

# BD2 – Pytania

TODO 1, 2.; 4.; 5.

<b>1. Zarządzanie transakcjami współbieżnymi</b>	<b>2</b>
1.1. Pojęcie transakcji, postulaty ACID. Poziomy izolacji transakcji i występowanie konfliktów. Przykłady ilustrujące problemy związane z przetwarzaniem transakcji na różnych poziomach izolacji.	2
1.1.1 Transakcja:	2
1.1.2 Właściwości ACID	3
1.1.3 Poziomy izolacji	3
1.1.4 PRZYKŁADY	9
1.2. Blokowanie dwufazowe (B2F) jako metoda zarządzania transakcjami współbieżnymi. Poprawność B2F.	13
1.2.1 Protokół blokowania dwufazowego B2F	13
1.2.2 Realizacja poziomów izolacji przez planistę B2F	15
1.2.3 Poprawność B2F	16
1.2.4 Twierdzenie:	16
1.3. Odtwarzanie baz danych. Algorytm UNDO/REDO. Problemy zatwierdzania i odtwarzanie transakcji rozproszonych.	17
1.3.1 Odtwarzanie baz danych (recovery)	17
1.3.2 Algorytm UNDO/REDO	18
1.4. Zarządzanie transakcjami z wykorzystaniem znaczników czasowych.	19
1.5. Blokowanie optymistyczne (patrz LINQ).	19
<b>2. Indeksowanie i haszowanie w bazach danych</b>	<b>20</b>
2.1. Budowa indeksów wielopoziomowych. Złożoność przeszukiwania i aktualizacji.	20
2.2. Dynamiczne tworzenie indeksu o strukturze B-drzewa przy dołączaniu/usuwaniu elementów. Przykłady.	22
2.3. Indeksy o strukturze B-drzewa. Wpływ indeksów na złożoność operacji wyszukiwania i modyfikowania (dołączanie i usuwanie) danych.	22
2.4. Obliczenia związane z szacowaniem wysokości B-drzewa – i związanej z tym zajętości pamięci – przy zadanych parametrach indeksu.	22
2.5. Algorytmy haszowania danych w bazach danych i ich wpływ na złożoność operacji wyszukiwania i modyfikowania (dołączanie i usuwanie) danych.	22
<b>3. Optymalizacja zapytań</b>	<b>22</b>
3.1. Optymalizacja selekcji. Przykład: przedyskutować strategię wykonania operacji selekcji w tabeli R z warunkiem $(A = a) \text{ AND } (B < b)$ . Jak stosowanie indeksów wpływa na efektywność wykonywania tej operacji.	22
3.2. Optymalizacja złączenia. Przedyskutować złączenie tabel R i S według warunku $R.A=S.B$ , według różnych strategii przy istnieniu i braku indeksów.	23
3.3. Optymalizacja heurystyczna. Podaj drzewo syntaktyczne przed i po zastosowaniu optymalizacji heurystycznej dla wybranego nietrywialnego przykładu.	27
<b>4. Entity Framework i LINQ</b>	<b>30</b>

4.1. Przetwarzanie relacyjnej bazy danych z poziomu języka C# i z wykorzystaniem ADO.NET, Entity Framework i LINQ (metody ORM).	30
4.1.2. Entity Framework:	32
4.1.3. LINQ:	34
4.2. Napisać w LINQ przykładowe zapytania wyrażone w SQL: selekcja, złączenie, grupowanie, stosowanie funkcji agregujących.	35
<b>5. Modele danych NoSQL i bazy danych NoSQL</b>	<b>36</b>
5.1. Model danych JSON i baza Azure DocumentDB	36
5.1.1. Model danych JSON:	36
5.1.2. Azure DocumentDB:	37
5.2. Przykładowe zapytanie do bazy dokumentów JSON	37
5.3. Składnia danych według modelu Key/Value i baza Azure Storage	39
5.4. Sposób reprezentacji danych o podanym schemacie ER w modelu JSON i Key/Value, np. dla Student-Egzamin-Przedmiot.	40
5.5. Dane BigData i metody ich przetwarzania (na przykładzie MS Azure)	40
<b>6. XML, XPath i XQuery</b>	<b>42</b>
6.1. Gramatyka XML i wyrażenia zgodne z tą gramatyką jako dokumenty XML.	42
6.2. Schematy XML: DTD i XML Schema	43
6.2.1. DTD	43
6.2.2. XML Schema	44
6.3. Model DOM – przykład, rodzaje wierzchołków.	45
6.4. Zapytania XPath: składnia, przykłady.	45
6.5. XQuery – ogólny schemat zapytań, przykłady zapytań	48

# 1. Zarządzanie transakcjami współbieżnymi

1.1. Pojęcie transakcji, postulaty ACID. Poziomy izolacji transakcji i występowanie konfliktów. Przykłady ilustrujące problemy związane z przetwarzaniem transakcji na różnych poziomach izolacji.

## 1.1.1 Transakcja:

**Transakcją** (T) nazywamy ciąg następujących operacji na wspólnej bazie danych:

R	czytanie (read)	$r_T[x]$	czytanie danej $x$ przez transakcję T
---	-----------------	----------	---------------------------------------

W	zapis (write)	$w_T[x]$	zapisanie danej $x$ przez transakcję $T$
A	odrzućcie (abort)	$a_T$	odrzućcie transakcji $T$
C	zatwierdzanie (commit)	$C_T$	zatwierdzenie transakcji $T$

gdzie  $x$  - jednostka danych na różnych poziomach granulacji tj.

- dana elementarna    - krotka    - zbiór krotek    - tabela

**Celem systemu zarządzania** transakcjami jest takie sterowanie operacjami w BD, aby były one wykonywane z możliwie wysokim współczynnikiem współbieżności i przeciwdziałanie naruszeniu spójności BD

**Zarządzanie współbieżnym dostępem:**

- Blokowanie 2-fazowe (B2F) - restrykcyjne, silne blokowanie stosowane w systemach z dobrze zdefiniowanymi transakcjami
- Blokowanie optymistyczne - lekkie - stosowane w "zwykłych" programach (bez transakcji)
- Blokowanie oparte na znacznikach czasowych - niektóre transakcyjne systemy rozproszone

### 1.1.2 Właściwości ACID

Transakcje oraz protokoły zarządzania nimi w BD muszą spełniać właściwości ACID:

<b>A</b>	<b>Atomowość (atomicity)</b>	Każda transakcja: <ul style="list-style-type: none"> <li>- jest pojedynczą i niepodzielną jednostką przetwarzania/odtworzenia</li> <li>- jest wykonywana w całości lub jej efekt nie jest widoczny w BD</li> <li>- nie zawiera podtransakcji</li> </ul>
<b>C</b>	<b>Spójność (consistency)</b>	<ul style="list-style-type: none"> <li>- po wykonaniu transakcji BD musi być spójna</li> <li>- jeśli transakcja narusza warunki spójności BD to jest odrzućca</li> </ul>
<b>I</b>	<b>Odizolowanie (isolation)</b>	<ul style="list-style-type: none"> <li>- zmiany wykonywane przez transakcję niezatwierdzoną są ukryte dla innych transakcji</li> <li>- wyjątek: tylko, gdy pozwala na to przyjęty poziom izolacji</li> </ul>
<b>D</b>	<b>Trwałość (duration)</b>	<ul style="list-style-type: none"> <li>- zmiany dokonane przez transakcję zatwierdzoną są trwałe w BD</li> <li>- w przypadku awarii systemu musi istnieć możliwość ich odtworzenia (backup)</li> </ul>

### 1.1.3 Poziomy izolacji

Wyróżniamy 4 poziomy izolacji transakcji:

- najniższy (0)
  - największa współbieżność
  - największe ryzyko - mogą wystąpić anomalie
- najwyższy (3)

- brak wszelkich anomalii
- najbardziej kosztowny (często ponoszenie tych kosztów jest niepotrzebne)

Wybór poziomu izolacji wiąże się z problemami:

- zbyt niski poziom
  - zapewnia zwiększenie współczynnika współbieżności
  - może doprowadzić do niekorzystnych cech związanych z zachowaniem spójności bazy danych
- zbyt wysoki poziom
  - może powodować nieuzasadnione opóźnianie wykonywania transakcji

Poziomy izolacji:

- T - operacje mogą być wykonywane współbieżnie
- N - brak współbieżności, dłuższy czas wykonywania transakcji, większa niezawodność przetwarzania i bezpieczeństwo spójności BD

Związki poziomów izolacji z problemami przetwarzania transakcji

Poziom izolacji	Brak odtwarzalności	Anomalia powtórnego czytania	Fantomy
0: READ UNCOMMITTED	T	T	T
1: READ COMMITED	N	T	T
2: REPEATABLE READ	N	N	T
3: SERIALIZABLE	N	N	N

A)

## Poziom izolacji 0 (READ UNCOMMITTED)

---

Poziom izolacji **READ UNCOMMITTED** dopuszcza czytanie danych niezatwierdzonych (*uncommitted*), tj. danych które zostały zmienione przez transakcję jeszcze aktywną. Popularnie określany jest jako *dirty read* („brudne czytanie”).

Za operacje konfliktowe uważa się tylko parę operacji zapisu, a dwie operacje, z których jedna jest operacją odczytu nie są operacjami konfliktowymi.

- Reguły współbieżności dla tego poziomu izolacji przedstawiono poniżej w tablicy (T oznacza, że operacje mogą być wykonywane współbieżnie, czyli nie są konfliktowe, N – oznacza brak współbieżności, a więc konfliktowość).

Reguły współbieżności dla poziomu izolacji 0 - **READ UNCOMMITTED**

	Read	Write
Read	T	T
Write	T	N

- ❑ Przyjęcie tego rodzaju współbieżności operacji może doprowadzić do braku odtwarzalności, kaskady odrzuceń, anomalii powtórnego czytania oraz do pojawiania się fantomów.
- ❑ Zaletą tego poziomu izolacji jest jednak to, że uzyskujemy wysoki współczynnik współbieżność transakcji.
- ❑ Ten poziom izolacji należy wybierać dla tych transakcji, które nie wykorzystują wczytanych danych do modyfikacji bazy danych.

B)

## Poziom izolacji 1 (READ COMMITTED)

---

- Poziom izolacji **READ COMMITTED** wprowadza zakaz czytania danych z transakcji niezatwierdzonych, a więc czytać można tylko dane zatwierdzone (*committed*)
- Przy tym poziomie izolacji dopuszczalne jest jednak zapisywanie danych w transakcjach nie zatwierdzonych.
- Jest to domyślny poziom izolacji w MS SQL Server.
- Dwie operacje, z których pierwsza jest operacją czytania, a druga operacją zapisu nie są więc konfliktowe w myśl tego poziomu izolacji. Transakcje mogą więc zapisywać w innych transakcjach.
- Reguły współbieżności dla poziomu izolacji **READ COMMITTED** przedstawiono poniżej w tablicy (T oznacza, że operacje mogą być wykonywane współbieżnie, czyli nie są konfliktowe, N - oznacza brak współbieżności, a więc konfliktowość).

Reguły współbieżności dla poziomu izolacji 1 – READ COMMITTED

	Read	Write
Read	T	T
Write	N	N

- ❑ Przyjęcie tego rodzaju współbieżności operacji eliminuje brak odtwarzalności oraz kaskadę odrzuceń.
- ❑ Na tym poziomie izolacji mogą jednak wystąpić zarówno anomalie powtórnego czytania, jak i zjawisko fantomów.

C)

## Poziom izolacji 2 (REPEATABLE READ)

- Poziom izolacji **REPEATABLE READ** wprowadza zakaz zapisywania w transakcjach niezatwierdzonych.
- Za konfliktowe uważa się takie pary operacji, gdzie co najmniej jedna jest operacją zapisu. Za niekonfliktowe uważa się tylko operacje czytania.
- Ten poziom izolacji eliminuje anomalię powtórnego czytania.
- Reguły współbieżności dla poziomu izolacji **REPEATABLE READ** przedstawiono poniżej w tablicy (T oznacza, że operacje mogą być wykonywane współbieżnie, czyli nie są konfliktowe, N - oznacza brak współbieżności, a więc konfliktowość).

Reguły współbieżności dla poziomu izolacji 2 – REPEATABLE READ

	Read	Write
Read	T	N
Write	N	N

- ❑ Przyjęcie tego rodzaju współbieżności operacji eliminuje brak odtwarzalności, kaskadę odrzuceń oraz anomalię powtórnego czytania.
- ❑ Na tym poziomie izolacji mogą pojawiać się fantomy.

D)

## Poziom izolacji 3 (SERIALIZABLE)

- Poziom izolacji **SERIALIZABLE** (szeregowalność) rozwiązuje problem fantomów.
- Należy uwzględnić formuły (warunki) definiujące zbiory danych, na których działają transakcje.
- **SERIALIZABLE** oznacza, że historia przetwarzania transakcji jest szeregowalna, a więc jest równoważna pewnej historii szeregowej (tj. takiej, gdzie wszystkie operacje jednej transakcji poprzedzają wszystkie operacje innej transakcji).



- Niech dane będą operacje  $\alpha[\phi]$  i  $\rho[\psi]$  pochodzące z dwóch różnych i aktywnych transakcji ( $\phi$  i  $\psi$  są formułami określającymi zbiory danych, na których działa operacja) oraz niech operacja  $\alpha$  poprzedza operację  $\rho$ , tzn.  $\alpha[\phi] < \rho[\psi]$ .

➤ Przyjmijmy oznaczenia:

- $X = \{x \mid \phi(x)\}$  - zbiór danych spełniających warunek  $\phi$  bezpośrednio **przed wykonaniem** operacji  $\rho[\psi]$ ,
- $Y = \{y \mid \psi(y)\}$  - zbiór danych spełniających warunek  $\psi$  bezpośrednio przed wykonaniem operacji  $\rho[\psi]$ ,
- $X' = \{x \mid \phi(x)\}$  - zbiór danych spełniających warunek  $\phi$  bezpośrednio **po wykonaniu** operacji  $\rho[\psi]$ .

Pojęcie współbieżności operacji rozszerzamy obecnie następująco:

1. Dwie operacje  $\text{Read}[\phi]$  i  $\text{Read}[\psi]$  są zawsze współbieżne.
2. Operacje,  $\alpha[\phi]$  i  $\text{Write}[\psi]$  są współbieżne, jeśli:
  - $X \cap Y = \emptyset$  oraz
  - $X = X'$ , tj.
    - zbiór  $Y$  na którym działa druga z tych operacji jest rozłączny ze zbiorem  $X$  związanym z wykonaniem pierwszej, oraz
    - wykonanie drugiej operacji nie zmieni zbioru związanego z wykonywaniem pierwszej.
3. Operacje,  $\text{Write}[\phi]$  i  $\text{Read}[\psi]$  są współbieżne, jeśli zbiór na którym działa druga z tych operacji jest rozłączny ze zbiorem związanym z wykonaniem pierwszej z nich. Formalnie:

$$X \cap Y = \emptyset.$$

- Reguły współbieżności dla poziomu izolacji SERIALIZABLE przedstawiono poniżej w tablicy (T oznacza, że operacje mogą być wykonywane współbieżnie, czyli nie są konfliktowe, N - oznacza brak współbieżności, a więc konfliktowość).

Reguły współbieżności dla poziomu izolacji 3 – SERIALIZABLE

	Read[ $\psi$ ]		Write[ $\psi$ ]	
	$X \cap Y = \emptyset$	$X \cap Y \neq \emptyset$	$X \cap Y = \emptyset \wedge X = X'$	$X \cap Y \neq \emptyset \vee X \neq X'$
Read[ $\phi$ ]	T	T	T	N
Write[ $\phi$ ]	T	N	T	N

- ❑ Przyjęcie tego rodzaju współbieżności operacji eliminuje wszystkie anomalie, w tym problem fantomów.
- ❑ Przy tym poziomie izolacji współbieżność transakcji jest najmniejsza, a efektywność przetwarzania bardzo niska. Należy więc stosować go tylko wtedy, gdy jest to naprawdę konieczne.



## E) SNAPSHOT

### Poziom izolacji SNAPSHOT

- Dane czytane przez transakcje są „transakcyjnie spójne” (ang. *transactionally consistent*), tzn. reprezentują spójną wersję danych istniejącą w chwili rozpoczęcia transakcji.
- Transakcje widzą tylko te modyfikacje, które zostały zatwierdzone przed rozpoczęciem transakcji.
- Efekt jest taki, jakby transakcja posiadała „fotkę” (ang. *snapshot*) danych zatwierdzonych w momencie jej wystartowania.

## F) PODSUMOWANIE

- Przyjęcie określonego poziomu izolacji może być źródłem problemów (anomalii) omówionych przy okazji historii przetwarzania transakcji. Może też eliminować te problemy.
- W poniższej tabelicy symbol T oznacza, że dany problem występuje przy rozważanym poziomie izolacji, N - że nie występuje.

Związek poziomów izolacji z problemami przetwarzania transakcji

Poziom izolacji	Brak odtwarzalności	Anomalia powtórnego czytania	Fantomy
0 : READ UNCOMMITTED	T	T	T
1 : READ COMMITTED	N	T	T
2 : REPEATABLE READ	N	N	T
3 : SERIALIZABLE	N	N	N

### 1.1.4 PRZYKŁADY

- Problemy związane z przetwarzaniem bazy danych przy różnych poziomach izolacji zilustrujemy teraz przykładami z MS SQL Server.
- Przypuśćmy, że w bazie danych istnieje tabela Procesor o następującej postaci:

Procesor		
Nazwa	Cena	Stan
200MMX	320	20
233MMX	370	50

- ❑ Na tabeli tej będą współbieżnie operowały dwie transakcje.
- ❑ Pierwsza z tych transakcji ma stały poziom izolacji, tj. domyślny READ COMMITTED, a poziom izolacji drugiej z nich ustalany jest za pomocą komendy SET TRANSACTION ISOLATION LEVEL

A) UNCOMMITTED

## Poziom izolacji READ UNCOMMITTED – przykład (anomalia brudnego czytania)

---

### Transakcja T1

- set transaction isolation level  
read committed; go
- begin transaction
- update Procesor set Cena = 300  
where Nazwa = '200MMX'  
*(T1 zmienia cenę)*
- rollback  
*(T1 wycofuje zmianę)*

### Transakcja T2

- set transaction isolation level  
**read uncommitted**; go
- begin transaction
- select Cena from Procesor  
where Nazwa = '200MMX'  
*(T2 czyta zmienioną cenę (300))*
- *T2 ma niepoprawną informację o cenie  
(300 zamiast 320)*

B) COMMITED

## Poziom izolacji READ COMMITTED – przykład (anomalia powtórnego czytania)

---

### Transakcja T1

- set transaction isolation level  
read committed; go
- begin transaction
- select Cena, Stan from Procesor  
where Nazwa = '200MMX'  
*(T1 czyta cenę i stan: (320, 20))*
- select sum(Cena\*Stan)  
from Procesor  
where Nazwa = '200MMX'  
*(T1 czeka na zakończenie T2)*
- *Wykonanie oczekującej operacji select  
dla T1. Wynik sprzeczny z poprzednią  
operacją select (zwracana wartość:  
310\*20 = 6200)*

### Transakcja T2

- set transaction isolation level  
read **committed**; go
- begin transaction
- update Procesor set Cena = 310  
where Nazwa = '200MMX'  
*(T2 zmienia cenę wczytaną przez T1)*
- commit

C) Repeatable read

## Poziom izolacji REPEATABLE READ – przykład (eliminacja anomalii powtórnego czytania)

### Transakcja T1

- set transaction isolation level  
**repeatable read**; go
- begin transaction
- select Cena, Stan from Procesor  
where Nazwa = '200MMX'  
*(T1 czyta cenę i stan: (320, 20))*
- select sum(Cena\*Stan)  
from Procesor  
where Nazwa = '200MMX'  
*(T1 oblicza wartość :  $320 * 20 = 6400$ )*
- commit

### Transakcja T2

- set transaction isolation level  
**repeatable read**; go
- begin transaction
- update Procesor set Cena = 310  
where Nazwa = '200MMX'  
*(T2 czeka na zakończenie T1)*
- Wykonanie oczekującej operacji update  
dla T2.

D) Rep. read (fantom)

## Poziom izolacji REPEATABLE READ – przykład (problem fantomów)

### Transakcja T1

- set transaction isolation level  
**repeatable read**; go
- begin transaction
- select Cena, Stan from Procesor  
where Nazwa = '200MMX'  
*(T1 czyta cenę i stan: (320, 20))*
- select sum(Cena\*Stan)  
from Procesor  
where Nazwa = '200MMX'  
*(T1 oblicza wartość :  $6400 + 2500$ )*

### Transakcja T2

- set transaction isolation level  
**repeatable read**; go
- begin transaction
- insert into Procesor  
values('200MMX', 250, 10)  
*(T2 dołącza nową krotkę – „fantom”)*
- commit

## E) Snapshot

### Poziom izolacji SNAPSHOT – przykład (czytanie fotki, wersji transakcyjnie spójnej)

#### Transakcja T1

- set transaction isolation level  
read committed; go
- begin transaction
- update Procesor set Cena = 300  
where Nazwa = '200MMX'  
*(T1 zmienia cenę)*

Results		Messages	
	Nazwa	Cena	Stan
1	200MMX	320,00	20
2	233MMX	370,00	50

300

#### Transakcja T2

- set transaction isolation level  
**snapshot**; go
- begin transaction
- select Cena from Procesor  
where Nazwa = '200MMX'  
*(T2 czyta starą spójną cenę (320))*



## F) Serializable

### Poziom izolacji SERIALIZABLE

- Przy poziomie izolacji SERIALIZABLE zapewnia szeregowalność transakcji.
- Eliminuje wszelkie anomalie.
- W szczególności czuwa, aby transakcja późniejsza T2 podczas modyfikacji nie wpływała w żaden sposób na dane czytane lub zapisywane przez wcześniejszą transakcję T1.

### Poziom izolacji SERIALIZABLE – przykład (eliminacja problemu fantomów)

#### Transakcja T1

- set transaction isolation level  
**serializable**; go
- begin transaction
- select Cena, Stan from Procesor  
where Nazwa = '200MMX'  
*(T1 czyta cenę i stan: (320, 20))*
- select sum(Cena\*Stan)  
from Procesor  
where Nazwa = '200MMX'  
*(T1 oblicza wartość : 6400)*
- commit

#### Transakcja T2

- set transaction isolation level  
**repeatable read**; go
- begin transaction
- insert into Procesor  
values('200MMX', 250, 10)  
*(T2 czeka na zakończenie T1, gdyż  
zmienia/wprowadza dane spełniające  
warunek selekcji realizowanej przez T1)*

## 1.2. Blokowanie dwufazowe (B2F) jako metoda zarządzania transakcjami współbieżnymi. Poprawność B2F.

### 1.2.1 Protokół blokowania dwufazowego B2F

#### Planista:

- a) to wyspecjalizowany moduł odpowiedzialny za zarządzanie transakcjami
- b) związany z menadżerem danych (MD)
- c) jego liczba jest równa liczbie lokalnych BD w systemie

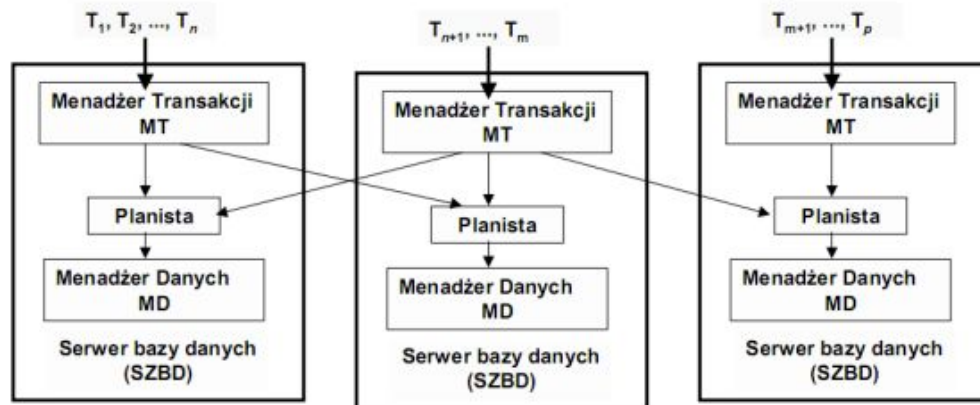
#### Działanie planisty:

##### Planista przyjmując operację O może:

- a) przekazać ją do wykonania (menadżerowi danych MD)
- b) umieścić ją w kolejce (gdy jest w konflikcie z inną operacją, czeka na zakończenie tamtej)

- c) odrzucić ją (wraz z całą transakcją) jeżeli zajdzie konieczność ze względu na zakleszczenie

## Przetwarzanie transakcji



### Protokół blokowania dwufazowego B2F

- **realizuje**: strategię planisty w komercyjnych SZBD - planista B2F
- **ma 2 fazy**: fazę wzrostu oraz fazę zmniejszania
- **zaleta**: każda historia przetwarzania transakcji utworzona przez planistę B2F jest poprawna
- **wada**: może prowadzić do powstania zakleszczeń (deadlock)

### Reguły planisty B2F

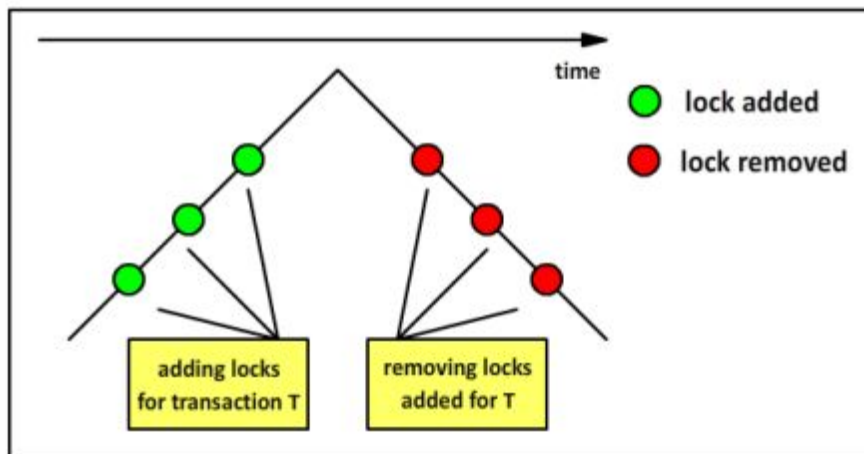
- 1) jeśli operacja  $p_i[x]$  może być wykonana, to planista:
  - a) zakłada blokadę (S dla odczytu lub X dla zapisu) na daną  $x$  dla transakcji  $T_i$
  - b) przekazuje ją do wykonania menadżerowi danych MD
 jeśli operacja ta nie może być wykonana to planista:
  - a) umieszcza ją w kolejce
- 2) zdjęcie blokady następuje, gdy MD powiadomi planistę o zakończeniu operacji
- 3) po zdjęciu blokady założonej dla transakcji  $T_i$ ,  $T_i$  nie można założyć innej blokady

W procesie wykonywania transakcji wyróżniamy 2 fazy:

- faza **zakładania blokad** (faza wzrostu)
- faza **zdejmowania blokad** (faza zmniejszania)

Sposób zdejmowania blokad należy do danego poziomu izolacji





### 1.2.2 Realizacja poziomów izolacji przez planistę B2F

#### Założenia:

- blokadę S(SHARED) dla tej samej danej może uzyskać dowolną liczba transakcji
- blokadę X(EXCLUSIVE) dla konkretnej danej może uzyskać tylko jedną transakcja
- S i X nie mogą być jednocześnie założone na tę samą daną dla dwóch różnych transakcji
- uzyskanie blokady S jest konieczne dla odczytania danej
- uzyskanie blokady X jest konieczne dla zapisania (modyfikacji, usunięcia) danej

#### Realizacja B2F dla poziomu izolacji 0

- zdejmowanie blokad natychmiast po zakończeniu operacji
- zdjęcie blokady S z zablokowanej danej - natychmiast po zakończeniu operacji Read
- na czas realizacji operacji Write, zablokowana dana zmienia blokadę X na S (można ją czytać, ale nie zapisać)
- całkowite zdjęcie blokady dla operacji Write po zatwierdzeniu transakcji

#### Realizacja B2F dla poziomu izolacji 1

- zdjęcie blokady S z zablokowanej danej - natychmiast po zakończeniu operacji Read
- zdjęcie blokady X z zablokowanej danej - po zatwierdzeniu transakcji

#### Realizacja B2f dla poziomu izolacji 2

- zdejmowanie blokad następuje dopiero po zatwierdzeniu transakcji

#### Realizacja B2F dla poziomu izolacji 3

- dodatkowe blokady sterowane przez warunki tzn. zablokowana jest możliwość takich aktualizacji (w tym dołączania), które wpływają na rekordy spełniające określony warunek.

### 1.2.3 Poprawność B2F

#### Poprawność B2F

Niech  $H = (o_1, o_2, \dots, o_m)$  będzie historią przetwarzania dla zbioru transakcji  $\Sigma = (T_1, T_2, \dots, T_n)$

Mówimy, że *operacja  $o_j$  poprzedza w  $H$  operację  $o_k$* , jeśli:

- $j < k$
- operacje te są konfliktowe
- zapisujemy jako:  $o_j < o_k$

Mówimy, że *transakcja  $T$  poprzedza w  $H$  transakcję  $T'$*  jeśli:

- zachodzi 1 z 2 warunków:
  - istnieją w  $H$  operacje  $o$  i  $o'$  (pochodzące odpowiednio z  $T$  i  $T'$ ), takie że  $o < o'$
  - istnieje transakcja  $T''$ , taka że  $T < T'' < T'$
- zapisujemy jako:  $T < T'$

Historię  $H$  nazywamy *poprawną*, jeśli dla każdej pary transakcji  $T, T' \in \Sigma$  spełniony jest warunek:

- jeśli  $T < T'$ , to nieprawda, że  $T' < T$

#### Twierdzenie i dowód poprawności B2F

### 1.2.4 Twierdzenie:

*Każda historia przetwarzania utworzona przez planistę B2F jest poprawna.*

#### Dowód:

Pokażemy, że w każdej historii przetwarzania  $H$  utworzonej przez planistę B2F, dla każdej pary transakcji  $T$  i  $T'$  zachodzi warunek:

*jeśli  $T < T'$ , to nieprawda, że  $T' < T$*

1. jeśli  $T < T'$ , to założenie blokady na  $T'$  musi być poprzedzone zdjęciem blokady na  $T$
2. **dowód nie wprost:**
  - a. założymy, że w  $H$  zachodzą:  $T < T'$  oraz  $T' < T$
  - b. uwzględniając punkt (1):
    - i. założenie blokady na  $T$  musi być poprzedzone zdjęciem blokady z tej samej transakcji
    - ii. sprzeczność z trzecią regułą B2F
3. **z punktu (2) wynika, że:**
  - a. historia przetwarzania  $H$  nie mogła by utworzona przez planistę B2F
  - b. twierdzenie jest poprawne

## 1.3. Odtwarzanie baz danych. Algorytm UNDO/REDO. Problemy zatwierdzania i odtwarzanie transakcji rozproszonych.

### 1.3.1 Odtwarzanie baz danych (recovery)

System bazy danych musi być w stanie odtworzyć swój poprawny stan w sposób automatyczny, a więc bez interwencji człowieka.

Operacja odtwarzania powinna być automatycznie inicjowana w chwili restartu systemu.

Organizacja systemu bazy danych i organizacja procesów jej przetwarzania musi być tak pomyślana, aby odtwarzanie takie było możliwe.

Odtwarzanie realizuje postulat trwałości ze zbioru postulatów ACID.

Odtwarzanie dotyczy awarii bazy danych wywołanych nieoczekiwanym przerwaniem przetwarzania transakcji (np. w wyniku awarii zasilania), a nie awarii krytycznych wywołanych np. uszkodzeniem pamięci dyskowej.

Omawianego w dalszym ciągu odtwarzania (recovery) nie należy mylić z odtwarzaniem z archiwum (restore).

## Algorytm UNDO/REDO

---

- Algorytm UNDO/REDO opisuje proces przetwarzania transakcji z uwzględnieniem możliwości odtworzenia bazy danych w przypadku awarii.
- Odtwarzanie bazy danych następuje w chwili restartu systemu po jego awarii.
- Algorytm UNDO/REDO jest stosowany wówczas, gdy w procesie odtwarzania konieczne jest wykonanie operacji UNDO dotyczącej przerwanych transakcji oraz operacji REDO dotyczącej zatwierdzonych transakcji.
- UNDO oznacza przywrócenie poprzednich (zatwierdzonych) wartości tym danym, które były zmieniane przez transakcje przerwane przed ich zatwierdzeniem.
- Operacja jest konieczna wtedy, gdy strategia wymiany stron w PCh prowadzona jest przez MPCh niezależnie od MO.
- Może więc się zdarzyć, że blok danych z PCh został zapamiętany w bazie danych w momencie, gdy transakcja zapisująca w nim dane nie została jeszcze zatwierdzona. W przypadku awarii należy więc te zmiany wycofać.

- REDO oznacza ponowne wykonanie operacji aktualizacji danych wykonanych przez transakcje zatwierdzone.
- Operacja ta jest konieczna wtedy, gdy MPCh prowadzi własną strategię wymiany stron.
- Może więc się zdarzyć, że mimo iż transakcja została zatwierdzona, to wykonane przez nią zmiany nie zostały jeszcze zrzucane z PCh do bazy danych. W przypadku awarii nastąpi utrata efektów pracy transakcji.

1.4. Zarządzanie transakcjami z wykorzystaniem znaczników czasowych.

1.5. Blokowanie optymistyczne (patrz LINQ).



## 2. Indeksowanie i haszowanie w bazach danych

### 2.1. Budowa indeksów wielopoziomowych. Złożoność przeszukiwania i aktualizacji.

#### Indeks wielopoziomowy

BlokID	Nazwisko	PTR
12	Andrzejewski	9
	Mazela	10
	Skrzypek	11

BlokID	Nazwisko	PTR
9	Andrzejewski	1
	Borak	2
	Karczewski	3

BlokID	Nazwisko	PTR
10	Mazela	4
	Musielak	5
	Płócieniak	6

BlokID	Nazwisko	PTR
11	Skrzypek	7
	Wojcieszek	8

Nazwisko	PTR
Andrzejewski	15
Banasiak	3
Błaszkievicz	20

Nazwisko	PTR
Borak	22
Filipiak	21
Grzesiak	23

Nazwisko	PTR
Karczewski	14
Kowalik	1
Lisiecki	11

Nazwisko	PTR
Mazela	24
Mazurek	18
Michalak	13

Nazwisko	PTR
Musielak	17
Nowiński	8
Panuś	7

Nazwisko	PTR
Płócieniak	4
Rogacki	2
Sklepik	9

Nazwisko	PTR
Skrzypek	16
Sołtysiak	10
Walkiewicz	6

Nazwisko	PTR
Wojcieszek	12
Wszedybył	5
Żakowski	19

RID	NrEwid	Nazwisko	Imie	Grupa	...
1	49435	Kowalik	Karol	I-2	...
2	49479	Rogacki	Piotr	I-3	...
3	49406	Banasiak	Ewa	I-1	...
4	49476	Płócieniak	Karol	I-3	...
5	49515	Wszedybył	Karol	I-4	...
6	49503	Walkiewicz	Adrian	I-4	...
7	77400	Panuś	Agnieszka	I-1	...
8	49462	Nowiński	Arkadiusz	I-3	...
9	79082	Sklepik	Aron	I-1	...
10	49491	Sołtysiak	Barbara	I-4	...
11	49446	Lisiecki	Jerzy	I-2	...
12	49510	Wojcieszek	Jerzy	I-2	...
13	49456	Michalak	Jan	I-3	...
14	77369	Karczewski	Arkadiusz	I-1	...
15	49402	Andrzejewski	Jan	I-4	...
16	83762	Skrzypek	Jan	I-4	...
17	49458	Musielak	Dariusz	I-3	...
18	77377	Mazurek	Piotr	I-1	...
19	49520	Żakowski	Piotr	I-4	...
20	49412	Błaszkievicz	Bartłomiej	I-2	...
21	49419	Filipiak	Dominik	I-2	...
22	49409	Borak	Filip	I-1	...
23	49423	Grzesiak	Karol	I-4	...
24	49454	Mazela	Grzegorz	I-3	...

Sołtysiak?

12 → 11 → 7 (10)  
(trzy elementy w bloku)

12:04

**Indeksy wielopoziomowe** – dla pierwszego poziomu tworzymy indeks podstawowy i nazywamy go indeksem drugiego poziomu. Analogicznie dla poziomu drugiego, gdzie tworzy się indeks poziomu trzeciego. Indeksy wielopoziomowe można konstruować z wykorzystaniem indeksów podstawowych, wtórnych i zgrupowanych, pod warunkiem jednak, że indeks pierwszego poziomu ma różne wartości i rekordy stałej długości.



## Indeksy Wielopoziomowe

### Poziom drugi

4	○
25	○
44	○
78	○

### Poziom pierwszy

4	○
9	○
11	○
16	○

25	○
28	○
38	○
40	○

44	○
58	○
70	○
74	○

78	○
	○
	○
	○

### Plik danych z polem klastrowania

4	
7	

9	
10	

11	
14	

16	
22	

25	
26	

28	
33	

38	
39	

40	

2.2. Dynamiczne tworzenie indeksu o strukturze B-drzewa przy dołączaniu/usuwaniu elementów. Przykłady.

2.3. Indeksy o strukturze B-drzewa. Wpływ indeksów na złożoność operacji wyszukiwania i modyfikowania (dołączanie i usuwanie) danych.

2.4. Obliczenia związane z szacowaniem wysokości B-drzewa – i związanej z tym zajętości pamięci – przy zadanych parametrach indeksu.

**Zadanie 2** (6 pkt.)

1) Omów budowę indeksu o postaci B-drzewa.

2) Oblicz **maksymalne** zużycie pamięci konieczne na zapamiętanie indeksu o postaci B-drzewa, przy następujących danych:

$N = 100\,000$  - liczba rekordów w pliku głównym,

$B = 1\text{kB}$  - wielkość bloku,

$P = 4B$  - wielkość pola wskaźnikowego,

$A = 8B$  - wielkość pola adresowego,

$X = 20B$  - wielkość pola klucza indeksowania.

Jaka jest wówczas **wysokość** B-drzewa?

## Zadanie 2

2)

Najpierw obliczamy  $m$ . Musi być to taka największa liczba całkowita, dla której

$$2m(P + A + X) + P \leq B$$

Stąd  $m = 15$ .

Przy minimalnym wypełnieniu bloków indeksu ich liczba jest równa:

$$1 + \left\lceil \frac{99999}{15} \right\rceil = 6668$$

Przy maksymalnym wypełnieniu bloków indeksu ich liczba jest równa:

$$\left\lceil \frac{100000}{30} \right\rceil = 3334$$

Liczba bloków,  $M$ , (o wielkości 1kB) pamięci konieczna na zapamiętanie tego indeksu ograniczona jest więc przedziałem

$$3334 \leq M \leq 6668.$$

Przy jakiej wysokości B-drzewa (indeksu) zapewniona jest możliwość zapamiętania takiej liczby bloków?

$h$	Przy minimalnym wypełnieniu		Przy maksymalnym wypełnieniu	
	Bloków na poziomie $h$	Łącznie w drzewie o wysokości $h$	Bloków na poziomie $h$	Łącznie w drzewie o wysokości $h$
1	1	1	1	1
2	2	3	31	32
3	32	35	961	993
4	512	547	29791	30784
5	8192	8739		

Indeks będzie więc miał 4 poziomy. Wypełnienie bloków nie będzie ani minimalne, ani maksymalne tylko pośrednie. (Dlaczego?)

2.5. Algorytmy haszowania danych w bazach danych i ich wpływ na złożoność operacji wyszukiwanie i modyfikowania (dołączanie i usuwanie) danych.

## 3. Optymalizacja zapytań

3.1. Optymalizacja selekcji. Przykład: przedyskutować strategię wykonania operacji selekcji w tabeli R z warunkiem ( $A = a$ ) AND ( $B < b$ ). Jak stosowanie indeksów wpływa na efektywność wykonywania tej operacji.

Oszacowanie liczby transmisji między pamięcią operacyjną a pamięcią dyskową.

	Struktura dostępu względem atrybutu A	Wyszukiwanie równościowe $A = c$ (warunek spełniony jest przez s rekordów, $s \geq 1$ )	Wyszukiwanie nierównościowe $A \theta c, \theta \in \{<, <=, >, >=\}$ (warunek spełniony jest przez połowę rekordów)
		(a)	(b)
UL	Plik nieuporządkowany, wyszukiwanie liniowe dla unikatowego atrybutu.	$\frac{b}{2}$	$b$
L	Plik nieuporządkowany, wyszukiwanie liniowe dla nieunikatowego atrybutu.	$b$	$b$
UC	Indeks klastrowy dla unikatowego atrybutu (UNIQUE-CLUSTERED)	$x + 1$	$x + \frac{b}{2}$
C	Indeks klastrowy dla nieunikatowego atrybutu (CLUSTERED)	$x + \frac{s}{bfr}$	$x + \frac{b}{2}$
UNC	Indeks nieklastrowy dla unikatowego atrybutu (UNIQUE-NONCLUSTERED)	$x + 1$	$x + \frac{b1}{2} + \frac{r}{2}$
NC	Indeks nieklastrowy dla nieunikatowego atrybutu (NONCLUSTERED)	$x + s$	$x + \frac{b1}{2} + \frac{r}{2}$

Przykład(brakuje danych żeby to wyprowadzić z poleceniu)

(q4):  $\sigma_{Plec = 'K' \text{ AND } Pensja > 4500 \text{ AND } IdDzialu = 500}$ (PRACOWNIK)

- $b = 2\ 000, r = 10\ 000, bfr = 5,$
- Indeks klastrowy, na nieunikatowym atrybucie Pensja, o parametrach:  
 $x = 3, s = 20$
- Indeks nieklastrowy, na nieunikatowym atrybucie IdDziału, o parametrach:  
 $x = 2, b1 = 4, s = 80.$
- Indeks nieklastrowy, na nieunikatowym atrybucie Płeć, o parametrach:  
 $x = 1, s = 5000$

1. Zaczynamy od Płeć = 'K', wtedy:

$$NC_a(Plec) = x + s = 1 + 5000 = 5001$$

2. zaczynamy od Pensja > 4500, wtedy:

$$C_b(Pensja) = x + b/2 = 3 + 1000 = 1003$$

2. Wyszukiwanie z użyciem indeksu:

$$NC_a(IdDzialu) = x + s = 2 + 80 = 82$$

Stosowanie indeksów znacznie wpływa na efektywność wykonania selekcji.

3.2. Optymalizacja złączenia. Przedyskutować złączenie tabel R i S według warunku  $R.A=S.B$ , według różnych strategii przy istnieniu i braku indeksów.

Oszacowanie liczby transmisji między pamięcią operacyjną a pamięcią dyskową.

Metody:

J1. złączenie metodą pętla zewnętrzna-pętla zagnieżdżona,

Koszt zależy od tego, która tabela będzie przeglądana w pętli zewnętrznej, a która w pętli wewnętrznej.

J2. złączenie metodą pętla zewnętrzna - selekcja dopasowanych krotek,

Koszt zależy od kosztu wyszukiwania zbioru rekordów z S dopasowanych do r.A.

J3. złączenie scalające plików posortowanych rosnąco wg. A i B,

W metodzie J3 każdy blok z pliku R i każdy blok z pliku S transmitowany jest dokładnie jeden raz.

$$C_{J3} = b_R + b_S + \frac{js \cdot |R| \cdot |S|}{bfr_{RS}}$$

## Przykład:

Dla tabel:

PRACOWNIK(IdPrac, Nazwisko, Imię, DataUr, Adres, Płeć, Pensja, IdKier, IdDziału)

DZIAŁ(IdDz, Nazwa, IdKier, FunduszPlac)

Oszacować koszty realizacji następujących operacji złączenia:

(q5): PRACOWNIK  $\bowtie_{IdDziału = IdDz}$  DZIAŁ

(q6): PRACOWNIK  $\bowtie_{IdPrac = IdKier}$  DZIAŁ

Przy założeniach:

- $js_{q5} = 1/125$ ,      gdyż IdDz jest kluczem głównym w tabeli DZIAŁ,
- $bfr_{PD} = 4$       rekordy wynikowe na blok.



## Rozwiązanie 1: (q5 metoda J1, Prac(Dział))

PRACOWNIK(IdPrac, Nazwisko, Imię, DataUr, Adres, Płeć, Pensja, IdKier, IdDziału)

DZIAŁ(IdDz, Nazwa, IdKier, FunduszPłac)

(q5): PRACOWNIK  $\bowtie_{IdDziału = IdDz}$  DZIAŁ

$r_p = 10\ 000$  – liczba rekordów,

$b_p = 2\ 000$  – liczba bloków,

$r_D = 125$  – liczba rekordów,

$b_D = 13$  – liczba bloków,

$js_{q5} = 1/125$ ,  $bfr_{PD} = 4$

J1: Z plikiem PRACOWNIK w pętli zewnętrznej:

$$C_{J1} = b_R + b_R \cdot b_S + \frac{js \cdot |R| \cdot |S|}{bfr_{RS}}$$

$$C_{J1} = b_P + b_P \cdot b_D + \frac{js_{q5} \cdot |PRACOWNIK| \cdot |DZIAŁ|}{bfr_{PD}} = 2000 + 2000 \cdot 13 + \frac{\frac{1}{125} \cdot 10000 \cdot 125}{4} = 30500$$

## Rozwiązanie 2: (q5 metoda J1, Dział(Prac))

PRACOWNIK(IdPrac, Nazwisko, Imię, DataUr, Adres, Płeć, Pensja, IdKier, IdDziału)

DZIAŁ(IdDz, Nazwa, IdKier, FunduszPłac)

(q5): PRACOWNIK  $\bowtie_{IdDziału = IdDz}$  DZIAŁ

$r_p = 10\ 000$  – liczba rekordów,

$b_p = 2\ 000$  – liczba bloków,

$r_D = 125$  – liczba rekordów,

$b_D = 13$  – liczba bloków,

$js_{q5} = 1/125$ ,  $bfr_{PD} = 4$

J1: Z plikiem DZIAŁ w pętli zewnętrznej

$$C_{J1} = b_R + b_R \cdot b_S + \frac{js \cdot |R| \cdot |S|}{bfr_{RS}}$$

$$C_{J1} = b_D + b_D \cdot b_P + \frac{js_{q5} \cdot |PRACOWNIK| \cdot |DZIAŁ|}{bfr_{PD}} = 13 + 13 \cdot 2000 + \frac{\frac{1}{125} \cdot 10000 \cdot 125}{4} = 28513$$



## Rozwiązanie 3: (q5 metoda J2, Prac(Dział))

PRACOWNIK(IdPrac, Nazwisko, Imię, DataUr, Adres, Płeć, Pensja, IdKier, IdDziału)

DZIAŁ(IdDz, Nazwa, IdKier, FunduszPłac)

(q5): PRACOWNIK  $\bowtie_{IdDziału = IdDz}$  DZIAŁ

$r_p = 10\ 000$  – liczba rekordów,  $b_p = 2\ 000$  – liczba bloków,

$r_D = 125$  – liczba rekordów,  $b_D = 13$  – liczba bloków,

$js_{q5} = 1/125$ ,  $bfr_{PD} = 4$

Indeks klastrowy, na unikalnym atrybucie IdDz, o parametrach:  $x = 1$ ,  $s = 1$ .

J2: Stosujemy metodą J2 z plikiem PRACOWNIK w pętli zewnętrznej i z indeksem klastrowym na pliku DZIAŁ wg unikalnego atrybutu IdDz:

$$C_{J2} = b_R + |R| \cdot C_{sel} + \frac{js \cdot |R| \cdot |S|}{bfr_{RS}}$$

$$C_{J2} = b_P + |PRACOWNIK| \cdot (x+1) + \frac{js_{q5} \cdot |PRACOWNIK| \cdot |DZIAŁ|}{bfr_{PD}} =$$

$$2000 + 10000 \cdot 2 + \frac{\frac{1}{125} \cdot 10000 \cdot 125}{4} = 24500$$

12:05

28

## Rozwiązanie 4: (q5 metoda J2, Dział(Prac))

PRACOWNIK(IdPrac, Nazwisko, Imię, DataUr, Adres, Płeć, Pensja, IdKier, IdDziału)

DZIAŁ(IdDz, Nazwa, IdKier, FunduszPłac)

(q5): PRACOWNIK  $\bowtie_{IdDziału = IdDz}$  DZIAŁ

$r_p = 10\ 000$  – liczba rekordów,  $b_p = 2\ 000$  – liczba bloków,

$r_D = 125$  – liczba rekordów,  $b_D = 13$  – liczba bloków,

$js_{q5} = 1/125$ ,  $bfr_{PD} = 4$

Indeks nieklastrowy, na unikalnym atrybucie IdKier, o parametrach:  $x = 2$ ,  $s = 1$ .

J2: Stosujemy metodą J2 z plikiem DZIAŁ w pętli zewnętrznej i z indeksem nieklastrowym na pliku PRACOWNIK wg nieunikalnego atrybutu IdDziału:

$$C_{J2} = b_R + |R| \cdot C_{sel} + \frac{js \cdot |R| \cdot |S|}{bfr_{RS}}$$

$$C_{J2} = b_D + |DZIAŁ| \cdot (x+s) + \frac{js_{q5} \cdot |PRACOWNIK| \cdot |DZIAŁ|}{bfr_{PD}} =$$

$$13 + 125 \cdot (2+80) + \frac{\frac{1}{125} \cdot 10000 \cdot 125}{4} = 12763$$

12:05

29

## Rozwiązania: (q5). Podsumowanie

- Najniższym kosztem charakteryzuje się przypadek (4).
- Uwaga: Jeśli założymy, że w systemie dostępnych jest co najmniej 15 buforów zamiast 3, to 13 z nich można by wykorzystać do pamiętania bloków pliku DZIAŁ. Wówczas koszt dla przypadku (2) zostałby znacząco zredukowany i wynosiłby:

$$C_{J1} = b_D + b_P + \frac{js_{q5} \cdot |PRACOWNIK| \cdot |DZIAL|}{bfr_{PD}} = 13 + 2000 + \frac{\frac{1}{125} \cdot 10000 \cdot 125}{4} = 4513$$

- I wówczas ta metoda byłaby najbardziej efektywna.

3.3. Optymalizacja heurystyczna. Podaj drzewo syntaktyczne przed i po zastosowaniu optymalizacji heurystycznej dla wybranego nietrywialnego przykładu.

Heurystyczna technika optymalizacji wykorzystuje reguły heurystyczne w celu modyfikowania wewnętrznej reprezentacji zapytania (drzewa zapytania) w celu zwiększenia oczekiwanej wydajności działania

Analizator składniowy najpierw generuje początkową reprezentację wewnętrzną, która jest optymalizowana zgodnie z regułami heurystycznymi (np. stosowanie operacji selekcji i projekcji przed operacją złączenia)

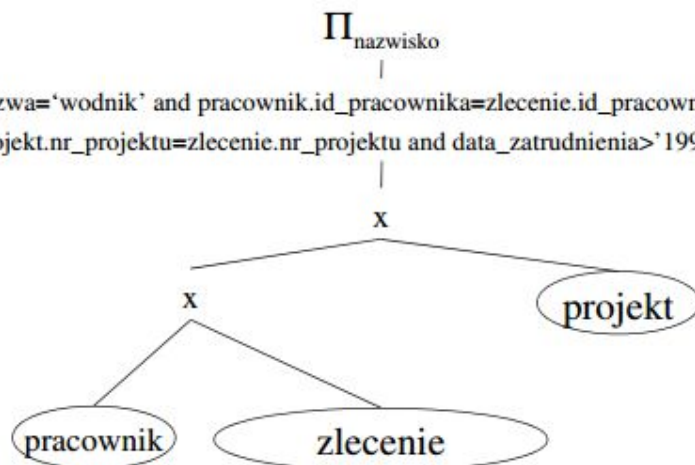
Otrzymujemy końcowe drzewo zapytania a następnie generuje się plan wykonania zapytania w celu wykonania grup operacji

**Przykład:**

Select nazwisko from pracownik, projekt, zlecenie

where nazwa='wodnik' and pracownik.id\_pracownika=zlecenie.id\_pracownika and  
projekt.nr\_projektu=zlecenie.nr\_projektu and data\_zatrudnienia>'1998-12-31';

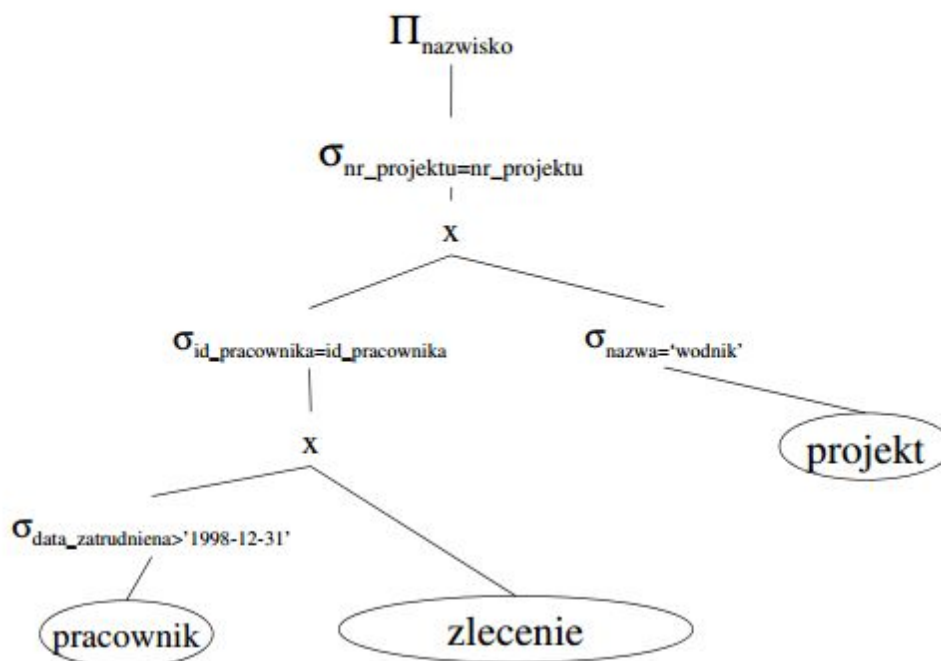
Początkowe drzewo zapytań:



22

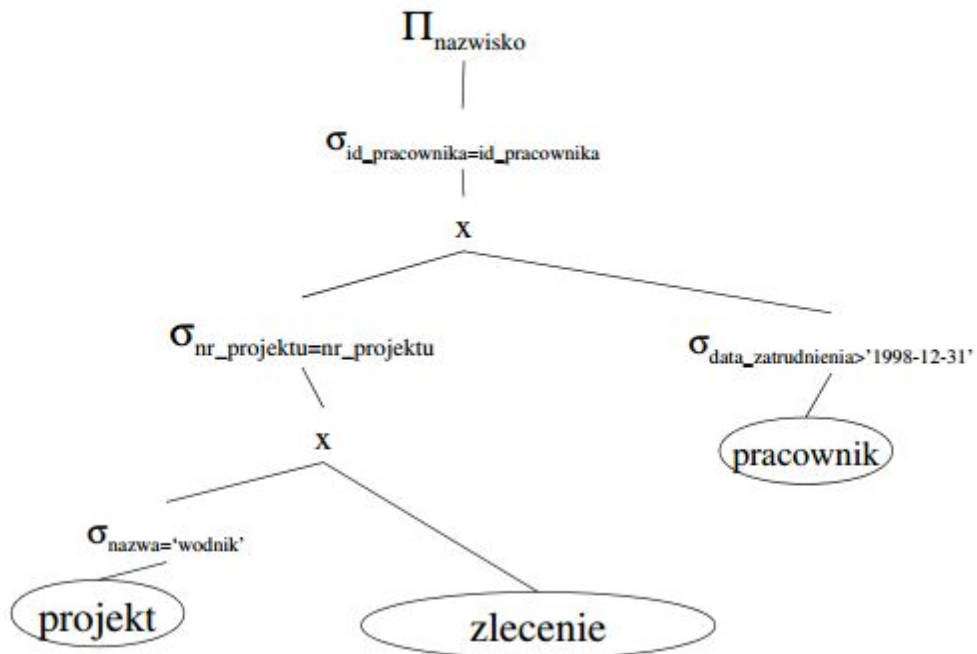
## Przykład cd

Przeniesienie operacji select w dół drzewa



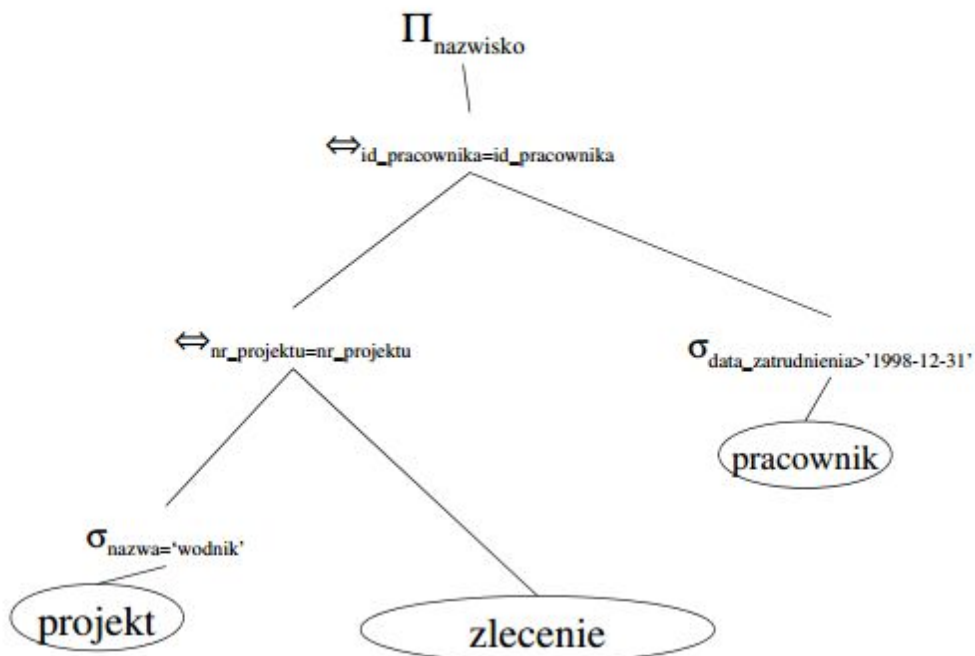
## Przykład cd

Zastosowanie bardziej restrykcyjnej operacji select jako pierwszej



## Przykład cd

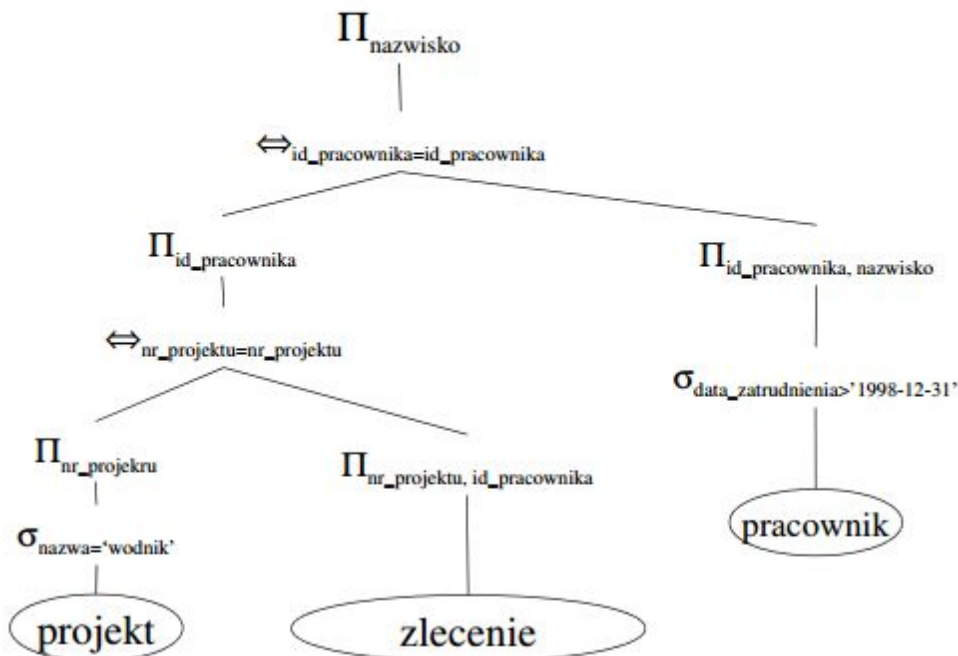
Zastąpienie iloczynu kartezjańskiego i select operacją join





# Przykład cd

## Redukcja liczby atrybutów



## 4. Entity Framework i LINQ

4.1. Przetwarzanie relacyjnej bazy danych z poziomu języka C# i z wykorzystaniem ADO.NET, Entity Framework i LINQ (metody ORM).

**4.1.1. ADO .NET:** (tego nie znalazłem na prezentacjach)

- zbiór bibliotek typu **obiekowego** które pozwalają zarządzać danymi w **platformie .NET**
- Przede wszystkim chodzi o **bazy danych**, jednak ADO .NET może też współpracować z **plikami tekstowymi**, arkuszami **Excela** czy plikami **XML**.
- **podstawowe** klasy: **DataSet** może przechowywać duże ilości danych w postaci **tabel** (obiekt klasy **DataTable**), ich **relacji** (obiekt klasy **DataRelation**) oraz **więzów spójności** (czyli modyfikatory NOT NULL, UNIQUE, PRIMARY KEY etc.)
- biblioteki znajdują się w **przestrzeni nazw System.Data** natomiast biblioteki do konkretnych źródeł danych znajdują się w **System.Data.ZrodloDanych**,

gdzie *ZrodloDanych* musimy użyć nazwy odpowiedniej biblioteki, np. **SqlClient** dla serwera MS-SQL, **OracleClient** dla Oracle SQL, **OleDb** dla danych o interfejsie OleDb, np. pliki programu Acces lub Excel (patrzcie na foto ale raczej foto żeby zrozumieć a nie zakuć :) )

## ADO.NET and .NET Base Class Library (Cont.)

Namespace	Description
System.Data	Classes, interfaces, delegates, and enumerations that define and partially implement the ADO.NET architecture
System.Data.Common	Classes shared by .NET Framework data providers
System.Data.Design	Classes that can be used to generate a custom-typed data set
System.Data.Odbc	The .NET Framework data provider for ODBC
System.Data.OleDb	The .NET Framework data provider for OLE DB
System.Data.Sql	Classes that support SQL Server-specific functionality
System.Data.OracleClient	The .NET Framework data provider for Oracle
System.Data.SqlClient	The .NET Framework data provider for SQL Server
System.Data.SqlServerCe	The .NET Compact Framework data provider for SQL Server Mobile
System.Data.SqlTypes	Classes for native SQL Server data types
Microsoft.SqlServer.Server	Components for integrating SQL Server and the CLR

- do **odczytu danych** ADO .NET udostępnia kolejno następujące klasy: **Connection** (do **nawiązaniu połączenia** w oparciu o connectionstring), **Command** (do **wykonania zapytania** w oparciu o zapytanie (xD) ) i **DataReader** (do **odczytu danych** gdy połączenie i zapytanie zostanie wykonane poprawnie)

Uwaga - nie istnieje ogólne polecenie command, datareader etc, tylko specyficzne, np. **SqlDataReader** etc - przykład niżej

Data Provider	Namespace	Connection Class
ODBC	System.Data.Odbc	OdbcConnection
OLE DB	System.Data.OleDb	OleDbConnection
Oracle	System.Data.OracleClient	OracleConnection
SQL Server	System.Data.SqlClient	SqlConnection
SQL Server CE	System.Data.SqlServerCe	SqlCeConnection



- przykład dla ms-sql  
<http://pastebin.com/CEMRjuq0>
- przykład dla bazy z programu Acces  
<http://pastebin.com/Gtb74PJq>
- przykład dla Oracle-SQL  
<http://pastebin.com/qvArjy8w>
- bardzo **profesjonalna** prezentacja po angielsku - 50 stron ale przejrzeć warto  
<http://www.slideshare.net/ngeamsoly/3-adonet>

#### 4.1.2. Entity Framework:

- jest komponentem .NET Framework
- jest zbiorem technologii ADO.NET wspierających tworzenie **oprogramowania zorientowanego na dane** (and. data-oriented)
- jest środowiskiem (frameworkiem) realizacji odwzorowań obiekty-relacje (**ORM** – Object-Relational Mapping), które umożliwia użytkownikowi pracę z **danymi relacyjnymi tak jak z obiektami**. Eliminuje to konieczność pisania większości kodu dostępu do danych.
- Korzystając z ER użytkownik wydaje **zapytania w języku LINQ**, a następnie operuje na nich jak na **silnie typowanych obiektach**. (silnie typowane obiekty to podział na typy i konsekwencja w używaniu tychże typów, czyli np. nie zrobimy  $6 * „3”$  ( $\text{int} * \text{string}$ ) - ale np. w php czy js dałoby się to zrobić bo tam nie ma silnie typowanych obiektów)
- EF jest rozszerzeniem środowiska ADO.NET dając użytkownikowi **automatyczny mechanizm** operowania na bazie danych **bez** konieczności korzystania z **DataReader i DataSet**.
- EF udostępnia **współbieżność optymistyczną** w celu przyspieszenia dostępu do danych; EF zapamiętuje w **pamięci podręcznej** zawartość i jeżeli zawartość bazy się nie zmieniła, dane są odczytywane lokalnie; **zmiany** w bazie danych sprawdzane są **na podstawie stempla czasowego** w kolumnie **RowVersion**, który jest **aktualizowany przy każdej operacji** na danej tabeli
- EF może działać **asynchronicznie** (wymagane użycie słowa async w funkcji oraz zwracania obiektu typu Task) oraz funkcję await, która oznacza, że wątek wywołujący może kontynuować swoje działanie do czasu zakończenia oznakowanego polecenia asynchronicznego; przykłady (do zerknięcia, nauka na pamięć raczej jest bez sensu):

## Asynchroniczność: czytanie

<http://www.entityframeworktutorial.net/EntityFramework6/async-query-and-save.aspx>

```
private static async Task<Student> GetStudent()
{
    Student student = null;
    using (var context = new StudiaBDEntities1())
    {
        Console.WriteLine("Start GetStudent...");
        student = await (context.Students.Where(s => s.IdStud == 1)
                                .FirstOrDefaultAsync<Student>());
        Console.WriteLine("Koniec GetStudent...");
        Console.WriteLine(student.IdStud.ToString() + " " + student.NazwiskoStud.Trim()
                                + " " + student.Kierunek);
    }
    return student; //tutaj student jest obiektem (krotką)
}
```

## Asynchroniczność: zapisywanie

<http://www.entityframeworktutorial.net/EntityFramework6/async-query-and-save.aspx>

```
private static async Task SaveStudent(Student editedStudent)
{
    using (var context = new StudiaBDEntities1())
    {
        context.Entry(editedStudent).State = EntityState.Modified;
        Console.WriteLine("Start SaveStudent...");
        int x = await (context.SaveChangesAsync());
        Console.WriteLine("Koniec SaveStudent...");
    }
}
```

- **transakcja** – zakres, w którym znajdują się **polecenia związane z operacjami na bazie danych**. W przypadku EF możemy wykonywać wiele operacji na bazie danych, ale **zostaną one wykonane dopiero w momencie wywołania funkcji `context.SaveChanges()`** – dzięki temu operacje te zostaną **zoptymalizowane** przez EF i zajmą mniej czasu i zasobów; *Database.BeginTransaction* i *Database.UseTransaction* to metody pozwalające **kontrolować transakcje**

użycie: trzy sposoby na stworzenie modelu danych.

- Database First - **baza danych istnieje** przed kodowaniem aplikacji. Za pomocą *ADO.NET Entity Data Model* dodajemy istniejące w bazie tabele, widoki oraz procedury składowane do projektu.
- Code First - **najpierw tworzymy klasy** w aplikacji, a następnie na podstawie tych klas chcemy utworzyć bazę danych
- Model First - najpierw **tworzymy model** bazy danych **za pomocą specjalnego designera**, a następnie na jego podstawie tworzymy bazę danych


### 4.1.3. LINQ:

- **powszechność** teorii **zorientowania obiektowego** wymusiło powstanie LINQ, dzięki któremu w wygodny sposób możemy dokonywać operacji **przeglądania, filtrowania i projekcji** (pobieranie kolumn) zbioru danych każdym języku programowania środowiska .NET z jednoczesnym **zachowaniem ich deklaratywności** (deklaratywność to opisywanie wyniku a nie poszczególnych kroków potrzebnych do jego osiągnięcia, przeciwieństwo imperatywności)
- sprawdzanie **składni podczas kompilacji** oraz przy użyciu **IntelliSense** (to to że podkreśla nam błędy podczas pisania w Visualu)
- główną ideą LINQ jest udostępnienie **tych samych metod do obsługi różnych typów danych** (np. kolekcji (np. list), dokumentów XML, baz danych)
- Wbudowanie mechanizmów SQL-owych do języków programowania platformy .NET (LINQ to SQL) zapewnia **silne typowanie dla danych relacyjnych** (czyli że dane z tabeli bazy danych będą określonego typu np. int, string, jakaś nasza klasa etc.) zachowując **wagę idei relacji** oraz **wydajność** przetwarzania zapytań w podstawowym serwerze danych.
- przykład „gołego” **LINQ** i LINQ w postaci wyrażenia **lambda** (osobiście preferuję to drugie bo jest bardziej ludzkie aczkolwiek oba działają tak samo wydajnie i zawsze są zamienne)

**Źródło danych:**  
`string[] Imiona = { "Katarzyna", "Jan", "Adam", "Piotr", "Paweł" };`

**Polecenie:**  
Wyszukaj imiona o długości większej niż 5, uporządkuj alfabetycznie i zamień na duże litery.

**Składnia zapytania LINQ:**  
`var q = from s in Imiona  
 where s.Length >= 5  
 orderby s  
 select s.ToUpper();  
foreach (string s in q)  
 Console.WriteLine(s);`



**Składnia wyrażenia lambda:**  
`var w =  
 Imiona.Where(s => s.Length >= 5).OrderBy(s=>s).Select(s=>s.ToUpper());  
foreach (string s in w)  
 Console.WriteLine(s);`

Trzy standardowe operatory zapytaniowe: **Where**, **OrderBy**, and **Select**. (jest ich ok 50 !)

- w celu użycia LinQ to SQL należy skorzystać z klasy **DataContext** w celu nawiązania połączenia (podajemy odpowiedni connectionstring) po czym korzystając z klasy **GetTable** pobrać odpowiednią tabelę (tabelę o takiej samej nazwie i polach musimy **mieć stworzoną jako klasę**)

A tutaj przykład LINQ od A do Z w obsłudze bazy danych.

<http://pastebin.com/H2ccuPD8>

## 4.2. Napisać w LINQ przykładowe zapytania wyrażone w SQL: selekcja, złączenie, grupowanie, stosowanie funkcji agregujących.

**selekcja:** pierwsze co robimy to mówimy, na jakiej tabeli pracujemy (*from cust in customers*) i określamy iterator (czyli *cust*) - teraz możemy zrobić selekcję opartą o konkretne pole tegoż iteratora

```
var selected = from cust in customers
               select cust.City;
```

**złączenie (wewnętrzne):** tak jak poprzednio, zaczynamy od zaznaczenia tabeli, na której pracujemy (*from cust in customers*) oraz określenia, którą tabelę dołączamy (*join dist in distributors*), następnie określamy warunek złączenia, w tym przypadku nazwy miast muszą być takie same (*on cust.City equals dist.City*) - zwróćcie uwagę, że przyrównujemy iteratory oraz korzystamy z słowa *equals* zamiast znaku *==*. Na samym końcu musimy powiedzieć, co ma zawierać tabela wynikowa robiąc selekcję nowych obiektów. Poniżej jest wersja minimum złączenia, można ją rozbudować o jeszcze jakiegoś np. *where*'a między linijką z *join* i *select*.

```
var joined =
    from cust in customers
    join dist in distributors on cust.City equals dist.City
    select new { CustomerName = cust.Name, DistributorName = dist.Name };
```

**Alternatywne złączenie wewnętrzne** - "naiwne" - określamy tabele, na których pracujemy, potem stawiamy warunek, do których krotek się odnosimy (do tych, gdzie City jest takie samo) następnie tworzymy nowe obiekty i je zwracamy jako kolekcję.

```
var joined =
    from cust in customers
    from dist in distributors
    where cust.City == dist.City
    select new { CustomerName = cust.Name, DistributorName = dist.Name };
```

**złączenie zewnętrzne:**

<http://www.dotnetlearners.com/linq/linq-to-sql-left-outer-join.aspx>

**grupowanie:** tutaj również wskazujemy tabelę, na której pracujemy (*from cust in customers*) i nazywamy iterator (*cust*) a potem grupujemy całą tabelę względem kolumny City

```
var grouped =  
    from cust in customers  
    group cust by cust.City;
```

**agregowanie:** w LINQ nie ma funkcji agregującej, jest ona tylko w wyrażeniach lambda; ponieważ (wiki) *(agregacja to) sytuacja, w której tworzy się nową klasę, używając klas już istniejących* oznacza to, że wystarczy zrobić odpowiednią selekcję, patrz wyżej przy *alternatywne złączenie wewnętrzne* - "naiwne".

## 5. Modele danych NoSQL i bazy danych NoSQL

### 5.1. Model danych JSON i baza Azure DocumentDB

#### 5.1.1. Model danych JSON:

- ❑ Dokument JSON jest obiektem o następującej składni:
  - ❑ *obiekt* jest nieuporządkowanym zbiorem (być może pustym) par (pól) *nazwa : wartość* ujętym w nawiasy klamrowe { },
  - ❑ *nazwa* jest typu String, a *wartość* jest *skalarem*, *obiektem* lub *wektorem* (array),
  - ❑ *wektor* jest uporządkowanym zbiorem (być może pustym) *wartości* ujętym w nawiasy prostokątne [ ],
  - ❑ *skalar* jest *stringiem*, *liczbą* lub stałą `true`, `false`, `null`.
- ❑ Nazwy pól w obiekcie mogą się powtarzać.
- ❑ Każdy dokument w kolekcji może mieć inny typ (strukturę).

przykład:

```
{ "id": "AndersenFamily",  
  "lastName": "Andersen",  
  "parents":  
  [ { "firstName": "Thomas" },  
    { "firstName": "Mary Kay" }  
  ],  
  "children":  
  [{ "firstName": "Henriette",  
    "gender": "female",  
    "age": 5,  
    "pets":  
    [{"givenName": "Fluffy"}]  
  }  
  ],  
  "address":  
  { "state": "WA",  
    "county": "King",  
    "city": "Seattle"  
  },  
  "creationDate": 1431620472,  
  "isRegistered": true  
}
```



### 5.1.2. Azure DocumentDB:

- Azure DocumentDB jest usługą **chmurową** o SQLowej bazie danych, która zarządza dokumentami o strukturze zgodnej z modelem JSON i wykorzystuje język JavaScript do utrzymywania spójności danych.
- jest przeznaczony do ogromnych **zbiorów danych** rzędy **nawet setek terabajtów** – wówczas różne kolekcje jednej bazy mogą być pamiętane na kilku maszynach
- każde **zapytanie i transakcja** musi być ograniczone do **jednej** kolekcji
- **replikacje** (przechowywanie tych samych danych na kilku serwerach) stosuje się w celu zapewnienia **odporności na awarie** oraz zmniejszenia **czasu odczytu** – powoduje to jednak **problem aktualizowania replik**, który rozwiązuje się poprzez określenie kompromisu pomiędzy wydajnością a spójnością zależnego od konkretnej aplikacji
- wysyłając zapytanie do serwera otrzymamy w **odpowiedzi JSONa**
- **.NET SDK for DocumentsDB** pozwala łatwo zarządzać bazą udostępniając wygodne metody do tworzenia baz i przetwarzania danych

### 5.2. Przykładowe zapytanie do bazy dokumentów JSON

- jako SQL:

```
SELECT *  
FROM Families f  
WHERE f.id = "AndersenFamily"
```

- po REST API (poniżej widzicie zawartość przykładowego komendy POST, adres pod który ono idzie etc - można coś takiego wysłać sobie wygodnie np. javascriptem) - oczywiście można tutaj przesłać również definicje procedur składowych, triggerów etc oraz użyć innych komend HTML aniżeli POST - również GET, PUT, DELETE itd nie uczcie się na pamięć - wystarczy, że zapamiętacie, że jest coś takiego jak query z zapytaniem i parameters z parametrami i z jest wysyłane pod jakiś konkretny adres

```
POST  
https://contosomarketing.documents.azure.com/dbs/XP0mAA==/colls/XP0m  
AJ3H-AA=/docs HTTP/1.1  
...  
x-ms-documentdb-isquery: True  
Content-Type: application/query+json  
{  
  "query": "SELECT * FROM Families f WHERE f.id = @familyId",  
  "parameters": [ {"name": "@familyId", "value": "AndersenFamily"} ]  
}
```

- za pomocą .NET SDK for DocumentDB dostępnego jako paczka Nugetowa - po odpowiedniej konfiguracji pozwala na wysyłanie zapytań w formie LINQ, lambda lub SQL

## Zapytanie LINQ

```
private static void QueryDocuments(string collectionLink)
{
    // LINQ Query by document FamilyName

    var query = from f in
        client.CreateDocumentQuery<Family>(collectionLink)
        where f.FamilyName == "Anderson"
        select f;

    Console.WriteLine("Family Name is - {0}",
        query.AsEnumerable().FirstOrDefault<Family>().FamilyName);
}
```

## LINQ Lambda

```
private static void QueryDocuments(string collectionLink)
{
    // LINQ Lambda with FamilyName attribute

    query = client.CreateDocumentQuery<Family>(collectionLink)
        .Where(f => f.FamilyName == "Wakefield");

    Console.WriteLine("Family Name is - {0}",
        query.AsEnumerable().FirstOrDefault<Family>().FamilyName);
}
```

## Zapytanie SQL

```
private static void QueryDocuments(string collectionLink)
{
    // SQL query for a family record by a Parent's FirstName
    // using intra-document JOINS, http://www.documentdb.com/sql/demo

    query = client.CreateDocumentQuery<Family>(collectionLink,
        new SqlQuerySpec
        {
            QueryText = "SELECT VALUE f FROM Families f
                JOIN p IN f.Parents
                WHERE (p.FirstName = @name)",
            Parameters = new SqlParameterCollection()
            {
                new SqlParameter("@name", "Susan")
            }
        });
    Console.WriteLine("5. Family Name is - {0}",
        query.AsEnumerable().FirstOrDefault<Family>().FamilyName);
}
```

## 5.3. Składnia danych według modelu Key/Value i baza Azure Storage

Każda dane (obiekt) według modelu Key/Value składa się z czterech elementów:

- ❑ **PartitionKey** – string jednoznacznie identyfikujący partycję,
- ❑ **RowKey** – string jednoznacznie identyfikujący krotkę (rekord) w partycji
- ❑ **Timestamp** – znacznik czasowy automatycznie aktualizowany przez system (czas ostatniej aktualizacji)
- ❑ **Value** – wartość danej o dowolnej strukturze – krotki z dowolnymi zagnieźdzeniami i z możliwością powtarzania atrybutów.

np.:

PartitionKey	RowKey	Timestamp	Description
Baseball	BBa1094	2013-10-31T18:42:02.2793386Z	The standard

- tabela Azure może przechowywać **maksymalnie 255 par Key/Value**, wielkość tabeli nie może przekroczyć **1 mb**
- obiekty w tablicy są **grupowane według kolumny PartitionKey**
- para **PartitionKey i RowKey stanowi klucz główny** krotki
- **klucz główny tworzy indeks klastrowy** (dzięki indeksowi klastrowemu, zamiast szukać czegoś w tabeli skanując ją po kolei, przechodzimy przez strukturę b-drzewa stworzonego na podstawie jednej kolumny, dzięki czemu dane szukamy szybciej i z mniejszym zużyciem zasobów)
- **znacznik czasowy jest automatycznie aktualizowany** i używany do zarządzania optymistyczną współbieżnością
- **odpowiedź** jest sformatowana jako **XML lub JSON**
- Azure Table **nie wspiera kluczy drugorzędnych**, nie ma też pojęcia schematu – **każda krotka może mieć inną strukturę**

### Azure Storage: ograniczenia

Całkowity rozmiar konta Azure Storage	500 TB
Liczba tablic	Ograniczone rozmiarem konta
Liczba partycji w tablicy	Ograniczone rozmiarem konta
Liczba jednostek w partycji	Ograniczone rozmiarem konta
Rozmiar pojedynczej krotki	Do 1 MB z maksymalną liczbą 255 własności (łącznie z PartitionKey, RowKey i Timestamp)
Rozmiar PartitionKey	String do 1 KB
Rozmiar RowKey	String do 1 KB
Rozmiar Entity Group Transaction	Transakcja może operować na 100 jednostkach a jej rozmiar musi być mniejsze niż 4 MB. Jedna EGT może tylko raz aktualizować jedną jednostkę.

## 5.4. Sposób reprezentacji danych o podanym schematu ER w modelu JSON i Key/Value, np. dla Student-Egzamin-Przedmiot.

nie wiem, w prezentacji tego nie ma a już mi się nie chce tego ogaraniać

## 5.5. Dane BigData i metody ich przetwarzania (na przykładzie MS Azure)

### 5.5.1. Dane Big Data

Big Data charakteryzuje się stosując tzw. parametry 4V. Parametry te oznaczają, że dane te charakteryzuje duża:

- **Volume (objętość)** – objętość danych; danych – konieczne jest **partycjonowanie danych na kilka serwerów** a obecne upowszechnianie komputerów powoduje **wykładniczy rozrost danych**
- **Variety (różnorodność)** – różnorodność struktur i źródeł (**teksty, obrazy**, audio, video, dane sensorowe, **logi, serwisy**, email, ...), problem z analizą – tradycyjne narzędzia analizy danych zorientowane są na dane strukturalne, dodatkowo dane przed użyciem muszą być poddane **kosztownym procesom czyszczenia** danych
- **Velocity (szybkość)** – szybkość napływania danych i konieczność **szybkiego przetwarzania** (strumieniowość) - często **w czasie rzeczywistym** (np. transakcje giełdowe, natężenie ruchudrogowego)
- **Variability/Veracity (niejednoznaczność, niepewność)** – niejednoznaczność (zmienność) znaczenia, zależna od kontekstu, **poprawność interpretacji przesądza o wartości** danych

### 5.5.2. Metody przetwarzania Big Data na przykładzie MS Azure



## Czym jest Windows Azure HDInsight?

1. **Windows Azure HDInsight** jest usługą firmy Microsoft, która implementuje i udostępnia w chmurze infrastrukturę klastrową zgodną z Apache Hadoop, z przeznaczeniem do zarządzania, analizy i raportowania danych Big Data.
2. Jądro usługi stanowi magazyn danych HDFS (Hadoop Distributed File System) i model programowania MapReduce.
3. HDFS stosuje replikację danych dla zapewnienia tolerancji błędów i wysokiej dostępności.
4. MapReduce służy do równoległego przetwarzania i analizy danych pamiętanych w HDFS lub Azure Blob Storage. MapReduce postrzega wszystkie swoje zadania jak przetwarzanie na parach klucz-wartość.
5. Dane w Azure Blob Storage pozostają dostępne również po usunięciu HDFS.

HDInsight wspiera:

- **Hive** – framework działający w środowisku Hadoop, służący do odpytywania i analizy danych. Udostępnia m.in. SQL-podobny język HiveQL, który może generować zadania MapReduce (transluje zapytania w ciąg zadań MapReduce). Nadaje się głównie do zadań z danymi ustrukturyzowanymi. Dla danych niestrukturalnych lepszy jest Pig. Windows Azure HDInsight dostarcza sterownik ODBC dla zapytań Hive z takich narzędzi jak Excel do Hadoop.
- **Pig** – platforma przetwarzania Big Data na klastrach Hadoop. Udostępnia język Pig Latin do tworzenia zadań MapReduce. Można go rozszerzać o funkcje w Java, Python C# i JavaScript.
- **Sqoop** – transferuje dane między Hadoop relacyjnymi (SQL-owymi) bazami danych lub innymi strukturalnymi magazynami. Można go wykorzystać aby zaimportować zewnętrzne źródło danych do HDFS. Potrafi także eksportować wybrane dane do zewnętrznych relacyjnych baz danych.

I co nieco z tego można zapamiętać:



---

## Zastosowania – uczenie się z danych

---

- Bogaty zestaw narzędzi do uczenia się z danych relacyjnych, gromadzonych np. w SQL Server, dostarcza SQL Server Analysis Services (SSAS) .
- Dla analizy danych nierelacyjnych (NoSQL) takich jak: informacja z sensorów, znaczniki z RFID, pliki logów serwerów, strumienie kliknięć produkowanych przez aplikacje webowe, obrazy z medycznych urządzeń diagnozujących – pomyślane są narzędzia z otoczenia Hadoop/MapReduce.
- Pisanie zadań MapReduce do analizy Big Data jest trudne. Łatwiejsze jest programowanie analizy danych w językach Pig i HiveQL. Programy takie automatycznie generują zadania MapReduce ukrywając jednocześnie ich złożoność przed użytkownikiem.
- HDInsight udostępnia zarówno Pig, jak i Hive. Oferowany jest też zestaw dodatków do Excela, które służą do tworzenia zadań MapReduce. Wyniki można przetwarzać i wizualizować korzystając z PowerPivot i innych narzędzi Excelowych.

## 6. XML, XPath i XQuery

### 6.1. Gramatyka XML i wyrażenia zgodne z tą gramatyką jako dokumenty XML.

**Definicja:** Gramatyką XML-ową nazywamy układ:

$$D = (\text{top}, \text{Lab}, \rho)$$

gdzie:

- Lab – zbiór etykiet (nazw),
- $\text{top} \in \text{Lab}$  – wyróżniona etykieta (szczytowa),
- $\rho$  – zbiór reguł (produkcji) o postaci:

$$l \rightarrow e,$$

gdzie  $l \in \text{Lab}$ , a  $e$  jest wyrażeniem regularnym nad zbiorem  $\text{Lab} - \{\text{top}\}$ :

$$e ::= \varepsilon \mid l \mid e? \mid e^* \mid e^+ \mid e_1 + e_2 \mid e_1 e_2 \mid (e)$$

$\varepsilon$  – ciąg pusty,

$l \in \text{Lab} - \{\text{top}\}$ ,

$e?$  – 0 lub 1 powtórzenie  $e$ ,

$e^*$  – dowolna liczba powtórzeń  $e$  (od 0 do nieskończoności),

$e^+$  – dowolna dodatnia liczba powtórzeń  $e$  (od 1 do nieskończoności),

$e_1 + e_2$  – alternatywa, wyrażenie  $e_1$  lub  $e_2$ ,

$e_1 e_2$  – sekwencja, po wyrażeniu  $e_1$  następuje wyrażenie  $e_2$ ,

$(e)$  – wyrażenie regularne w nawiasie.

**Przykład, opis osób:**

*Produkcje gramatyki XML-owej*

osoby	→ osoba+
osoba	→ nazwisko imię
nazwisko	→ ε
imię	→ ε

*Wyrażenie języka XML-owego (dana XML-owa):*

```
<osoby>
  <osoba>
    <nazwisko>Nowak</nazwisko>
    <imię>Ewa</imię>
  </osoba>
  <osoba>
    <nazwisko>Kubiak</nazwisko>
    <imię>Jan</imię>
  </osoba>
</osoby>
```

## 6.2. Schematy XML: DTD i XML Schema

### 6.2.1. DTD

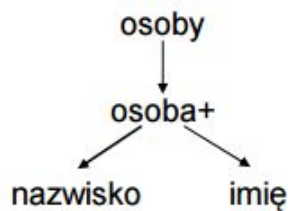
**DTD** (Document Type Definition)

- element szczytowy deklarowany jest jako DOCTYPE,
- produkcje są deklarowane jako ELEMENT,
- wyrażenie puste jest deklarowane jako #PCDATA,
- dodatkowo: atrybuty

**Przykład:**

**Gramatyka:**

osoby → osoba+  
osoba → nazwisko imię  
nazwisko → ε  
imię → ε



**DTD:**

```
<!DOCTYPE osoby[
  <!ELEMENT osoby (osoba+)>
  <!ELEMENT osoba (nazwisko imię)>
  <!ELEMENT nazwisko (#PCDATA)>
  <!ELEMENT imię (#PCDATA)>
]>
```

**DTD, a XML**

<pre> &lt;!DOCTYPE bib [   &lt;ELEMENT bib (książka*)&gt;   &lt;ELEMENT książka (#PCDATA   autor)*&gt;   &lt;!ATTLIST książka     tytuł CDATA #REQUIRED     cena CDATA #REQUIRED&gt;   &lt;ELEMENT autor (#PCDATA   nazwisko   adres)*&gt;   &lt;ELEMENT nazwisko (#PCDATA)&gt;   &lt;ELEMENT adres (#PCDATA)&gt; ]&gt; </pre>	<pre> &lt;bib&gt;   &lt;książka tytuł="XML" cena="100"&gt;     &lt;autor&gt;       &lt;nazwisko&gt;Jan Nowak&lt;/nazwisko&gt;       Prezes XCon     &lt;/autor&gt;     &lt;autor&gt;Ewa Maj&lt;/autor&gt;   &lt;/książka&gt;   &lt;książka tytuł="SQL" cena="200"&gt;     Polecam     &lt;autor&gt;Ewa Maj&lt;/autor&gt;     &lt;autor&gt;       &lt;nazwisko&gt;Maria Kubiak&lt;/nazwisko&gt;       &lt;adres&gt;Warszawa&lt;/adres&gt;     &lt;/autor&gt;   &lt;/książka&gt; &lt;/bib&gt; </pre>
--	--

**DTD określa strukturę dokumentu – brak kontroli nad danymi**

## 6.2.2. XML Schema

**XML Schema** (XML Schema definition language – XSD) umożliwia definiowanie struktury i typów danych w dokumencie XML

- Typy złożone i proste (dla elementów, atrybutów i wartości)
- Wyrażenia regularne w definicji typów

<pre> &lt;xs:element nazwa="Towar" minOccurs="0" maxOccurs="unbounded"&gt;   &lt;xs:complexType&gt;     &lt;xs:sequence&gt;       &lt;xs:element nazwa="NazwaTow" type="xs:string"/&gt;       &lt;xs:element nazwa="LiczbaSztuk"&gt;         &lt;xs:simpleType&gt;           &lt;xs:restriction base="xs:positiveInteger"&gt;             &lt;xs:maxExclusive value="100"/&gt;           &lt;/xs:restriction&gt;         &lt;/xs:simpleType&gt;       &lt;/xs:element&gt;       &lt;xs:element nazwa="CenaPLN" type="xs:decimal"/&gt;       ...     &lt;/xs:sequence&gt;     &lt;xs:attribute nazwa="IdTow" type="IDTOW" use="required"/&gt;   &lt;/xs:complexType&gt; &lt;/xs:element&gt; </pre>	<pre> &lt;Towar IdTow="872-AA"&gt;   &lt;NazwaTow&gt;LaserJet 5MP&lt;/NazwaTow&gt;   &lt;LiczbaSztuk&gt;1&lt;/LiczbaSztuk&gt;   &lt;CenaPLN&gt;1847.84&lt;/CenaPLN&gt;   &lt;DataZak&gt;2004.06.20&lt;/DataZak&gt; &lt;/Towar&gt; </pre>
<pre> &lt;xs:simpleType nazwa="IDTOW"&gt;   &lt;xs:restriction base="xs:string"&gt;     &lt;xs:pattern value="d{3}-[A-Z]{2}" /&gt;   &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt; </pre>	

## 6.3. Model DOM – przykład, rodzaje wierzchołków.

Wg modelu **DOM – Document Object Model (W3C)**, dokument XML przedstawiany jest jako drzewo o 7 typach wierzchołków. **Kolejność wierzchołków w drzewie jest istotna!** Są to wierzchołki typu:

- **root** (korzeń) lub **document** – korzeń drzewa,
- **element** (element),
- **atrybut** (attribute),
- **tekst** (text),
- **instrukcja przetwarzania** (processing instruction),
- **przestrzeń nazw** (spacenazwisko),
- **komentarz** (comment).

Ograniczymy się do pierwszych **czterech**.

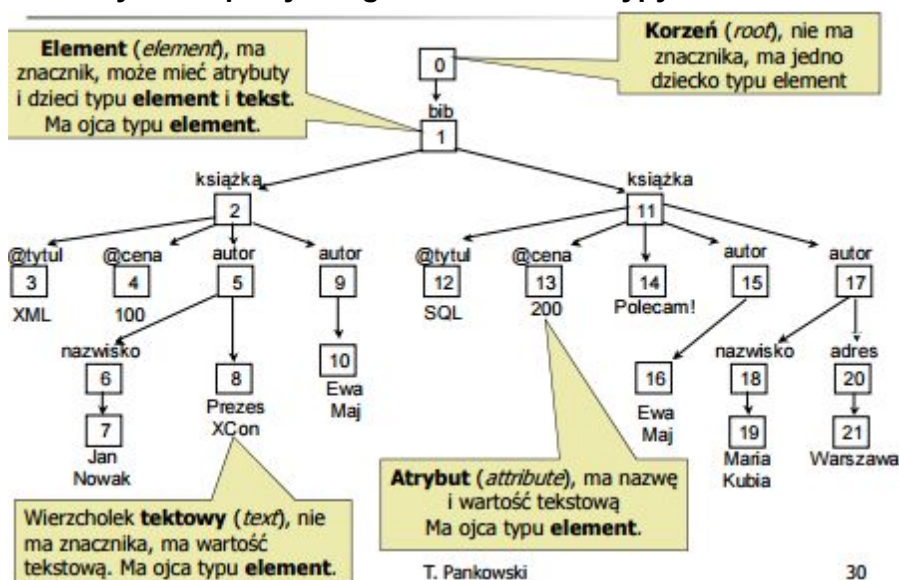
Każdy wierzchołek ma **jednoznaczny** identyfikator.

Przykład:

- **Dokument XML**

<pre>&lt;bib&gt;   &lt;książka tytuł="XML" cena="100"&gt;     &lt;autor&gt;       &lt;nazwisko&gt;Jan Nowak&lt;/nazwisko&gt;       Prezes XCon     &lt;/autor&gt;     &lt;autor&gt;Ewa Maj&lt;/autor&gt;   &lt;/książka&gt;</pre>	<pre>&lt;książka tytuł="SQL" cena="200"&gt;   Polecam   &lt;autor&gt;Ewa Maj&lt;/autor&gt;   &lt;autor&gt;     &lt;nazwisko&gt;Maria Kubiak&lt;/nazwisko&gt;     &lt;adres&gt;Warszawa&lt;/adres&gt;   &lt;/autor&gt; &lt;/książka&gt; &lt;/bib&gt;</pre>
---	---

- **Drzewo danych dla powyższego dokumentu i 4 typy wierzchołków**



## 6.4. Zapytania XPath: składnia, przykłady.

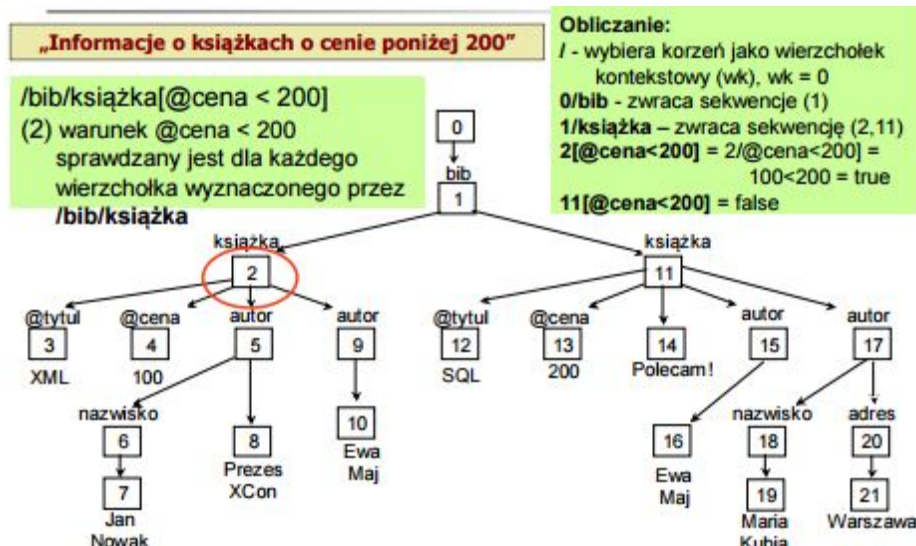
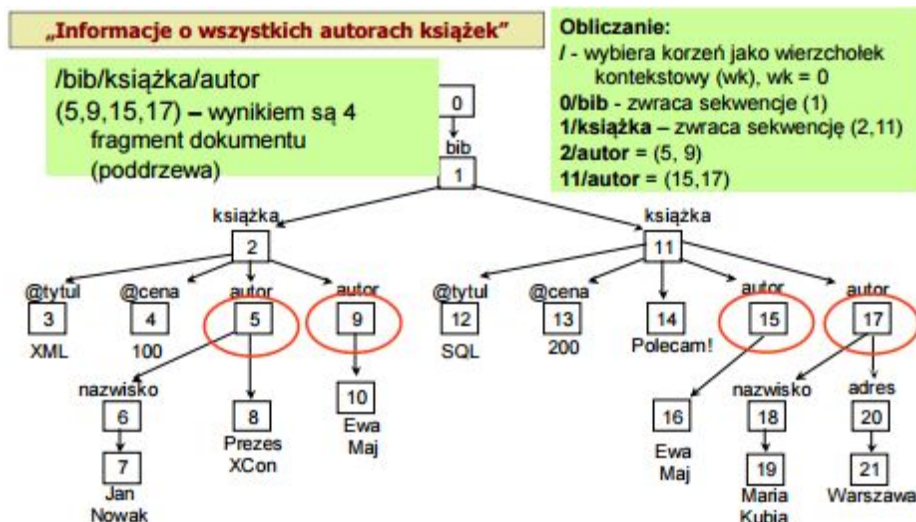
Język XPath

- pozwala wybierać fragment dokumentu XML-owego



- wyrażenie XPath definiuje ścieżkę w drzewie danych podając etykiety (znaczniki elementów lub nazwy atrybutów) wierzchołków, kierunki przechodzenia drzewa i warunki, jakie mają spełniać wybierane wierzchołki,
- każdy wybrany wierzchołek określa poddrzewo, którego jest korzeniem,
- bieżący wierzchołek nazywamy wierzchołkiem kontekstowym,
- jeśli wyrażenie E wyznaczyło sekwencję S i  $x \in S$ , to parę (S,x) nazywamy kontekstem wykonywania kolejnego wyrażenia.

## Zapytania



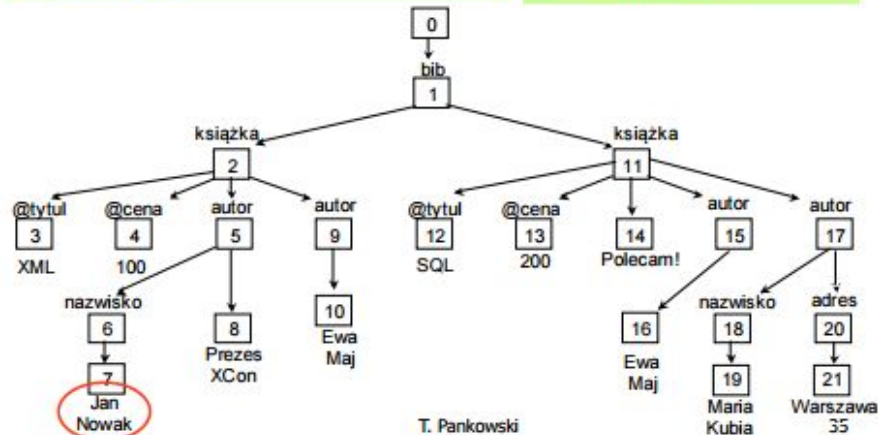


**„Nazwiska autorów książek o cenie poniżej 200”**

`/bib/książka[@cena < 200]/autor/nazwisko/text()`  
„Jan Nowak”

**Obliczanie:**

`/bib/książka[@cena < 200] = (2)`  
`2/autor/nazwisko = 6`  
`6/text() = „Jan Nowak”`



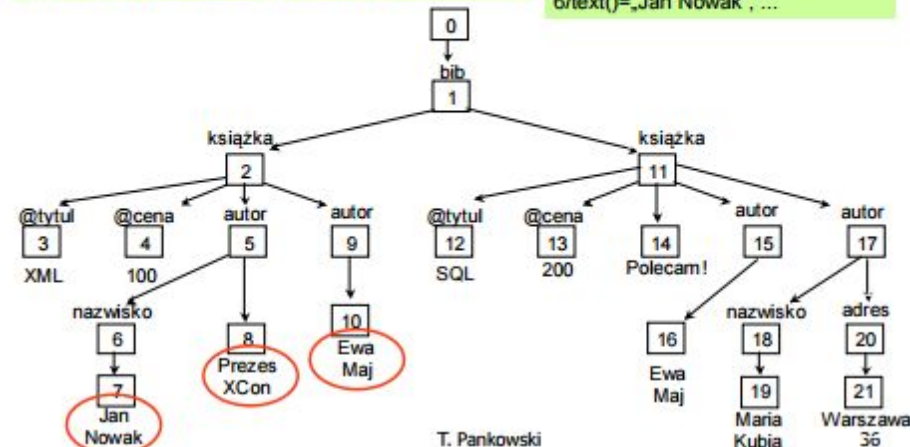
T. Pankowski

**„Nazwiska autorów książek o cenie poniżej 200”**

`/bib/książka[@cena < 200]/autor/text()`  
„Jan Nowak”, „prezes XCon”, „Ewa Maj”

**Obliczanie:**

`/bib/książka[@cena < 200] = (2)`  
`2/autor = (5,9,6,8,10,7)`  
`5/text() = „prezes XCon”, „Ewa Maj”`  
`6/text() = „Jan Nowak”, ...`



T. Pankowski

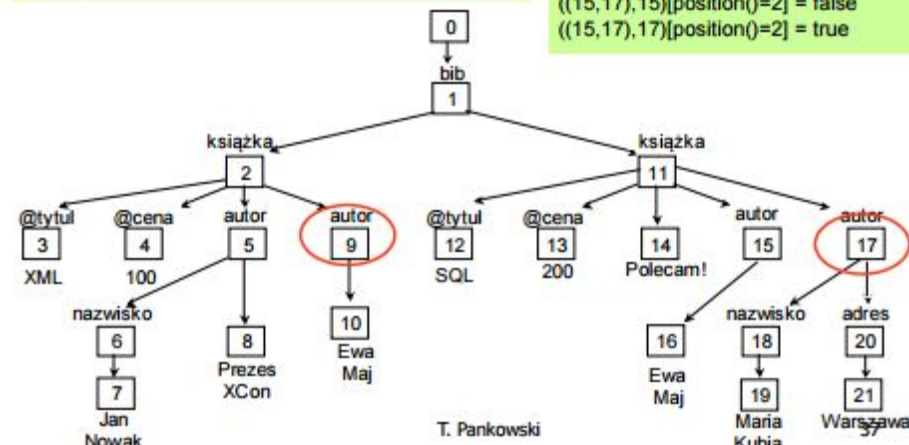
## XPath – zapytanie 5

**„Nazwiska drugich autorów książek”**

`/bib/książka/autor[position()=2]`  
(9,17)

**Obliczanie, cały kontekst (S,x):**

`(S,x)position()` – pozycja x w S  
`/bib = (1)`  
`((1,1)/książka = (2,11))`  
`((2,11,2)/autor = (5,9))`  
`((2,11,11)/autor = (15,17))`  
`((5,9,5)[position()=2] = false)`  
`((5,9,9)[position()=2] = true)`  
`((15,17,15)[position()=2] = false)`  
`((15,17,17)[position()=2] = true)`



T. Pankowski

## Wyrażenia XPath

```
Path ::= /Step1 / Step2 / ... / Stepn  
Step ::= Axis :: Node-test [Predicate]*  
Axis ::= child | attribute | parent | self | descendant-or-self | ...  
Predicate ::= Path | position() <math>\Theta</math> n | last() <math>\Theta</math> n | position() <math>\Theta</math> last() | ...
```

Przykład:

```
/bib//*[position() = last()]
```

postać rozwinięta:

```
/child::bib/descendant-or-self::node()/child::*[position() = last()]
```

## 6.5. XQuery – ogólny schemat zapytań, przykłady zapytań


**XQuery** - język zapytań, który:

- wybiera elementy/atributy z dokumentu wejściowego,
- łączy dane z wielu dokumentów wejściowych,
- umożliwia modyfikację danych,
- wylicza nowe dane,
- umożliwia budowanie dokumentu wynikowego
- dodaje nowe elementy/atributy do wyniku,
- sortuje wynikowy dokument

**Przykłady dla dokumentu:**

```
<bib>  
  <ksiazka cena="55.00">  
    <tytul>XML</tytul>  
    <autor>Nowak</autor>  
    <autor>Kubiak</autor>  
    <wydawnictwo>PWN</wydawnictwo>  
  </ksiazka>  
  <ksiazka cena="70.00">  
    <tytul>SQL</tytul>  
    <autor>Nowak</autor>  
    <autor>Kubiak</autor>  
    <autor>Lipski</autor>  
    <wydawnictwo>PWN</wydawnictwo>  
  </ksiazka>  
</bib>
```

**Polecenie: Podaj autorów książek w cenie mniejszej niż 60 zł**

Wyrażenie ścieżkowe	Wyrażenie FLWOR
<p>Wyrażenie ścieżkowe: doc("bib.xml")/bib/ksiazka[@cena&lt;60]/autor</p> <p><b>SQL Server:</b> select OpisXML.query('/bib/ksiazka[@cena&lt;60]/autor') from Bib;</p> <p>Wynik: &lt;autor&gt;Nowak&lt;/autor&gt; &lt;autor&gt;Kubiak&lt;/autor&gt;</p> <p>Wynik nie jest dokumentem XML. Jest sekwencją dwóch fragmentów. </p>	<p>Wyrażenie FLWOR: for \$k in doc("bib.xml")/bib/ksiazka where \$k/@cena&lt;60 return \$k/autor</p> <p><b>SQL Server:</b> select OpisXML.query('for \$k in /bib/ksiazka where \$k/@cena&lt;60 return \$k/autor') from Bib</p> <p>Wynik: &lt;autor&gt;Nowak&lt;/autor&gt; &lt;autor&gt;Kubiak&lt;/autor&gt;</p>
Konstruowanie dokumentu	
<p>SELECT OpisXML.query('&lt;wynik&gt;{ for \$k in /bib/ksiazka where \$k/@cena&lt;60 return \$k/autor} &lt;/wynik&gt;') FROM Bib</p> <p>Wynik: &lt;wynik&gt; &lt;autor&gt;Nowak&lt;/autor&gt; &lt;autor&gt;Kubiak&lt;/autor&gt; &lt;/wynik&gt;</p>	<p>SELECT OpisXML.query('&lt;wynik&gt;{ for \$a in /bib/ksiazka[@cena&lt;60]/autor return &lt;nazwisko&gt; { \$a/text() } &lt;/nazwisko&gt;} &lt;/wynik&gt;') FROM Bib</p> <p>Wynik: &lt;wynik&gt; &lt;nazwisko&gt;Nowak&lt;/nazwisko&gt; &lt; nazwisko&gt;Kubiak&lt;/nazwisko&gt; &lt;/wynik&gt;</p>

**I kolejny przykład:**

Dla każdego autora PWN podaj wykaz jego książek:

```
SELECT OpisXML.query('
<wynik>{
  for $a in distinct-values(/bib/ksiazka[wydawnictwo/text()='PWN']/autor/text())
  return
    <wykaz>
      { $a }
      { for $t in /bib/ksiazka[autor/text()=$a]/tytul/text()
        return <tytul>{$t}</tytul> }
    </wykaz>}
</wynik>')
FROM Bib
```

**distinct-values** = funkcja eliminująca duplikaty

```
<wynik>
  <wykaz>Kubiak<tytul>XML</tytul><tytul>SQL</tytul></wykaz>
  <wykaz>Nowak<tytul>XML</tytul><tytul>SQL</tytul></wykaz>
  <wykaz>Lipski<tytul>SQL</tytul></wykaz>
</wynik>
```