PODSTAWY SQL

WYKŁAD 4

MGR INŻ. BARTŁOMIEJ DETTLAFF

SENIOR JAVA DEVELOPER

AGENDA

- Transakcje
- ACID
- Podzapytania
- Podzapytanie kontra łączenie
- Like, Between, Distinct, In

Czym są transakcje?

Transakcje to mechanizm, który pozwala na grupowanie jednego lub więcej poleceń SQL w jednej jednostce pracy. Transakcje są używane do zapewnienia integralności danych w bazie danych.

ACID - co to oznacza?

Akronim opisujący cztery podstawowe właściwości transakcji w systemach baz danych, które gwarantują niezawodność przetwarzania danych.

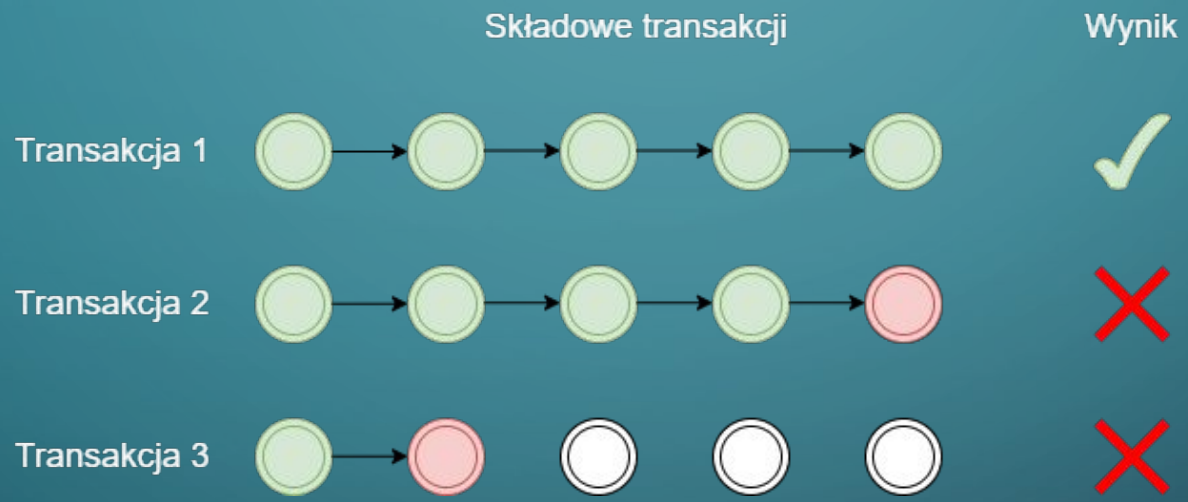
Atomicity (Atomowość)

Gwarantuje, że wszystkie operacje w transakcji są wykonywane kompletnie albo wcale.

Oznacza to, że jeśli którykolwiek element transakcji zawiedzie, cała transakcja jest anulowana (cofana). Jeśli wszystkie elementy są pomyślne, transakcja jest zatwierdzana.

Przykład: Jeśli transakcja bankowa przenosi środki między dwoma kontami, obie operacje (odjęcie środków z jednego konta i dodanie ich do drugiego) muszą zostać zakończone pomyślnie. Jeśli jedna z nich zawiedzie, obie są cofane.

Atomicity (AtomowoŚĆ) - prezentacja graficzna



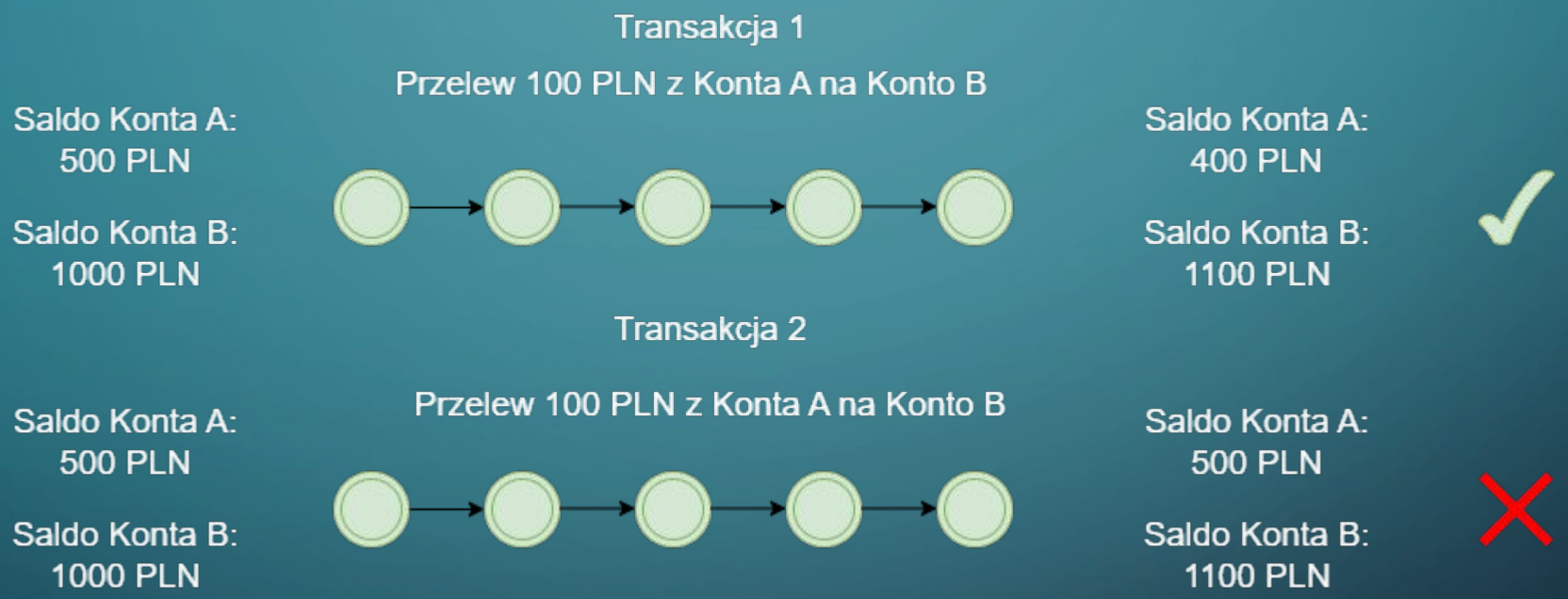
Consistency (Spójność)

Zapewnia, Że baza danych zawsze przechodzi od jednego spójnego stanu do drugiego.

Transakcja przekształca bazę danych z jednego stanu spełniającego pewne reguły integralności w inny taki stan. Jeśli transakcja jest zakończona pomyślnie, baza danych musi być w spójnym stanie.

Przykład: Jeśli suma środków na dwóch kontach musi pozostać stała, transakcja przenoszenia środków między kontami musi zapewnić zachowanie tej reguły.

Consistency (SpójnoŚĆ) - prezentacja graficzna



Isolation (Izolacja)

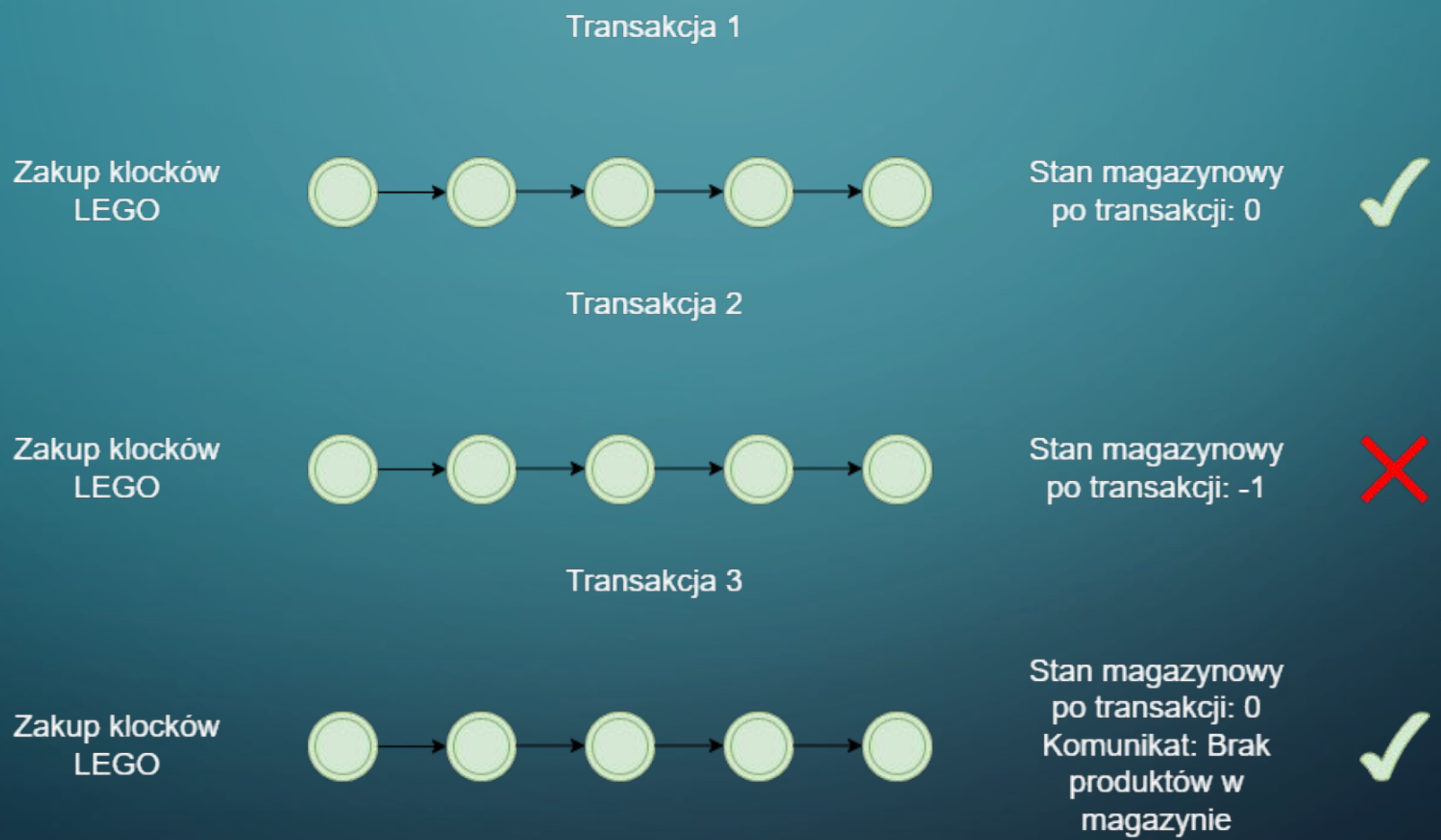
Zapewnia, Że jednoczesne transakcje nie zakłócają się nawzajem.

Efekt jednej niezakończonej transakcji jest niewidoczny dla innych równoległych transakcji.

Ochrona przed problemami takimi jak brudne odczyty, niespójne odczyty (non-repeatable reads) oraz fantomy (phantom reads) zależy od poziomu izolacji wybranego dla transakcji.

Przykład: Jeśli dwie osoby jednocześnie próbują zakupić ostatni produkt w sklepie internetowym, izolacja zapewni, Że tylko jedna z nich dokona zakupu, a druga otrzyma komunikat o braku dostępności produktu.

Isolation (Izolacja) - prezentacja graficzna



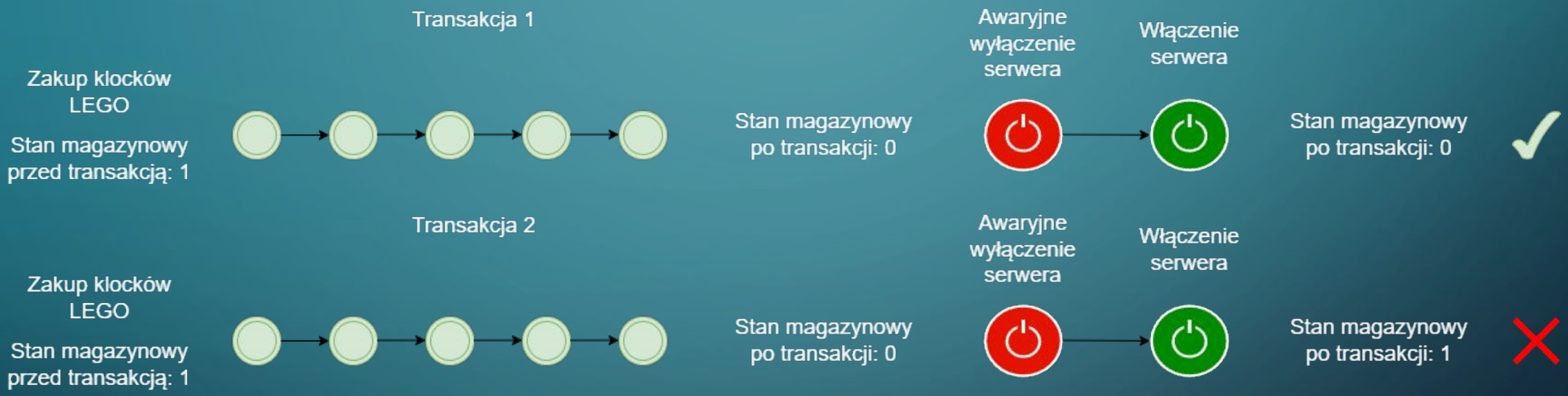
Durability (Trwałość)

Gwarantuje, że raz zatwierdzone zmiany w transakcji są trwałe i nie zostaną utracone w przypadku awarii, takich jak przerwanie zasilania czy awaria systemu.

W praktyce oznacza to, że dane są zapisywane na dysku w taki sposób, że mogą być odzyskane nawet po awarii.

Przykład: Jeśli transakcja bankowa została zatwierdzona i Środki zostały przekazane, nie można ich stracić nawet w przypadku awarii systemu.

Durability (TrwałoŚĆ) - prezentacja graficzna



Przykładowy scenariusz transakcji - bankowość

Przeniesienie środków między kontami: Gdy klient przenosi środki z jednego konta na drugie, musisz najpierw odjąć kwotę z jednego konta, a następnie dodać ją do drugiego. Jeśli jedno z tych działań by się nie powiodło (na przykład z powodu awarii systemu), transakcja SQL zapewniłaby, że obie operacje są anulowane, aby zapobiec niespójności stanu kont.

Przykładowy scenariusz transakcji - rezerwacje

Rezerwacja miejsca: Jeśli klient rezerwuje pokój w hotelu lub miejsce w samolocie, system musi zaktualizować dostępność i jednocześnie zarejestrować rezerwację dla klienta. Jeśli rezerwacja nie zostanie zarejestrowana prawidłowo, miejsce nie powinno zostać zarezerwowane.

Przykładowy scenariusz transakcji - produkcja

Aktualizacja magazynu: Gdy produkt jest produkowany lub dostarczany, może to wpłynąć na różne części magazynu (np. surowce, produkty końcowe). Wszystkie te aktualizacje muszą zostać dokonane spójnie.

Przykładowy scenariusz transakcji - zakupy

Składanie zamówienia: Gdy klient składa zamówienie, kilka rzeczy musi się stać jednocześnie - aktualizacja stanu magazynowego, zapisanie szczegółów zamówienia, potwierdzenie płatności itp. Jeśli jedna z tych operacji się nie powiedzie, całe zamówienie powinno zostać anulowane.

Polecenia związane z transakcjami cz. 1

START TRANSACTION lub BEGIN: Rozpoczyna nową transakcję. Po tym poleceniu wszystkie kolejne operacje są częścią tej samej transakcji, aż do jej zakończenia.

COMMIT: Zatwierdza wszystkie operacje od czasu rozpoczęcia transakcji. Po wykonaniu tego polecenia, wszystkie zmiany w transakcji stają się trwałe.

ROLLBACK: Anuluje wszystkie operacje od czasu rozpoczęcia transakcji. Jeżeli pojawią się problemy podczas wykonywania operacji (na przykład błąd), możemy użyć tego polecenia, aby przywrócić stan bazy danych do momentu przed rozpoczęciem transakcji.

Polecenia związane z transakcjami cz. 2

SAVEPOINT nazwa_savepointa: Ustawia punkt kontrolny w transakcji, do którego można potem wrócić. Dzięki temu, jeśli chcemy cofnąć tylko część operacji, ale nie całą transakcję, możemy to zrobić.

ROLLBACK TO SAVEPOINT nazwa_savepointa: Cofa zmiany do określonego punktu kontrolnego (savepoint) w transakcji.

RELEASE SAVEPOINT nazwa_savepointa: Usuwa określony punkt kontrolny. Jeśli nie planujemy wracać do danego punktu kontrolnego, możemy go usunąć.

SET TRANSACTION: Pozwala na zmianę właściwości transakcji, takich jak poziom izolacji.

Poziomy izolacji

Poziomy izolacji transakcji określają, jak transakcje oddziałują na siebie nawzajem, szczególnie w kontekście konkurencyjnego dostępu do danych. W SQL standard określa cztery poziomy izolacji, każdy z nich oferuje różne gwarancje i jest podatny na różne typy anomalii.

Poziomy izolacji - macierz zależności

	Dirty read	Non-repeatable reads	Phantom
Read uncommited	Możliwe	Możliwe	Możliwe
Read commited	Nie występują	Możliwe	Możliwe
Repeatable read	Nie występują	Nie występują	Możliwe
Serializable	Nie występują	Nie występują	Nie występują

Anomalie

Mogą wystąpić w bazach danych podczas współbieżnego (równoległego) wykonywania transakcji.

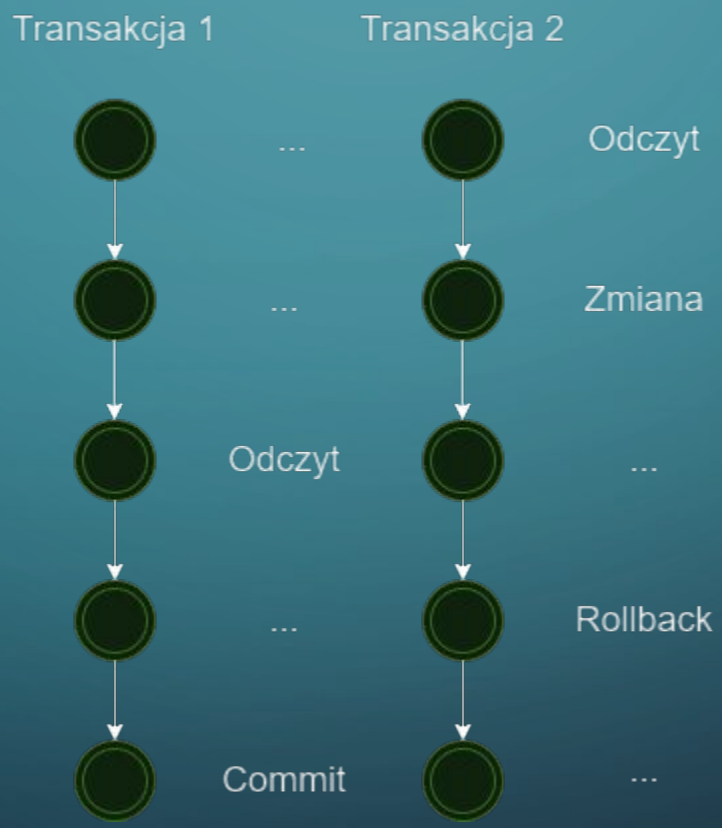
- Brudne odczyty
- Niespójne odczyty
- Fantomy

Dirty Reads

Występuje, gdy jedna transakcja odczytuje dane zmienione przez inną transakcję, która jeszcze nie została zatwierdzona (commit).

Problem polega na tym, Że jeśli ta druga transakcja zostanie ostatecznie cofnięta (rollback), pierwsza transakcja odczytała dane, które nigdy nie zostały oficjalnie zapisane w bazie, czyli "brudne" dane.

Dirty Reads - wizualizacja

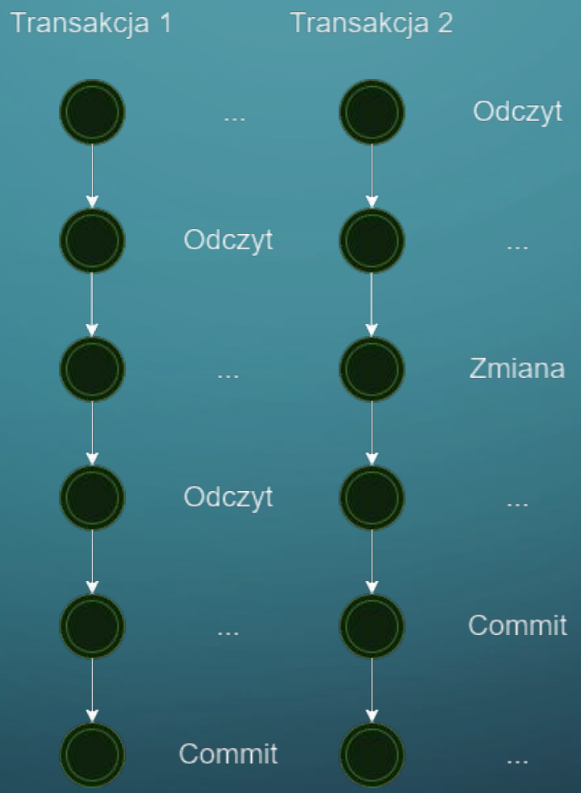


Non-Repeatable Reads

Występuje, gdy w trakcie trwania jednej transakcji, odczytywany wielokrotnie ten sam wiersz daje różne wyniki, ponieważ wiersz ten został zmodyfikowany przez inną transakcję.

Przykład: Gdy transakcja A odczytuje wiersz, następnie transakcja B modyfikuje ten wiersz i dokonuje commitu, a następnie transakcja A ponownie odczytuje ten sam wiersz i otrzymuje różne dane.

Non-Repeatable Reads - wizualizacja



Phantom Reads

Występują, gdy w trakcie trwania jednej transakcji nowe wiersze są dodawane lub usuwane przez inną transakcję, powodując, że kolejne zapytania w pierwotnej transakcji zwracają różne zestawy wierszy.

Przykład: Gdy transakcja A odczytuje zbiór wierszy spełniających pewne kryterium, a transakcja B w międzyczasie dodaje nowe wiersze spełniające to kryterium i dokonuje commitu. Gdy transakcja A ponownie wykona to samo zapytanie, zobaczy dodatkowe "fantomowe" wiersze.

Phantom Reads - wizualizacja

Read Uncommitted

Najniższy poziom izolacji.

Transakcje mogą odczytywać niezatwierdzone dane z innych transakcji (tzw. brudne odczyty).

Jest podatny na wszystkie główne problemy: brudne odczyty, niespójne analizy (non-repeatable reads) oraz fantomy (phantom reads).

Read Committed

Transakcje mogą odczytywać tylko zatwierdzone dane z innych transakcji.

Zabezpiecza przed brudnymi odczytami.

Nadal jest podatny na niepowtarzalne odczyty oraz fantomy.

Repeatable Read

Oferuje wszystkie gwarancje poziomu "Read Committed" i dodatkowo zapewnia, że jeśli transakcja odczytuje wiersz danych, ten wiersz nie zostanie zmieniony przez inną transakcję do czasu zakończenia pierwszej transakcji.

Zabezpiecza przed brudnymi odczytami i niepowtarzalnymi odczytami.

Jednakże w niektórych systemach może nadal być podatny na fantomy.

Serializable

Najwyższy poziom izolacji.

Gwarantuje, że transakcje są wykonywane w sposób szeregowy, tak jakby były wykonywane jedna po drugiej, co eliminuje wszystkie powyższe anomalie.

Zapobiega brudnym odczytom, niepowtarzalnym odczytom oraz fantomom.

Podzapytania

Podzapytania, nazywane również subquery, są zapytaniami osadzonymi wewnątrz innego zapytania. Mogą one dostarczać dane, które zostaną użyte przez zewnętrzne zapytanie do dalszego przetwarzania. Podzapytania mogą pojawiać się w różnych miejscach zapytania głównego, w zależności od tego, jakiego rodzaju wartości są potrzebne.

Podzapytania - przykład 1, do filtrowania

Załóżmy, że chcemy znaleźć wszystkich klientów, którzy złożyli zamówienie. Możemy to osiągnąć poprzez sprawdzenie, czy CustomerID danego klienta pojawia się w tabeli Orders.

```
SELECT CustomerID, CustomerName

FROM Customers

WHERE CustomerID IN (

SELECT DISTINCT CustomerID FROM Orders)
```

Podzapytania - przykład 2, jako kolumna

Zapytanie ile zamówień zostało wykonanych przez poszczególne firmy.

```
SELECT CustomerName,
(SELECT COUNT(*) FROM Orders
WHERE Orders.CustomerID = Customers.CustomerID) AS OrderCount
FROM Customers;
```

Podzapytania - przykład 3, jako inna tabela

Zapytanie ile zamówień zostało wykonanych przez poszczególne firmy.

```
SELECT CustomerName, OrderCount
FROM Customers,
(SELECT CustomerID, COUNT(*) as OrderCount FROM Orders GROUP BY CustomerID) as Subquery
WHERE Customers.CustomerID = Subquery.CustomerID;
```

Podzapytania versus łączenia cz. 1

	Podzapytanie	Łączenie (join)
Definicja	Podzapytania to zapytania osadzone wewnątrz innego zapytania. Mogą one dostarczać wartości do zewnętrznego zapytania.	Łączenia łączą wiersze z dwóch lub więcej tabel na podstawie wspólnego kryterium (zwykle klucza obcego i głównego klucza).
Zastosowanie	Podzapytania są często używane, gdy potrzebujemy filtrować lub sortować dane na podstawie wartości, które są wynikiem innego zapytania.	Idealne do łączenia powiązanych tabel i prezentowania skonsolidowanych informacji.

Podzapytania versus łączenia cz. 2

	Podzapytanie	Łączenie (join)
Złożoność	Mogą być bardziej czytelne dla początkujących, gdyż często prezentują krok po kroku proces przetwarzania danych.	Mogą być mniej intuicyjne dla początkujących, ale są bardziej standardowym podejściem do łączenia tabel. W przypadku wielu związków między tabelami, zapytania mogą stać się skomplikowane.
Wydajność	W niektórych przypadkach mogą być mniej wydajne niż łączenia, szczególnie gdy operujemy na dużych zbiorach danych.	Zwykle bardziej wydajne niż podzapytania, zwłaszcza w systemach RDBMS zoptymalizowanych do wykonywania operacji join.

Proponowane zastosowanie

Scenariusz	Opis	Rozwiązanie
Dla pojedynczych wartości	Jeśli jest potrzeba uzyskania pojedynczej wartości (np. średniej, maksimum) z innej tabeli jako części kryterium filtrującego.	Podzapytanie
Dla skomplikowanych relacji	Jeśli łączone jest wiele tabel w celu uzyskania skomplikowanego zestawu wyników.	Łączenie (join)
Dla optymalizacji	Jeśli istnieją problemy z wydajnością.	Podzapytanie/Łączenie (join)

Like

Klauzula LIKE służy do przeszukiwania wzorców w tekście. Jest używana głównie w połączeniu z kolumnami typu tekstowego, takimi jak VARCHAR czy CHAR.

Dwa symbole specjalne:

% - Reprezentuje zero, jedno lub wiele znaków.

_ - Reprezentuje dokładnie jeden znak.

Like - przykłady

Znajdź wszystkie produkty zaczynające się na literę "A":

```
SELECT * FROM Products WHERE ProductName LIKE 'A%';
```

Znajdź produkty mające dokładnie dwa znaki w nazwie:

```
SELECT * FROM Products WHERE ProductName LIKE '__';
```

Znajdź produkty, które w nazwie mają literę "a" jako drugi znak:

```
SELECT * FROM Products WHERE ProductName LIKE '_a%';
```

Distinct

Klauzula DISTINCT jest używana do eliminowania duplikatów z wyników zapytania.

Distinct - przykłady

Znajdź wszystkie unikatowe kraje klientów:

SELECT DISTINCT Country FROM Customers;

Znajdź unikatowe kombinacje kraju i miasta klientów:

SELECT DISTINCT Country, City FROM Customers;

Liczba unikatowych krajów klientów:

SELECT COUNT(DISTINCT Country) FROM Customers;

In

Klauzula IN jest używana do filtrowania wyników w oparciu o listę wartości.

In - przykłady

Znajdź wszystkich klientów z USA, UK lub Kanady:

```
SELECT * FROM Customers WHERE Country IN ('USA', 'UK', 'Canada');
```

Znajdź wszystkie zamówienia z produktami o ProductID 1, 5 lub 10:

```
SELECT * FROM Orders WHERE ProductID IN (1, 5, 10);
```

Znajdź wszystkich klientów, którzy złożyli zamówienie:

```
SELECT * FROM Customers WHERE CustomerID IN (SELECT DISTINCT
CustomerID FROM Orders);
```

Between

Klauzula BETWEEN jest używana do filtracji wyników w zakresie określonych wartości. Działa na liczby, daty i teksty. Klauzula BETWEEN jest włączająca, co oznacza, Że obejmuje obie określone wartości graniczne.

Between - przykłady

Znaleźć produkty, których cena znajduje się w zakresie 10 do 50:

SELECT * FROM Products WHERE Price BETWEEN 10 AND 50;

Odpowiednik bez użycia BETWEEN:

SELECT * FROM Products WHERE Price >= 10 AND Price <= 50;

Znaleźć zamówienia złożone między 1 stycznia 2020 roku a 31 grudnia 2020 roku:

SELECT * FROM Orders WHERE OrderDate BETWEEN '2020-01-01' AND '2020-12-31';

Aby znaleźć produkty, których nazwa zaczyna się na litery od "A" do "C" (włącznie):

SELECT * FROM Products WHERE ProductName BETWEEN 'A%' AND 'C%';