadano uprawnienie INSERT, UPDATE lub DELETE do całej tabeli, to jest utworzona perspektywa (to, które kolumny są w niej widoczne, zależy tylko od uprawnienia SELECT; w szczególności, jeżeli do tej tabeli nie jest nadane uprawnienie SELECT, to perspektywa ma wartości NULL we wszystkich kolumnach), roli nadaje się odpowiednie uprawnienie (INSERT, UPDATE, DELETE) do tej perspektywy.

Jeżeli danej roli nadano uprawnienie INSERT, UPDATE do niektórych kolumn tabeli, to roli nadaje się uprawnienie INSERT lub UPDATE tylko do tych kolumn perspektywy (wykorzystuje się tu mechanizmy Oracle, które pozwalają na nadawanie uprawnienia INSERT lub UPDATE tylko do niektórych kolumn).

Jeżeli uprawnienie SELECT jest ograniczone przez warunki wierszowe, to warunki te uwzględnione są w klauzuli WHERE perspektywy w ten sposób, że perspektywa wyświetla tylko te wiersze, które spełniają warunki wierszowe (za kontrolę uprawnienia odpowiedzialna jest konstrukcja perspektywy).

Jeżeli uprawnienie INSERT, UPDATE lub DELETE jest ograniczone przez warunki wierszowe, to uprawnienie takie jest egzekwowane przez odpowiednie wyzwalacze INSTEAD utworzone na perspektywie. Wyzwalacze te skonstruowane są w ten sposób, że nie pozwalają na wykonanie operacji, gdy nie są spełnione odpowiednie warunki wierszowe związane z tą operacją.

Na rysunku 1 przedstawiono model zależności między poszczególnymi tabelami i perspektywami *Właściciela tabel*, perspektywami *Administratora ról* oraz synonimami użytkownika.



Rys. 1. Kontrola uprawnień do tabel i perspektyw

USOS_PROD_TAB jest *Właścicielem tabel*, natomiast TAB_1, TAB_2, TAB_3, TAB_n (znajdujące się wewnątrz USOS_PROD_TAB) to tabele i perspektywy *Właściciela tabel*.

ADMINROL jest *Administratorem ról*, natomiast P_1_1, P_1_2, P_1_n to perspektywy *Administratora ról*. UZYTKOWNIK to zwykły użytkownik, natomiast TAB_1, TAB_2, TAB_n (znajdujące się wewnątrz UZYTKOWNIK) to prywatne synonimy użytkownika. ROLA_1 jest rolą zdefiniowaną przez *Administratora ról.* Strzałki przerywane między tabelami i perspektywami *Właściciela tabel* i perspektywami *Administratora ról* pokazują, na których tabelach i perspektywach *Właściciela tabel* oparte są poszczególne perspektywy *Administratora ról*. Strzałki przerywane między perspektywami *Administratora ról* i synonimami użytkownika pokazują, do których perspektyw *Administratora ról* odnoszą się synonimy użytkownika. Strzałki ciągłe reprezentują nadanie uprawnień, uprawnienia do poszczególnych perspektyw *Administratora ról* nadane są roli ROLA_1, natomiast ROLA_1 nadana jest użytkownikowi UZYTKOWNIK. Strzałka przerywana pomiędzy rolą i użytkownikiem oznacza, że ta rola jest jego rolą domyślną.

W przedstawionej sytuacji roli ROLA_1 nadano pewne uprawnienia do tabel *Właściciela tabel* TAB_1, TAB_2 i TAB_n. Utworzone zostały odpowiednie perspektywy *Administratora Ról* P_1_1, P_1_2, P_1_n oparte na tych tabelach. Uprawnienia do perspektyw P_1_1, P_1_2, P_1_n nadano roli ROLA_1. Uprawnienia roli do poszczególnych tabel TAB_1, TAB_2, TAB_n są egzekwowane przez odpowiednią konstrukcję perspektyw P_1_1, P_1_2, P_1_n (i ewentualnie wyzwalaczy na tych perspektywach) oraz odpowiednie nadanie roli ROLA_1 uprawnień do tych perspektyw. W omawianej sytuacji użytkownikowi UZYTKOWNIK nadano rolę ROLA_1 i rola ta jest jego rolą domyślną (obrazują to dwie strzałki od ROLA_1 do UZYTKOWNIK). Ponieważ rola ROLA_1 jest rolą domyślną dla użytkownika UZYTKOWNIK, użytkownik ten ma zdefiniowane odpowiednie synonimy prywatne (TAB_1, TAB_2, TAB_n). Synonimy te odwołują się do perspektyw *Administratora ról*, ale mają nazwy takie same, jak nazwy tabel *Właściciela tabel*. Na przykład użytkownik ma synonim prywatny TAB_1 odnoszący się do perspektywy ADMINROL.P_1_1.

Należy zwrócić uwagę, że w roli ROLA_1 nie ma żadnego uprawnienia do tabeli TAB_3 *Właściciela tabel*, w związku z tym nie została utworzona odpowiednia perspektywa *Administratora ról* oraz nie istnieje odpowiedni synonim prywatny użytkownika.

3.3 Kontrola uprawnień do sekwencji

Uprawnienia do sekwencji nadawane są wszystkim użytkownikom. Pobieranie wartości z sekwencji nie stanowi zagrożenia dla systemu i dlatego nie musi być chronione.

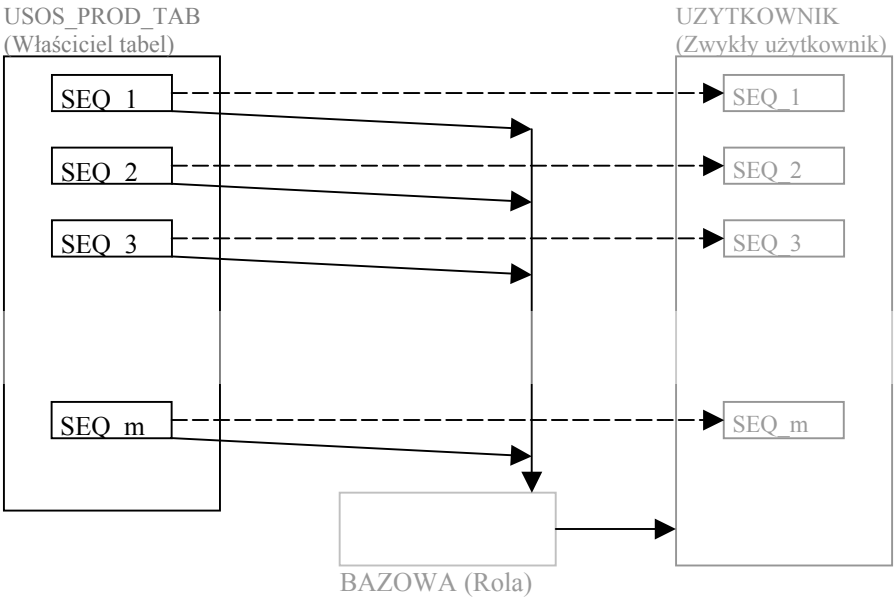
Prawo SELECT do wszystkich sekwencji systemu USOS nadane jest roli BAZOWA (por. p. 3.6). Rolę BAZOWA nadaje się wszystkim użytkownikom. Aby każdy użytkownik mógł korzystać z sekwencji bez poprzedzania jej nazwy nazwą konta *Właściciela tabel*, dla wszystkich użytkowników tworzy się synonimy prywatne odnoszące się do sekwencji *Właściciela tabel* i mające nazwy takie jak nazwy sekwencji.

Administrator ról nie ma wpływu na nadawanie uprawnienia do sekwencji poszczególnym użytkownikom. Rolę BAZOWA, która ma uprawnienia do wszystkich sekwencji, nadaje się automatycznie każdemu nowemu użytkownikowi.

Na rysunku 2 przedstawiono sposób kontroli uprawnień do sekwencji *Właściciela tabel*.

USOS_PROD_TAB jest *Właścicielem tabel*, natomiast SEQ_1, SEQ_2, SEQ_3, SEQ_m (znajdujące się wewnątrz USOS_PROD_TAB) to sekwencje *Właściciela tabel*. UZYTKOWNIK to zwykły użytkownik, natomiast SEQ_1, SEQ_2, SEQ_3, SEQ_m (znajdujące się wewnątrz UZYTKOWNIK) to synonimy użytkownika. Rola BAZOWA jest to predefiniowana rola nadawana wszystkim użytkownikom (por. p. 3.6). Strzałki przerywane między sekwencjami *Właściciela tabel* i synonimami użytkownika pokazują, do których sekwencji odwołują się poszczególne synonimy użytkownika. Strzałki ciągłe reprezentują nadanie uprawnień. Uprawnienia do poszczególnych sekwencji *Właściciela tabel* są nadane roli BAZOWA, a rola BAZOWA jest nadana użytkownikowi UZYTKOWNIK.

W przedstawionej sytuacji roli BAZOWA nadano uprawnienia SELECT do wszystkich sekwencji *Właściciela tabel* (rola BAZOWA powinna mieć zawsze nadane uprawnienie SELECT do wszystkich sekwencji). Użytkownik ma utworzone odpowiednie synonimy SEQ_1, SEQ_2, SEQ_3, SEQ_m, synonimy te odwołują się do sekwencji *Właściciela tabel*. Na przykład użytkownik ma synonim prywatny SEQ_1 odnoszący się do sekwencji USOS_PROD_TAB.SEQ_1.



Rys. 2. Kontrola uprawnień do sekwencji

3.4 Kontrola uprawnień do pakietów

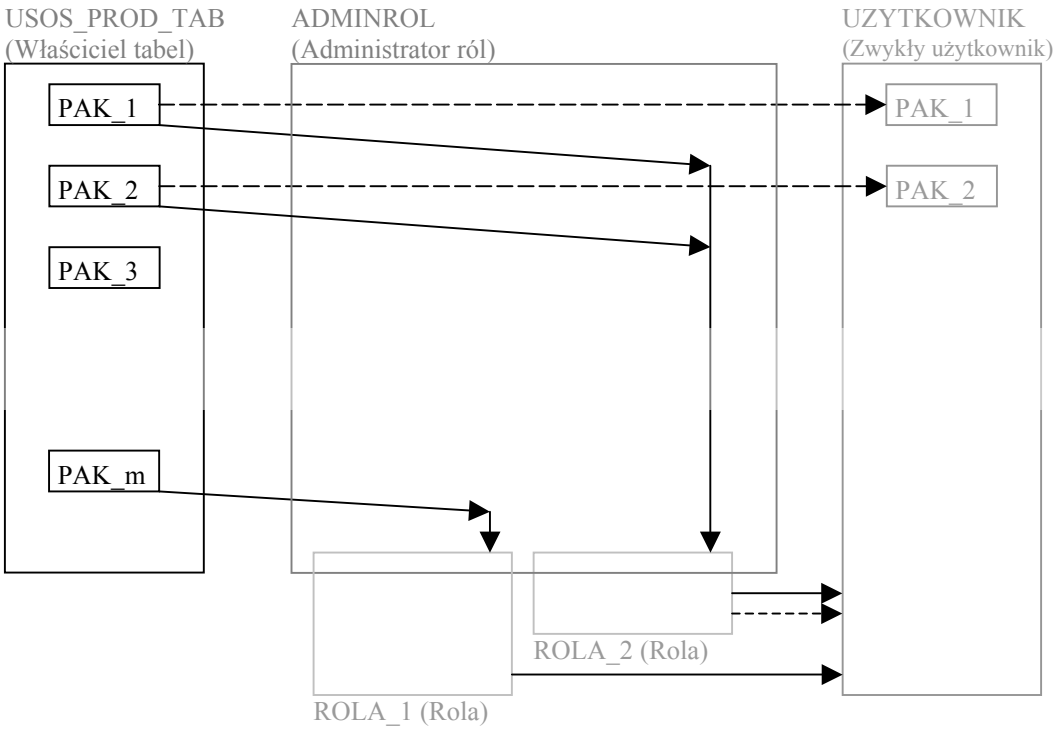
Kontrola uprawnień do pakietów odbywa się z wykorzystaniem mechanizmów Oracle. Uprawnienie wykonywania wybranych pakietów nadaje się wybranym rolom. Prawo EXECUTE dla wybranych pakietów nadaje się użytkownikowi poprzez nadanie tego uprawnienia wybranej roli, a następnie nadanie tej roli użytkownikowi. Gdy rola, która

ma nadane uprawnienie EXECUTE dla wybranych pakietów, staje się rolą domyślną użytkownika, tworzy się dla niego synonimy odnoszące się do pakietów *Właściciela tabel* i mające nazwy identyczne z nazwami pakietów.

Za kontrolę uprawnień do pakietów jest odpowiedzialny *Administrator ról* – to on określa, jakiej roli nadać uprawnienia do poszczególnych pakietów oraz jaką rolę nadać każdemu użytkownikowi.

Na rysunku 3 przedstawiono sposób kontroli uprawnień do pakietów *Właściciela tabel*.

USOS_PROD_TAB jest *Właścicielem tabel*, natomiast PAK_1, PAK_2, PAK_3, PAK_m (znajdujące się wewnątrz USOS_PROD_TAB) to pakiety *Właściciela tabel*. ADMINROL to *Administrator ról*. UZYTKOWNIK to zwykły użytkownik, natomiast PAK_1, PAK_2 (znajdujące się wewnątrz UZYTKOWNIK) to prywatne synonimy użytkownika. Role ROLA_1 i ROLA_2 to role zdefiniowane przez *Administratora ról*. Strzałki ciągłe reprezentują nadanie uprawnień. Uprawnienie EXECUTE do poszczególnych pakietów nadano rolom ROLA_1 i ROLA_2, natomiast role ROLA_1 i ROLA_2 nadano użytkownikowi UZYTKOWNIK. Strzałka przerywana między rolą ROLA_2 i użytkownikiem oznacza, że jest to jego rola domyślna.



Rys. 3. Kontrola uprawnień do pakietów

W przedstawionej sytuacji roli ROLA_1 nadano uprawnienie do pakietu PAK_m, natomiast roli ROLA_2 do pakietów PAK_1 i PAK_2. Obie role (ROLA_1 i ROLA_2) nadano użytkownikowi UZYTKOWNIK, ale tylko jedna z nich – ROLA_2 jest dla niego

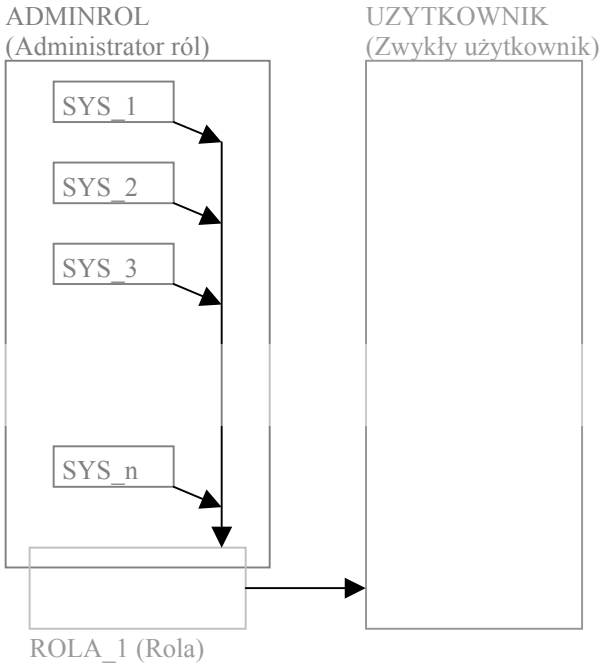
rolą domyślną. Użytkownik ma synonimy PAK_1 i PAK_2 odnoszące się do pakietów *Właściciela tabel* USOS_PROD_TAB.PAK_1 i USOS_PROD_TAB.PAK_2.

Należy zwrócić uwagę, że choć użytkownik UZYTKOWNIK ma uprawnienie do pakietu PAK_m (nadane przez rolę ROLA_1), to jednak nie ma on zdefiniowanego synonimu do tego pakietu, ponieważ rola ROLA_1 nie jest jego rolą domyślną.

Warto też zauważyć, że przy kontroli uprawnień do pakietów *Właściciela tabel*, *Administrator ról* nie ma żadnych pośrednich obiektów (jak to ma miejsce przy uprawnieniach do tabeli – por. p. 3.2), jednak ma on wpływ na nadawanie uprawnień do poszczególnych pakietów (w przeciwieństwie do uprawnień do sekwencji, gdzie uprawnienia nadawane są automatycznie – por. p. 3.3). Na rysunku zaznaczono *Administratora ról*, aby podkreślić, że to on nadaje uprawnienia do pakietów.

3.5 Kontrola uprawnień systemowych

Kontrola uprawnień systemowych odbywa się z wykorzystaniem mechanizmów Oracle. Uprawnienia systemowe nadaje się rolom. Uprawnienie systemowe nadaje się użytkownikowi poprzez nadanie tego uprawnienia roli, a następnie nadanie tej roli użytkownikowi. Za kontrolę uprawnień jest odpowiedzialny *Administrator ról*.



Rys. 4. Kontrola uprawnień systemowych

Na rysunku 4 przedstawiono, w jaki sposób przebiega kontrola uprawnień systemowych. ADMINROL to *Administrator ról*, natomiast SYS_1, SYS_2, SYS_3, SYS_n to uprawnienia systemowe *Administratora ról*. UZYTKOWNIK to zwykły użytkownik.

Przekazanie uprawnień jest reprezentowane przez strzałki ciągłe. W przedstawionej sytuacji poszczególne uprawnienia systemowe nadane są roli ROLA_1, natomiast rola ROLA_1 jest nadana użytkownikowi UZYTKOWNIK. Rola ROLA_1 nie jest rolą domyślną użytkownika UZYTKOWNIK (brak strzałki przerywanej między rolą ROLA_1 i użytkownikiem UZYTKOWNIK). Warto zwrócić uwagę na to, że przy nadawaniu uprawnień systemowych nie ma znaczenia czy rola, poprzez którą przekazuje się uprawnienia systemowe jest rolą domyślną użytkownika. Z nadawaniem uprawnień systemowych nie wiąże się też konieczność tworzenia synonimów dla użytkownika.

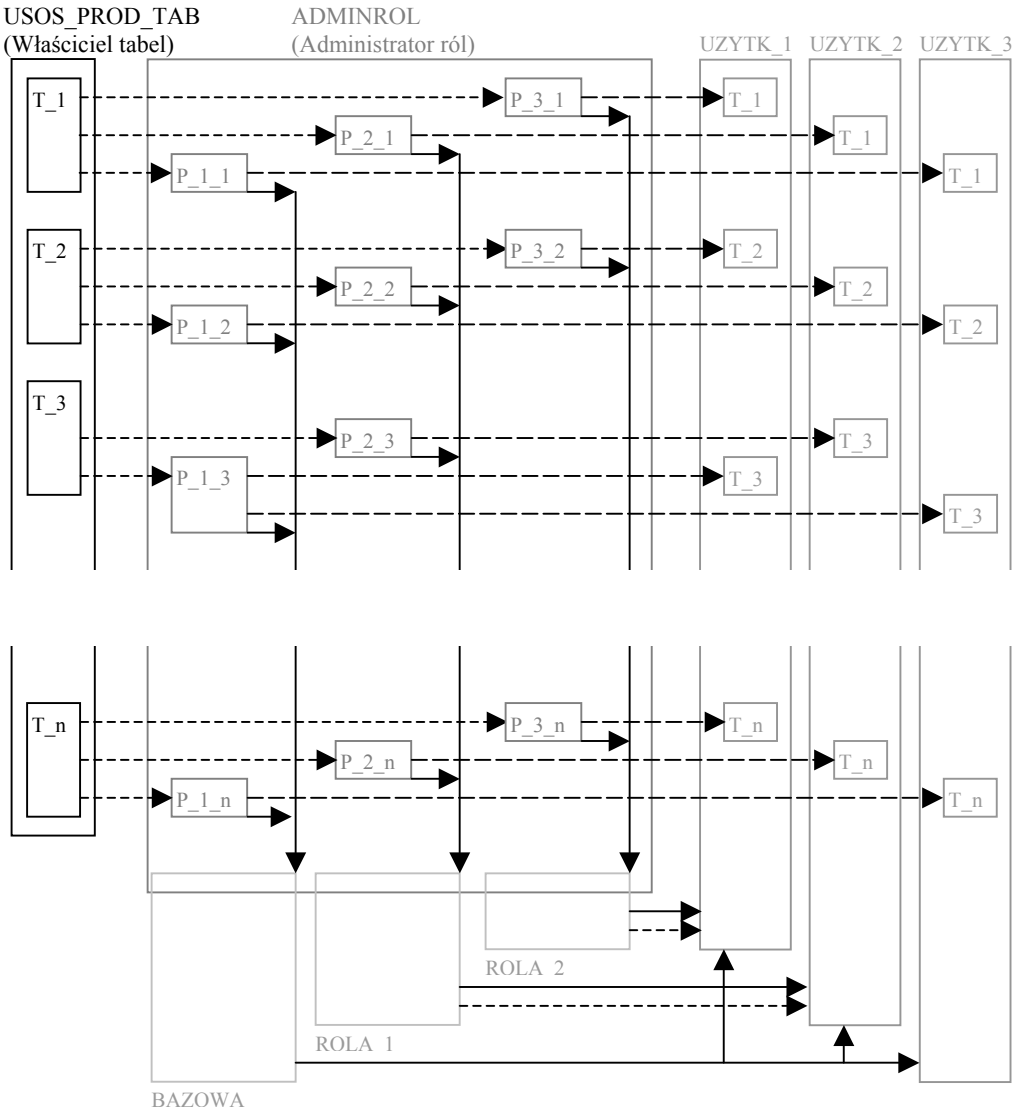
3.6 Rola BAZOWA

Rola BAZOWA to predefiniowana rola nadawana automatycznie (bez udziału *Administratora ról*) każdemu użytkownikowi. Zawiera uprawnienia, które powinien mieć każdy użytkownik (nawet nie mający żadnej roli nadanej przez *Administratora ról*). Dzięki niej jest też możliwe prawidłowe działanie formularzy uruchamianych przez użytkowników, którzy mogą nie mieć żadnych uprawnień do tabel używanych w formularzach (nie są zgłaszane błędy przy odwoływaniu do tabel, do których użytkownik nie ma uprawnień).

Kontrola uprawnień przy odwołaniach do tabel *Właściciela tabel* poprzez rolę BAZOWA, odbywa się w sposób analogiczny jak w innych rolach. Dla każdej tabeli i perspektywy *Właściciela tabel* tworzone są perspektywy, poprzez które użytkownicy odwołują się do poszczególnych obiektów. Konstrukcja tych perspektyw jest taka, że przekazują one zawsze jeden wiersz z wartościami NULL we wszystkich kolumnach.

Na rysunku 5 przedstawiono model zależności między tabelami i perspektywami *Właściciela tabel*, perspektywami *Administratora ról* oraz synonimami użytkownika. Pokazano również, w jaki sposób wykorzystuje się rolę BAZOWA i kiedy użytkownicy korzystają z perspektyw utworzonych dla tej roli.

USOS_PROD_TAB to *Właściciel tabel*, natomiast T_1, T_2, T_3, T_n (znajdujące się wewnątrz USOS_PROD_TAB) reprezentują tabele i perspektywy *Właściciela tabel*. ADMINROL jest *Administratorem ról*. P_1_1, P_1_2, P_1_3, P_1_n reprezentują perspektywy *Administratora ról* utworzone dla roli BAZOWA. Perspektywy P_2_1, P_2_2, P_2_3, P_2_n to perspektywy *Administratora Ról* utworzone dla roli ROLA_1. Perspektywy P_3_1, P_3_2, P_3_n to perspektywy *Administrator ról* utworzone dla roli ROLA_2. UZYT_1, UZYT_2, UZYT_3 to zwykli użytkownicy, natomiast T_1, T_2, T_3, T_n to prywatne synonimy użytkowników. Strzałki przerywane między tabelami *Właściciela tabel* i perspektywami *Administratora ról* pokazują, na jakich tabelach są oparte poszczególne perspektywy. Strzałki ciągłe prowadzące od perspektyw *Administratora ról* do poszczególnych ról (BAZOWA, ROLA_1, ROLA_2) reprezentują nadanie rolom uprawnień do tych perspektyw. Strzałki ciągłe prowadzące od ról (BAZOWA, ROLA_1, ROLA_2) do użytkowników reprezentują nadanie ról użytkownikom. Strzałka przerywana pomiędzy rolą i użytkownikiem oznacza, że dana rola jest rolą domyślna użytkownika. Strzałki przerywane między perspektywami *Administratora ról* i synonimami użytkowników oznaczają obiekty, do jakich odwołują się poszczególne synonimy.



Rys. 5. Synonimy do perspektyw z roli domyślnej oraz do perspektyw roli BAZOWA

W przedstawionej sytuacji UZYT_3 nie ma nadanej żadnej roli (poza rolą BAZOWA, która jest nadana wszystkim użytkownikom), dlatego też jego synonimy odwołują się do perspektyw roli BAZOWA (perspektywy roli BAZOWA wyświetlają jeden pusty wiersz). UZYT_2 ma nadaną rolę ROLA_1, rola ta jest też jego rolą domyślną, ponieważ rola ta ma uprawnienia do wszystkich przedstawionych na rysunku tabel *Właściciela tabel*, UZYT_2 nie ma żadnych synonimów odnoszących się do perspektyw roli BAZOWA. UZYT_1 ma nadaną rolę ROLA_2, rola ta jest też jego rolą domyślną. Rola ROLA_2 nie ma uprawnień do wszystkich tabel *Właściciela tabel*, rola ta nie ma nadanego uprawnienia do tabeli TAB_3, dlatego też w dostępie do tabeli T_3 użytkownik ten korzysta z perspektywy P_1_3 utworzonej dla roli BAZOWA. W dostępie do pozostałych tabel korzysta z perspektyw jego roli domyślnej.

4. Podsumowanie

W niniejszej pracy przedstawiono elastyczny system kontroli uprawnień użytkowników bazy danych, który powstał w ramach prac na Uniwersyteckim Systemem Obsługi Studiów. Otrzymany system jest jednak niezależny od USOS i można go zastosować dla dowolnego schematu bazy danych.

Jednym z najciekawszych aspektów tego rozwiązania jest zastosowanie w nim wyzwalaczy INSTEAD jako narzędzia programowania aktualizowalnych perspektyw. Dzięki tym wyzwalaczom możliwe było aktualizowanie dowolnie złożonych perspektyw i uniezależnienie możliwości ich aktualizacji od składni ich definicji. W standardzie SQL [ISO1992] określono dość surowe ograniczenia syntaktyczne na perspektywy, które można aktualizować. Z tego powodu praktyczne znaczenie tak pojmowanej aktualizacji perspektyw jest marginalne. Jedynym ogólnym rozwiązaniem są wyzwalacze INSTEAD, dzięki którym każdą perspektywę można uczynić aktualizowalną.

Otrzymany system jest bardzo elastyczny, ale niestety stanowi to również jego wadę. Ta elastyczność powoduje, że definiowanie uprawnień poszczególnych ról jest trudne i wymaga dobrej znajomości struktury bazy danych. Obecnie prowadzone są prace nad rozwiązaniem tego problemu.

W systemach, w których wdrożono już opisywane rozwiązanie, zaobserwowano, że wiele ról jest do siebie podobnych (np. role dla poszczególnych wydziałów różniące się między sobą tylko kodem wydziału w warunkach). Dalszy rozwój systemu kontroli uprawnień będzie polegał na dodaniu możliwości parametryzacji ról i tworzenia ról będących sumami uprawnień innych ról.

Bibliografia

[DADA2000] Date, C. J, Darwen, H., *SQL. Omówienie standardu języka*, WNT, Warszawa, 2000.

[ISO1992] International Organization for Standardization (ISO), *Database Language SQL*. ISO/IEC 9075:1992.

[KORU2000] Korkuć, J., Rudzki, R., *Uniwersytecki System Obsługi Studiów*. Praca magisterska, Instytut Informatyki, Uniwersytet Warszawski, wrzesień 2000. http://usos.mimuw.edu.pl/PraceMagisterskie/korkuc-rudzki/korkuc-rudzki.html.

[MAKA2002] Makaroś, M., *Uniwersytecki System Obsługi Studiów. Role*. Praca magisterska, Instytut Informatyki, Uniwersytet Warszawski, wrzesień 2002. http://usos.mimuw.edu.pl/PraceMagisterskie/makaros/makaros.zip.

[MINC2002] Mincer-Daszkiewicz, J., *Tworzenie i wdrażanie produkcyjnego oprogramowania w środowisku akademickim.* Wybrane problemy inżynierii oprogramowania. Red. Jerzy R. Nawrocki i Bartosz Walter. IV KKIO, Tarnowo Podgórne k. Poznania, 15-18 października 2002, str. 299-314.

[USOS2002] *Dystrybucja USOS 1.14*,http://usos.mimuw.edu.pl/Dystrybucje/dystrybucja.html.