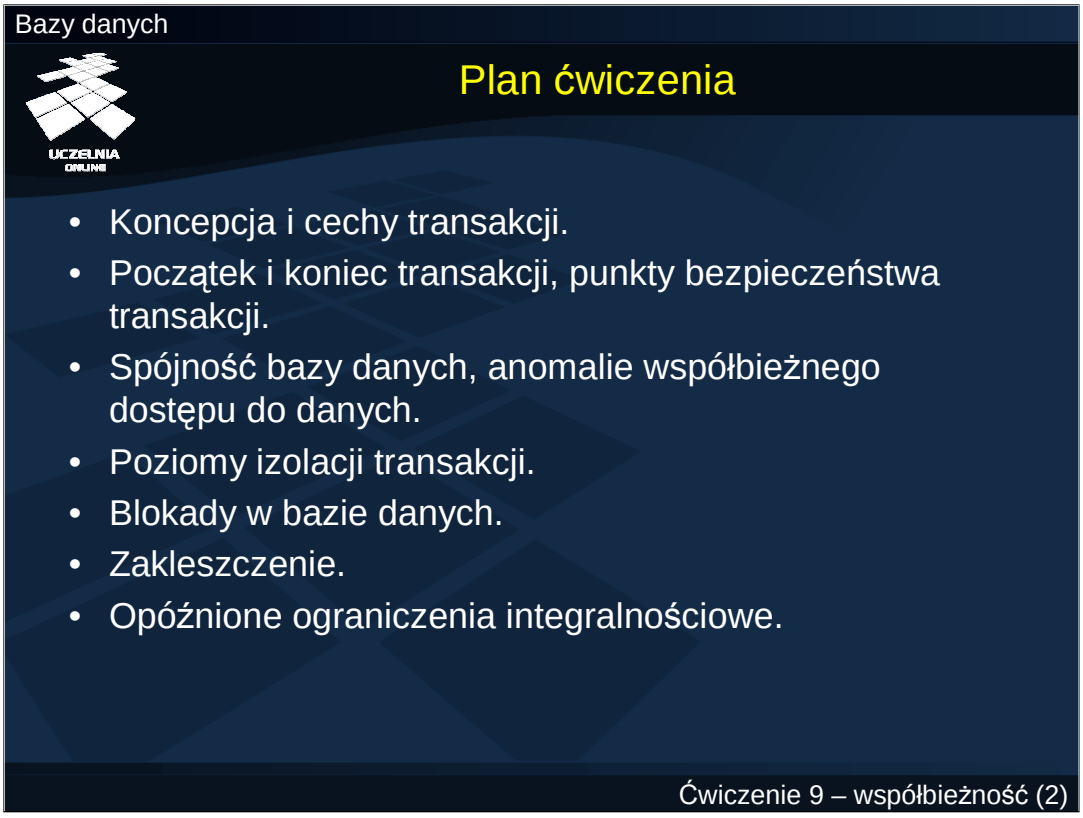
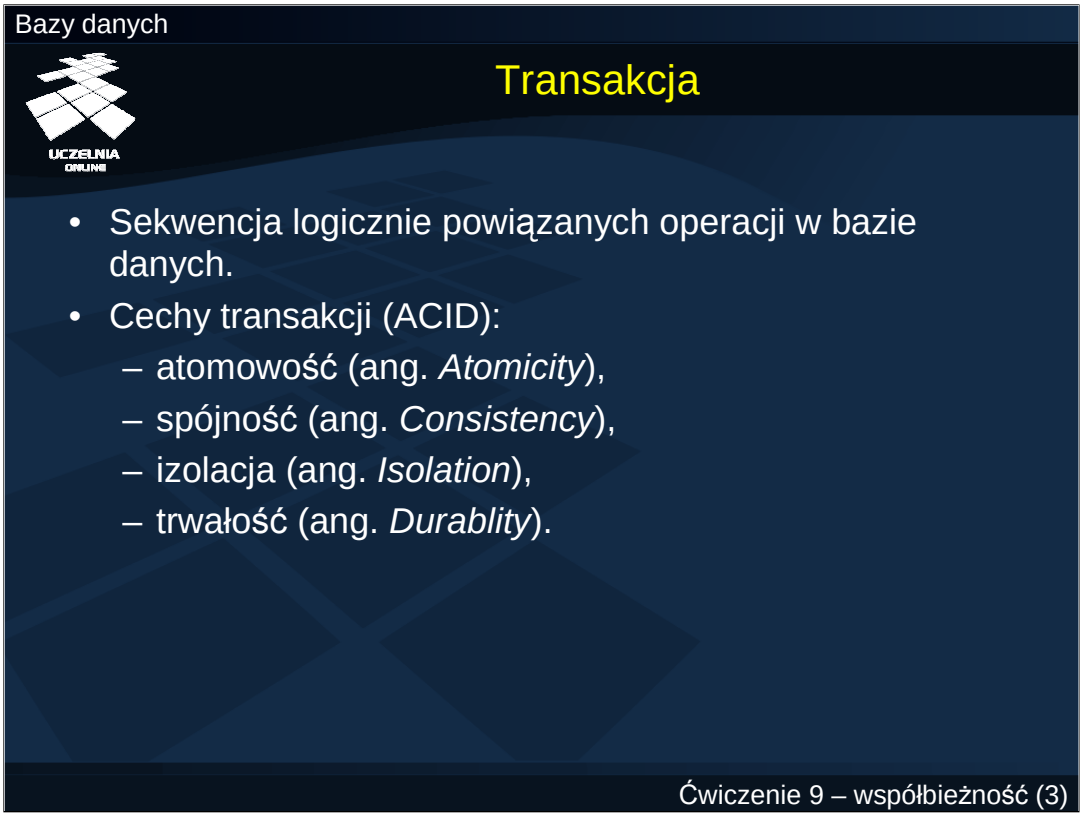
Niniejsze ćwiczenie zaprezentuje zagadnienia związane z problemami wynikającymi ze współbieżnego dostępu użytkowników do bazy danych.

Wymagania:

Umiejętność tworzenia zapytań w języku SQL, znajomość operacji z grup DML i DDL.



Na wstępie ćwiczenia omówiona zostanie koncepcja i cechy transakcji w bazie danych. Przedstawione zostaną sytuacje jawnego i niejawnego rozpoczynania i kończenia transakcji oraz mechanizm punktów bezpieczeństwa. Następnie scharakteryzowany zostanie termin „spójność bazy danych” oraz zbiór anomalii, jakie mogą mieć miejsce przy współbieżnym dostępie do danych. Dalej omówione zostaną poziomy izolacji transakcji oraz blokady w bazie danych jako mechanizmy zapobiegające występowaniu anomalii. Następnie przedstawimy sytuację, w której dochodzi do zakleszczenia transakcji w bazie danych. Na końcu ćwiczenia zaprezentowane zostaną opóźnione ograniczenia intergralnościowe relacji.



Transakcja jest sekwencją logicznie powiązanych operacji w bazie danych. W przypadku posługiwania się językiem SQL operacje, wchodzące w skład transakcji, to kolejne polecenia SQL, wydawane przez użytkownika w trakcie sesji.

Każda transakcja charakteryzuje się czterema własnościami, często określanymi jako cechy ACID od pierwszych liter angielskich określeń. Są to:

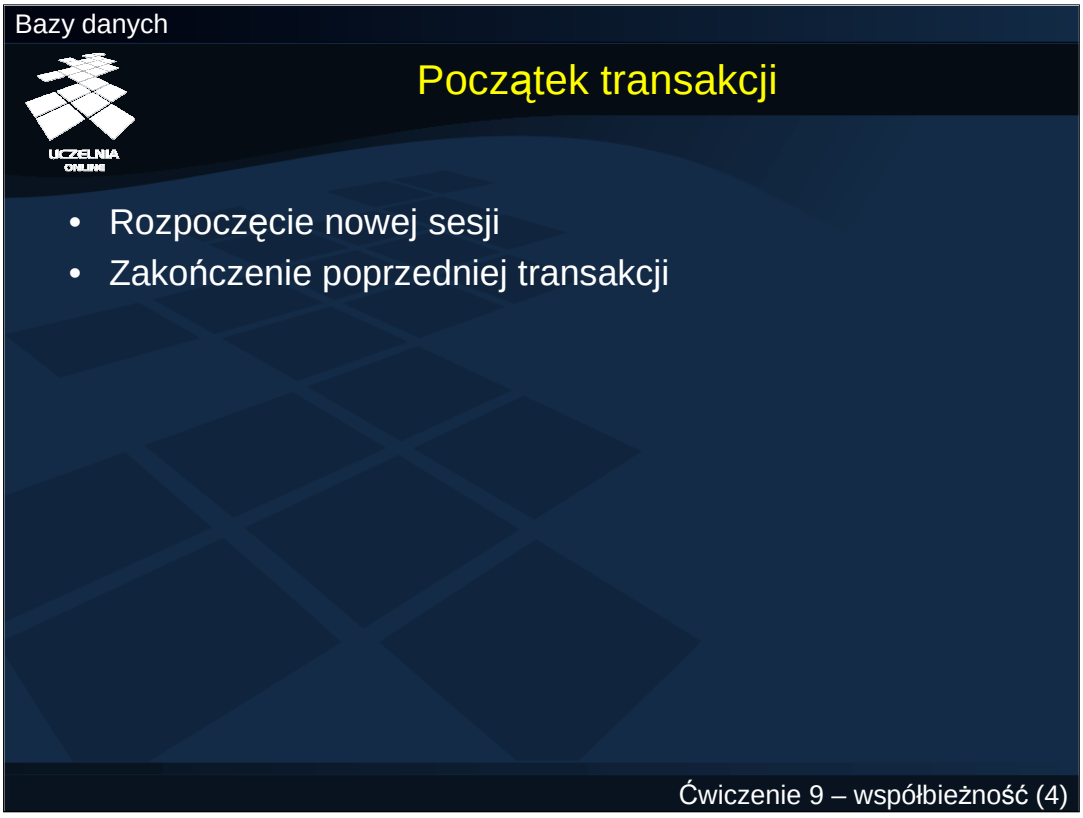
(1) Atomowość – sekwencja operacji w ramach transakcji jest niepodzielna: albo wszystkie operacje zostaną wykonane z powodzeniem, albo żadna z nich.

(2) Spójność – baza danych w momencie rozpoczęcia transakcji jest w stanie spójnym, gdy transakcja zostaje zakończona, baza danych również musi zachować spójność. Termin „spójność bazy danych” scharakteryzujemy na jednym z następnych slajdów.

(3) Izolacja – transakcje, wykonywane równolegle w bazie danych, są od siebie logicznie odseparowane. Oznacza to, że transakcja nie doświadcza w trakcie swego działania efektów, wprowadzanych do bazy danych przez inne transakcje.

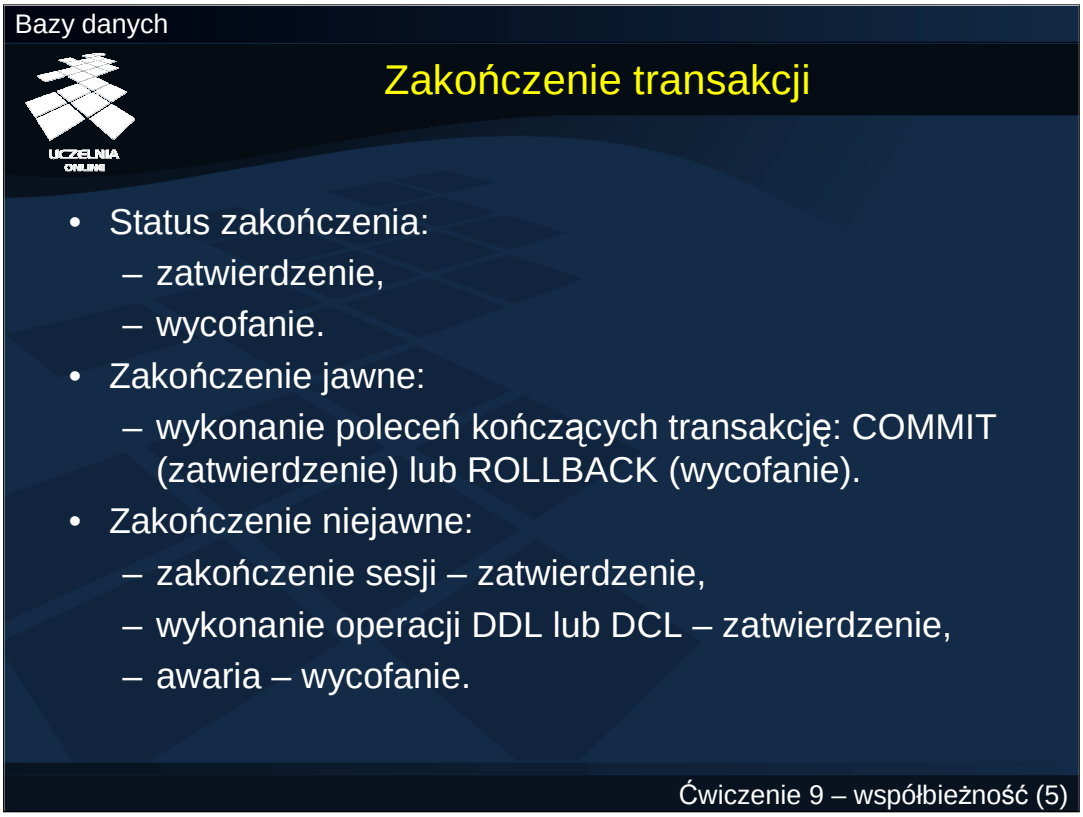
(4) Trwałość – zmiany wprowadzone przez zakończoną z powodzeniem transakcję muszą być trwałe, nie mogą zostać utracone nawet w przypadku awarii systemu.

Każda transakcja ma początek i koniec. Omówimy teraz po kolei momenty rozpoczęcia i zakończenia transakcji.



Gdy użytkownik przyłącza się do bazy danych, rozpoczyna nową transakcję. Wszystkie operacje, które wykonuje od momentu rozpoczęcia pracy, są operacjami w ramach transakcji.

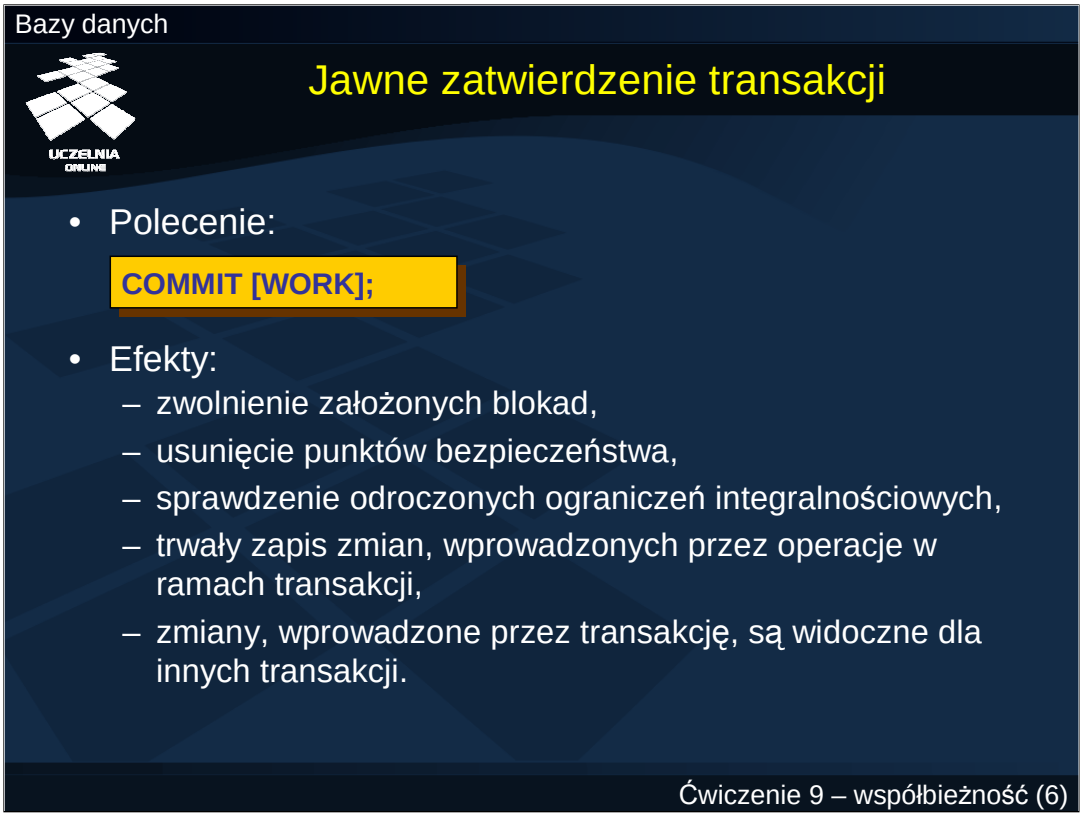
Użytkownik może jawnie rozpocząć nową transakcję, żądając zakończenia transakcji bieżącej. Charakterystykę momentu zakończenia transakcji przedstawimy na następnym slajdzie.



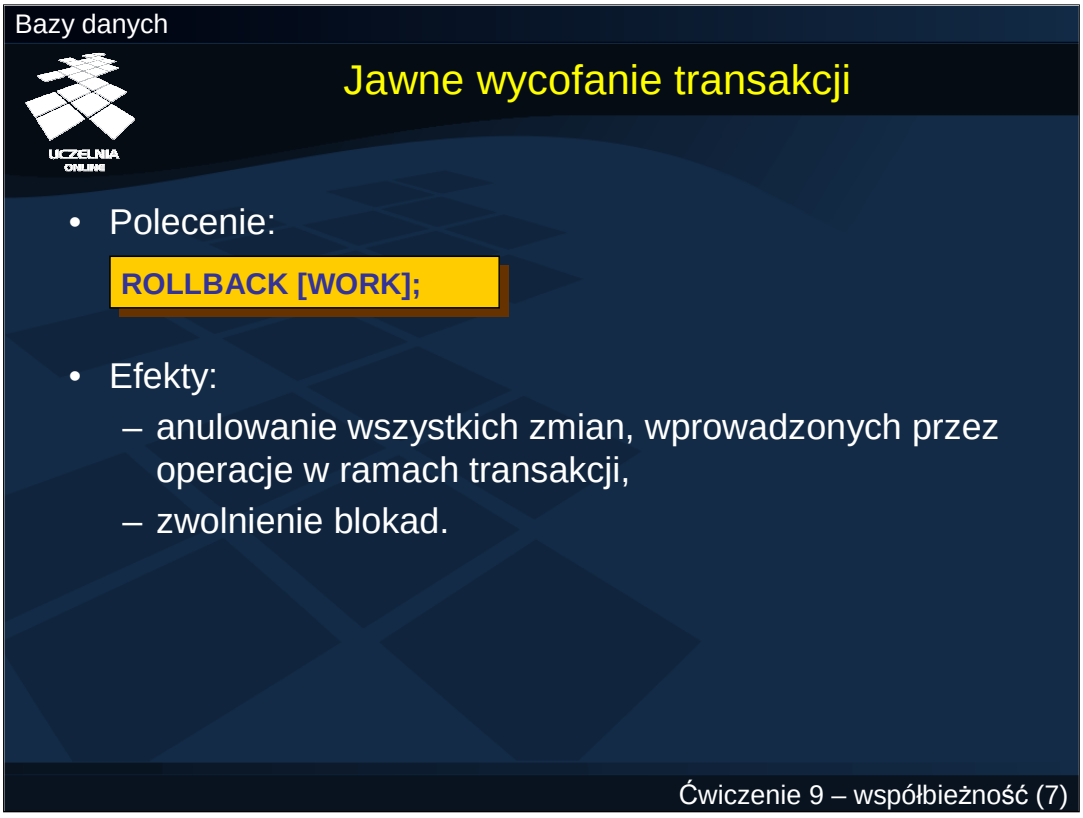
Transakcja może zostać zakończona w jednym z dwóch stanów. Zatwierdzenie transakcji oznacza, że transakcja kończy się powodzeniem i wszystkie operacje, jakie zostały w ramach transakcji zrealizowane, mają zostać trwale zapisane w bazie danych. Taką transakcję określamy jako „zatwierdzoną”. Z kolei wycofanie transakcji oznacza niepowodzenie jej działania, wszystkie zmiany, wprowadzone przez operacje w ramach transakcji, muszą zostać anulowane. Taka transakcja jest określana mianem „wycofanej”.

Transakcja może zostać zakończona jawnie lub niejawnie. Zakończenie jawne następuje na wyraźne żądanie użytkownika, który wykonuje polecenie COMMIT, zatwierdzające transakcję, lub polecenie ROLLBACK, wycofujące transakcję. Oba polecenia zostaną przedstawione na kolejnych slajdach.

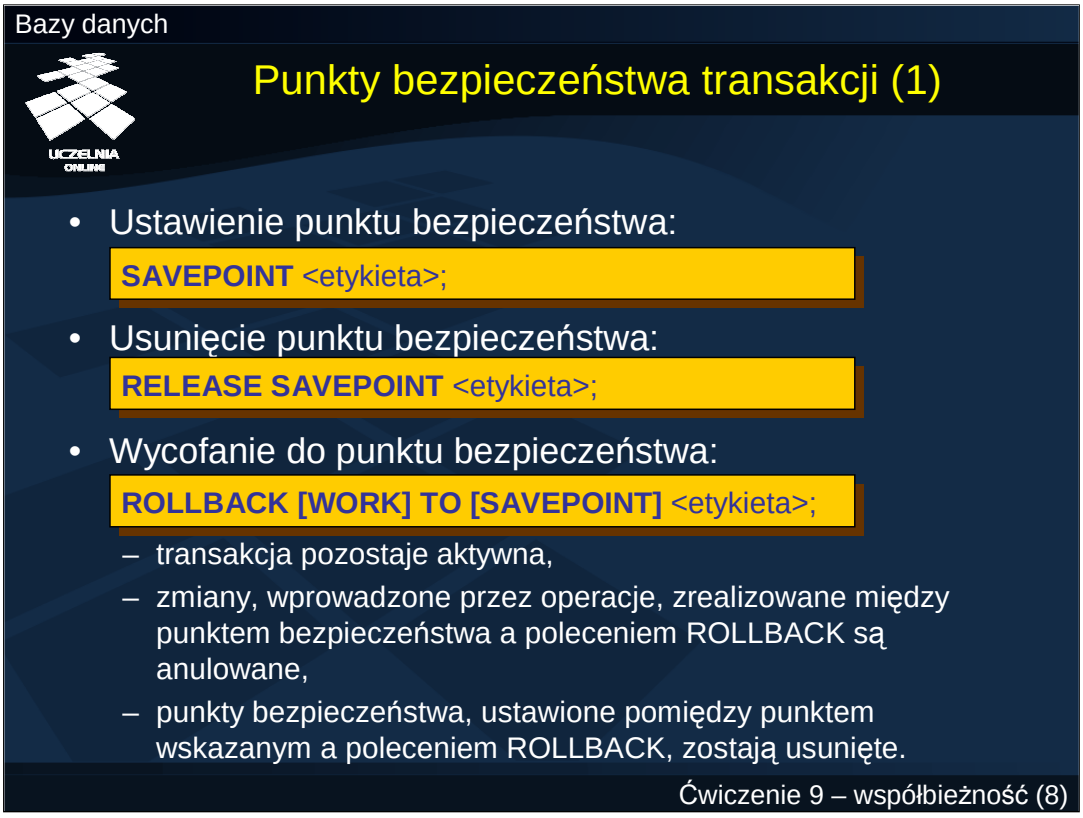
Niejawne zakończenie transakcji realizowane jest w następujących sytuacjach. Gdy użytkownik żąda zakończenia sesji, bieżąca transakcja użytkownika zostaje niejawnie zatwierdzona. Jeśli w ramach swojej sesji użytkownik uruchomi polecenie z grup DDL lub DCL, wówczas przed wykonaniem polecenia bieżąca transakcja sesji użytkownika zostaje zatwierdzona, a po zakończeniu wykonywania polecenia rozpoczynana jest nowa transakcja. Jeśli nastąpi awaria, np. utrata połączenia między aplikacją klienta a serwerem bazy danych, wówczas bieżąca transakcja zostaje wycofana.



Użytkownik jawnie zatwierdza transakcję poleceniem COMMIT WORK, przy czym słowo WORK jest opcjonalne. W momencie zatwierdzenia transakcji następuje zwolnienie blokad, założonych przez operacje w ramach transakcji, usunięcie punktów bezpieczeństwa, sprawdzenie odroczonych ograniczeń integralnościowych (blokady, punkty bezpieczeństwa i ograniczenia odroczone zostaną omówione w dalszej części ćwiczenia). Zmiany, wprowadzone przez operacje w ramach transakcji zostają trwale zapisane w bazie danych i są widoczne dla innych transakcji. Po zatwierdzeniu bieżącej transakcji rozpoczęta zostaje nowa transakcja.



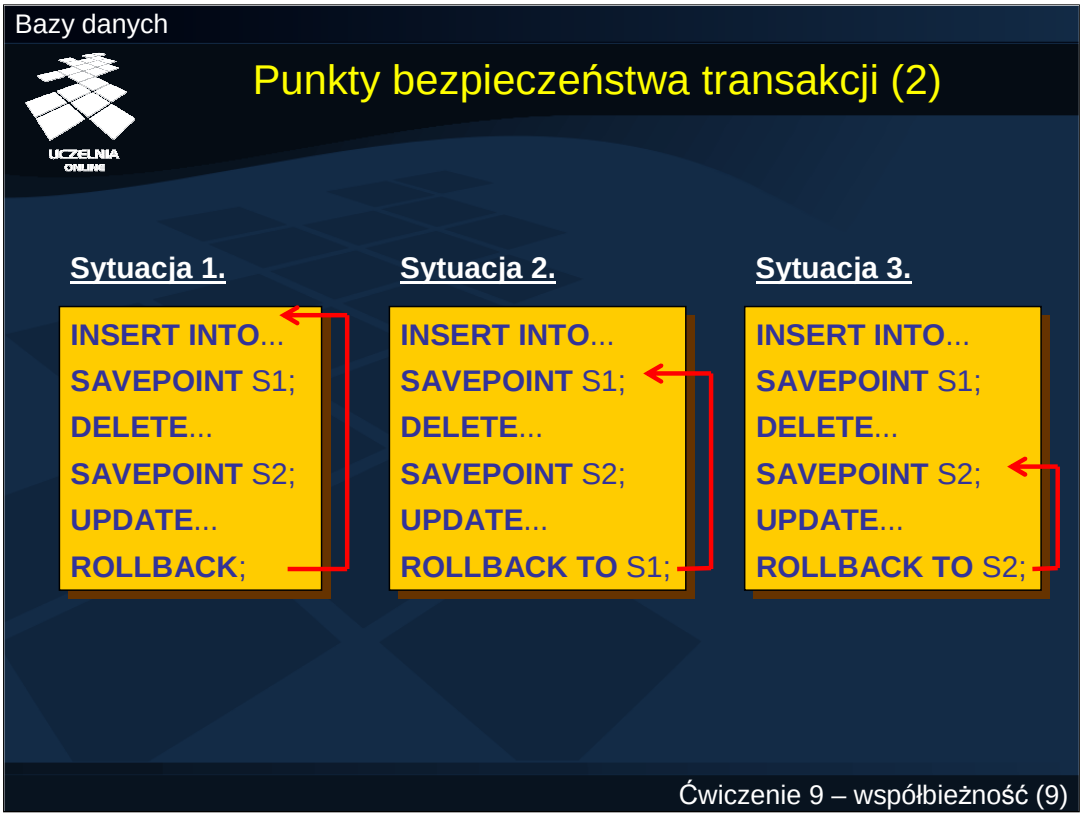
Polecenie ROLLBACK WORK (słowo WORK jest opcjonalne) wycofuje bieżącą transakcję. W tym momencie zostają anulowane wszystkie zmiany, wprowadzone do bazy danych przez operacje w ramach transakcji i następuje zwolnienie założonych blokad. Po wycofaniu bieżącej transakcji rozpoczynana jest nowa transakcja.



Transakcja może zostać podzielona na etapy przy użyciu tzw. punktów bezpieczeństwa. Dzięki punktom bezpieczeństwa istnieje możliwość cofnięcia się do określonego momentu w historii aktywnej transakcji. Cofnięcie powoduje anulowanie zmian, które zostały wprowadzone przez operacje transakcji wykonane od momentu utworzenia punktu bezpieczeństwa, do którego nastąpiło cofnięcie, do momentu zażądania przez użytkownika cofnięcia do punktu bezpieczeństwa. Transakcja nadal pozostaje aktywna (nie zostaje wycofana w całości!) i może być kontynuowana.

Utworzenie punktu bezpieczeństwa realizuje się poleceniem SAVEPOINT <etykieta>, gdzie <etykieta> jest unikalną w ramach bieżącej transakcji nazwą punktu bezpieczeństwa. Punkt bezpieczeństwa można usunąć z historii transakcji poleceniem RELEASE SAVEPOINT <etykieta>. Polecenie to nie wpływa na przebieg transakcji i operacje w ramach transakcji. Transakcja może posiadać wiele punktów bezpieczeństwa.

Wycofanie transakcji do wskazanego punktu bezpieczeństwa realizuje polecenie ROLLBACK TO SAVEPOINT <etykieta>. Efektem polecenia jest wycofanie zmian, wprowadzonych przez operacje aktywnej transakcji od momentu utworzenia punktu bezpieczeństwa o podanej etykiecie do momentu wykonania polecenia. Jeśli wycofujemy się do punktu bezpieczeństwa wcześniejszego niż ostatnio utworzony, wszystkie punkty bezpieczeństwa utworzone później zostają usunięte. Sytuację tą wyjaśnią przykłady na slajdzie następnym.



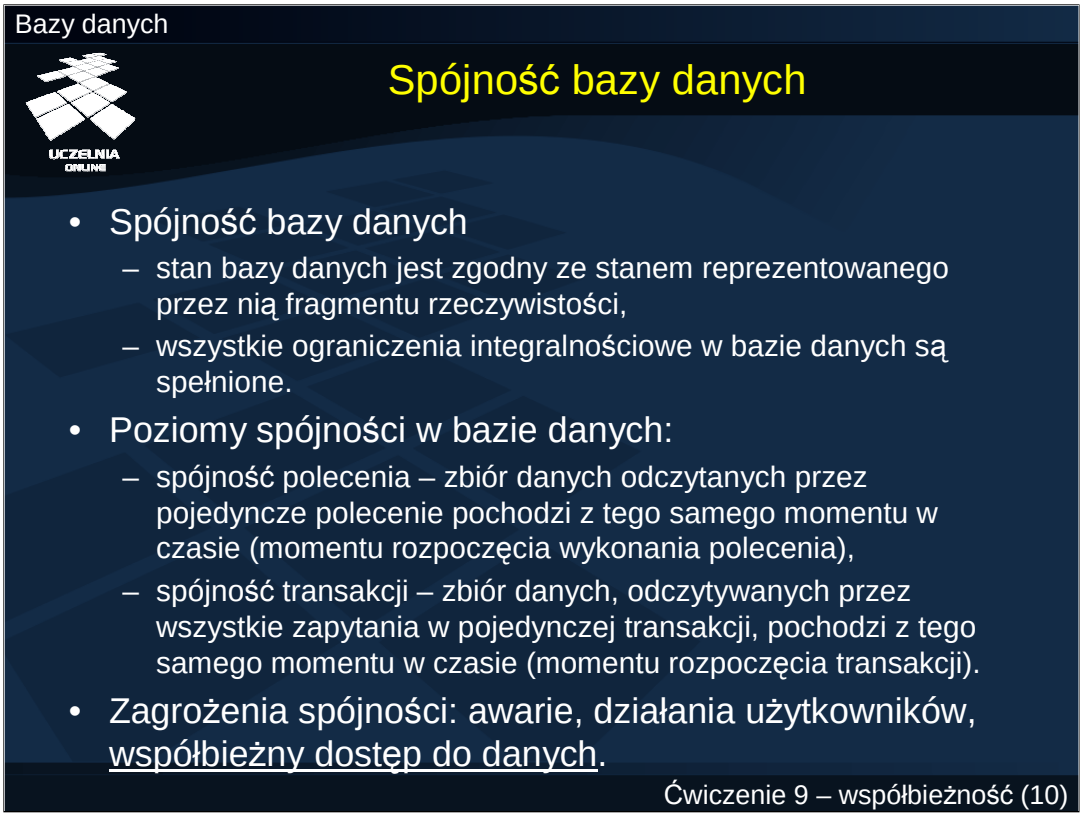
Bieżący slajd wyjaśnia działanie punktów bezpieczeństwa transakcji.

Przykładowa transakcja składa się z sekwencji poleceń. Pierwsze polecenie wstawia rekord do pewnej relacji. Po wykonaniu tej operacji utworzono punkt bezpieczeństwa S1. Następnie usunięto rekordy z pewnej relacji i utworzono punkt bezpieczeństwa S2. Ostatnie polecenie uaktualnia rekordy pewnej relacji. Teraz omówimy trzy różne sytuacje.

W sytuacji 1. wykonano polecenie ROLLBACK. Oznacza to wycofanie wszystkich zmian, wprowadzonych przez transakcję, a więc efektów poleceń: INSERT, DELETE i UPDATE, oraz usunięcie punktów bezpieczeństwa S1 i S2. Transakcja zostaje wycofana, przestaje być aktywna. Rozpoczyna się nowa transakcja.

W sytuacji 2. wykonano polecenie ROLLBACK TO SAVEPOINT S1, powodując anulowanie efektów operacji UPDATE i DELETE oraz usunięcie punktu bezpieczeństwa S2. Transakcja pozostaje aktywna, zmiany wprowadzone przez polecenie INSERT nie zostają wycofane. W ramach transakcji mogą być wykonywane następne operacje.

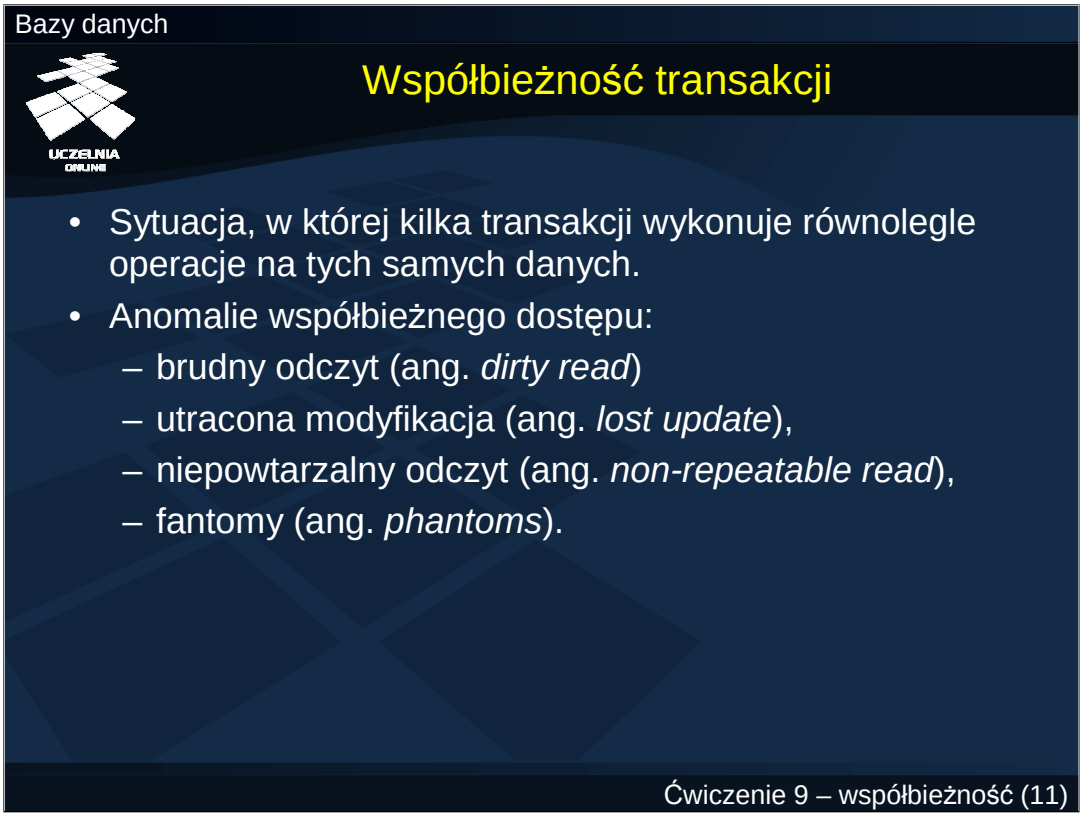
W sytuacji 3. wydano polecenie ROLLBACK TO SAVEPOINT S2, cofając się w historii transakcji do momentu sprzed wykonania polecenia UPDATE (efekty tego polecenia zostają utracone). Transakcja pozostaje aktywna, zmiany wprowadzone przez polecenia INSERT i DELETE nie zostają wycofane, punkt bezpieczeństwa S1 pozostaje zdefiniowany.



Scharakteryzujemy teraz termin „spójność bazy danych”. Można podać dwie definicje spójności. Pierwsza z nich mówi, że baza danych jest spójna jeśli jej stan jest zgodny ze stanem fragmentu rzeczywistości, którą baza danych reprezentuje. Inna definicja spójności określa bazę danych jako spójną wtedy, gdy wszystkie ograniczenia integralnościowe, zdefiniowane w bazie danych, są spełnione.

Baza danych może posługiwać się jednym z dwóch poziomów spójności. Spójność polecenia gwarantuje, że wszystkie dane, które odczytuje pojedyncze polecenie z bazy danych, pochodzą z tego samego momentu w czasie – momentu rozpoczęcia wykonania polecenia. Czyli nie może zajść sytuacja, w której rozpoczynamy wykonanie zapytania do bazy danych, w trakcie realizacji zapytania dane w bazie zostają zmodyfikowane, przez co część danych, odczytanych przez zapytanie, pochodzi sprzed modyfikacji, a część jest już zmieniona. Z kolei spójność transakcji gwarantuje, że wszystkie polecenia, odczytujące w ramach pojedynczej transakcji dane z bazy danych, będą odczytywały stan z momentu rozpoczęcia transakcji. Najczęściej domyślnym poziomem spójności jest poziom polecenia, w niektórych systemach zarządzania bazami danych można wybrać dodatkowo poziom spójności transakcji.

Baza danych może utracić spójność w wyniku różnych działań. Mogą do tego przyczynić się awarie, powodujące utratę części danych, działania użytkowników, którzy omyłkowo zmodyfikują czy usuną dane, wreszcie współbieżny dostęp do danych. Ten ostatni czynnik będzie stanowił przedmiot dalszych rozważań.



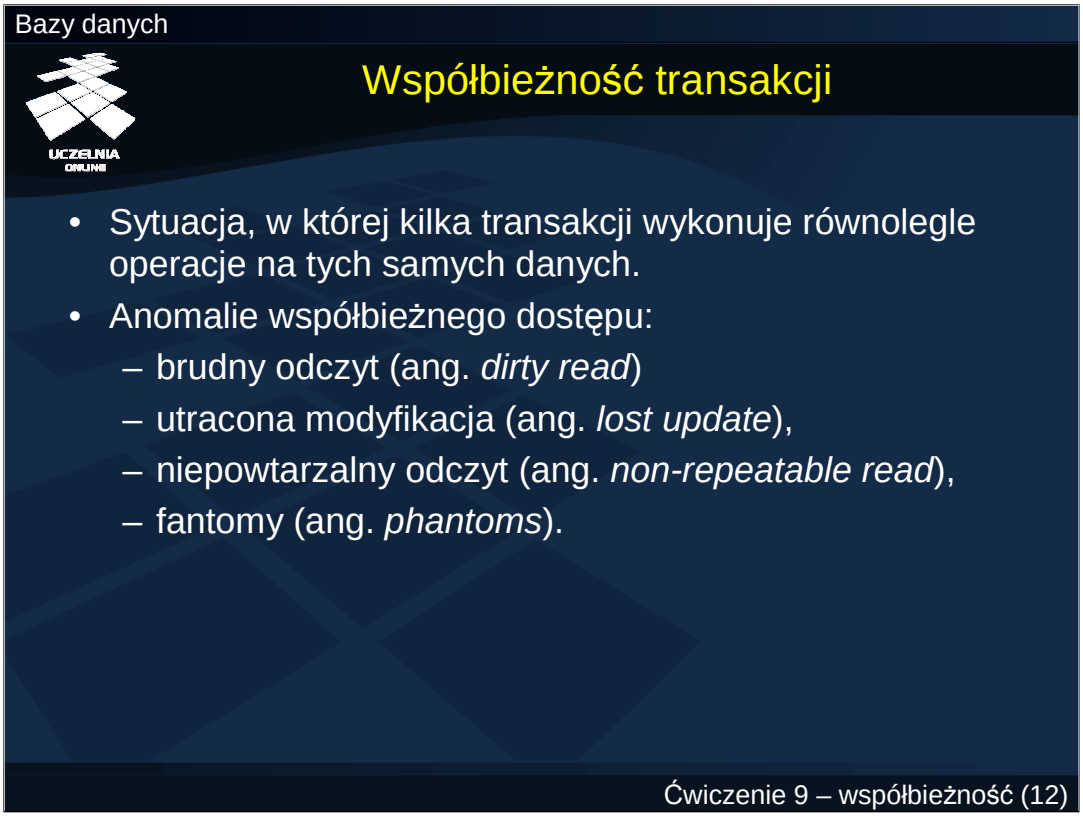
Często w bazie danych jednocześnie pracuje wielu użytkowników, uruchamiających polecenia w ramach swoich transakcji. Transakcje wykonywane są równolegle, często również zdarza się, że w ramach różnych transakcji użytkownicy żądają wykonania operacji na tych samych danych. Np. użytkownik ALA chce usunąć wszystkie rekordy ze swojej relacji PRACOWNICY, tymczasem równolegle użytkownik OLEK chce podnieść płace podstawowe wszystkim pracownikom z relacji PRACOWNICY użytkownika ALA. Konflikty w dostępie do danych operacji z równoległych transakcji mogą spowodować utratę spójności bazy danych. Może dojść do wystąpienia tzw. anomalii współbieżnego dostępu. Anomalie były szeroko omawiane w ramach wykładu z przedmiotu, tutaj przypomnimy w skrócie ich istotę.

Z anomalią o nazwie „brudny odczyt” mamy do czynienia wtedy, gdy transakcja odczytuje dane, zmienione przez inną transakcję, która potem zostaje wycofana. Tak więc pierwsza transakcja odczytała dane, które już nie istnieją.

Anomalia o nazwie „utracona modyfikacja” powstaje w sytuacji, gdy dwie transakcje równolegle przystępują do aktualizacji tych samych danych i zmiany, wprowadzone przez jedną z transakcji, zostają nadpisane przez zmiany z drugiej transakcji.

„Niepowtarzalny odczyt” to sytuacja, w której transakcja wielokrotnie wykonuje tą samą operację odczytu rekordu z relacji bazy danych i otrzymuje różne wartości atrybutów rekordu, gdyż równolegle inna transakcja modyfikuje wartości atrybutów tego samego rekordu.

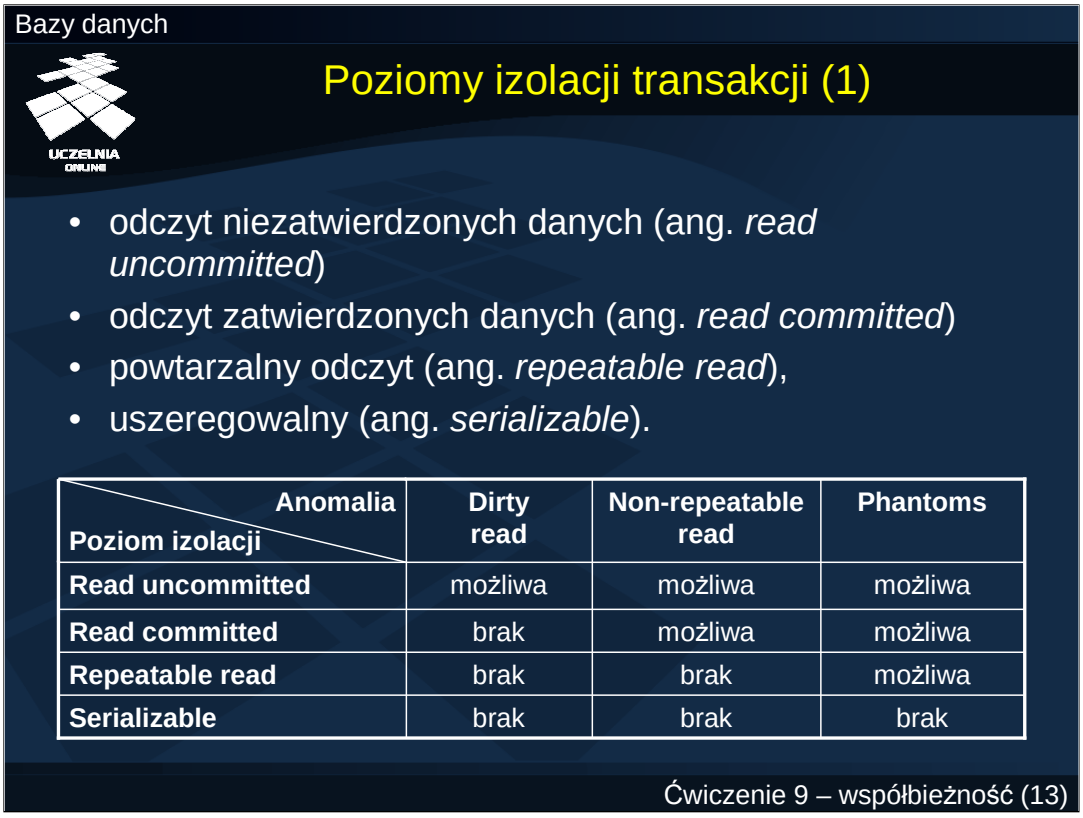
Ciąg dalszy na następnym slajdzie.



Ciąg dalszy z poprzedniego slajdu.

Ostatnia z anomalii, „fantomy”, jest podobna do niepowtarzalnego odczytu: transakcja wielokrotnie wykonuje to samo zapytanie do bazy danych i otrzymuje różne zbiory rekordów. Dzieje się tak dlatego, że równolegle inna transakcja wstawia nowe rekordy czy też modyfikuje istniejące w ten sposób, że zbiór rekordów, spełniających kryteria zapytania pierwszej transakcji, wciąż się powiększa.

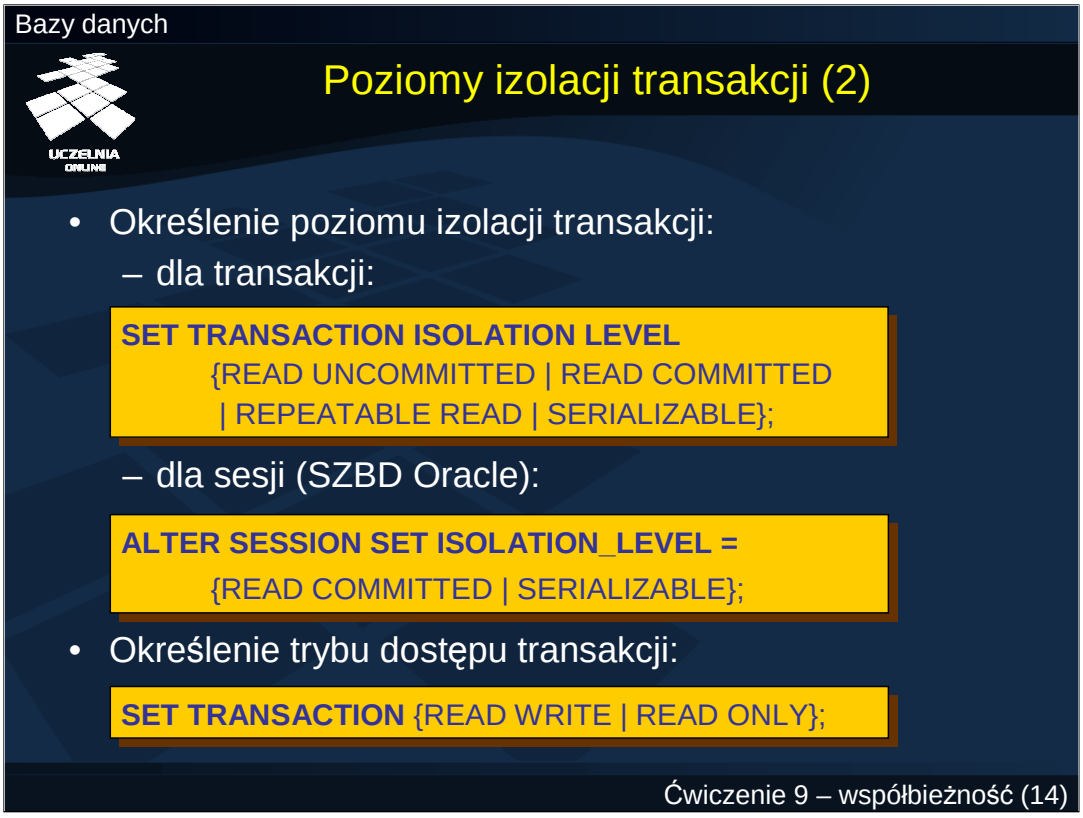
Anomalie „brudny odczyt” i „utracona modyfikacja” są poważnymi zagrożeniami dla spójności, natomiast „niepowtarzalny odczyt” i „fantomy” są anomaliami tolerowanymi w większości systemów zarządzania bazami danych.



Poziomy izolacji transakcji określają, jaki stopień wzajemnego odseparowania równolegle wykonywanych transakcji ma zostać zastosowany w bazie danych. Standard ANSI SQL definiuje cztery poziomy izolacji transakcji. Anomalia „utracona modyfikacja” nie występuje w żadnym z poziomów izolacji.

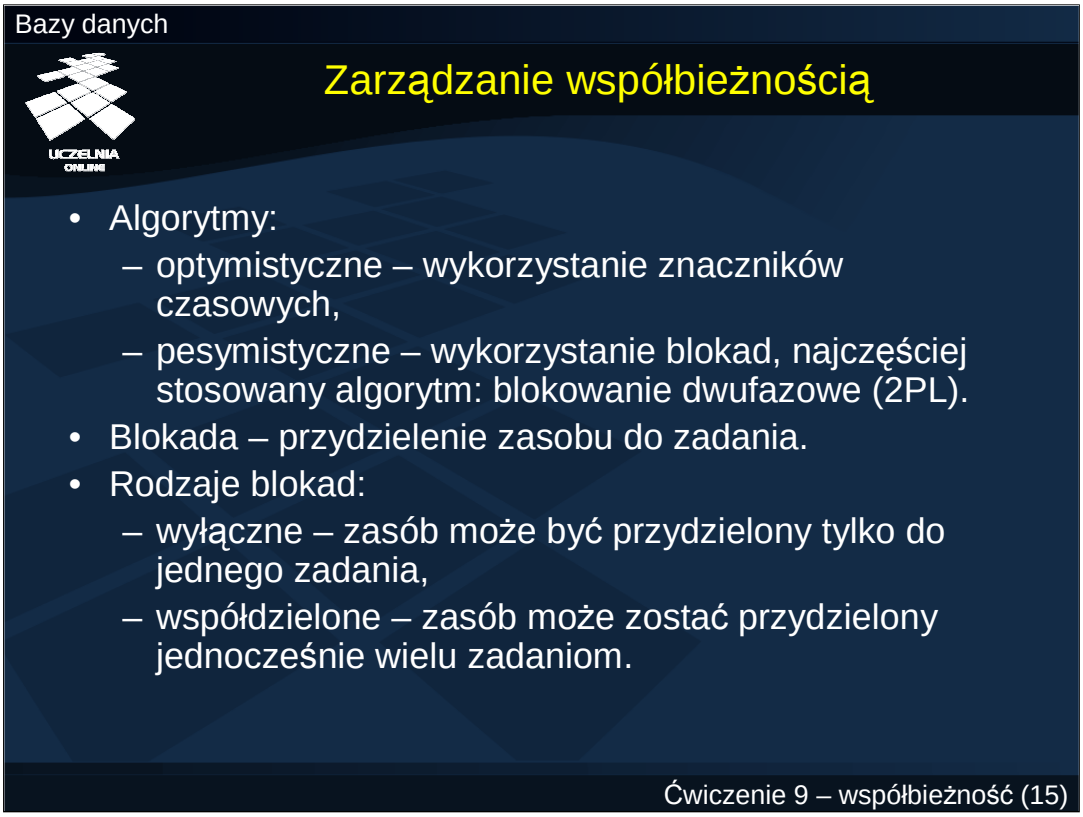
Najniższym poziomem izolacji jest „odczyt danych niezatwierdzonych”, co oznacza, że zmiany, wprowadzone przez operacje w ramach aktywnej transakcji (jeszcze nie zatwierdzonej), są widoczne dla innych, równolegle realizowanych transakcji. W poziomie tym mogą wystąpić wszystkie omówione wcześniej anomalie (z wyjątkiem „utraconej modyfikacji”). Wyższy poziom, „odczyt danych zatwierdzonych” oznacza, że zmiany, wprowadzone przez operacje w ramach transakcji, są widoczne dla innych transakcji dopiero po zatwierdzeniu transakcji. Poziom ten eliminuje anomalię „brudnego odczytu”, dopuszczając występowanie „niepowtarzalnego odczytu” i „fantomów”. Kolejny poziom, „powtarzalny odczyt”, dodatkowo zapobiega anomalii „niepowtarzalnego odczytu”. W najwyższym poziomie o nazwie „uszeregowany”, nie występuje żadna z anomalii.

Należy mieć świadomość, że wraz ze wzrostem poziomu izolacji transakcji maleje stopień współbieżności. Największą współbieżność uzyskuje się dla poziomu „odczyt danych niezatwierdzonych”, jednak przy występowaniu anomalii, najmniejsza współbieżność to poziom „uszeregowany” jednak kosztem najniższej współbieżności. W praktyce rozsądnym kompromisem jest wybór poziomu „odczyt zatwierdzonych danych”.



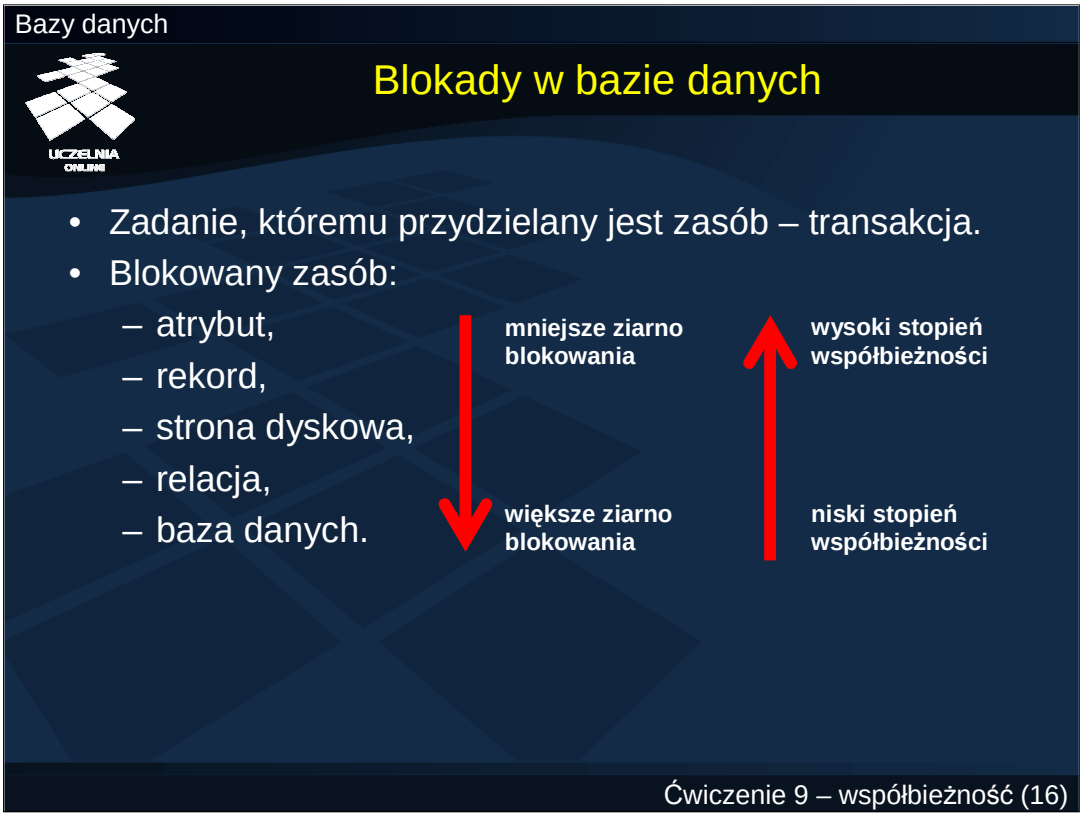
Użytkownik może sam określić stosowany poziom izolacji transakcji. Dla bieżącej transakcji realizuje się to poleceniem SET TRANSACTION ISOLATION LEVEL, po którym podaje się nazwę żądanego poziomu. Poziomem tym będzie posługiwać się tylko bieżąca transakcja, następna zostanie wykonana w poziomie domyślnym dla danego systemu zarządzania bazą danych. W SZBD Oracle można określić jedynie poziom READ COMMITTED i SERIALIZABLE, pozostałe poziomy izolacji nie są dostępne (domyślnym poziomem w SZBD Oracle jest READ COMMITTED). W SZBD Oracle można dodatkowo określić poziom izolacji dla wszystkich transakcji w sesji poleceniem ALTER SESSION SET ISOLATION_LEVEL = <poziom>.

Dodatkowo dla transakcji można określić jej tryb dostępu: tylko do odczytu, wówczas transakcja nie może wykonać żadnego polecenia modyfikującego stan bazy danych, lub normalna, z odczytem i zapisem, czyli dopuszczone są wszystkie operacje w ramach transakcji. Służy do tego polecenie SET TRANSACTION READ ONLY dla transakcji tylko do odczytu i SET TRANSACTION READ WRITE dla transakcji normalnej. Przy transakcji tylko do odczytu baza danych zapewnia dodatkowo spójność na poziomie transakcji.

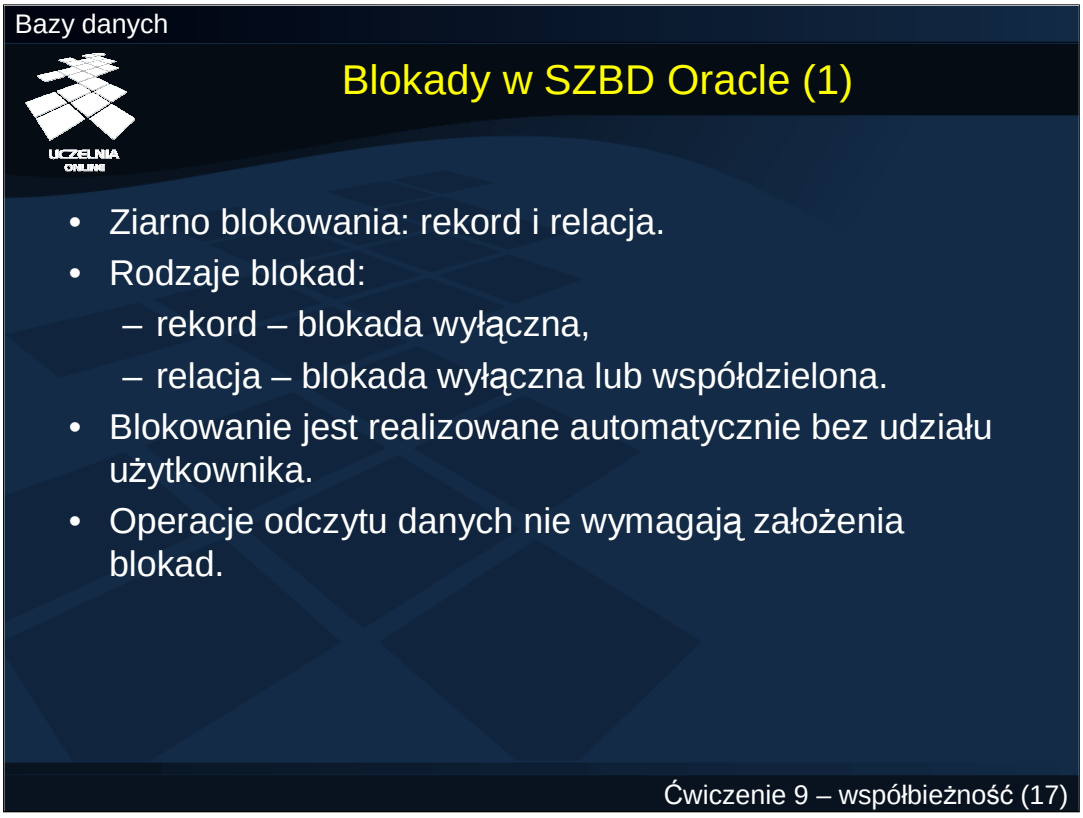


Aby uniknąć występowania anomalii współbieżnego dostępu i zapewnić żądany poziom izolacji transakcji, system zarządzania bazą danych musi zastosować odpowiedni algorytm zarządzania współbieżnym zbiorem transakcji. Wyróżnia się tu dwa podejścia. Algorytmy optymistyczne dopuszczają wysoki stopień współbieżności, jednak kosztem występowania wielu konfliktów między transakcjami. Poprawność wykonania operacji w ramach transakcji jest sprawdzana przez zastosowanie znaczników czasowych operacji. Na tej podstawie system zarządzania bazą danych określa, czy transakcja może zostać zatwierdzona czy musi zostać odrzucona z racji możliwości wprowadzenia niespójności do bazy danych. Z kolei algorytmy pesymistyczne próbują zapobiegać występowaniu anomalii przez eliminację konfliktów między operacjami równoległych transakcji. Mechanizmem tutaj wykorzystywanym są blokady. Najczęściej stosowanym algorytmem pesymistycznym jest blokowanie dwufazowe (ang. two-phase locking). Algorytm ten jest szeroko omawiany w ramach wykładu z przedmiotu.

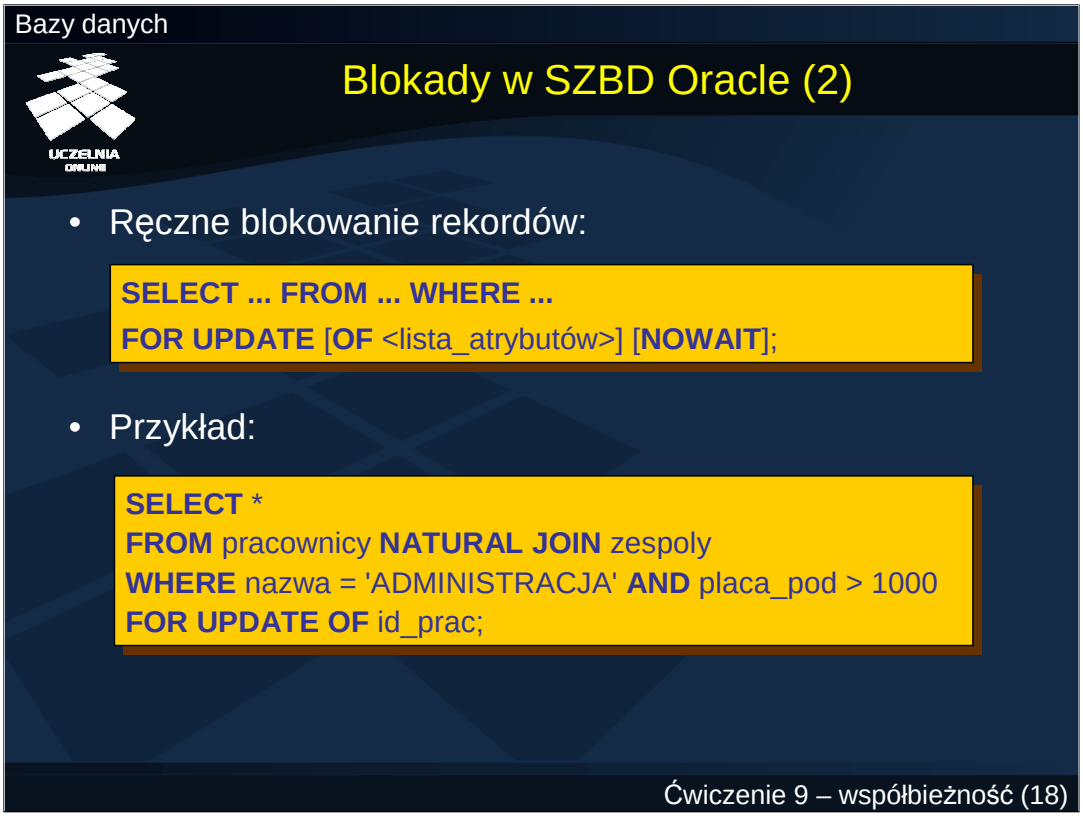
Blokada to przydzielenie określonego zasobu do zadania. Zasób może zostać przydzielony do zadania na wyłączność, mówimy wówczas, że zasób został zablokowany w trybie wyłącznym. Jeśli jednocześnie kilka zadań może uzyskać dostęp do zasobu, wówczas zasób jest blokowany w trybie współdzielonym.



W bazach danych zadaniem, które żąda przydziału zasobu, jest transakcja, natomiast blokowanym zasobem może być atrybut, rekord relacji, strona dyskowa, w której relacja składuje rekordy, cała relacja lub nawet cała baza danych. Rodzaj blokowanego zasobu jest często określany jako „ziarno blokowania”. Najmniejszym ziarnem blokowania będzie atrybut, największym – cała baza danych. Powstaje problem – jak wybrać odpowiednie ziarno blokowania. Należy mieć świadomość, że im większe ziarno blokowania, tym niższy stopień współbieżności w dostępie do zasobu. Jeśli ziarnem blokowania byłaby cała baza danych, wówczas jednocześnie w bazie mogłaby pracować tylko jedna transakcja. Z kolei im mniejsze ziarno blokowania, tym większy stopień komplikacji algorytmu zarządzania współbieżnością, co przekłada się na zwiększone wykorzystanie zasobów systemowych i możliwość degradacji wydajności systemu. Przy wyborze ziarna blokowania trzeba wypracować rozsądny kompromis.



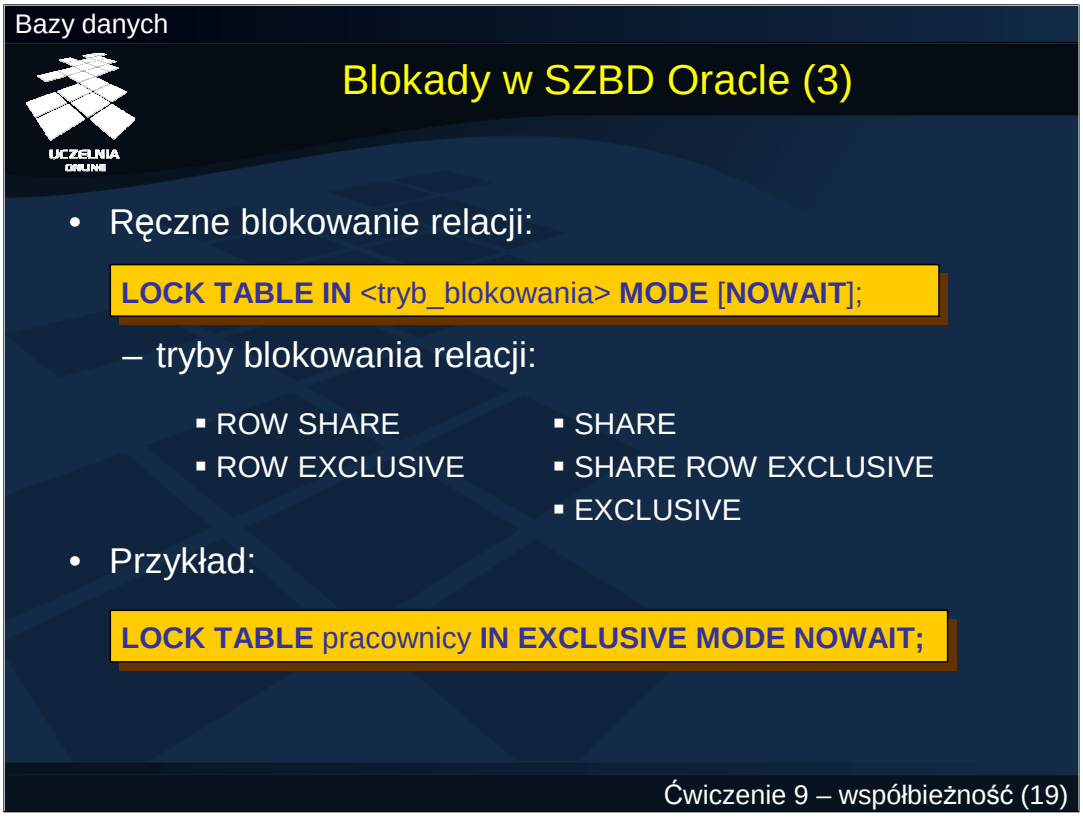
SZBD Oracle korzysta z dwóch ziaren blokowania: blokady dla rekordu i blokady dla całej relacji. Pojedynczy rekord może zostać zablokowany jednocześnie tylko przez jedną transakcję. Z kolei relacja może być zablokowana jednocześnie przez wiele transakcji, mamy wówczas do czynienia ze współdzieloną blokadą relacji, lub przez jedną transakcję, mówimy wtedy o wyłącznej blokadzie relacji. Blokady zakładane są automatycznie w momencie wydawania przez użytkownika poleceń modyfikacji danych. Istnieje również możliwość ręcznego zakładania blokad – polecenia temu służące zostaną zaprezentowane na następnym slajdzie. Ciekawą cechą SZBD Oracle jest brak blokad przy operacjach odczytu danych – powoduje to, że zapytania nigdy nie są blokowane i mogą zostać zawsze wykonane. Blokady zakładane są tylko przy operacjach INSERT, UPDATE i DELETE.



Bieżący i następny slajd przedstawiają polecenia, pozwalające na ręczne zakładanie blokad na rekordach i relacjach.

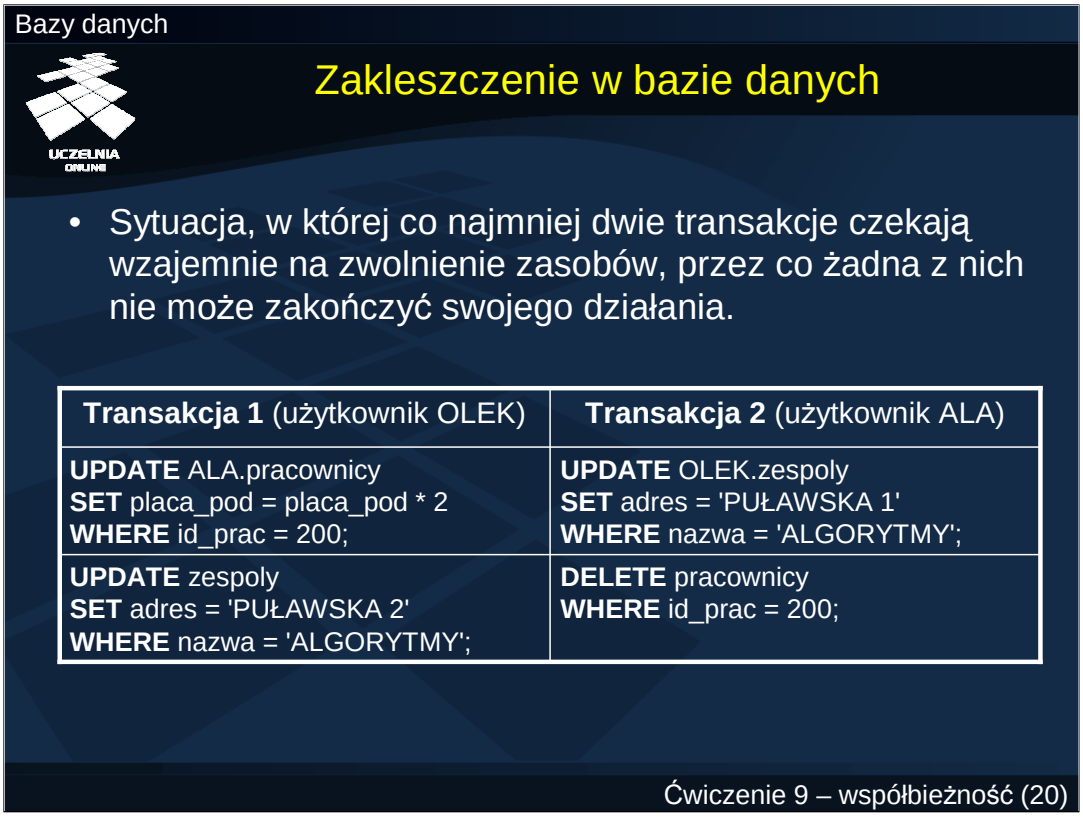
Jeśli do zapytania SQL dołączymy klauzulę FOR UPDATE, na rekordach, które zostaną zwrócone przez zapytanie, system spróbuje założyć blokady, oczywiście o ile rekordy nie są już zablokowane przez inną transakcję. W przypadku uzyskania blokad rekordy zostają przydzielone w trybie wyłącznym do transakcji, dodatkowo na relacji zostaje założona blokada ROW SHARE. Jeśli rekordy zostały wcześniej zablokowane przez inną transakcję, polecenie jest wstrzymywane do czasu, aż blokady na rekordach zostaną zwolnione. Jeśli do polecenia dodamy klauzulę NOWAIT, wówczas w przypadku niemożności zablokowania rekordów polecenie jest przerywane a na konsoli zostaje wypisany komunikat o wystąpieniu błędu. Rozszerzonej postaci zlecenia FOR UPDATE z dodatkową częścią OF <lista_atrybutów> używa się w przypadku, gdy zapytanie odwołuje się do wielu relacji a zablokowane mają zostać rekordy tylko wybranych relacji. W takim przypadku na liście atrybutów po słowie OF należy podać po jednym, dowolnym atrybucie z każdej relacji, której rekordy mają zostać zablokowane. Jeśli klauzula OF z listą atrybutów zostanie pominięta, zablokowane zostaną rekordy wszystkich relacji, biorących udział w zapytaniu.

W zaprezentowanym na slajdzie przykładzie zapytanie łączy rekordy relacji PRACOWNICY i ZESPOLY. Blokady zostaną założone tylko na tych rekordach relacji PRACOWNICY, które wejdą w skład zbioru wynikowego zapytania.



Ręczną blokadę na relacji można uzyskać, stosując polecenie LOCK TABLE IN <tryb> MODE, gdzie <tryb> to jeden z trybów, wymienionych na slajdzie. Nie wchodząc w szczegóły, blokada ROW SHARE jest najlżejszą blokadą relacji, zakładana jest automatycznie przy wykonaniu polecenia SELECT... FOR UPDATE, blokada ROW EXCLUSIVE zakładana jest na relacji, na której wykonano polecenie INSERT, UPDATE lub DELETE. Pozostałe blokady relacji są zakładane w specjalnych sytuacjach. Wszystkie blokady są blokadami współdzielonymi z wyjątkiem blokady EXCLUSIVE, która powoduje zablokowanie całej relacji w trybie wyłącznym przez transakcję.

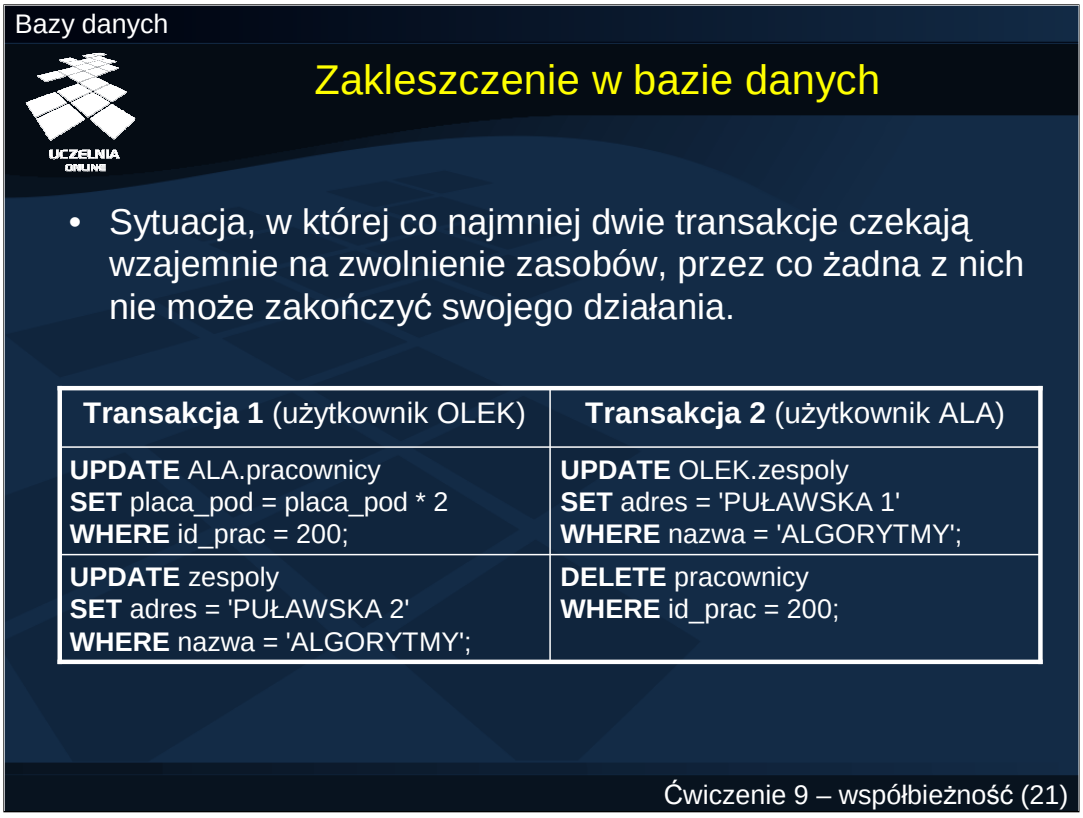
W zaprezentowanym na slajdzie przykładzie transakcja próbuje założyć blokadę wyłączną na relacji PRACOWNICY, jeśli blokada nie zostanie uzyskana, polecenie zostanie przerwane (zapewnia to klauzula NOWAIT).



Poważnym problemem, występującym w bazach danych ze zbiorem współbieżnych transakcji, jest możliwość wystąpienia zakleszczenia. Zakleszczenie to sytuacja, w której aktywne transakcje nie mogą zakończyć swojego działania, gdyż nawzajem blokują sobie zasoby (rekordy i relacje). Zanalizujmy zaprezentowany przykład. Zakładamy, że użytkownicy ALA i OLEK przyznali sobie nawzajem prawa odczytu, modyfikacji i usuwania rekordów ze swoich relacji PRACOWNICY i ZESPOLY.

Użytkownik OLEK rozpoczął swoją transakcję, oznaczmy ją jako transakcja 1, w niej wykonuje pierwsze polecenie, które modyfikuje wartość płacy podstawowej w rekordzie relacji PRACOWNICY ze schematy użytkownika ALA, opisującym pracownika o identyfikatorze 200. Przed modyfikacją rekord zostaje zablokowany w trybie wyłącznym przez transakcję 1. W międzyczasie użytkownik ALA rozpoczął swoją transakcję, oznaczmy ją jako transakcja 2, w pierwszym poleceniu zmodyfikował wartość atrybutu ADRES relacji ZESPOLY ze schematu użytkownika OLEK w rekordzie opisującym zespół o nazwie ALGORYTMY. Rekord zostaje zablokowany przez transakcję 2. Użytkownik OLEK w swojej transakcji 1 kontynuuje pracę i próbuje zmodyfikować adres zespołu o nazwie ALGORYTMY w swojej własnej relacji ZESPOLY. Jednak transakcja nie może uzyskać blokady – rekord został wcześniej zablokowany przez transakcję 2. Transakcja 1 zostaje zawieszona, czekając na zwolnienie blokady.

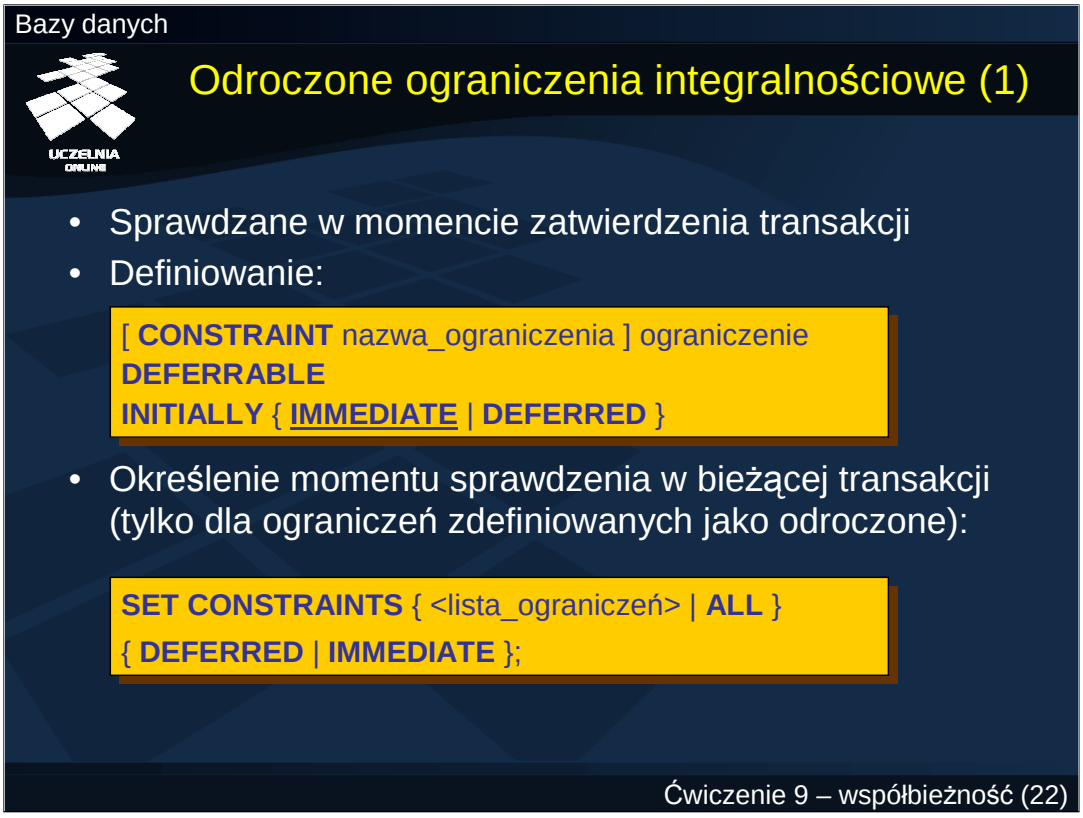
Ciąg dalszy na następnym slajdzie.



Ciąg dalszy z poprzedniego slajdu.

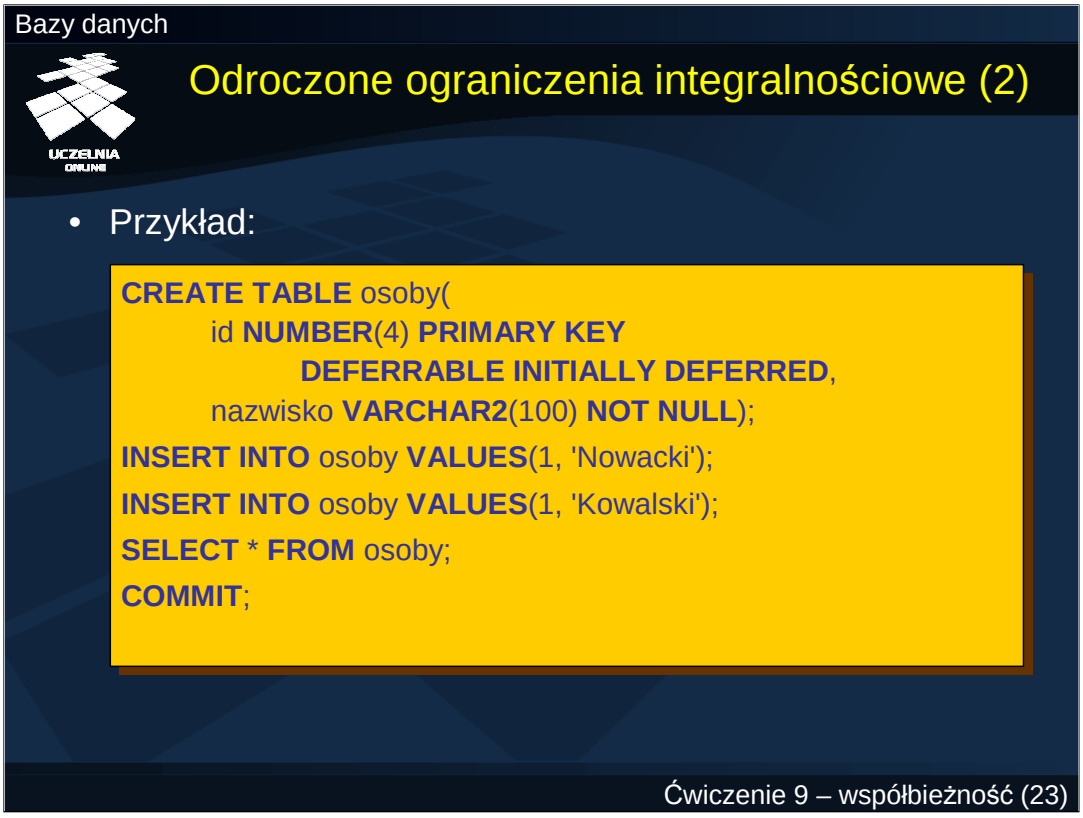
Tymczasem użytkownik ALA w transakcji 2 chce usunąć ze swojej relacji PRACOWNICY rekord opisujący pracownika o identyfikatorze 200. Operacja jednak nie może zostać zrealizowana, gdyż transakcja nie uzyskuje blokady na rekordzie, który ma zostać usunięty – rekord ten został wcześniej zablokowany przez transakcję 1. Transakcja 2 zostaje również zawieszona i czeka na zakończenie transakcji 1.

Obie transakcje zostają zawieszone, czekając na zdjęcie blokad. Żadna nie może kontynuować pracy, gdyż nie może uzyskać blokad. Przez to nie może również zwolnić blokad, na które czeka druga transakcja. Wystąpiło zakleszczenie. System zarządzania bazą danych wykrywa zakleszczenie i rozwiązuje je, wycofując jedno z czekających poleceń zawieszonych transakcji. Dalej to użytkownik musi zdecydować, co dalej ma się dziać z transakcjami: czy je wycofać, czy może kontynuować, jednak wykonując inne polecenia.



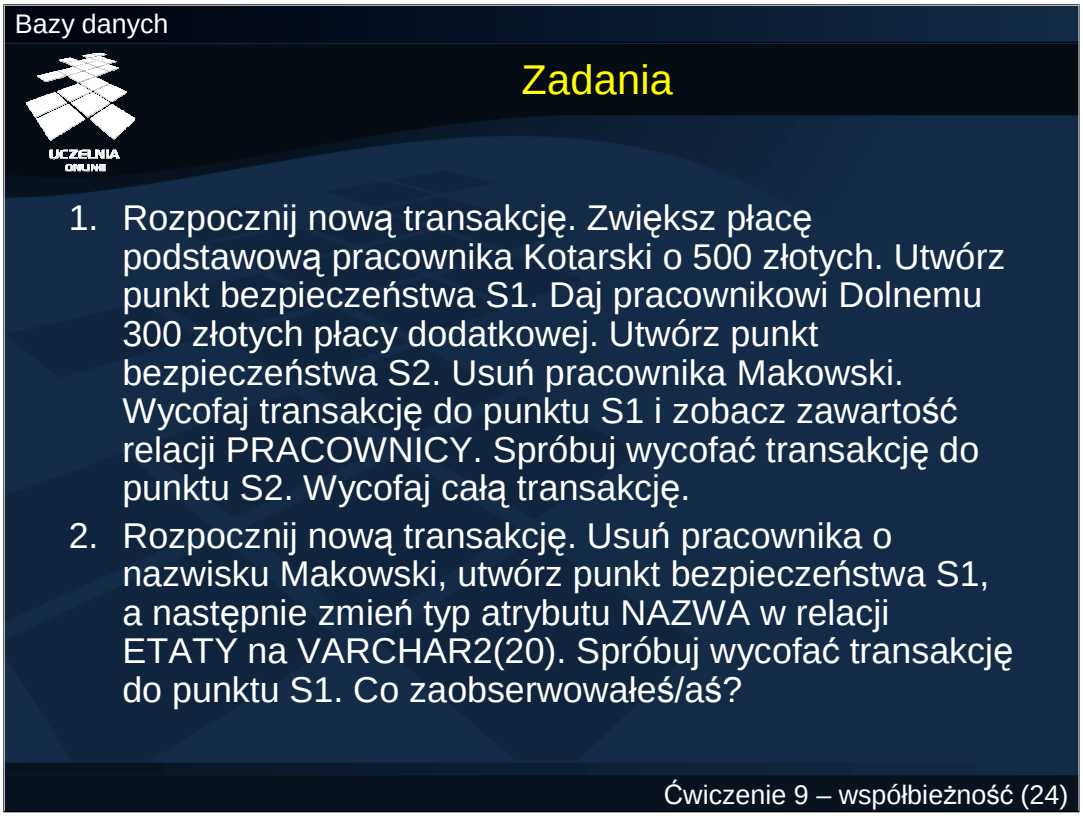
Zwykłe ograniczenia integralnościowe sprawdzane są zaraz po wykonaniu polecenia modyfikującego dane relacji. Można jednak zdefiniować tzw. odroczone ograniczenia integralnościowe, sprawdzane dopiero w momencie zatwierdzenia transakcji, której jedną z operacji była modyfikacja danych relacji. Jeśli stan danych relacji w momencie zatwierdzenia transakcji nie spełnia zasad narzuconych przez ograniczenia intergralnościowe, wówczas cała transakcja zostaje wycofana.

Aby utworzyć ograniczenie odroczone w poleceniu CREATE TABLE lub ALTER TABLE za definicją ograniczenia dodajemy klauzulę DEFERRABLE. Klauzula ta ustala cechę odroczonej weryfikacji ograniczenia. Następnie musimy określić, czy definiowane ograniczenie będzie zaraz po utworzeniu weryfikowane z opóźnieniem (ustala to klauzula INITIALLY DEFERRED) czy też ma się zachowywać jak standardowe ograniczenie, weryfikowane natychmiast po wykonaniu polecenia modyfikacji danych (ustala to klauzula INITIALLY IMMEDIATE). Dla ograniczenia utworzonego z klauzulą DEFERRED można przełączać moment weryfikacji ograniczenia, takiej operacji nie można jednak wykonać dla zwykłego ograniczenia integralnościowego. Realizuje się to poleceniem SET CONSTRAINTS, w którym podaje się nazwy ograniczeń, które mają zmienić moment weryfikacji, i określa moment sprawdzenia (DEFERRED dla sprawdzenia odroczonego i IMMEDIATE dla weryfikacji natychmiastowej). Zamiast listy ograniczeń można podać słowo ALL, wówczas wszystkim ograniczeniom, utworzonym z opcją DEFERRABLE, zostanie zmieniony moment weryfikacji.



Na bieżącym slajdzie zaprezentowano przykład definicji i działania ograniczenia odroczonego.

Pierwsze polecenie tworzy relację OSOBY z atrybutami ID i NAZWISKO, klucz podstawowy relacji, założony na atrybucie ID, jest ograniczeniem odroczonym weryfikowanym z opóźnieniem. Po utworzeniu relacji wykonano dwa identyczne polecenia INSERT, wstawiające do relacji OSOBY dwa rekordy o takich samych wartościach atrybutu ID. Oba polecenia zakończą się sukcesem, mimo że drugie polecenie wstawia rekord, który powiela wartość klucza podstawowego. Wykonane następnie zapytanie potwierdza powielenie wartości w atrybucie ID. Następnie wykonano polecenie zatwierdzenia transakcji. W tym momencie następuje sprawdzenie, czy dane w relacji OSOBY są zgodne z kluczem podstawowym relacji. Wartość klucza podstawowego jest powielona w obu rekordach, tak więc weryfikacja klucza podstawowego zakończyła się niepowodzeniem. Cała transakcja, a więc operacje wstawienia dwóch rekordów do relacji OSOBY, zostaje wycofana.



Bieżący slajd rozpoczyna zbiór zadań, których celem jest utrwalenie wiadomości o mechanizmach zarządzania współbieżnością w bazie danych.

Zadania

3. Utwórz relację TEST z jednym atrybutem ID typu number(5). Zdefiniuj na atrybucie ID klucz podstawowy, którego weryfikacja ma być przeprowadzana po zakończeniu transakcji modyfikującej tabelę TEST. Wstaw do tabeli TEST kilka rekordów z tą samą wartością atrybutu ID. Spróbuj zatwierdzić transakcję. Co zaobserwowałeś/aś? Zmień w bieżącej sesji tryb sprawdzania ograniczenia na natychmiastowy. Spróbuj teraz wstawić kilka rekordów z tą samą wartością atrybutu ID.

Zadania

4. Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji wykonaj polecenie, zwiększające płacę podstawową pracownika o nazwisku Nowak o 10%, w drugiej sesji temu samemu pracownikowi zwiększ płacę podstawową o 20%. Co zaobserwowałeś w drugiej sesji? Zatwierdź zmiany w pierwszej sesji i odczytaj płacę pracownika Nowak. W drugiej sesji odczytaj płacę pracownika Nowak, a następnie zatwierdź zmiany. Odczytaj ponownie płacę pracownika Nowak w pierwszej sesji.

Zadania

5. Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji określ poziom izolacji transakcji na serializable i odczytaj informacje o pracownikach. W drugiej sesji zwiększ dwukrotnie płacę podstawową pracownikowi o nazwisko Opolski. Zatwierdź zmianę. Sprawdź ponownie, jaki obraz relacji PRACOWNICY widzi pierwsza z uruchomionych sesji. Spróbuj w tej sesji zwiększyć płacę podstawową pracownika Opolski o 50%. Co zaobserwowałeś/aś? Wycofaj wprowadzone zmiany.

Zadania

6. Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji zablokuj rekordy opisujące pracowników zespołu o identyfikatorze 20. W drugiej sesji spróbuj zablokować rekordy opisujące pracowników na etacie PROFESOR w trybie bez czekania. Wycofaj transakcję z pierwszej sesji i ponów operację blokowania z drugiej sesji. Następnie w tej samej sesji zablokuj całą relację PRACOWNICY w trybie EXCLUSIVE. Zatwierdź transakcję.
7. Spróbuj doprowadzić do zakleszczenia trzech transakcji.

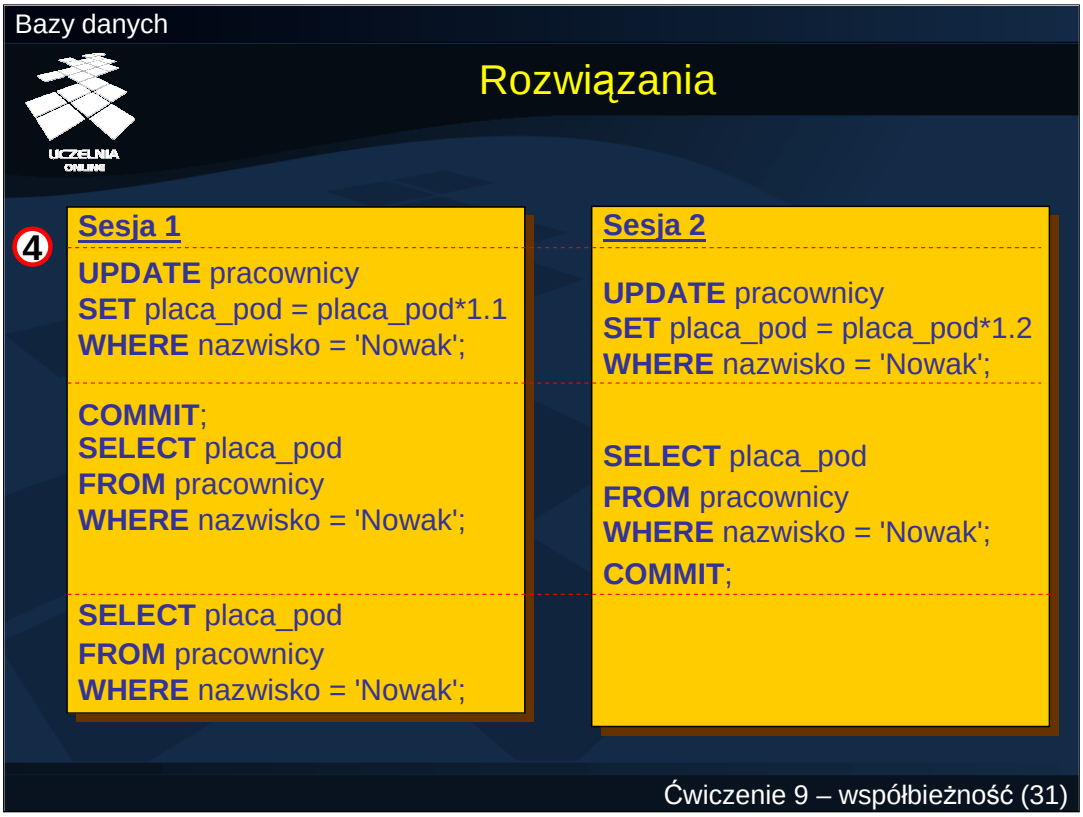
Bieżący slajd przedstawia rozwiązanie zadania (1), którego treść zacytowano poniżej.

(1) Rozpocznij nową transakcję. Zwiększ płacę podstawową pracownika Kotarski o 500 złotych. Utwórz punkt bezpieczeństwa S1. Daj pracownikowi Dolnemu 300 złotych płacy dodatkowej. Utwórz punkt bezpieczeństwa S2. Usuń pracownika Makowski. Wycofaj transakcję do punktu S1 i zobacz zawartość relacji PRACOWNICY. Spróbuj wycofać transakcję do punktu S2. Wycofaj całą transakcję.

Bieżący slajd przedstawia rozwiązania zadań (2) i (3), których treść zacytowano poniżej.

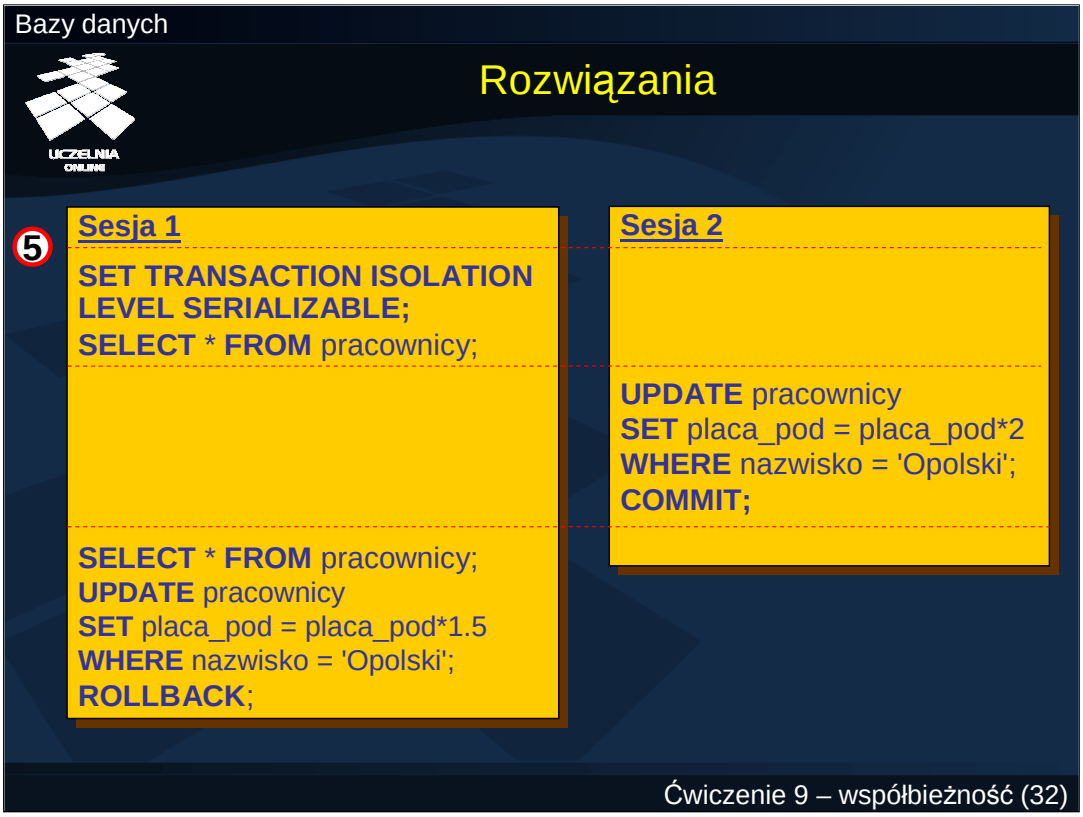
(2) Rozpocznij nową transakcję. Usuń pracownika o nazwisku Makowski, utwórz punkt bezpieczeństwa S1, a następnie zmień typ atrybutu NAZWA w relacji ETATY na VARCHAR2(20). Spróbuj wycofać transakcję do punktu S1. Co zaobserwowałeś/aś?

(3) Utwórz relację TEST z jednym atrybutem ID typu number(5). Zdefiniuj na atrybucie ID klucz podstawowy, którego weryfikacja ma być przeprowadzana po zakończeniu transakcji modyfikującej tabelę TEST. Wstaw do tabeli TEST kilka rekordów z tą samą wartością atrybutu ID. Spróbuj zatwierdzić transakcję. Co zaobserwowałeś/aś? Zmień w bieżącej sesji tryb sprawdzania ograniczenia na natychmiastowy. Spróbuj teraz wstawić kilka rekordów z tą samą wartością atrybutu ID.



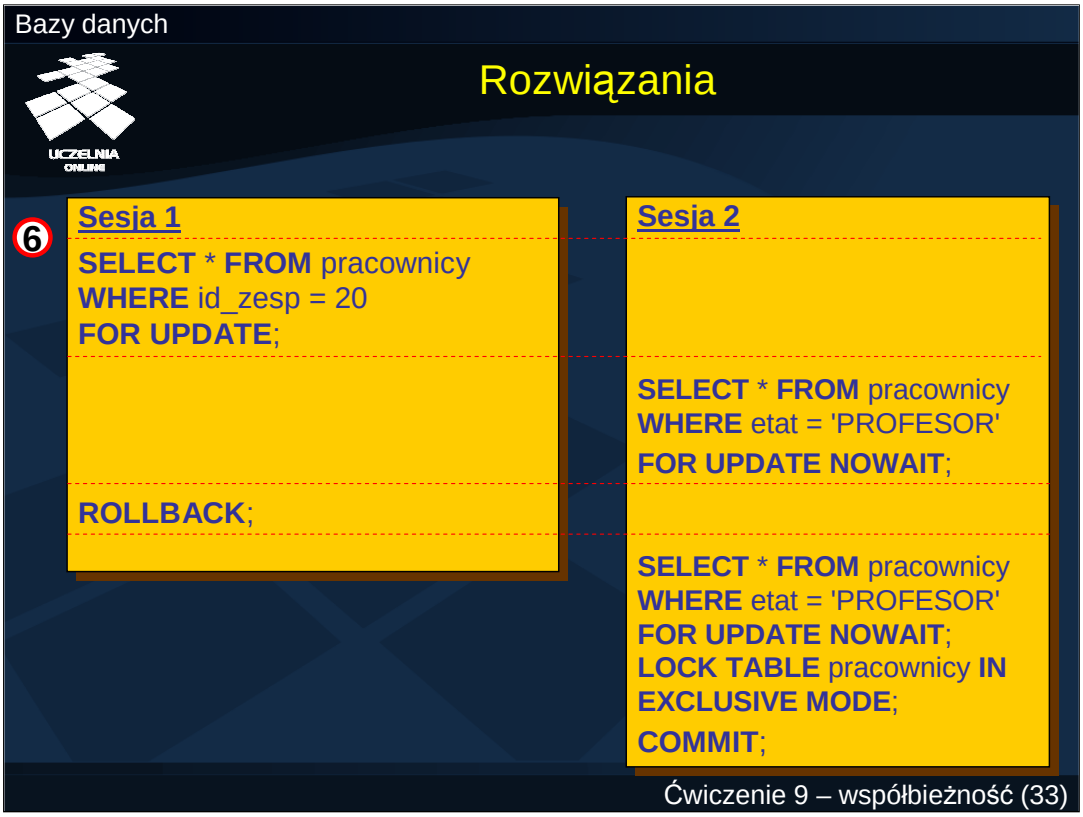
Bieżący slajd przedstawia rozwiązanie zadania (4), którego treść zacytowano poniżej.

(4) Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji wykonaj polecenie, zwiększające płacę podstawową pracownika o nazwisku Nowak o 10%, w drugiej sesji temu samemu pracownikowi zwiększ płacę podstawową o 20%. Co zaobserwowałeś w drugiej sesji? Zatwierdź zmiany w pierwszej sesji i odczytaj płacę pracownika Nowak. W drugiej sesji odczytaj płacę pracownika Nowak, a następnie zatwierdź zmiany. Odczytaj ponownie płacę pracownika Nowak w pierwszej sesji.



Bieżący slajd przedstawia rozwiązanie zadania (5), którego treść zacytowano poniżej.

(5) Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji określ poziom izolacji transakcji na serializable i odczytaj informacje o pracownikach. W drugiej sesji zwiększ dwukrotnie płacę podstawową pracownikowi o nazwisko Opolski. Zatwierdź zmianę. Sprawdź ponownie, jaki obraz relacji PRACOWNICY widzi pierwsza z uruchomionych sesji. Spróbuj w tej sesji zwiększyć płacę podstawową pracownika Opolski o 50%. Co zaobserwowałeś/aś? Wycofaj wprowadzone zmiany.

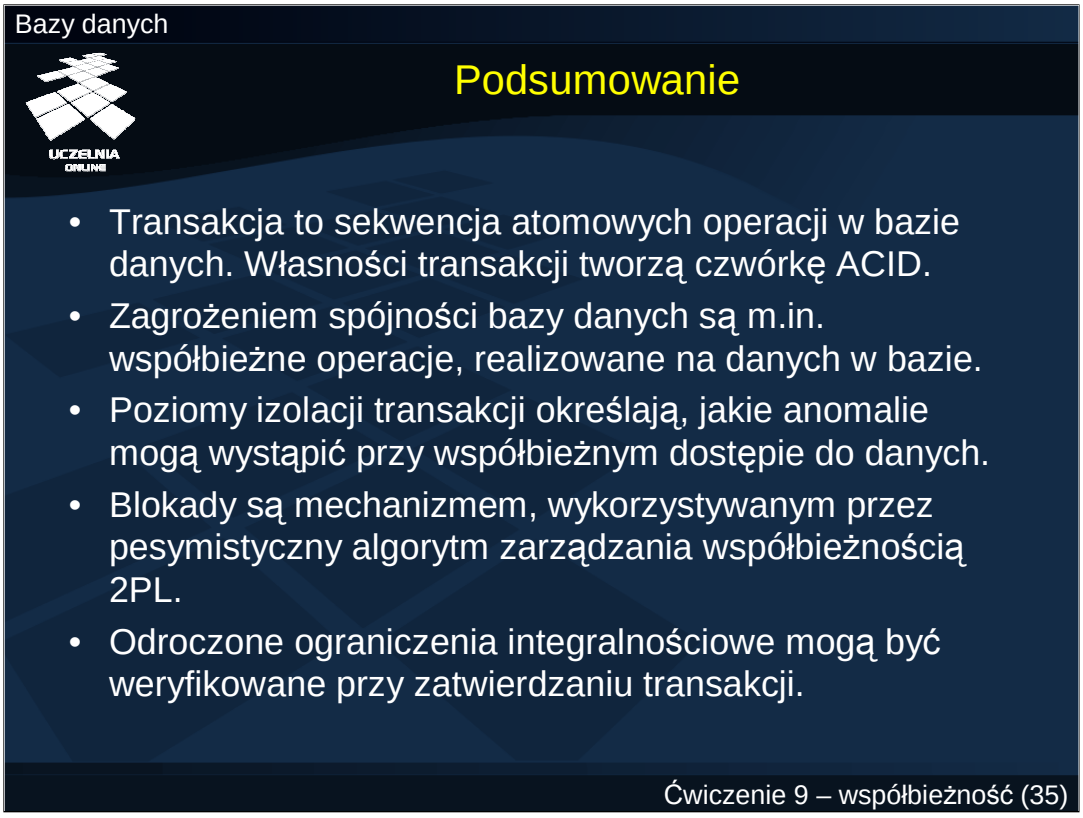


Bieżący slajd przedstawia rozwiązanie zadania (6), którego treść zacytowano poniżej.

(6) Uruchom dwie sesje tego samego użytkownika. W pierwszej sesji zablokuj rekordy opisujące pracowników zespołu o identyfikatorze 20. W drugiej sesji spróbuj zablokować rekordy opisujące pracowników na etacie PROFESOR w trybie bez czekania. Wycofaj transakcję z pierwszej sesji i ponów operację blokowania z drugiej sesji. Następnie w tej samej sesji zablokuj całą relację PRACOWNICY w trybie EXCLUSIVE. Zatwierdź transakcję.

Bieżący slajd przedstawia rozwiązanie zadania (7), którego treść zacytowano poniżej.

(7) Spróbuj doprowadzić do zakleszczenia trzech transakcji.



W bieżącym ćwiczeniu zaprezentowano koncepcję transakcji w bazie danych jako podstawowej jednostki pracy użytkownika bazy danych. Następnie zdefiniowano pojęcie spójności bazy danych i wskazano sytuacje, które mogą powodować utratę spójności przez bazę danych. Dalej zaprezentowano anomalie współbieżnego dostępu i poziomy izolacji transakcji, które mają za zadanie zapewnienie eliminacji anomalii. W kolejnej części ćwiczenia zaprezentowano koncepcję blokad w bazie danych. Na zakończenie zaprezentowano odroczone ograniczenia integralnościowe, które mogą być weryfikowane z opóźnieniem, w momencie zatwierdzania transakcji.

Każde z omówionych w ćwiczeniu zagadnień zostało utrwalone przez serię zadań.