SQL: UPDATE, DELETE, DDL
Transakcje

Robert A. Kłopotek

r.klopotek@uksw.edu.pl

Wydział Matematyczno-Przyrodniczy. Szkoła Nauk Ścisłych, UKSW

Instrukcja UPDATE

- Umożliwia modyfikację istniejących rekordów w tabeli
- Można aktualizować jeden lub wiele rekordów (a nawet wszystkie!)
- Składnia:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Instrukcja UPDATE - przykłady

- ```
  UPDATE customers
  SET city='Warszawa'
  WHERE country='Poland'
  ```
- ```
  UPDATE customers SET …
  WHERE customer_id IN
  (
      SELECT customer_id FROM orders
      WHERE …
  )
  ```
- ```
  UPDATE customers
  SET city='Warszawa' ;
  ```
  ```

Instrukcja DELETE

- Umożliwia usuwanie istniejących rekordów
- Można usunąć jeden lub wiele rekordów (a nawet wszystkie!)
- Składnia:

```
DELETE FROM table_name
WHERE condition;
```

Instrukcja DELETE - przykłady

- ```
  DELETE FROM customers
  WHERE city='Warszawa'
  ```
- ```
  DELETE FROM customers c
  WHERE NOT EXIST
  (
      SELECT * FROM orders o
      WHERE o.customer_id = c.customer_id
  )
  ```
- ```
  DELETE FROM customers
  ```
- ```
  DELETE * FROM customers
  ```

SQL - DLL

- DDL – data description language
- Bardzo ważna część języka SQL jest poświęcona zarządzaniem strukturami danych i obiektami, takimi jak:
  - Tabelami
  - Ograniczeniami
  - Indeksami
  - Bazami danych oraz tabelami
  - Widokami
  - Wyzwalaczami (triggerami)
  - Transakcji
  - Procedurami, funkcjami
  - …

CREATE TABLE, CREATE DATABASE

- Składnia:

```
CREATE TABLE TableName (ColumnName
ColumnType [Constraint] [,…])

CREATE DATABASE databasename
```

- Przykład:

```
CREATE TABLE osoby2
(
id_osoby int not null AUTO_INCREMENT,
primary key (id_osoby),
imie varchar(40) not null,
nazwisko varchar(60) not null,
data_ur datetime null,
wiek int
);
```

ALTER TABLE

- Instrukcja pozwala na modyfikację struktury istniejącej tabeli
- Składnia:

```
ALTER TABLE TableName [ADD|ALTER]
COLUMN ColumnName ColumnType
```

- Przykład:

```
ALTER TABLE persons ADD COLUMN
birthdate datetime
```

Instrukcja DROP

- Pozwala na usunięcie różnych rodzajów obiektów z bazy danych
- Składnia:
  - `DROP TABLE | INDEX | PROCEDURE |… ObjectName`
  - `ALTER TABLE TableName DROP COLUMN ColumnName`
- Przykłady:
  - `DROP TABLE persons`
  - `ALTER TABLE persons DROP COLUMN birthdate`

Ograniczenia

- Ograniczenia mogą być definiowane dla kolumn kiedy tabela jest tworzone przy pomocy CREATE TABLE lub podczas modyfikacji już istniejącej ALTER TABLE
- ```
  CREATE TABLE table_name (
      column1 datatype constraint,
      column2 datatype constraint,
      column3 datatype constraint,
      ....
  )
  ```
  ```

Wybrane ograniczenia

- NOT NULL – wymusza, że kolumna nie może przyjmować wartości NULL
- UNIQUE - wymusza, że kolumna ma unikalne wartości
- PRIMARY KEY – kombinacja NOT NULL i UNIQUE. Unikalnie identyfikuje rekordy w tabeli (powinno być użyte podczas łączenia tabel)
- FOREIGN KEY – unikalnie identyfikuje rekord w innej tabeli (powinno być użyte podczas łączenia tabel)
- CHECK - wymusza, że wszystkie wartości w kolumnie spełniają pewien warunek
- DEFAULT – ustawia wartość domyślną, jeśli nie podamy jej podczas wstawiania
- INDEX – utworzenie indeksu na kolumnie umożliwia szybsze wyszukiwanie i łączenie tabel (wstawianie i usuwanie jest zwykle wolniejsze)

UNIQUE – przykłady MySQL

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

```
ALTER TABLE Persons
DROP INDEX UC_Person;
```

CHECK, DEFAULT – przykłady MySQL

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND
City='Sandnes')
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

AUTO INCREMENT – przykłady MySQL

```
CREATE TABLE Persons (
    ID int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

```
ALTER TABLE Persons AUTO_INCREMENT=100;
```

AUTO INCREMENT – przykład MS SQL

```
CREATE TABLE Persons (
    ID int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

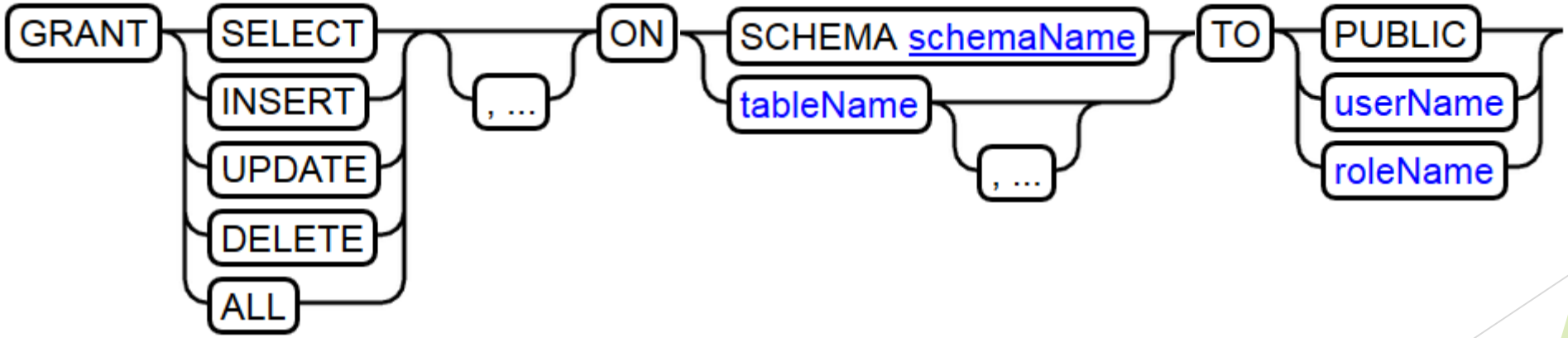
```
IDENTITY(start at value, increment by)
```

CREATE, ALTER, DROP USER - MySQL

- ```
  CREATE USER [IF NOT EXISTS]
      user [auth_option] [, user [auth_option]] ...
      [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
      [WITH resource_option [resource_option] ...]
      [password_option | lock_option] ...
  ```
- ```
  ALTER USER 'jeffrey'@'localhost'
   IDENTIFIED WITH sha256_password BY 'new_password'
   REQUIRE SSL WITH MAX_CONNECTIONS_PER_HOUR 20
      AND MAX_USER_CONNECTIONS 2
   PASSWORD EXPIRE INTERVAL 180 DAY;
  ```
- ```
  DROP USER [IF EXISTS] user [, user] ...
  ```
  ```

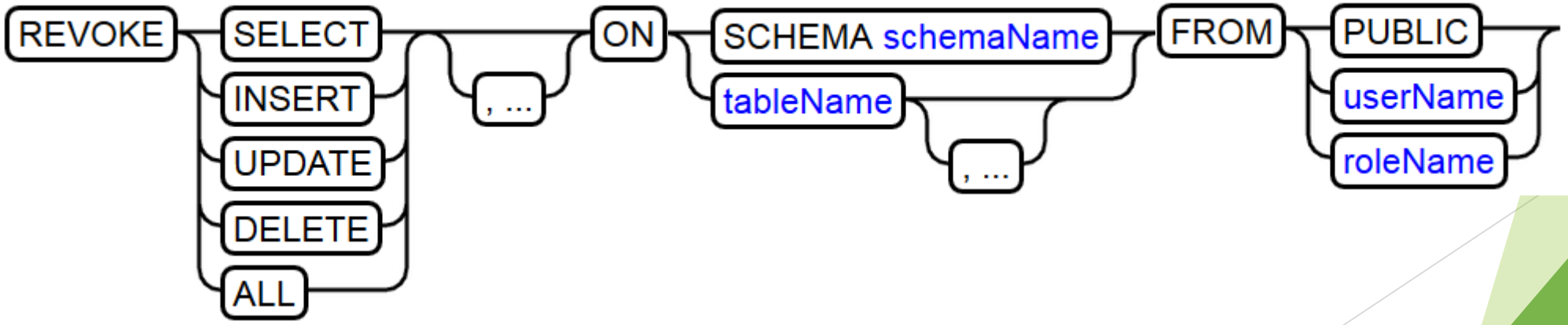
GRANT

- Przyznaje prawa dostępu do tabeli dla użytkownika lub roli, np.:
  `GRANT SELECT ON osoby TO nazwa_uzytkownika`



REVOKE

- Przyznaje prawa dostępu do tabeli dla użytkownika lub roli, np.:
  `REVOKE SELECT ON osoby FROM nazwa_uzytkownika`



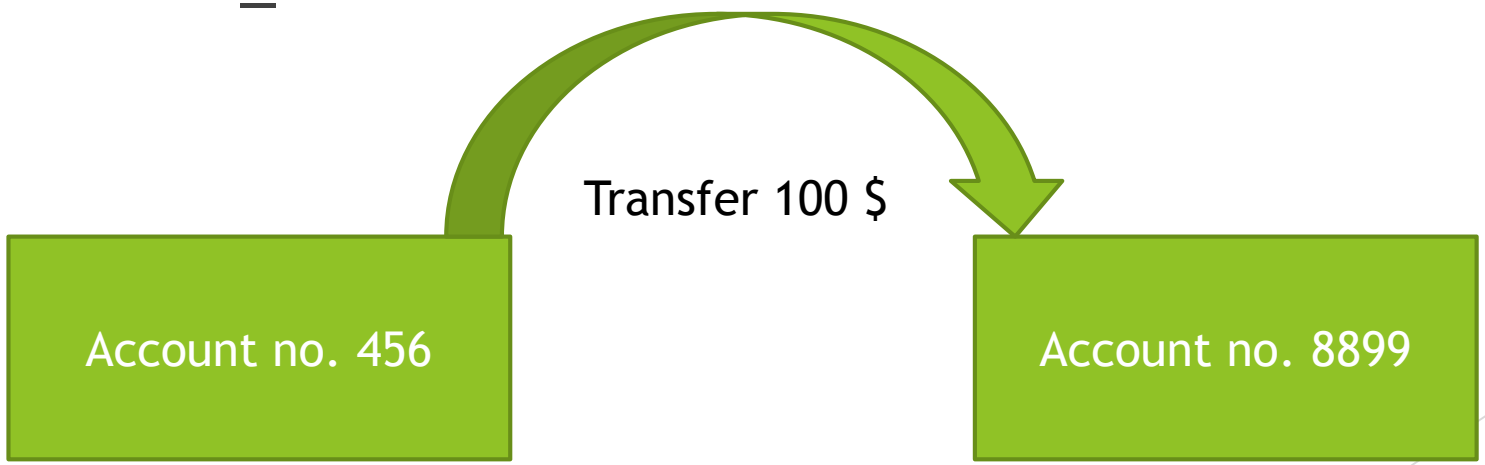
DDL - komentarz

- Instrukcje DDL bardzo zależą od konkretnej implementacji języka SQL w danym SZBD (DBMS)
- Różne opcje ją dostępne przez różnych dostawców
- Zawsze należy sprawdzić w dokumentacji jakie opcje są dostępne w konkretnym DBMS (zwłaszcza podczas projektowania systemu bazodanowego)
- Wszystkie operacje DDL są zależnie od ograniczeń, np. nie możemy usunąć kolumny, która tworzy relację bez usunięcia najpierw tej relacji

Przetwarzanie transakcyjne

- Przykład - mamy serię zapytań:

```
UPDATE Accounts set balance=balance-100 where
account_id=456
UPDATE Accounts set balance=balance+100 where
account_id=8899
```

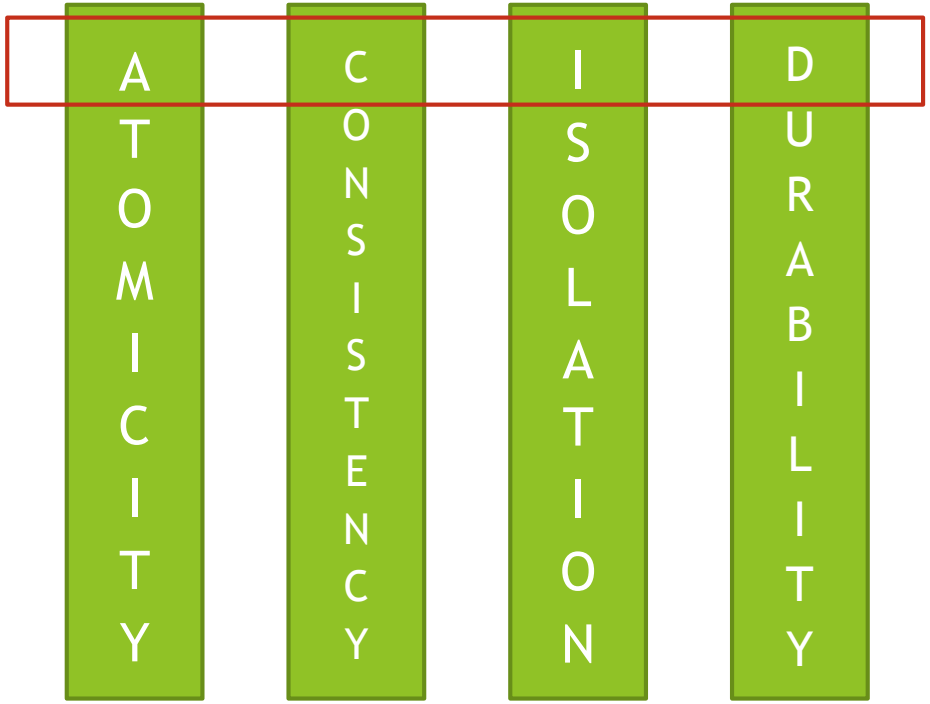


Możliwe problemy

- Obie instrukcje UPDATE powinny zakończyć się powodzenie albo żadna
- Wynik tego zapytania powinien być stały. Co się stanie jeśli podczas wykonywania będziemy mieli awarię prądu?
- Jedna seria zapytań nie powinna wpływać na drugą serię

Przetwarzanie transakcyjne

- Przetwarzanie danych może być uznaje jako transakcyjne, jeśli spełnione są warunki ACID



Transakcja oraz cechy ACID

CECHA	OPIS
Atomicity (**niepodzielność)**	Transakcja rozumiana jako sekwencja instrukcji SQL musi być traktowana jako jedna jednostka – albo wszystko zostało wykonane albo wszystko jest anulowane.
Consistency (**spójność**)	Niezależnie jaki jest wynik transakcji baza danych musi zachować spójność. W szczególności ograniczenia muszą zostać nienaruszone.
Isolation (**izolacja**)	Niedokończone transakcje nie mogą mieć wpływu na transakcje, które są w takcie, np. dóki transakcja przelewu nie jest zatwierdzona nikt nie może tych pieniędzy wypłacić.
Durability (trwałość)	Wynik transakcji, które zostały zakończone, musi być trwały, w szczególności zmiany na danych nie mogą być skasowane lub utracone

Transakcja - przykład

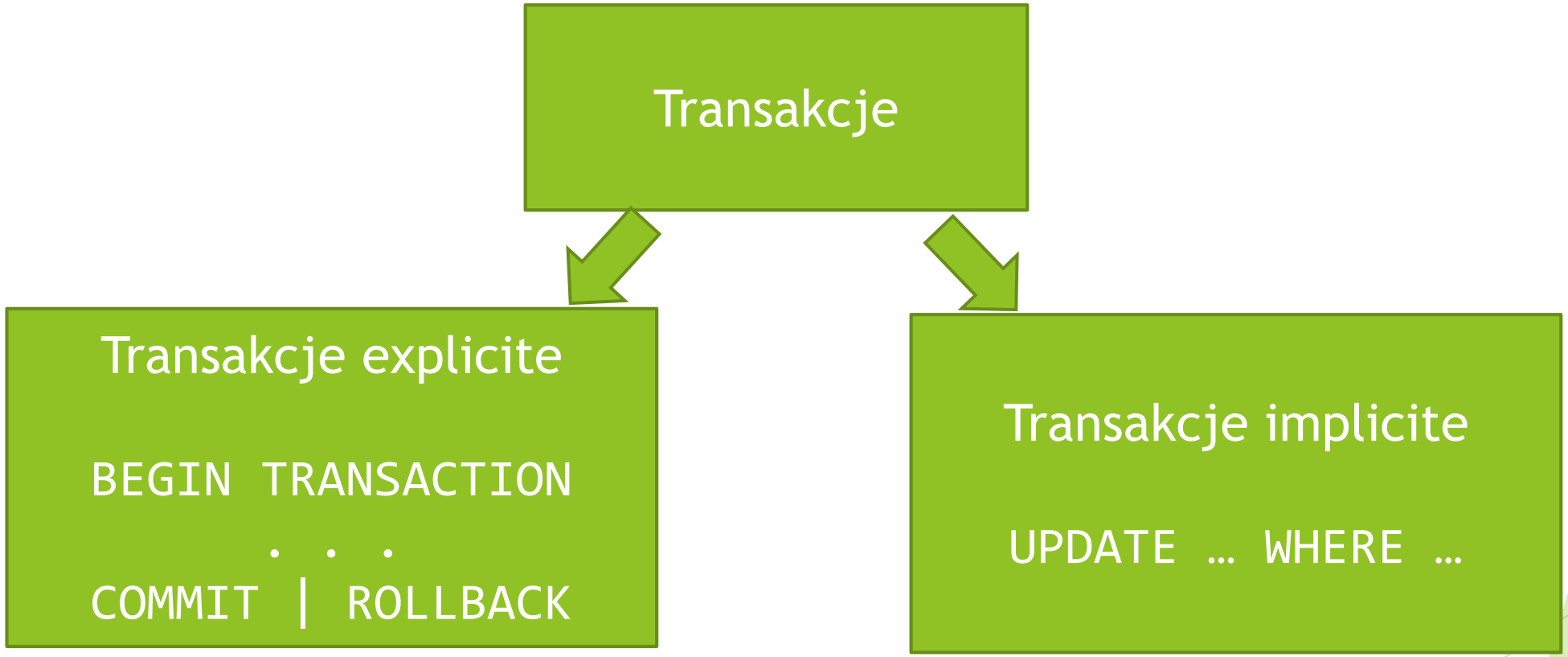
- **BEGIN TRANSACTION**

  UPDATE Accounts set balance=balance-100 where account_id=456
  UPDATE Accounts set balance=balance+100 where account_id=8899

  COMMIT

Poprzez BEGIN TRANSACTION oraz COMMIT oznaczamy zakres transakcji, czyli sekwencja zapytań SQL powinna być przetwarzana zgodnie z ACID

Podział transakcji



Transakcje implicite to są pojedyncze zapytania SQL, które mają zagwarantowane to, że się wykonają jako transakcja, czyli wszystkie wiersze, które spełniają warunek WHERE, będą zmodyfikowane przez UPDATE albo żaden.

Instrukcje transakcji

- BEGIN TRANSACTION – początek transakcji
- COMMIT – zatwierdzenie wszystkich zmian
- ROLLBACK – cofniecie wszystkich zmian do początku bieżącej transakcji
- Niektóre implementacje SQL pozwalają na:
  - *nested transactions* – transakcji zagnieżdżone
  - *save points* – punkty zapisu, czyli punkty w bardzo długich transakcjach, które będą służyły do cofnięcia zmian do jakiegoś punktu (a nie do początku transakcji)

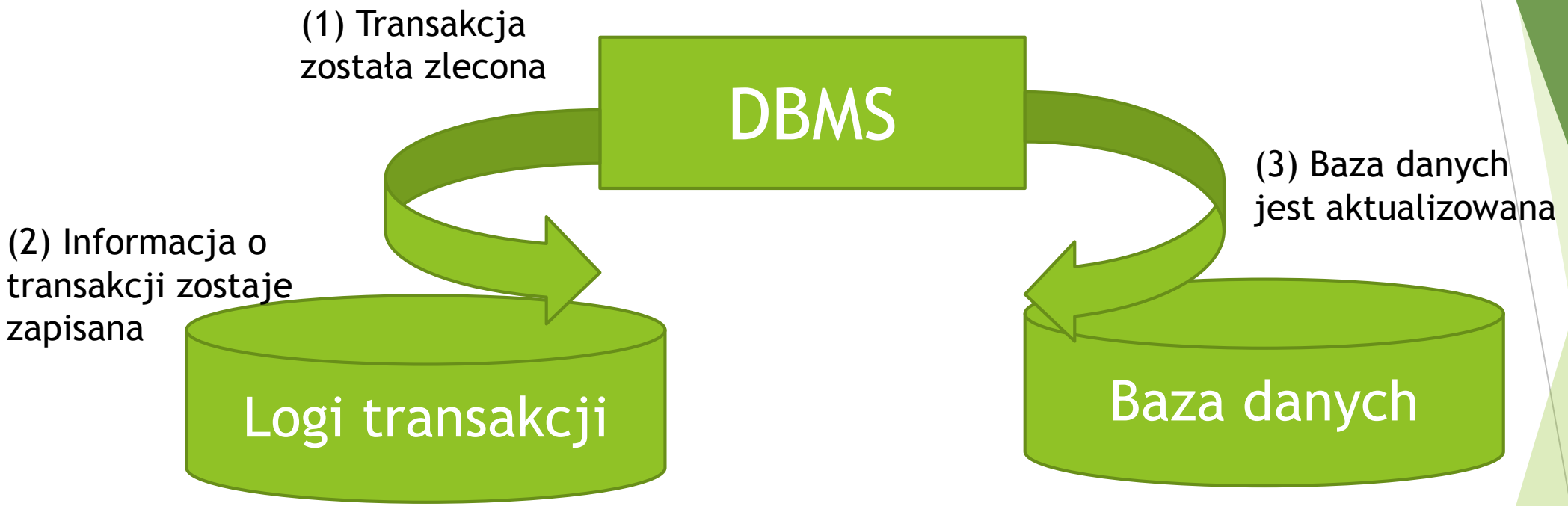
Transakcje - dyskusja

- Przetwarzanie transakcyjne na wpływ na wszystkie instrukcje modyfikacji danych, w szczególności instrukcje CRUD (INSERT, UPDATE and DELETE)
- ROLLBACK może być użyty przez programistę aby anulować transakcję oraz zrezygnować z zatwierdzenia wszystkich zapytań w transakcji
- Instrukcje DDL zwykle NIE SĄ częścią transakcji

Implementacja przetwarzania transakcyjnego w DBMS

- Nie ma prostego rozwiązania – cechy ACID musza być ściśle przestrzegane
- Wyobraźmy sobie nagły brak zasilania:
  - Żaden wynik zatwierdzonej transakcji nie może być utracony (durability)
  - Częściowa aktualizacja rekordów nie jest dozwolona (atomicity)
  - Częściowe wyniki nie mogą być użyte przez inne działające transakcje (isolation)

Transakcje - rozwiązanie



Logi transakcji pomagają zidentyfikować rekordy dotknięte oraz rozwiązywać oczekujące transakcji w przypadku restartu DBMS po nagłej awarii (kiedy nie przeprowadzono poprawnego zamknięcia)

Po nagłej awarii serwer może zawierać niespójne dane, np. tylko część aktualizacji w transakcji została naniesiona. W takim przypadku logi transakcji zostaną użyte do rozwiązania każdej z transakcji przed ponownym startem DBMS

Transakcje - implementacja

- Średnio ponad 99% transakcji zostaje zatwierdzonych
- Stąd z powodów wydajnościowych często aktualizacje są od razu nanoszone. W przypadku anulowania transakcji oryginalna zawartość zostaje przywrócona.
- W środowisku z wieloma użytkownikami, wiele transakcji może się odbywać w tym samym czasie, dlatego może dojść do wielu problemów z izolacja transakcji kosztem przepustowości

Pytania?