

Tematy zajęć:

- 37. T-SQL-skład.
- 38. Definicja transakcji i jej właściwości.
- 39. Poziomy izolowania transakcji. Współbieżność.

Omówienie materiału na podstawie bazy Northwind

37. T-SQL-skład.

Czym jest Transact-SQL?



Transact-SQL, często nazywany po prostu **T-SQL** to rozszerzenie języka **SQL** stosowane obecnie w rozwiązaniach bazodanowych firmy Microsoft. T-SQL jest **językiem proceduralnym**. Rozszerza możliwości tradycyjnego pisania zapytań SQL. Możemy za jego pomocą stosować typowe elementy programowania jak **pętle**, **instrukcje warunkowe**, możemy także tworzyć własne funkcje, procedury i operować na **zmiennych**. Transact-SQL możemy łączyć ze zwykłym językiem SQL, co rozszerza nam możliwości korzystania z bazy danych.

W języku T-SQL ciąg poleceń bezpośrednio kierowany do serwera powinien kończyć się słowem kluczowym GO. Ich wykonanie na poziomie edytora nastąpi po wybraniu przycisku Execute lub wciśnięciu klawisza F5 na klawiaturze. Pojedyncza instrukcja może kończyć się średnikiem, ale brak średnika nie jest błędem. Instrukcje mogą być pisane w jednej linii. Jednak dla większej czytelności kodu zaleca się umieszczanie każdej z nich w oddzielnej linii i stosowanie wcięć.

Transact-SQL daje nam przede wszystkim dodatkowe możliwości. Samego SQL nie można do końca nazwać językiem programowania, T-SQL już tak. Za pomocą T-SQL możesz **automatyzować** swoje zapytania na bazie danych, tworzyć dynamiczne rozwiązania, a nawet zaprogramować bazę tak, by Twoja praca wykonywała się sama. Transact posiada m.in. takie elementy programistyczne jak:

- Pętle i instrukcje warunkowe
- Rozbudowaną obsługę transakcji
- Obsługę błędów / wyjątków
- Możliwość tworzenia własnych funkcji i procedur
- Zaawansowane funkcjonalności bazodanowe jak kursory, wyzwalacze itp.

http://www.informatyka.orawskie.pl/?pl_t-sql,192

Zmienne

Zmienne deklarowane przez użytkownika są zmiennymi lokalnymi i istnieją tylko w obrębie skryptu. Muszą być poprzedzone znakiem @. Deklaracja zmiennych jest realizowana za pomocą instrukcji:

DECLARE @zmienna typ_danych

W jednej instrukcji można deklarować wiele zmiennych.

Przykład

DECLARE @Imie varchar (20), @Nazwisko varchar (20);

W podanym przykładzie zostały zadeklarowane dwie zmienne lokalne: @ imie, @nazwisko.

Zmienne systemowe

Zmienne systemowe oznaczone są dwoma znakami @@.

Przykłady zmiennych systemowych:

@@ERROR – zwraca numer ostatniego błędu (0 – brak błędu),

@@FETCH_Status – ustala, czy kursor pobrał wiersz (0 – pobrał),

@@IDENTITY – zwraca ostatnio wygenerowaną wartość,

@@ROWCOUNT - zwraca liczbę wierszy, dla których została wykonana instrukcja SQL
@@VERSION - zwraca informację o wersji SQL,
@@TRANCOUNT - zwraca liczbę otwartych transakcji.

Zmiennej można przypisać wartość za pomocą instrukcji SELECT:

```
SELECT @zmienna=wyrażenie [FROM tabela]
```

Przykład

```
SELECT @Imie='Anna', @Nazwisko='Nowak'
```

Innym sposobem przypisania wartości do zmiennej jest użycie instrukcji SET. Instrukcją SET można przypisać tylko jedną wartość.

```
SET @Imie=' Anna ';
```

```
SET @Nazwisko=' Nowak ';
```

Zmiennym można przypisywać wartości zwrócone przez zapytanie. Zapytanie takie musi jednak zwracać dokładnie jeden wiersz.

```
SELECT @Imie=imie, @Nazwisko=nazwisko
```

```
FROM Klient
```

```
WHERE id_klienta=5;
```

Instrukcja warunkowa

Instrukcja warunkowa używana jest do określania sposobu wykonania kodu. Ogólna postać instrukcji wygląda następująco:

```
IF warunek_logiczny polecenie  
ELSE  
Polecenie
```

Przykład 1 instrukcji warunkowej:

```
DECLARE @Imie varchar (20), @Nazwisko varchar (20);
```

```
SELECT @Imie='Kamil', @Nazwisko='Stoch';
```

```
IF NOT EXISTS (SELECT TOP 1 * FROM Klient WHERE imie=@Imie AND nazwisko=  
@Nazwisko)
```

```
INSERT INTO Klient (imie, nazwisko)
```

```
VALUES ('Kamil', 'Stoch');
```

Wynikiem będzie sprawdzenie, czy klient Kamil Stoch został dodany w bazie, a jeżeli nie został dodany - dodanie go do tabeli Klient.

Przykład 2 instrukcji warunkowej:

```
IF object_id ('v_klienci_bezfax') IS NOT NULL
```

```
DROP VIEW v_klienci_bezfax1;
```

```
GO
```

```
CREATE VIEW v_klienci_bezfax AS SELECT CompanyName, City, Country FROM  
CUSTOMERS WHERE Fax IS NULL;
```

Funkcja systemowa `object_Id` (nazwa_obiektu) zwraca numer identyfikacyjny obiektu. Została tutaj użyta do sprawdzenia, czy podany obiekt istnieje. Jeżeli tak, obiekt ten zostanie usunięty a następnie utworzony.

Wyrażenie CASE

Wyrażenie CASE może być użyte wewnątrz niektórych poleceń języka SQL. Należą do nich polecenia: SELECT, UPDATE, DELETE, SET oraz klauzule: IN, WHERE, ORDER BY, HAVING. Wyrażenie CASE sprawdza dla każdego wiersza wartość otrzymaną w zapytaniu i porównuje ją z wynikiem wyrażenia. W zależności od wyniku zwraca wartość zapisaną po słowie kluczowym THEN.

CASE

```
{WHEN wyrażenie_logiczne THEN wyrażenie_wynikowe}
[ELSE wyrażenie_wynikowe]
END
```

Przykład na CASE

```
USE Northwind
SELECT [UnitPrice],
CASE
WHEN [UnitPrice] < 30 THEN 'Tani produkt'
WHEN [UnitPrice] < 100 THEN 'W średniej cenie'
ELSE 'Produkt drogi'
END
FROM Products;
```

Wyrażenie CASE sprawdza dla każdego wiersza tabeli Products wartość pola [UnitPrice] i porównuje ją z wartościami 30 i 100. W zależności od wyniku porównania zwraca wartość zapisaną po słowie kluczowym THEN.

Wyrażenia tablicowe (CTE)

Wyrażenia tablicowe umożliwiają definiowanie wirtualnych tabel, do których można się odwoływać wielokrotnie za pomocą ich nazwy. W celu utworzenia wyrażenia tablicowego należy użyć polecenia **WITH** w postaci:

```
WITH nazwa_wirtualnej_tabeli AS (zapytanie)
```

Po wykonaniu zapytania zostanie utworzona wirtualna tabela, z którą można pracować jak ze zwykłą tabelą.

Wyrażenia tablicowe mogą być stosowane:

- do tworzenia zapytań cyklicznych,
- jako odpowiedniki widoków, jeżeli nie ma potrzeby przechowywania definicji metadanych,
- przy wielokrotnym odwoływaniu się do tabeli wynikowej w tym samym zapytaniu.

Zaletą stosowania wyrażeń tablicowych jest większa czytelność złożonych zapytań. Mogą one służyć do budowania prostych bloków logicznych, z których tworzone są bardziej złożone zapytania. Mogą być definiowane i używane w funkcjach i procedurach składowanych oraz w wyzwalaczach i widokach.

Wykonanie wyrażenia tablicowego następuje po podaniu polecenia SELECT:

```
SELECT lista_kolumn
FROM nazwa_wirtualnej_tabeli
```

Przykład CTE

```
USE Northwind
WITH CTE_Customer AS
(SELECT CustomerID, CompanyName, Country, (Address+ ' '+City) AS dane_adresowe
FROM Customers)
SELECT CustomerID, CompanyName, dane_adresowe
FROM CTE_Customer
WHERE Country <> 'Germany';
```

Najpierw została utworzona wirtualna tabela *CTE_Customer*, a następnie na podstawie tej tabeli w zapytaniu zostały wybrane dane do pokazania. Wyrażenia tablicowe, wykonywane są w pamięci, nie są nigdzie zapisywane. Po wykonaniu są zwalniane z pamięci.

Procedury składowane

Procedury składowane to polecenie lub kilka poleceń, które są wykonywane jako całość w jednym bloku. Mogą mieć zadeklarowane zarówno parametry wejściowe, jak i wyjściowe. Mogą również zawierać polecenia kontrolowania kodu, takie jak IF lub WHILE. Zaleca się, aby wszystkie powtarzalne czynności w bazie danych były definiowane przy użyciu procedur składowanych.

Zdefiniowana przez użytkownika procedura jest tworzona w bieżącej bazie danych. Wyjątkiem są procedury tymczasowe — one zawsze są tworzone w folderze *tempdb*.

Definiowanie procedury składowanej wygląda:

```
CREATE PROCEDURE Nazwa_procedury
(
@nazwa_parametru_1 typ_parametru, ..., @nazwa_parametru_n typ_parametru
)
AS
Treść_procedury
```

Przykład procedury:

Mamy utworzyć procedurę, która po podaniu CustomerID klienta będzie zwracała jego dane.

```
CREATE PROCEDURE Dane_klientow
(@CustomerID nchar(5) )
AS
SELECT CustomerID, CompanyName, ContactName, Address, City, Country, Phone FROM
Customers
WHERE CustomerID=@CustomerID;
GO
--wywołanie procedury
EXEC Dane_klientow 'ANATR';
--lub 2 sposób wywołania
EXECUTE Dane_klientow @CustomerID='ANATR';
```

Results

Messages

	CustomerID	CompanyName	ContactName	Address	City	Country	Phone
1	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	Mexico	(5) 555-4729

Funkcje składowane

Funkcje składowane działają podobnie jak procedury składowane, ale sposób ich wykorzystania jest inny. Jedną z cech takiej funkcji jest to, że należy zadeklarować typ zwracanego wyniku. Instrukcja SELECT wykorzystywana w procedurze może zwracać pewną wartość, ale nie musi, natomiast w funkcji jest to wymagane. Dlatego zalecane jest, aby **procedury** były wykorzystywane do zapisywania, modyfikowania i usuwania rekordów, czyli działań, które nie zwracają wyniku, **funkcje** zaś do obliczeń typu: liczenie średniej, przeliczanie wartości itp.

Podstawowa postać funkcji to:

```
CREATE FUNCTION nazwa_funkcji
(
    @nazwa_parametru_1 typ_parametru,      @nazwa_parametru_n typ parametru
)
RETURNS typ_zwracany AS
BEGIN
    Treść_funkcji
RETURN wartość_zwracana
END;
```

Podczas definiowania funkcji trzeba zdefiniować nagłówek (nazwa, lista parametrów, typ zwracanej wartości) i treść funkcji. Ostatnim poleceniem treści powinna być instrukcja RETURN, która określa wartości zwracane przez funkcję.

Przykład funkcjiⁱⁱ

```
CREATE FUNCTION dbo.LiczbaDniRoboczych
(
    -- Funkcja nie uwzględnia świąt, innych dni wolnych poza sobotą i
niedzielą
    -- określenie parametrów wejściowych – są opcjonalne
    -- u nas konieczne są dwa, określające zakres dat
    @StartDate datetime,
    @EndDate datetime
)
-- określenie typu zwracanej wartości (to obowiązkowo)
RETURNS int
AS
-- ciało funkcji
BEGIN

    RETURN  (DATEDIFF(dd, @StartDate, @EndDate) + 1)
            -(DATEDIFF(wk, @StartDate, @EndDate) * 2)
            -(CASE WHEN DATENAME(dw, @StartDate) = DATENAME(dw,6) THEN 1
ELSE 0 END)
            -(CASE WHEN DATENAME(dw, @EndDate) = DATENAME(dw,5) THEN 1 ELSE
0 END)

END;
```

Nasza funkcja zawiera dwa parametry – początek i koniec zakresu dat. W wyniku jej działania, zwracana jest wartość liczbową całkowitą – typ integer.

Wywoływanie funkcji skalarnych

Korzystanie ze skalarnych funkcji użytkownika jest bardzo proste, ale trzeba pamiętać o kilku zasadach. Przede wszystkim **wywołujemy je zawsze za pomocą nazwy przynajmniej dwuczłonowej**, zawierającej schemat (u nas jest to dbo) w którym została utworzona.

```
SELECT OrderId, OrderDate, ShippedDate,  
       dbo.LiczbaDniRoboczych(OrderDate, ShippedDate) as WorkingDays FROM  
dbo.Orders  
WHERE dbo.LiczbaDniRoboczych(OrderDate, ShippedDate) > 20 ORDER BY WorkingDays  
DESC;
```

Przykładowe funkcje wbudowane

AVG – obliczanie wartości średniej,
SUM – sumowanie wartości,
COUNT – zliczanie liczby wystąpień,
GETDATE – zwracanie wartości bieżącej daty i czasu,
DAY – zwracanie dnia miesiąca jako liczby całkowitej,
YEAR – zwracanie roku jako liczby całkowitej,
MONTH – zwracanie miesiąca jako liczby całkowitej.

Wyzwalacze - Triggers

Wyzwalacze (ang. Triggers) bazy danych są to procedury wywoływane przez system przy zajściu odpowiedniego zdarzenia dotyczącego tabel w bazie danych. Będą więc miały budowę podobną do procedur składowanych, lecz nie będziemy mogli ich uruchomić samodzielnie, poleceniem `EXECUTE`, lecz będą uruchamiane automatycznie

Wyzwalacze (triggers) dzielą się na kilka grup:

- **Wyzwalacze DML** (Data Manipulation Language – język manipulowania danymi) są wykonywane w momencie, gdy użytkownik próbuje wykonać operacje na danych dotyczących tabeli lub widoku za pośrednictwem instrukcji należących do tej grupy używalności języka SQL, czyli:
 - INSERT
 - UPDATE
 - DELETE
- **Wyzwalacze DDL** (Data Definition Language – język definiowania danych) są wykonywane w momencie, gdy użytkownik próbuje wykonać operacje bezpośrednio na obiektach bazodanowych za pośrednictwem instrukcji należących do tej grupy używalności języka SQL, czyli:
 - CREATE
 - ALTER
 - DROP
- **Wyzwalacze logowania** są wykonywane w momencie, gdy użytkownik próbuje się zalogować podczas nawiązywania sesji za pośrednictwem instrukcji logowania, czyli:
 - LOGON

Tworzenie wyzwalaczy

Ogólna konstrukcja tworzenia wyzwalaczy wygląda w taki sposób:

```
CREATE OR REPLACE TRIGGER nazwa_triggera
BEFORE/AFTER
DELETE/UPDATE/INSERT/INSERT OR UPDATE OR DELETE
ON nazwa_tabeli
instrukcje które mają zostać wykonane.
```

Przykład wyzwalacza:

```
CREATE TRIGGER pierwszy_wyzwalacz
ON
    dbo.Customers
AFTER INSERT, UPDATE, DELETE
AS
print 'Brawo Twój pierwszy Wyzwalacz zadziałał';
```

```

2 CREATE TRIGGER pierwszy_wyzwalacz
3 ON
4     dbo.Customers
5 AFTER INSERT, UPDATE, DELETE
6 AS
7     print 'Brawo Twój pierwszy Wyzwalacz zadziałał';
8
9     select * from Customers;
10
11 update Customers SET Country='WIELKA BRYTANIA' WHERE Country='UK';

```

Messages

Brawo Twój pierwszy Wyzwalacz zadziałał

(7 rows affected)

Completion time: 2021-01-08T22:53:23.6102349+01:00

```

13 USE Northwind;
14 select * from Customers;

```

161 %

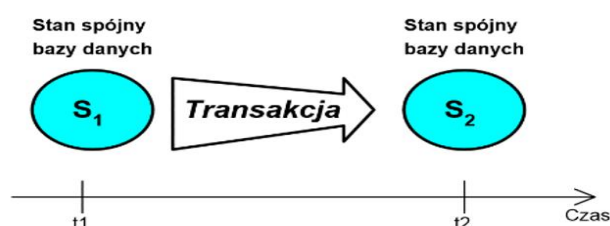
	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	NULL	12209	Germany	030-0074321	030-0076545
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	NULL	05021	Mexico	(5) 555-4729	(5) 555-3745
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	NULL	05023	Mexico	(5) 555-3932	NULL
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	NULL	WA1 1DP	WIELKA BRYTANIA	(171) 555-7788	(171) 555-6750
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	NULL	S-958 22	Sweden	0921-12 34 65	0921-12 34 67
6	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	NULL	68306	Germany	0621-08460	0621-08924
7	BLONP	Blondesdssl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	NULL	67000	France	88.60.15.31	88.60.15.32
8	BOLID	Bóldo Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	NULL	28023	Spain	(91) 555 22 82	(91) 555 91 99
9	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	NULL	13008	France	91.24.45.40	91.24.45.41
10	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745
11	BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	NULL	EC2 5NT	WIELKA BRYTANIA	(171) 555-1212	NULL
12	CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires	NULL	1010	Argentina	(1) 135-5555	(1) 135-4892
13	CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993	México D.F.	NULL	05022	Mexico	(5) 555-3392	(5) 555-7293

38. Definicja transakcji i jej właściwości.

39. Poziomy izolowania transakcji. Współbieżność.

Transakcja – zbiór operacji na bazie danych, które stanowią w istocie pewną całość i jako takie powinny być wykonane wszystkie lub żadna z nich. Warunki jakie powinny spełniać transakcje bardziej szczegółowo opisują zasady **ACID** (*atomicity, consistency, isolation, durability* – niepodzielność, spójność, izolacja, trwałość).

- **Transakcja** – jest sekwencją logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny. Transakcja jest atomową jednostką pracy systemu, która może być wykonana w całości, albo nie wykonana w ogóle.



- **Stan spójny bazy danych** – to stan, w którym wszystkie ograniczenia integralnościowe wynikające ze schematu bazy danych oraz inne ograniczenia są spełnione.

Przykładem transakcji może być transakcja bankowa jaką jest przelew. Muszą tu zostać dokonane 2 operacje – zabranie pieniędzy z jednego konta oraz dopisanie ich do drugiego. W przypadku niepowodzenia żadna z tych operacji nie powinna być zatwierdzona, gdyż zajęcie tylko jednej powodowałoby nieprawidłowości w bazie danych (pojawienie się lub zniknięcie pieniędzy).

Przykład 3 transakcji- transakcja przelewu 100 zł z konta 1 na konto 2,
Zamówienie towaru i modyfikacja stanu ilości produktu,
Rezerwacja lotu.

<pre> BEGIN UPDATE konta SET stan=stan-100 WHERE id_konta=1; UPDATE konta SET stan=stan+100 WHERE id_konta=2; COMMIT; </pre>	<pre> BEGIN INSERT INTO zamowienia(nr_zamowienia,ilo sc,nr_produktu) VALUES (5,15,2); UPDATE produkt SET ilosc=ilosc-15 /*usunięcie ze stanu produktu 15 sztuk*/ WHERE nr_produktu=2; COMMIT; </pre>	<pre> -- przykład transakcji rezerwacji lotu BEGIN UPDATE Loty SET wolneMiejsca=wolneMiejsca-1 WHERE idlotu =X; UPDATE Konta SET saldo = saldo - Z WHERE klient=Y; UPDATE Konta SET saldo = saldo +Z WHERE klient=A; INSERT INTO Pasażerowie(lot,klient) VALUES (X,Y); COMMIT; // zatwierdzenie transakcji </pre>
---	--	--

Transakcja składa się zawsze z 3 etapów:

- rozpoczęcia
- wykonania
- zamknięcia

W systemach baz danych istotne jest, aby transakcja trwała jak najkrócej, ponieważ równolegle może być dokonywanych wiele transakcji i część operacji musi zostać wykonana w pewnej kolejności. Każdy etap transakcji jest logowany, dzięki czemu w razie awarii systemu (dzięki zawartości logów) można odtworzyć stan bazy danych sprzed transakcji, która nie została zamknięta.

Część systemów baz danych umożliwia używanie **punktów pośrednich** (ang. *save point*), są to zapamiętane w systemie etapy transakcji, do których w razie wystąpienia błędu można się wycofać, bez konieczności anulowania wszystkich wykonanych działań.

Właściwości transakcji -ACID

Transakcje opisuje zasada **ACID** - atomowość (**A**tomicity), spójność (**C**onsistency), izolacja (**I**solation) i trwałość (**D**urability):

- **Atomowość** - transakcja może być albo wykonana w całości albo w całości niewykonana.
- **Spójność** - transakcja nie zmienia spójności (integralności) bazy danych, czyli wykonanie transakcji nie doprowadzi do utraty spójności danych. Jeżeli baza danych była spójna przed wykonaniem transakcji, to jest spójna również po jej zakończeniu.
- **Izolacja** - transakcja dzieje się niezależnie od innych wykonywanych operacji, w tym od innych transakcji.
- **Trwałość** - w przypadku awarii systemu baz danych, np. w wyniku odcięcia elektryczności, transakcja będzie albo wykonana w całości albo wcale nie wykonana.

W systemach baz danych realizujących standard SQL następujące polecenia dotyczą transakcji:

- `BEGIN` lub `BEGIN WORK` – rozpoczęcie transakcji;
- `COMMIT` – zatwierdzenie zmian wykonanych w obrębie transakcji;
- `ROLLBACK` – odrzucenie zmian wykonanych w obrębie transakcji;
- `SAVEPOINT nazwa/` lub `SAVE TRANSACTION` – zdefiniowanie punktu pośredniego o określonej nazwie;
- `RELEASE SAVEPOINT nazwa` – skasowanie punktu pośredniego (nie wpływa w żaden sposób na stan transakcji);
- `ROLLBACK TO SAVEPOINT nazwa` – wycofanie transakcji do stanu zapamiętanego w podanym punkcie pośrednim.

W SQL Server transakcja może być wykonywana na trzy sposoby:

- *Explicit* — jawnie. Rozpoczęcie transakcji jest realizowane za pomocą polecenia **BEGIN TRANSACTION**.
- *Autocommit* — automatycznie. Operacje wykonywane na serwerze są standardowo traktowane jako transakcje, w związku z czym nie ma potrzeby ich rozpoczynania poleceniem **BEGIN TRANSACTION**. Po poprawnym wykonaniu każde z poleceń jest automatycznie zatwierdzane.
- *Implicit* — niejawnie. Transakcje są wywoływane przez programy użytkowe działające na bazie danych.

Punkty przywracania

W większości serwerów bazodanowych można wycofać nie tylko całą transakcję, ale także jej część. W tym celu należy utworzyć punkty przywracania za pomocą instrukcji `SAVE TRANSACTION`.

Przykład - Punkty przywracania

```
BEGIN TRANSACTION;
INSERT INTO pracownik (nazwisko, imie, miasto)
VALUES ('Nowak', 'Stanisław', 'Kraków');
SAVE TRANSACTION t1;
INSERT INTO pracownik (nazwisko, imie, PESEL)
VALUES ('Kowalska', 'Anna', 'Poznań');
ROLLBACK TRANSACTION t1;
```

W podanym przykładzie instrukcja **BEGIN TRANSACTION** uruchomi transakcję. Po dodaniu do tabeli *pracownik* danych jednego pracownika instrukcja **SAVE TRANSACTION t1** utworzy punkt przywracania. Po dodaniu do tabeli *pracownik* danych kolejnego pracownika transakcja zostanie odwołana (instrukcja **ROLLBACK**), ale nie zostaną odrzucone wszystkie zmiany dokonane podczas trwania transakcji, tylko zmiany wprowadzone po zdefiniowanym punkcie przywracania. W rezultacie, mimo że transakcja została odwołana, w tabeli *pracownik* zostaną zapisane dane pracownika o nazwisku *Nowak*, natomiast dane pracownika o nazwisku *Kowalska* zostaną anulowane.

Współbieżność transakcji

Współbieżność transakcji występuje w sytuacji kiedy kilka transakcji wykonuje operacje na tych samych danych.

Podczas wykonywania współbieżnych transakcji mogą wystąpić niepożądane anomalie. System Zarządzania Baz Danych powinien dostarczać mechanizmów, które zapobiegają wystąpieniu niepożądanych anomalii jednocześnie nie ograniczając wydajności bazy danych.

Anomalie współbieżnego dostępu

Podczas wykonywania równoległe transakcji na tych samych danych, w zależności od poziomu izolacji transakcji mogą występować następujące anomalie:

- brudny odczyt,
- utracona modyfikacja,
- niepowtarzalny odczyt,
- fantomy.

Brudny odczyt

Brudny odczyt (ang. Dirty read / Uncommitted dependency) to sytuacja, w której transakcja odczytuje dane zmienione przez inną transakcję, która później zostaje wycofana, a więc odczytane dane przez pierwszą transakcję są nieprawdziwe.

Utracona modyfikacja

Utracona modyfikacja (ang. Lost update) to sytuacja, w której dwie transakcje równoległe próbują zmienić te same dane i zmiany wprowadzane przez jedną transakcję mogą być nadpisane przez drugą transakcję.

Niepowtarzalny odczyt

Niepowtarzalny odczyt (ang. Non-repeatable read / Inconsistent Analysis) powstaje w sytuacji gdy jedna z transakcji wykonuje kilkakrotnie te same zapytania na danych, które w międzyczasie są zmieniane przez inną transakcję, co powoduje, że to samo zapytanie nie generuje tych samych rezultatów.

Fantomy

Fantomy (ang. Phantoms) to sytuacja podobna do niepowtarzalnego odczytu. Jedna transakcja wykonuje kilkakrotnie to samo zapytanie ale otrzymuje różne zbiory wynikowe, ponieważ inna transakcja w tym samym czasie usuwa lub dodaje nowe dane, które spełniają warunki zapytania. To powoduje, że za każdym razem wynikiem zapytania może być inna liczba wierszy.

Blokada –przydzielenie zasobu do zadania

Blokowanie danych stosujemy w celu zagwarantowania integralności i spójności danych w trakcie realizowania transakcji.

Rodzaje blokad:

- **Wyłączne X** (ang. *eXclusive*) są zakładane na modyfikowanych obiektach i domyślnie utrzymywane do zakończenia całej transakcji. Użytkownicy modyfikujący dane blokują innych użytkowników.
- **Współdzielone S** (ang. *Shared*) są zakładane domyślnie na odczytywanych obiektach tylko na czas wykonania zapytania. Jeżeli dane zostały zablokowane w trybie S, to możliwe jest założenie na nie blokady S przez inne procesy.

Zakleszczenie (ang. Deadlock)- stan w którym dwie lub więcej transakcji znajduje się w stanie wzajemnego oczekiwania na zwolnienie blokady w celu wykonania dalszych operacji.

Przykład: Rozpocznij nową transakcję. Zwiększ płacę podstawową Kowalskiemu o 1000 złotych. Utwórz punkt bezpieczeństwa P1. Daj pracownikowi Zygmunt 500 zł płacy dodatkowej. Utwórz punkt bezpieczeństwa P2. Usuń pracownika Czerwonka. Wycofaj transakcję do punktu P1 i zobacz zawartość tabeli pracownicy. Wycofaj transakcję do punktu P2. Wycofaj całą transakcję.

```
BEGIN TRANSACTION;
UPDATE pracownicy SET placa_pod=placa_pod+1000 WHERE nazwisko='Kowalski';
SAVE TRANSACTION P1;
UPDATE pracownicy SET placa_dod=500 WHERE nazwisko='Zygmunt';
SAVE TRANSACTION P2;
DELETE pracownicy WHERE nazwisko='Czerwonka';
ROLLBACK TRANSACTION P1;
SELECT * FROM Pracownicy;
ROLLBACK TRANSACTION P2;
ROLLBACK;
```

W celu przetestowania tej transakcji załóż tabelę z potrzebnymi kolumnami i wpisz odpowiednie rekordy, a następnie wykonaj transakcję.

ⁱ <https://tomaszkenig.pl/kurs-sql-server/wprowadzenie-do-transact-sql/>

ⁱⁱ <https://www.sqlpedia.pl/tag/funkcje-skalarne/>