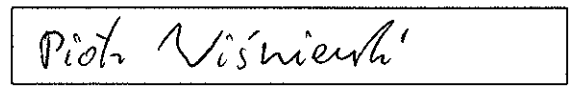
Optymalizacja zapytań w architekturach z odwzorowaniami obiektowo-relacyjnymi

Autoreferat

Piotr Wiśniewski

Uniwersytet Mikołaja Kopernika
Wydział Matematyki i Informatyki
Chopina 12/18, 87-100 Toruń



Grudzień 2013

Spis treści

1 Wprowadzenie

Odwzorowania obiektowo-relacyjne (ang. Object Relational Mappings) są warstwą pośrednią istotnie ułatwiającą zarządzanie obiektami trwałymi. Korzystając z nich programista obiektowego języka programowania może skupić się na logice przetwarzania obiektów, pracując z kolekcjami i mechanizmami utrwalania dostarczonymi przez system ORM. Mechanizmy odwzorowań obiektowo-relacyjnych pomimo osiągnięcia pewnego poziomu dojrzałości wciąż są przedmiotem badań naukowych [1, 2, 3].

Twórcy najpopularniejszych systemów odwzorowań obiektowo-relacyjnych ograniczyli się do podstawowych operacji bazodanowych, pomijając wiele rozszerzeń i technik optymalizacyjnych, które mogłyby zostać zintegrowane z tymi odwzorowaniami. Tymczasem systemy zarządzania bazami danych dynamicznie się rozwijają i są wzbogacane o kolejne mechanizmy wykraczające poza podstawowy zestaw operacji. Rozszerzenia te nawet jeśli zostają usystematyzowane standardem, to i tak nie w sposób ścisły. Rosnąca luka pomiędzy aspektami objętymi odwzorowaniami obiektowo-relacyjnymi, a możliwościami współczesnych systemów zarządzania obiektowo-relacyjnymi, stała się inspiracją do prac badawczych zaprezentowanych w niniejszym autoreferacie.

Główna idea stojąca za prezentowanymi badaniami, to wykorzystanie warstwy odwzorowań obiektowo-relacyjnych jako miejsca zanurzenia algorytmów optymalizacji komunikacji z bazą danych. Optymalizacja ta jest dokonywana w kilku aspektach. W pierwszej kolejności postanowiłem zintegrować systemy ORM z wybranymi rozszerzeniami systemów zarządzania bazami danych. Wyniki te zostały zawarte w pracach [R1–R8,F]. Następnym aspektem jest udostępnienie znanych technik optymalizacyjnych z poziomu systemów ORM. Rezultaty zostały zaprezentowane w pracach [P1–P3]. Dla lepszego zrozumienia odwzorowań obiektowo-relacyjnych podjęte zostały przeze mnie również badania teoretyczne w zakresie modeli danych i języków zapytań do tych modeli, których wyniki przedstawione są w pracach [T1–T4].

W pracy [T1] pokazujemy, że modele danych wykorzystywane w nowoczesnych językach programowania są znacznie bliższe modelowi relacyjnemu, niż to się powszechnie uważa. Z drugiej strony relacyjne bazy danych niezależnie od dostawcy oparte są na tym samym modelu relacyjnym. Mnogość dialektów języka SQL nie zmienia faktu, że zapytania konstruowane są za pomocą tych samych przekształceń logicznych danych. W efekcie odwzorowania obiektowo-relacyjne nie muszą istotnie spowalniać systemów z nich korzystających. Pewne nieporozumienia co do spadku wydajności aplikacji poprzez

zastosowanie odwzorowań obiektowo-relacyjnych mogą brać się z subtelności wokół kolekcji. Patrycja Węgrzynowicz w [4] prezentuje antywzorce powodujące drastyczny spadek wydajności w komunikacji z bazą aplikacji opartej na Hibernate oraz jak zastąpić je efektywnymi konstrukcjami.

Innym przykładem słabej wydajności aplikacji opartych o systemy ORM jest istotna nadmiarowość komunikacji z bazą, wynikająca z braku wsparcia dla rozszerzeń możliwości baz danych. Dobrym przykładem takich rozszerzeń są zapytania rekurencyjne, które pojawiły się w formie rekurencyjnych wyrażeń tabelarycznych w standardzie SQL:99. Od tamtego czasu systemy Microsoft SQL Server, IBM DB2 PostgreSQL i kilka innych zaimplementowały je, każdy na swój własny sposób. Tymczasem ORACLE, który jako pierwszy wprowadził zapytania rekurencyjne w oparciu o klauzulę `CONNECT BY`, dopiero w roku 2009 wprowadził rekurencyjne wyrażenia tabelaryczne.

Użytkownicy odwzorowań obiektowo-relacyjnych żyją zazwyczaj w odcięciu od tych udogodnień, przekonani, że wystarczy dane pobrać z bazy, a całą ich obsługę można dokonać po stronie języka programowania. Podejście takie, o ile rzeczywiście doprowadza do zamierzonych wyników, o tyle wymaga czasem nieadekwatnego narzutu czasowego. W obliczu takiej obserwacji podjąłem się badań w kierunku integracji zapytań rekurencyjnych z odwzorowaniami obiektowo-relacyjnymi. Pierwsze eksperymentalne próby zostały przedstawione w pracach [R2] i [R3]. W pracy [R4] zawarte zostały wyniki eksperymentu w połączeniu z różnymi systemami baz danych. Jakość uzyskanego w ten sposób przyspieszenia przerosła moje oczekiwania. Na próbce około 4500 rekordów udało się uzyskać przyspieszenie kilkudziesięciokrotne.

Wyniki te inspirowały mnie do dalszych poszukiwań scenariuszy optymalizacji obsługi bazy. W pracach [R5–R8] rozwiązaliśmy kolejne nasuwające się problemy wokół danych rekurencyjnych, w tym zadbaliśmy o użytkowników baz nie wspierających takich zapytań, w szczególności użytkowników MySQL. Równolegle do tych badań zajęliśmy się problemem wykorzystania danych nadmiarowych w celu optymalizacji zapytań analitycznych. Wyniki te zostały zaprezentowane w pracach [P1–P3].

Spośród rozszerzeń niedocenianych przez programistów bardzo użyteczne wydały mi się również indeksy funkcyjne. Niestety okazało się, że wiele baz, które wspierają to rozszerzenie, nie posiada mechanizmów detekcji możliwości ich wykorzystania. Mając już spore doświadczenie z pracy [R5] na temat konstrukcji języka HQL, opracowaliśmy mechanizm wykrywania możliwości użycia indeksów funkcyjnych, który został zaprezentowany w [F].

2 Dorobek naukowy

Spośród dorobku naukowego jako jednolity nurt prac badawczych wybrałem publikacje, które dotyczą tematyki optymalizacji komunikacji z bazą danych w aplikacjach opartych o architekturę ORM:

R1 Marta Burzańska, Krzysztof Stencel, Piotr Wiśniewski, *Pushing Predicates into Recursive SQL Common Table Expressions*, Lecture Notes in Computer Science Volume 5739, 2009, pp 194-205

R2 Marta Burzańska, Krzysztof Stencel, Patrycja Suchomska, Aneta Szumowska, Piotr Wiśniewski *Recursive Queries Using Object Relational Mapping*, Lecture Notes in Computer Science Volume 6485, 2010, pp 42-50

R3 Piotr Wiśniewski, Aneta Szumowska, Marta Burzańska, Aleksandra Boniewicz, *Hibernate the Recursive Queries - Defining the Recursive Queries using Hibernate ORM*, ADBIS 2011, CEUR Workshop Proceedings, vol 789, pp 190-199

R4 Aneta Szumowska, Marta Burzańska, Piotr Wiśniewski, Krzysztof Stencel, *Efficient Implementation of Recursive Queries in Major Object Relational Mapping Systems*, Lecture Notes in Computer Science Volume 7105, 2011, pp 78-89

R5 Aneta Szumowska, Marta Burzańska, Piotr Wiśniewski, Krzysztof Stencel *Extending HQL with Plain Recursive Facilities*, Advances in Intelligent Systems and Computing Volume 186, 2013, pp 265-272

R6 Boniewicz, A., Stencel, K., Wiśniewski, P. *Unrolling SQL:1999 recursive queries*, Communications in Computer and Information Science,Vol 352, Springer Berlin Heidelberg (2012) pp 345–354

R7 Aleksandra Boniewicz, Piotr Wiśniewski, Krzysztof Stencel *On Materializing Paths for Faster Recursive Querying*, Advances in Intelligent Systems and Computing Volume 241, 2014, pp 105-112

R8 Aleksandra Boniewicz, Piotr Wiśniewski, Krzystof Stencel, *On Redundant Data for Faster Recursive Querying Via ORM Systems*, Annals of Computer Science and Information Systems, Volume 1, IEEE No: CFP1385N-ART, pp 1451 – 1458

T1 Piotr Wiśniewski, Marta Burzańska, Krzysztof Stencel, *The Impedance Mismatch in Light of the Unified State Model*, Fundamenta Informaticae, Volume 120, Number 3-4 / 2012, pp 359-374

T2 Piotr Wiśniewski, Krzysztof Stencel, *Universal query language*, Proceedings of the 21th International Workshop on Concurrency, Specification and Programming, Berlin, Germany, September 26-28, 2012, pp 394-403

T3 Piotr Wiśniewski, *A Multi-set Relational Algebra in View of Universal Query Language*, Lecture Notes in Computer Science Volume 7709, 2012, pp 233-240

T4 Piotr Wiśniewski, Krzysztof Stencel, *Universal Query Language for Unified State Model*, accepted at Fundamenta Informaticae vol of memorial of Manfred Kudlek, vol. 129(102), 2014, pp 177-192

P1 Michał Gawarkiewicz, Piotr Wiśniewski *Partial Aggregation Using Hibernate*, Lecture Notes in Computer Science Volume 7105, 2011, pp 90-99

P2 Michał Gawarkiewicz, Piotr Wiśniewski, Krzysztof Stencel, *Enhanced segment trees in object-relational mapping*, Proceedings of the 6th Balkan Conference in Informatics (BCI '13). ACM, New York, NY, USA, pp 122-128

P3 Piotr Wiśniewski, Krzysztof Stencel, *Query Rewriting Based on Meta-Granular Aggregation*, CS&P 2013

F Aleksandra Boniewicz, Michał Gawarkiewicz, Piotr Wiśniewski, *Automatic Selection of Functional Indexes for Object Relational Mapping System*, International Journal of Software Engineering and Its Applications Vol. 7, No. 4, July 2013 pp 189 - 196

W powyższych pracach mój wkład autorski przekracza 50%. Odpowiednie oświadczenia są zawarte w załączniku.

Nowość tematyki w świetle doktoratu

W listopadzie 2001 roku uzyskałem stopień doktora nauk matematycznych na podstawie rozprawy doktorskiej na temat *Twierdzenie Laskera - Noether*

dla przemiennych i noetherowskich algebr modułowych nad punktową algebrą Hopfa. Badania dotyczyły zagadnień algebraicznych w dziedzinie nauk matematycznych. Prace badawcze przedstawione powyżej jako dorobek naukowy dotyczą zagadnień informatycznych w dziedzinie nauk technicznych.

3 Rekurencja w relacyjnych bazach danych

Pierwszym aspektem, którym zająłem się w ramach optymalizacji komunikacji z bazą danych w odwzorowaniach obiektowo-relacyjnych, była obsługa danych hierarchicznych i grafowych. Relacyjne bazy danych pozwalają w prosty sposób składować takie dane. Realizowane jest to poprzez klucz obcy łączący tabelę samą ze sobą, bądź też oddzielną tabelę z krawędziami grafu. Niestety możliwość ta nie od razu została odzwierciedlona w języku SQL. Pierwszym systemem, w którym zapytania rekurencyjne zostały zaimplementowany był ORACLE. W roku 1985 zaproponowano w nim klauzulę `CONECT BY`. Przykład jej wykorzystania pokazano na listingu 1.

Listing 1: Przykład użycia `CONNECT BY`

```
SELECT sname, fname FROM Emp
  WHERE level > 1
  START WITH sname = 'Smith'
  CONNECT BY bossId = PRIOR empId
```

Wiele lat później inne systemy zarządzania bazami danych wprowadziły bardziej elastyczne podejście oparte na wyrażeniach tabelarycznych, zwane *rekurencyjnymi wyrażeniami tabelarycznymi* (ang. *Recursive Common Table Expression*). Rozwiązanie to weszło do standardu SQL:99. Niestety nawet wśród liczących się systemów zarządzania bazami danych nie wszystkie posiadają implementację możliwości odpytywania danych rekurencyjnych. Ciekawy przegląd możliwości liczących się systemów czytelnik znajdzie w [5]. Badania nad efektywnością zapytań rekurencyjnych od długiego czasu cieszą się zainteresowaniem nauki [6, 7, 8, 9, 10, 11]. W naszym zespole badawczym odnieśliśmy wrażenie niedocenienia przez użytkowników tego typu zapytań i postanowiliśmy przyjrzeć się im bliżej, zarówno od strony efektywności użycia, jak i możliwości ich wykorzystania w odwzorowaniach obiektowo-relacyjnych.

Rozważmy jako przykład prostą tabelę reprezentującą hierarchię pracowników firmy przedstawioną w tabeli 1. W tabeli tej kolumna `bossId` reprezentuje samozłączenie tejże tabeli.

Jeśli dane te dotyczą pracowników firmy typu *network marketing*, ciekawym pytaniem jest, kto pracuje w drzewie rozpiętym przez np. *Johna Smitha*. Listing 2 zawiera odpowiednie zapytanie zgodne ze standardem SQL:99.

Tablica 1: Tabela Emp jako przykład danych hierarchicznych

empId	fname	sname	bossId
1	John	Smith	
2	Bruce	Willis	1
3	Marilyn	Monroe	
4	Angelina	Jolie	3
5	Brad	Pitt	4
6	Hugh	Grant	4
7	Colin	Firth	3
8	Keira	Knightley	6
9	Sean	Connery	1
10	Pierce	Brosnan	3

Listing 2: Znajdź podwładnych Johna Smitha

```
WITH RECURSIVE rcte (
    SELECT sname, fname, empId
        FROM Emp WHERE sname = 'Smith' AND fname = 'John'
  UNION
    SELECT e.sname, e.fname, e.empId
        FROM Emp e JOIN rcte r ON (e.bossId = r.empId))
SELECT sname, fname FROM rcte;
```

W zapytaniu tym możemy wyróżnić trzy konstrukcje typu SELECT. Pierwsza z nich nazywana *ziarnem* znajduje początki drzew, które zostaną zwrócone przez zapytanie. Druga z nich zwana częścią *rekurencyjną* wykorzystuje jako jedno ze źródeł danych definiowane wyrażenie. Ta część jest kluczowa. Rekordy zwrócone przez ten SELECT ponownie wracają do niego jako elementy źródła danych. Teoretycznie dzieje się tak dopóki nowe rekordy są dostarczane, co w szczególności może oznaczać zapętlenie. Trzeci SELECT, będący poza wyrażeniem tabelarycznym zwany częścią *konsumpcyjną* używa wyrażenia tabelarycznego jako źródła danych. Zapytanie 2 będzie wiodącym przykładem w niniejszym rozdziale.

3.1 Relokacja predykatów

Pierwszym wynikiem uzyskanym dla optymalizacji zapytań rekurencyjnych jest prezentowana w [R1] metodyka przepisywania zapytań rekurencyjnych

w celu przeniesienia warunku do zapytania.

Zaproponowany w pracy algorytm wyszukuje kolumny, których wartości nie ulegają zmianie w kroku rekurencyjnym.

Listing 3: Wyznacz podwładnych Smitha

```
WITH RECURSIVE rcte (
   SELECT sname, fname, empId, sname as master_name
      FROM Emp
 UNION
   SELECT e.sname, e.fname, e.empId,
          r.master_name as master_name
      FROM Emp e JOIN rcte r ON (e.bossId = r.empId))
SELECT sname, fname FROM rcte
   WHERE master_name = 'Smith'
```

W przypadku zapytania 3 kolumną taką jest `master_name`. Część konsumpcyjna zapytania filtruje wyniki poprzez tę kolumnę. W takiej sytuacji nasz algorytm przenosi sprawdzanie warunku do wnętrza wyrażenia rekurencyjnego. Powyższe zapytanie w wyniku optymalizacji otrzyma postać jak na listingu 4.

Listing 4: Zoptymalizowane zapytanie wyznaczające podwładnych Smitha

```
WITH RECURSIVE rcte (
   SELECT sname, fname, empId, sname as master_name
      FROM Emp
      WHERE sname = 'Smith'
 UNION
   SELECT e.sname, e.fname, e.empId,
          r.master_name as master_name
      FROM Emp e JOIN rcte r ON (e.bossId = r.empId))
SELECT sname, fname FROM rcte
```

Praca zawiera też wyniki testów wydajnościowych, z których wynika, że powyższa optymalizacja skraca czas realizacji zapytań o połowę.

3.2 Integracja zapytań rekurencyjnych z systemami odwzorowań obiektowo-relacyjnych

Odwzorowania obiektowo-relacyjne, jak to było wspomniane we wstępie są narzędziem istotnie uproszczającym architekturę systemu. Pozwalają one

modelować aplikacje na poziomie obiektowym. Odwzorowanie tego modelu na model relacyjny odbywa się automatycznie, jednakże z możliwością ingerencji ze strony projektanta. Niestety projektanci systemów ORM skupili się na wsparciu tylko tych aspektów, które są realizowane przez znakomitą większość baz danych. W efekcie odpytywanie danych reprezentujących graf wymaga wysłania serii zapytań. Na listingu 5 przedstawiono kod wyznaczający pracowników znajdujących się w drzewie rozpiętym przez *Smitha* w języku Python z wykorzystaniem biblioteki SQLObject.

Listing 5: Przeszukiwanie listy w Pythonie z wykorzystaniem SQLObject

```
l = Emp.select(Emp.q.sname=="Smith")
for e in l:
    l += Emp.select(Emp.q.bossId == e)
```

Konstrukcja ta, będąc elegancką, niestety jest bardzo nieefektywna. W jej wyniku do bazy jest wysyłane dokładnie tyle zapytań, ile na koniec elementów będzie znajdować się w liście `l`.

Dostrzegając potencjał w zapytaniach rekurencyjnych postanowiłem sprawdzić możliwości przybliżenia ich użytkownikom systemów ORM. W pracy [R2] zaproponowaliśmy czytelny interfejs dla systemu SQLObject, pozwalający wyrazić zapytanie rekurencyjne jako prostą klasę. Klasy te zachowują stylistykę klas trwałych SQLObject oraz analogiczną metodę wysyłania zapytań do bazy. W efekcie generowane jest automatycznie zapytanie rekurencyjne. Spodziewaliśmy dużego przyspieszenia w porównaniu do klasycznego wędrowania po danych, jednakże przyspieszenie rzędu pięćdziesiąt razy na próbce rekordów 4500 w tabeli zawierającej graf, przeszło nasze oczekiwania.

Dobre przyjęcie tych wyników zainspirowało mnie do rozszerzenia tych wyników na systemy powszechnie używane tj. Hibernate dla języka Java [R3, R4] oraz system django [R5]. W pracy [R5] opublikowaliśmy również wyniki prezentujące jakość przyspieszenia w kontekście współpracy z różnymi bazami danych dla kilku przykładów zestawów danych. Testy były przeprowadzane dla baz ORACLE, IBM DB2 oraz PostgreSQL. Niezależnie od wyboru bazy uzyskiwane przyspieszenia były podobnej klasy. W tabeli 2 zaprezentowano wyniki kontekście biblioteki Hibernate i bazy PostgreSQL. W kolumnie *Hibernate loop* zawarte są czasy realizacji kodu analogicznego do 5 w wersji dla Javy w oparciu o Hibernate, natomiast w kolumnie *Hibernate RCTE* zaprezentowane są czasy uzyskane przez rozszerzenie zaproponowane w pracach [R3, R4, R5]

Tablica 2: Porównanie czasu zapytania o hierarchię

	Hibernate loop	Hibernate RCTE	RATIO
900 rec.	3033 ms	143 ms	**4.71 %**
1800 rec.	8669 ms	266 ms	**2.61 %**
2700 rec.	17550 ms	271 ms	**1.54 %**
3500 rec.	28391 ms	405 ms	**1.43 %**
4500 rec.	41500 ms	414 ms	**1.00 %**

System Hibernate wprowadza własny język zapytań HQL. Wielu użytkowników używa tego języka w celu zaawansowanego przeszukiwania danych. Zapytania te są tłumaczone przez Hibernate do języka SQL w zgodzie z dialektem aktualnie podpiętej bazy danych. Próba integracji wyrażeń tabelarycznych wydała mi się zbyt głęboką ingerencją, postanowiłem więc ograniczyć się do pierwotnej propozycji rekurencji w wydaniu ORACLE opartej o klauzulę `CONNECT BY`. Wyniki zostały zaprezentowane w pacy [R5]

3.3 Wsparcie dla systemów nieimplementujących zapytań rekurencyjnych

Jednym z najpopularniejszych systemów zarządzania bazami danych jest MySQL. Niestety jego użytkownicy pozbawieni są możliwości używania zapytań rekurencyjnych, co nie zmienia faktu składowania w tych bazach różnorakich struktur grafowych. Poza naiwnym spacerowaniem po grafie jak w przykładzie 5 społeczność MySQL wypracowała techniki rozwijania zapytań oraz wspierania ich danymi nadmiarowymi. Mając interfejsy do definiowania zapytań rekurencyjnych naturalnym było opracowanie algorytmów generujących właściwe konstrukcje dla baz niewspierających zapytań rekurencyjnych.

W pracy [R6] przedstawiono metody generowania rozwinięć zapytań rekurencyjnych do dwóch postaci. Pierwsza *rozwinięcie wszerz* (ang. *horizontal unrolling*) buduje zapytanie poprzez wielokrotne samozłączenie tabeli. Na listingu 6 przedstawiono zapytanie, jakie zostanie wysłane do bazy zamiast zapytania rekurencyjnego z listingu 2.

Listing 6: Rozwinięcie rekurencji wszerz do trzeciego poziomu

```
SELECT *
FROM Emp l0
LEFT JOIN Emp l1 on (l0.empId = l1.bossId)
```

```
LEFT JOIN Emp l2 on (l1.empId = l2.bossId)
LEFT JOIN Emp l3 on (l2.empId = l3.bossId)
WHERE l1.sname = 'Smith'
```

Podstawową wadą tego podejścia jest ograniczenie poziomów rekurencji pewną stałą. Kolejną wadą jest rozmiar danych powstający w ten sposób. Każda ścieżka rekurencyjne jest w pełni przedstawiona w wierszu zawierającym jej koniec.

Użytkownik, który oczekuje wyników dokładnie takich jakie zwróciłoby zapytanie 2, może wykorzystać drugą zaprezentowaną w pracy metodę, nazywaną *rozwinięciem w głąb* (ang. *vertical unrolling*). W pierwszym kroku wysyłane jest zapytanie analogiczne do ziarna zapytania rekurencyjnego. Wyniki jego złożone są w tabeli tymczasowej `tmpr0`, następnie jeśli zostały utworzone jakieś rekordy wykonywane jest zapytanie analogiczne do części rekurencyjnej, które jako jedno ze źródeł danych używa tabeli `tmp`$(n-1)$, a wyniki wrzuca do tabeli tymczasowej `tmp`n. Proces ten kontynuowany jest tak długo, jak długo nowopowstałe tabele tymczasowe są niepuste. Po utworzeniu serii tabel tymczasowych generowane jest zapytanie zbierające wyniki z nich wszystkich za pomocą klauzul `UNION`. Zapytania, które zostaną wysłane do bazy zaprezentowane są na listingu 7

Listing 7: Rozwinięcie w głąb

```
CREATE TEMPORARY TABLE tmp(0)
  SELECT empId, fname, sname, bossId
  FROM emp
  WHERE sname = 'Smith';
...
CREATE TEMPORARY TABLE tmp(n)
  SELECT e.empId, e.fname, e.sname, e.bossId
  FROM Emp e, tmp(n - 1) t;
...

SELECT sname, fname
FROM (
    SELECT * FROM tmp(0)
  UNION
    SELECT * FROM tmp(1)
  UNION
    ...
```

Rysunek 1: Ścieżki rozpoczynające się od rekordu o identyfikatorze 8

```
 baseId | upId | distance |
--------+------+----------+
      8 |  8   | 0        |
      8 |  6   | 1        |
      8 |  4   | 2        |
      8 |  3   | 3        |
```

```
  UNION
    SELECT * FROM tmp(N)
);
```

Porównanie czasów rozwijania z naiwnym spacerowaniem po grafie jest zaprezentowane w tabeli 3.

Tablica 3: Rozwijanie zapytań w MySQL

Record count	Hierarchy depth	Direct loop	Horizontal unrolling	Vertical unrolling
5000	5	18.1s	0.6s	1.1s
5000	20	10.1s	1.2s	5.4s
10000	5	17.1s	0.7s	1.3s
10000	20	19.3s	2.7s	8.9s

Powyższy wynik zainspirował mnie do przyjrzenia się metodom wykorzystania danych nadmiarowych. Problem za nimi idący, to wpływ na efektywność zapytań modyfikujących dane. W pracy [R7] zaprezentowana została podstawowa metoda tzw. *pełnych ścieżek*. W dodatkowej tabeli składowane są zarówno ścieżki bezpośrednie jak i pośrednie. Na rysunku 1 zaprezentowany został przykład ścieżek rozpoczynających się z rekordu o numerze 8.

Jak dla każdej metody opartej o redundantne dane, problemem staje się ich zainicjowanie oraz utrzymanie. Opracowane rozszerzenie dostarcza automatycznych narzędzi, które utworzą odpowiednią tabelę do składowania ścieżek oraz wyzwalacze synchronizujące zawartość tej tabeli z danymi ba-

Tablica 4: Czas potrzebny do inicjalizacji

Rec. \ depth	5 levels	10 levels	15 levels	20 levels
1 000	01.62	01.18	01.20	01.22
10 000	03.85	06.34	08.41	15.73
100 000	43.68	01:04.82	01:25.28	02:10.72
1 000 000	08:22.51	25:05.28	46:52.04	01:23:03.00

zowymi. Przykładowe czasy utworzenia takich danych zawarte są w tabeli 4.

Metoda ta charakteryzuje się pewną uniwersalnością, o ile przyjmiemy, że struktura grafu nie podlega zbyt wielu modyfikacjom i graf ten jest acykliczny. Pewną wariacją na temat pełnych ścieżek jest metoda ścieżek logarytmicznych, w których składowane są jedynie ścieżki o długościach będących potęgami dwójki. Metoda ta jest zainspirowana pracami [6, 7, 8]. Metoda ta wraz z techniką *zmaterializowanych ścieżek* oraz *zagnieżdżonych podzbiorów* zostały zaprezentowane w [R8].

W metodyce zmaterializowanych ścieżek składowane są pełne ścieżki od węzła do korzenia. Ścieżka ta jest dodana jako dodatkowa kolumna w tabeli, jak to zostało przedstawione na rysunku 2

Rysunek 2: Przykład tabeli rozszerzonej o zmaterializowane ścieżki

```
 eid | fname |   sname   | bid | paths
-----+-------+-----------+-----+------
   6 | Hugh  | Grant     |   3 | [4,3]
   7 | Colin | Firth     |   3 | [3]
   8 | Keira | Knightley |   6 | [6,4,3]
```

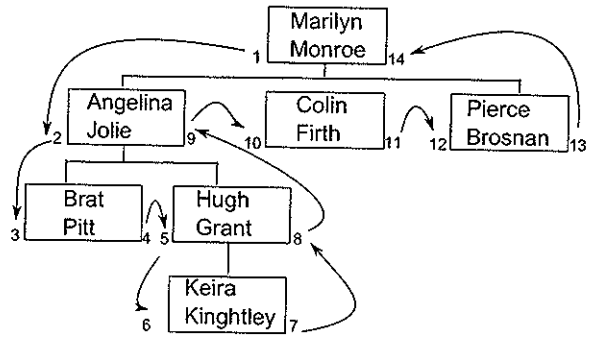
Istotną wadą tego podejścia jest rezygnacja z pierwszej postaci normalnej tabeli.

Ostatnia z prezentowanych metod polega na dodaniu dwóch kolumn `l,r` tak by zachowany był warunki:

1. `l` < `r` dla dowolnej krotki

2. Jeśli a$\neq$ b i a jest w drzewie rozpiętym przez b, to `b.l` $<$ `a.l` oraz `b.r` $>$ `a.r`.

Rysunek 3: Wartości nadmiarowych kolumn `left` i `right` w metodzie *zagnieżdżonych podzbiorów* dla przykładowych danych



Wkładem prac [R7] i [R8] są algorytmy automatycznego generowania opisanych metod oraz przegląd ich wydajności w oparciu o testy wydajności.

4 Uniwersalny model stanu

Pracując nad optymalizacją komunikacji z bazą danych w aplikacjach opartych o odwzorowania obiektowo-relacyjne, postanowiłem bliżej przyjrzeć się modelom danych używanych przez języki programowania i bazy danych. Próba jednolitego ujęcia różnych modeli dała mi głębsze spojrzenie na możliwości i ograniczenia odwzorowań obiektowo-relacyjnych, szczególnie w kontekście nowoczesnych obiektowych języków programowania, takich jak Java, C# i Python. Otrzymane rezultaty tych rozważań zaprezentowałem w publikacjach [T1–T4].

4.1 Stan obiektu

W prowadzanych rozważaniach ograniczamy się jedynie do modelowania stanów obiektów. Problematyka modelowania cyklu życia obiektu jest wciąż otwartym zagadnieniem. Przyjęte podejście nawiązuje do badań Kazimierza Subiety nad teorią i konstrukcją obiektowych języków zapytań [12, 13].

Podstawowe definicje, na których są oparte omawiane prace zostały wprowadzone w [T1]. Pozwolę je sobie przytoczyć dla ułatwienia prezentacji głównych wyników tych czterech prac.

Definicja. Wprowadźmy następujące oznaczenia:

1. Niech V będzie zbiorem wartości atomowych.
2. Wyróżnijmy w nim podzbiór dopuszczalnych *nazw zewnętrznych* $N \subset V$, ze specjalną nazwą $\eta \in N$ zwaną *nazwą pustą.*
3. Ze zbioru wartości wyróżnimy jeszcze zbiór $I \subset V$ zwany *zbiorem identyfikatorów* ze specjalną wartością $\mathbf{nil} \in I$ zwaną *adresem pustym.*
4. Ponadto założymy, że $N \cap I = \emptyset$.

Zbiór V zawiera wszystkie możliwe wartości proste (zgodnie z wcześniejszymi złożeniami napisy też traktujemy jako wartości proste). Zbiór N zawiera wszystkie nazwy zewnętrzne obiektów. Nazwy zewnętrzne są używane przez programistów do pobierania obiektów z bazy danych. Wyróżniona nazwa η jest przypisywana obiektom nieposiadającym własnej nazwy. Obiekty o nazwie η mogą być zlokalizowane tylko za pomocą referencji.

Zbiór I zwiera wewnętrzne identyfikatory obiektów. Identyfikatory wewnętrzne mogą być rozumiane jako adresy obiektów. Zbiór I zawiera również identyfikator **nil** reprezentujący adres pusty.

W celu uproszczenia zapisu wprowadźmy ponadto techniczne oznaczenie. Dla dowolnego zbioru S przez $S^* = \bigoplus_{t \in \mathcal{N}} S$ oznaczmy zbiór skończonych ciągów o wyrazach z S.

Definicja. *Stanem obiektu rzędu* 0 nazywamy trójkę $(i, n, v) \in I \times N \times (V \cup I)$. Przez O_0 rozumiemy zbiór wszystkich możliwych stanów rzędu 0. Zbiór ten będziemy również oznaczać symbolem S_0.

Ponadto wprowadzimy oznaczenie $S_{-1} = \emptyset$. Następnie dla dowolnej liczby naturalnej t wprowadzimy indukcyjnie definicję stanu rzędu t.

Definicja. Niech $t \in \mathcal{N}$ będzie dodatnią liczbą naturalną. Przyjmijmy, że mamy zdefiniowane stany obiektów rzędu k dla dowolnego $k < t$. Przez S_{t-1} oznaczyliśmy zbiór stanów obiektów rzędu mniejszego niż t. Stanem obiektu rzędu t nazywamy trójkę $(i, n, L) \notin S_{t-1}$, gdzie $i \in I, n \in N$ oraz $L \in S_{t-1}^*$ są skończonymi ciągami elementów S_{t-1}. Zbiór wszystkich możliwych stanów rzędu t oznaczymy przez O_t. Przez $S_t = O_t \cup S_{t-1}$ oznaczymy zbiór wszystkich stanów obiektu rzędu nie większego niż t. Elementy listy L będą nazywane podobiektami stanu obiektu o.

Zauważmy, że jeśli $o = (i, n, L) \in O_t$ dla pewnego $t > 0$, to istnieje $o' \in L$ takie, że $o' \notin S_{t-2}$. W przeciwnym wypadku mielibyśmy $o \in S_{t-1}$.

W niniejszym rozdziale interesuje nas stan obiektu w ustalonej chwili, a nie ciąg stanów osiąganych przez obiekt w czasie jego życia. Pojęcie stanu obiektu i obiektu będzie stosowane wymiennie. W myśl przedstawionej definicji każdy obiekt ma identyfikator, nazwę oraz wartość. Wartość ta może być wartością prostą (jak to się dzieje w przypadku obiektów rzędu 0) albo skończoną listą obiektów niższego stopnia. Dla $t > 0$ dowolny obiekt $o \in O_t$ rzędu t ma co najmniej jeden podobiekt rzędu $t - 1$. Liczba t jest zatem maksymalną głębokością zagnieżdżenia podobiektów obiektu o. Ta obserwacja pozwala nam uznać powyższą definicję za poprawną.

Definicja. $S_\infty = \bigcup_{t>0} S_t$ zbiorem wszystkich obiektów naturalnego rzędu.

Definicja. Dla dowolnego $o \in S_\infty$ oznaczmy $Ref(o = (i, n, x)) = i$.

Zdefiniujmy funkcję $\mathcal{S} : S_\infty \to 2^{S_\infty}$, której wartością jest zbiór wszystkich

bezpośrednich i pośrednich podobiektów danego obiektu. Funkcję tę definiujemy indukcyjnie:

1. Jeśli $o \in O_0$, kładziemy $\mathcal{S}(o) = \{o\}$.

2. Przypuśćmy, że $t > 0$ i mamy zdefiniowane $\mathcal{S}$ dla dowolnego $o' \in S_{t-1}$ oraz niech $o = (i, n, (o_1, ..., o_k)) \in O_t$. Wówczas:

$$\mathcal{S}(o) = \{o\} \cup \bigcup_{j=1}^{k} \mathcal{S}(o_j)$$

Kolejną przydatną funkcją będzie $\mathcal{I} : S_\infty \to 2^I$, której wartością jest zbiór zawierający adres danego obiektu oraz adresy wszystkich jego podobiektów. Funkcję tę definiujemy następująco:

$$\mathcal{I}(o) = \bigcup_{o' \in \mathcal{S}(o)} \{Ref(o')\}$$

Zauważmy, że dla dowolnego $o \in S_\infty$ zbiory $\mathcal{S}(o), \mathcal{I}(o)$ są skończone. Wprowadźmy jeszcze pojęcie stanu właściwego. Definicja tego stanu właściwego jest indukcyjna ze względu na rząd obiektu. Każdy stan obiektu rzędu 0 jest właściwy. Załóżmy, że dla $t > 0$ zdefiniowaliśmy już, które stany zawarte w S_{t-1} są właściwe. Niech $o = (i, n, (o_1, ..., o_k)) \in O_t$. Stan o jest *właściwy*, gdy dla dowolnego $j = 1, 2, \ldots, k$ stan o_j jest właściwy oraz spełnione są dwa następujące warunki:

$$Ref(o) \notin \mathcal{I}(o_j) \quad \text{dla każdego} \quad j = 1, \ldots, k \tag{1}$$

$$\mathcal{I}(o_j) \cap \mathcal{I}(o_{j'}) = \emptyset \quad \text{dla każdego} \quad j, j' = 1, \ldots, k; j \neq j' \tag{2}$$

Definicja ta oznacza, że stan jest właściwy wtedy i tylko wtedy, gdy nie powtarzają się w nim identyfikatory obiektów. Zbiór wszystkich stanów właściwych oznaczamy przez $\mathbf{S} \subset S_\infty$.

4.2 Modele obiektów

Na gruncie powyższych definicji w [T1] wyrażone są rozmaite modele danych. Przyjrzyjmy się im na kilku przykładach:

$$
\begin{aligned}
o \;= (i, person, \{ & \\
& (i_0, id, 1) \\
& (i_1, name, \text{"Jan"}) \\
& (i_2, sname, \text{"Kowalski"}) \\
& (i_3, address, 2) \\
\} \;\;) & \in S_1
\end{aligned}
$$

Stan o reprezentuje krotkę $(1, \text{"Jan"}, \text{"Kowalski"}, 2)$ tabeli postaci:

```
table person: id, name, sname, address
```

W pracy [T1] wprowadzamy *Relacyjną równoważność* stanów i pokazujemy, jak krotki relacyjnej bazy danych oraz tabele wyrażane są jako klasy abstrakcji na zbiorach stanów obiektów ze zbiorów S_1 (krotki) i S_2 (tabele). Przy czym rozważany w pracy model relacyjny oparty jest na modelu zaproponowanym przez Grefena w [14]

Ten sam obiekt o może wyrażać obiekt klasy `Person` w języku Java czy też innych językach programowania.

Podjęta w pracy dyskusja pokazuje, że w obiekty w nowoczesnych językach programowania takich jak Java, Python, C# dobrze modelują się poprzez stany klasy S_1. Wynika to z faktu, że pola klas w tych językach zawierają albo wartości proste, albo referencje do innych obiektów.

Tymczasem w bardziej tradycyjnych językach, jak C++, dopuszczających zagnieżdżone obiekty złożone sprawa komplikuje się. Rozważymy następujący przykład:

```
class Address {
    String street;
    String zip_code;
    String city;
};

class Person {
    String name;
    String sname;
    Address address;
};
```

Stany obiektów tak zdefiniowanej klasy `Peron` będą modelowane jako stany S_2. I dla dowolnego naturalnego n możemy rozważać taką klasę, której obiekty nie dadzą się modelować poprzez stany S_n.

W pracy [T1] dalej analizowane są również różne modele baz obiektowych. Rozważania te podsumowane są podziałem modeli na płaskie i zagnieżdżone.

Modele płaskie W tej klasie znajdują się model relacyjny oraz modele danych nowoczesnych języków programowania. Obiekty tych modeli są rzędu 1, natomiast zbiory encji/tabele są wyrażone jako stany rzędu 2.

Modele zagnieżdżone Klasa ta zawiera modele obiektowych baz danych, model dokumentów XML oraz model danych języka C++.

Klasyfikacja ta podkreśla w szczególności niezgodność impedancji na poziomie modelu pomiędzy językiem C++ a modelem relacyjnym, z drugiej strony konstytuując systemy odwzorowań obiektowo-relacyjnych dla nowoczesnych języków programowania takich jak Java, C#, Python etc.

4.3 Uniwersalny język zapytań

Mając model stanu, rozpocząłem rozważania nad językiem zapytań. Proponowany język zapytań, nazwany UQL, jest definiowany jako półgrupa zawarta w zbiorze wszystkich funkcji na S_∞, generowana przez rodziny operatorów prezentowanych w [T2], [T3] oraz uzupełniona i uporządkowana w [T4]. Zaprezentowanie spójnego podzbioru istotnie przekracza formułę autoreferatu. Stąd w niniejszym tekście pojawią się tylko nieliczne elementy. Kompletną informację czytelnik znajdzie w dołączonych publikacjach.

W pracy [T2] zaprezentowany został podstawowy zbiór operatorów pozwalających na wyrażenie przykładowych zapytań SQL. Najprostsze zapytania wymagają operatorów odwzorowania, filtracji oraz spłaszczania.

Operator odwzorowania

Operator odwzorowania służy do wykonania funkcji na wszystkich podobiektach danego obiektu. Funktor $map : \mathbf{S}^\mathbf{S} \to \mathbf{S}^\mathbf{S}$ definiujemy następująco. Niech $f : \mathbf{S} \to \mathbf{S}$ będzie dowolną funkcją. Wówczas:

$$map(f)(o) = \begin{cases} (i, n, (f(o_1), \ldots, f(o_n))) & \text{jeśli } o = (i, n, (o_1, o_2, \ldots, o_k)) \notin O_0 \\ (i, n, \mathbf{nil}) & \text{jeśli } o \in O_0. \end{cases}$$

Operator filtracji

Filtrowanie jest funktorem. Niech $p : \mathbf{S} \to V$ będzie funkcją zwracającą wartości logiczne. Wówczas:

$$\mathit{filter}(p)(o = (i, n, (o_1, o_2, \ldots, o_k))) = (i, n, (o_{i_1}, o_{i_2}, \ldots, o_{i'_k}))$$

Przy czym ciąg $(o_{i_1}, o_{i_2}, \ldots, o_{i'_k})$ jest podciągiem ciągu $(o_1, o_2, \ldots, o_k)$ zawierającym te elementy o_j, dla których $\mathit{eval}_1(p(o_j)) = \mathit{true}$.

Spłaszczanie

Operator *flatten* zamienia podobiekty obiektu złożonego na ich zawartość. W szczególności podobiekty te muszą być obiektami złożonymi, aby podmiana taka miała sens. Jeśli jednak któryś z podobiektów lub też sam obiekt jest atomowy, to wynikiem będzie obiekt atomowy z wartością pustą.

Przyjmijmy $o = (i, n, (o_1, o_2, \ldots, o_k))$. Jeśli dla każdego $t = 1 \ldots k$ $o_t \notin O_0$ i $o_t = (i_t, n_t, (o_{t,1}, \ldots, o_{t,j_t}))$, to

$$\mathit{flatten}(o) = (i, n, (o_{1,1}, o_{1,2}, \ldots, o_{1,j_1}, o_{2,1}, \ldots, o_{k,j_k}))$$

W przeciwnym wypadku kładziemy:

$$\mathit{flatten}(o) = (i, n, \mathbf{nil})$$

Przykładowe zapytanie

Rozważmy następujące zapytanie.

```
SELECT firstname, lastname
FROM emp
WHERE lastname = 'Schmidt'
```

Zapytanie to wyraża się w UQL w następujący sposób:

$$\begin{array}{l} \mathit{map}(\\ \quad \mathit{filter}_{name='firstname' \vee name='lastname'}(\\ \quad\quad \mathit{filter}_{\exists on: name(on)='lastname' \wedge value(on)='Schmidt'}(\\ \quad\quad\quad \mathit{flatten}(\\ \quad\quad\quad\quad \mathit{filter}_{name='emp'}(o) \\ \quad\quad\quad) \\ \quad\quad) \\ \quad) \\ ) \end{array}$$

Kolejną pracą dotyczącą UQL jest [T3]. W pracy tej pokazaliśmy, jak w UQL wyrazić operatory algebry relacyjnej proponowanej przez Grefena w [14]. Operatory rozszerzonej algebry niezbędne do zapytań agregacyjnych nie zostały objęte pracą [T3].

W pracy [T4] uporządkowaliśmy wyniki dotyczące języka UQL oraz rozszerzyliśmy je o zastosowania w języku OQL. W szczególności zaprezentowaliśmy, jak realizowane są zapytania zawierające złączenia zależne (ang. Dependent Join) oraz wyrażenia ścieżkowe (ang. Path Expressions). Ponadto pokazaliśmy sposób wyrażania operatorów algebry obiektowej [15] w UQL.

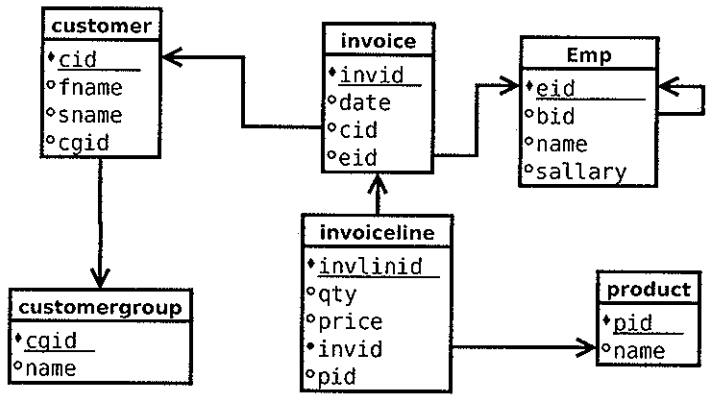
5 Wykorzystanie zmaterializowanych sum częściowych w zapytaniach analitycznych

Integracja zapytań rekurencyjnych z systemem odwzorowań obiektowo-relacyjnych Hibernate zainspirowały mnie do poszukiwań kolejnych technik optymalizacyjnych, które warto zintegrować z systemami ORM. Techniki te co prawda są osiągalne na poziome administracji bazą danych, jednakże integracja ich z Hibernate niesie za sobą wymierne korzyści:

- przeniesienie decyzji z wdrożenia do etapu projektu,
- gwarancja wykonania w każdej instancji,
- znaczne uproszczenie, bowiem skomplikowane mechanizmy zastąpione są prostym interfejsem.

Przypuśćmy, że mamy przykładową bazę danych zgodną ze schematem przedstawionym na rysunku 4.

Rysunek 4: Przykładowy schemat bazy sprzedaży



Częstym wyzwaniem dla takiej bazy są zapytania agregacyjne potrzebne do raportów. Rozważmy przykładowe zapytanie zaprezentowane w listingu 8. W przypadku małych danych, czas realizacji takiego zapytania jest akceptowalny, jednakże przy większych zasobach staje się on problematyczny. Oczywiście można zbudować hurtownię danych skrojoną na potrzeby użytkownika, jednakże zazwyczaj takie podejście generuje duże koszty.

Listing 8: Łączna sprzedaż dzienna dla wybranych klientów

```
SELECT fname, sname, date, SUM(quantity * price)
  FROM cust JOIN invoice USING (cid)
            JOIN invline USING (invid)
  WHERE cid IN [.....]
  GROUP BY sname, fname, date
```

Przypuśćmy że użytkownik posiada bazę danych, w której czas realizacji takiego zapytania wynosi około godziny. W eksperymentach prowadzonych przeze mnie taki czas otrzymałem dla bazy zajmującej 100GB i zawierającej miliard rekordów w tabeli `invline`. Więcej na temat wyników testów zaprezentowane jest w podrozdziale 5.3. W takich przypadkach naturalną techniką optymalizacji jest agregacja danych dla zapytań analitycznych.

5.1 Składowanie sum częściowych

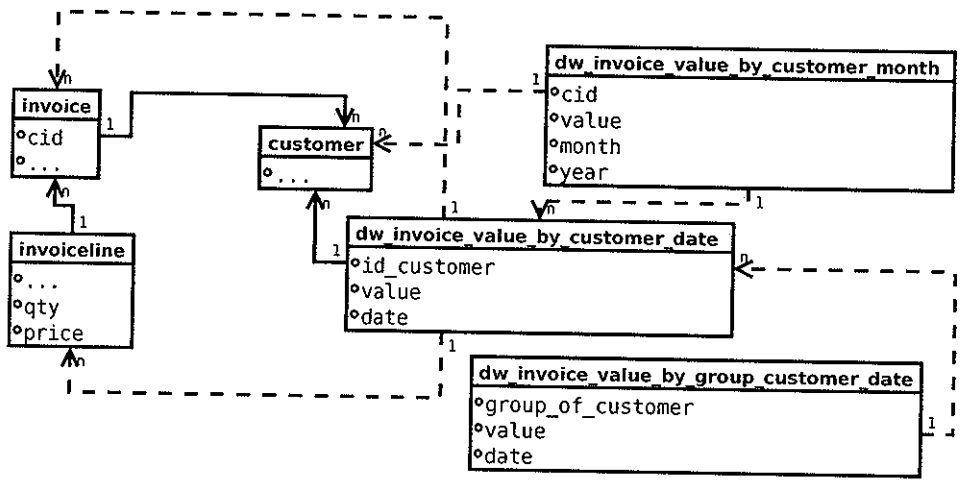
W pracy [P1] zaproponowany został prosty interfejs, w którym poprzez adnotacje w Javie dodane do definicji klasy trwałej, projektant precyzuje jakie agregacje będą dla niego użyteczne. Przykład został zaprezentowany na listingu 9. Rozwiązanie zaproponowane w pracy wzbogaca automatycznie klasę o metody odpytywania zapytań analitycznych. Po stronie bazy natomiast automatycznie tworzone są dodatkowe tabele do składowania agregacji oraz niezbędne wyzwalacze do synchronizacji z danymi bazowymi.

Listing 9: Adnotacje klasy `Invoice`

```
@Entity
public class Invoice {
    private Long id;
    @DWDim
    private Date date;
    @DWDim
    private Customer customer;
    @DWAgr(function="SUM(quantity*price)")
    private List<InvoiceLine> invoiceLines;
    ...
```

W efekcie dodania adnotacji schemat bazy danych zostanie rozszerzony o tabelę `dw_invoice_value_by_customer_date` pokazaną na rysunku 5. Na-

tomiast zamiast zapytania 8 zaproponowany interfejs wyśle zapytanie zaprezentowane na listingu 10.



Rysunek 5: Schemat rozszerzony o sumy częściowe

Listing 10: Łączna sprzedaż dzienna dla wybranych klientów z użyciem agregacji

```
SELECT fname, sname, date, value
    dw_invoice_value_by_customer_date USING (cid)
WHERE cid IN [......]
```

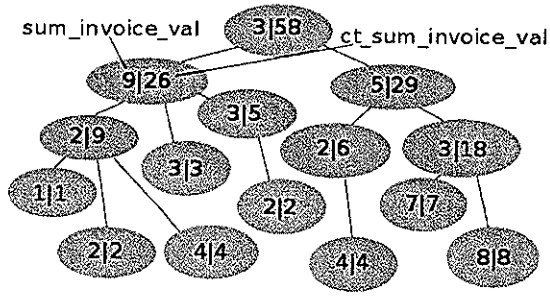
Czas realizacji takiego zapytania przy bazie zawierającej około miliarda rekordów sprzedaży liczony jest w sekundach, podczas, gdy oryginał wyraża się w godzinach. Oczywiście utrzymanie odpowiednich tabel wiąże się z istotnym narzutem na czas przetwarzania zapytań aktualizujących. Do problemu tego narzutu powrócimy w podrozdziale 5.3.

5.2 Wykorzystanie drzew licznikowych

W pracy [P2] podjęte zastało zagadnienie, będące niejako hybrydą problemu rekurencji i zapytań analitycznych. Schemat z rysunku 4 zawiera naturalną hierarchię pracowników. Jeśli baza obsługuję sieć sprzedaży bezpośredniej, interesującym pytaniem jest sprzedaż łączna dokonana przez podsieć pracowników rozpinaną przez wskazanego pracownika.

Podejście z pracy [P1] zostało w [P2] rozszerzone o bieżące zliczanie i składowanie sum. Idea została oparta na drzewach licznikowych (ang. *segment trees*), gdzie w danym węźle trzymana jest nie tylko wartość sprzedaży danego pracownika, ale też suma sprzedaży całego poddrzewa rozpiętego przez niego, jak to zostało przedstawione na rysunku 6.

Rysunek 6: Rozszerzone drzewa licznikowe



Istotny problem, z jakim należało się zmierzyć w tym kontekście, to zakleszczenia w okolicy korzeni drzew. Każdy węzeł drzewa przy aktualizacji wyzwala aktualizację węzła swojego rodzica. W efekcie aktualizacja korzenia odbywa się przy okazji aktualizacji każdego węzła w drzewie. W pracy zaproponowane zostało rozwiązanie pozwalające zaoszczędzić wielokrotnej aktualizacji poprzez wprowadzenie opóźnienia do aktualizacji rodzica. W pracy tej proponuje się metodę nazwaną *ring update*, w której w momencie modyfikacji węzła jego rodzic trafia do kolejki, o ile jeszcze się tam nie znajdował. Kolejka jest obsługiwana przez osobny wątek w oddzielnych transakcjach. Rozwiązanie to co prawda dopuszcza niespójne wyniki, ale z drugiej strony, otrzymane w ten sposób sumy i tak są zazwyczaj bliższe prawdzie niż wartości uzyskane z wielogodzinnego zapytania agregującego dane bezpośrednio z wierszy sprzedaży. W efekcie uzyskujemy *spójność docelową* (ang. eventual consistency).

5.3 Wykorzystanie grafu metagranul w przepisywaniu zapytań

Metody zaproponowane w pracy [P1] istotnie wspomagają zapytania analityczne, jednakże jeśli zastosujemy je wielokrotnie, to narzut na operacje aktualizujące dane może szybko przekroczyć dopuszczalne granice.

Problem ten zainspirował mnie do poszukiwań metod pośrednich. Postanowiłem eksperymentować z przepisywaniem zapytań analitycznych na zapytania oparte na sumach częściowych już zagregowanych, a następnie metodę tę zalgorytmizować. I tak zapytanie 11 o sumy sprzedaży dla grup klientów w danych miesiącach zamiast obliczać z danych pierwotnych co w teście trwało około godziny, można policzyć z agregacji dla sprzedaży wg klienta i dnia, jak to zostało zaprezentowane w zapytaniu 12.

Listing 11: Łączna sprzedaż w grupach klientów w lipcu 2011

```
SELECT year (invoice.date) AS year,
    month(invoice.date) AS month,
    cg.name, sum(invoiceline.qty * invoiceline.price)
  FROM customergroup cg JOIN customer USING(cgid)
    JOIN invoice USING(cid)
    JOIN invoiceline USING (invid)
  GROUP BY year (invoice.date),
     month(invoice.date), cg.name
  HAVING year = 2011 and month = 7
```

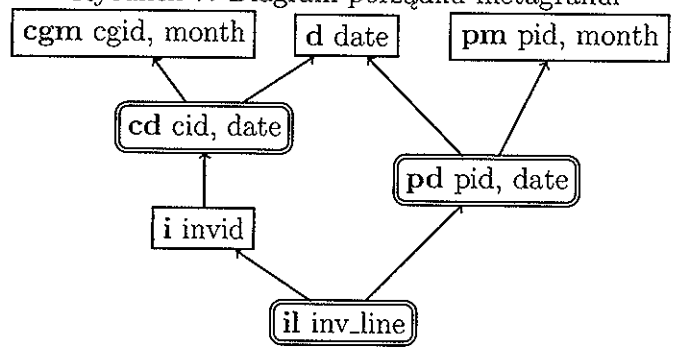
Listing 12: Łączna sprzedaż w grupach klientów w lipcu 2011 w oparciu o metagranulę cd

```
SELECT year (cd.date) AS year,
         month(cd.date) AS month,
         cg.name, sum(cd.value)
  FROM customergroup cg JOIN customer USING(cgid)
    JOIN dw_invoice_value_by_customer_date cd USING (cid)
  GROUP BY year (cd.date),
     month(cd.date), cg.name
  HAVING year = 2011 and month = 7
```

Test pokazał, że podejście to redukuje czas realizacji zapytania z godziny do około minuty. Zaproponowany algorytm został zaprezentowany w pracy [P3]. Poniżej przedstawię idee, na których się on opiera. Dla ustalenia uwagi przyjmijmy, że interesują nas zapytania o łączną wartość sprzedaży.

Poziom grupowania niezbędny do policzenia zapytania agregującego pojawiającego się w aplikacji nazywać będziemy *metagranulą*. W przypadku zapytania 11 jest to para `month(date)`, `cgid` (miesiąc i grpupa klientów). Metagranula wyznaczone jest poprzez pola występujące w klauzulach `GROUP`

Rysunek 7: Diagram porządku metagranul



BY oraz WHERE. Okazuje się, że poziomy te tworzą naturalny częściowy porządek, który jest przestawiony na rysunku 7.

Metagranulę a nazwiemy niemniejszą niż metagranulę b jeśli grupowanie b jest podziałem w grupowaniu a. W konsekwencji agregacje dla metagranuli a można policzyć z agregacji dla metagranuli b.

W diagramie tym grupowanie ze względu na miesiąc sprzedaży i grupę klientów cgm jest powyżej grupowania ze względu na klienta i dzień sprzedaży, co oznacza, że zapytanie 11 może być wyrażone poprzez agregacje dla sprzedaży dziennej dla klientów cd, w szczególności może zostać przepisane na zapytanie 12.

Drugim kluczowym pojęciem w pracy jest *metagranula właściwa*. Metagranulę nazywamy *właściwą*, jeśli agregacje przez nie wyznaczone zostały złożone w bazie. Celem algorytmu przedstawionego w pracy [P3] jest wskazanie dla danej metagranuli m maksymalnej metagranuli właściwej $\widehat{m}$, spełniającej warunek $\widehat{m} \leq m$. Taka metagranula zawsze istnieje, gdyż elementem najmniejszym w zbiorze metagranul jest tabela bazowa, z której budowane są agregacje. Jeśli metagranula m jest właściwa, to $\widehat{m} = m$

W diagramie 7 pogrubionymi owalami oznaczone są metagranule, które w bazie testowej uczynione są metagranulami właściwymi. Przy tym wyborze jasne jest, że $\widehat{\texttt{cgm}} = \texttt{cd}$, $\widehat{\texttt{pm}} = \texttt{pd}$ oraz $\widehat{\texttt{i}} = \texttt{il}$. W przypadku gdy algorytm ma kilka nieporównywalnych metagranul właściwych do wyboru, jak to jest w przypadku obliczania $\widehat{\texttt{d}}$ wybrana zostanie spośród nich metagranula o minimalnej liczebności. W przykładowej bazie liczebność metagranuli pd wynosiła około 60 000 000 rekordów, podczas gdy metagranula cd zawierała 95 000 000 rekordów, stąd $\widehat{\texttt{d}} = \texttt{pd}$. Testy pokazały, że wybór był słuszny,

zapytanie o sumy grupowane względem dat przepisane na metagranule pd liczyły się średnio prawie dwa razy szybciej niż przepisane na metagranulę cd.

Ważnym elementem pracy [P3] są testy prezentujące obciążenie wynikające ze składowania metagranul. Testy te prowadzone były na 3 bazach: `plain`, `med` oraz `full`. Bazy te w danych podstawowych zawierają dane zgodne ze schematem prezentowanym na rysunku 4. Dane te zawierają około 100 000 000 faktur posiadających w sumie ponad 1 000 000 000 rekordów linii faktur. Baza `plain` zajmuje około 100GB na dysku twardym. Baza `med` jest rozszerzeniem bazy `plain` o składowanie metagranul `cd` i `pd`, jako metagranul właściwych. W szczególności wyposażona została w wyzwalacze, które w czasie rzeczywistym uaktualniają dane w tabelach agregacyjnych. Dodanie tych dwóch tabel zwiększyło rozmiar bazy do 110 GB. Baza `full` posiada zmaterializowane wszystkie metagranule z diagramu 7, oczywiście każda została dołączona z zestawem wyzwalaczy synchronizujących ich zawartość względem danych bazowych. Jej rozmiar to 121GB.

W tabeli 5 zaprezentowano średnie czasy wstawiania paczek z fakturami. Pierwsza kolumna zawiera liczby wstawianych faktur. Każda faktura ma średnio 13 linii. Druga kolumna określa, czy przed ładowaniem danych wyczyszczono bufory bazy danych.

Tablica 5: Czas ładowania faktur wraz z synchronizacją metagranul

Invoices	Bufor	*plain*	*med*	*full*
10 000	zimny	2m 58.335s	29m 06.670s	37m 56.663s
10 000	gorący	3m 03.308s	9m 05.212s	13m 01.924s
15 000	zimny	4m 30.261s	20m 59.395s	36m 30.021s
20 000	zimny	6m 32.502s	29m 59.064s	44m 38.321s

Wyniki te dowodzą, że stosując przepisywanie zapytań na dobrze wybrane metagranule otrzymujemy rozsądny kompromis redukujący czas zapytań do akceptowalnego poziomu zachowując przy tym możliwie rozsądny czas wstawiania nowych danych. Specyfikacja sprzętu testowego jest podana w pracy [P3].

6 Automatyka indeksów funkcyjnych

Kolejną techniką optymalizacyjną, którą postanowiłem zintegrować z systemami odwzorowań obiektowo-relacyjnych jest wsparcie dla indeksów funkcyjnych. Wiele współczesnych systemów zarządzania bazami danych implementuje indeksy funkcyjne. Jednakże tylko część z nich posiada wsparcie takich indeksów na poziomie narzędzi do strojenia bazy danych.

Z punktu widzenia projektanta systemu wsparcie indeksów funkcyjnych oczekiwane jest na dwóch poziomach. Tworząc projekt chciałby on mieć możliwość wyrażenia explicite, że indeks taki powinien się zostać zbudowany. Z drugiej strony potrzebuje narzędzi, które taki indeks mogłoby zasugerować. Rozwiązania obydwu tak postawionych problemów zostały przedstawione w [F].

W celu czytelniejszej prezentacji indeksów funkcyjnych i algorytmu ich wyszukiwania posłużę się uproszczoną wersją schematu danych z rysunku 4 przedstawioną na rysunku 8. Wszystkie przykłady w niniejszym rozdziale będą prezentowane w kontekście tej wersji schematu danych.

Rozważmy przykładowe zapytanie zaprezentowane na listingu 13.

Listing 13: Dziesięć faktur o najwyższej wartości

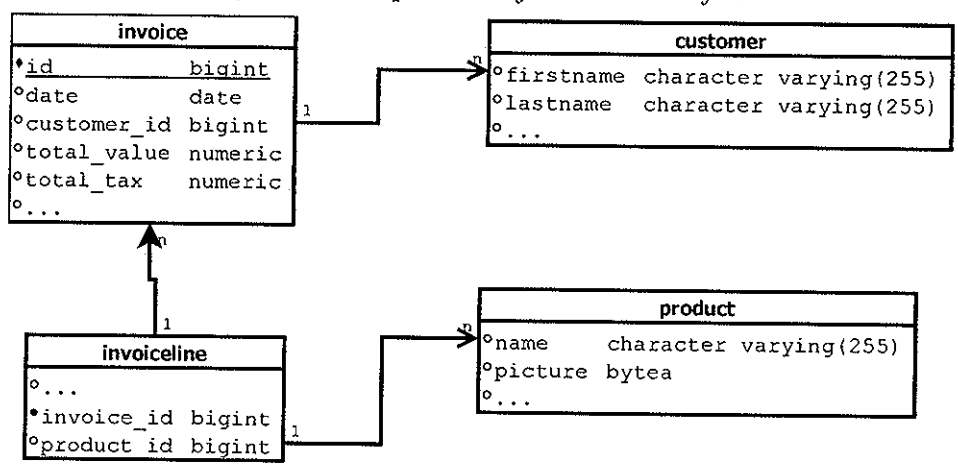
```
SELECT * FROM invoice
  ORDER BY (total_value + total_tax) DESC
  LIMIT 10
```

Ewaluacja tego zapytania może wykorzystać indeks funkcyjny na wartości `(total_value + total_tax)`, o ile indeks taki będzie istniał w bazie. Aby umożliwić definiowanie tego typu indeksów w pracy [F] zaproponowaliśmy prosty interfejs w postaci adnotacji dołączanych do klasy trwałej w języku Java. Na listingu 14 przedstawiona została adnotacja deklarująca omawiany przykład indeksu, zaimplementowany przez nas jako rozszerzenie biblioteki Hibernate.

Listing 14: Deklaracja indeksu funkcyjnego

```
@Entity
@Table(name = "invoice")
@FunctionalIndex(expr="(totalValue+totalTax)")
public class Invoice implements Serializable {
  ...
}
```

Rysunek 8: Uproszczony schemat danych

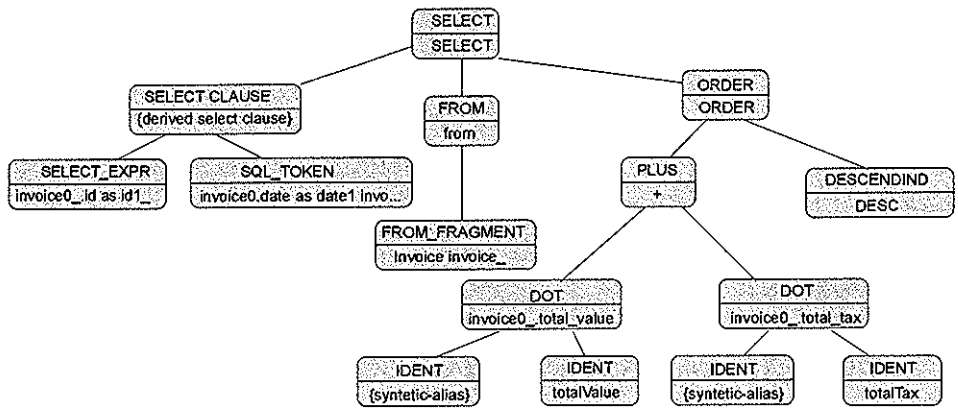


Rozszerzenie omawiane w [F] wygeneruje automatycznie odpowiedni indeks funkcyjny w bazie.

Problem automatycznego wyszukania właściwych indeksów funkcyjnych został rozwiązany poprzez analizę drzew AST zapytań HQL występujących w kodzie projektu. Do parsowania zapytań wykorzystany został moduł z biblioteki Hibernate. Na rysunku 9 zaprezentowane jest drzewo zapytania o faktury posortowane względem wartości, dostarczone przez parser Hibernate.

Hibernate w trakcie budowy drzewa dodaje informacje symboliczne o tabelach. Algorytm analizujący drzewo przeszukuje klauzule WHERE i ORDER BY, pod kątem wyrażeń arytmetycznych na polach. Jeśli wykryje wyrażenie oparte na polach pochodzących z jednej tabeli proponuje założenie na niej indeksu funkcyjnego. Jeśli użytkownik akceptuje propozycję odpowiednia adnotacja zostaje dołożona do definicji klasy, jak to zostało zaprezentowane w listingu 14.

Rysunek 9: Przykład drzewa zapytania sparsowanego przez Hibernate



7 Planowane badania

Praca [P3] otwiera nowe problemy badawcze, którymi planuję się zająć w najbliższym czasie.

Pierwszy z nich to wybór optymalnego zestawu metagranul do utrwalenia. Dla danego zbioru metagranul właściwych $\mathcal{P}$ zawartego w zbiorze wszystkich metagranul M, możemy rozważać koszt opisany wzorem:

$$\Phi(\mathcal{P}) = |\mathcal{P}| \sum_{m \in M} |(\widehat{m})| fr(m),$$

gdzie $fr(m)$ jest częstością odpytywania metagranuli m, natomiast $(\widehat{m})$ oznacza najkorzystniejszą metagranulę właściwą do przepisania zapytania opartego o metagranulę m.

Najprostszym sposobem wyznaczenia optymalnego, ze względu na funkcję kosztu, zbioru metagranul właściwych jest metoda typu *brute force*, sprawdzająca koszt dla wszystkich możliwych podzbiorów. Niestety takie podejście jest bardzo obciążające. Dla grafu zawierającego 30 metagranul otrzymujemy miliard możliwości do przetestowania. Zamiast poszukiwać rozwiązania optymalnego zamierzam dostosować algorytm genetyczny do potrzeb poszukiwania podzbioru generującego najmniejszy koszt. Jakość tego podejścia zostanie zweryfikowana poprzez porównanie z wynikami otrzymanymi metodą *brute force* na wybranych przykładach referencyjnych.

Kolejnym problemem jest identyfikacja wszystkich metagranul, które zamierzamy rozważać. Problem ten planuję rozwiązać na dwa sposoby. Użyt-

kownicy systemu odwzorowań obiektowo-relacyjnych Hibernate wyrażają zapytania grupujące w języku HQL. Już na etapie projektowania aplikacji możemy użyć wewnętrznego parsera biblioteki Hibernate do analizy drzewa składniowego, podobnie jak analizujemy te drzewa w pracy [F]. Pozwoli to na wyłuskanie interesujących metagranul oraz zbudowanie ich grafu na etapie projektu. Drugi sposób przeznaczony jest dla użytkowników systemu PostgreSQL. Analizując ślad bazy danych możemy rejestrować zapytania oraz ich częstość. PostgreSQL jest systemem z otwartym kodem i pierwsze próby wykorzystania wbudowanego parsera pokazują, że możliwe jest jego użycie w celu analizy tak zebranych zapytań.

Dzięki rozbudowie przestrzeni dyskowej w komputerze służącym testom proponowane rozwiązania planujemy testować na bazach 350 GB, a następnie na bazach rzędu 1TB zawierających ponad 10 miliardów rekordów. Dzięki testom dla takich rozmiarów danych adresaci dostarczancyh rozwiązań będą mogli łatwiej ocenić, czy proponowana metodyka jest adekwatna do posiadanych przez nich baz danych.

8 Pozostałe wyniki autora uzyskane po obronieniu doktoratu

Okres po obronie rozprawy doktorskiej był z jednej strony czasem na podsumowanie wyników rozprawy i przygotowaniu ich do publikacji, z drugiej okresem szukania własnego miejsca na polu nauki. Poza publikacjami głównego nurtu badań przedstawionymi w rozdziale 2 w tym czasie opublikowane zostały następujące wyniki:

M1 Tyc, A., Wiśniewski, P. *The Lasker–Noether theorem for commutative and noetherian module algebras over a pointed Hopf algebra*, Journal of Algebra 267 (2003) 58 - 95

M2 Neusel, M.D. *Connected Hopf algebras with Dixmier basis and infinite primary decomposition* (2005), preprint dostępny pod adresem http://Hopf.math.purdue.edu/Neusel-Wisniewski/piotr.pdf

SB1 Burzańska, M., Wiśniewski, P.: *Pysbql-python-like query language constructed using stack base approach*, Annales UMCS, Informatica, (2007) 143–151

SB2 Burzańska, M., Wiśniewski, P. *L-value and r-value concept-proposition to solve ref & deref chaos in sbql languages family*, Pol. J. Environ. Stud. 18 (2007) 143–151

SB3 Burzanska, M., Stencel, K., Wisniewski, P. *Intermediate structure reduction algorithms for stack based query languages*, FGIT-ASEA, Volume 117 of Communications in Computer and Information Science, Springer (2010) 317–326

SN Ćwiszewski, A., Wiśniewski, P. *Coverage verification algorithm for sensor networks*, Computer Applications for Bio-technology, Multimedia, and Ubiquitous City, Volume 353 of Communications in Computer and Information Science, Springer (2012) 397–405

TU1 Stencel K., Wiśniewski P. *Uniwersalny model stanu obiektu*, rozdział w *Krzysztof Stencel: Obiektowe i półstrukturalne bazy danych* , Wydawnictwo PJWSTK, Warszawa 2012

TU2 Stencel K., Wiśniewski P. *Uniwersalny język zapytań*, rozdział w *Krzysztof Stencel: Obiektowe i półstrukturalne bazy danych* , Wydawnictwo PJWSTK, Warszawa 2012

8.1 Wyniki w zakresie działań algebr Hopfa

Moją najważniejszą publikacją matematyczną jest [M1]. Formalnie proces publikacyjny tego tekstu przebiegał po obronie rozprawy doktorskiej, stąd znalazł się on w niniejszym zestawieniu, jednakże wyniki zawarte w nim pochodzą z rozprawy doktorskiej.

Pewne drobne wyniki, które udało mi się uzyskać zostały zaprezentowane na konferencji w Getyndze w 2003r. Następnie podczas pobytu w Lubbock zostały rozszerzone we współpracy z Marą Neusel i przygotowane w formie preprintu [M2].

8.2 Zagadnienia informatyczne

Pierwszym obszarem poszukiwań w zakresie baz danych były próby związane z podejściem stosowym w bazach danych i konstrukcji obiektowych języków zapytań, dla których punktem wyjścia były prace [16, 12, 13]. W roku 2005 do rozważań tych zaprosiłem moją ówczesną magistrantkę Martę Burzańską (z d. Rogińską). W wyniku wspólnych rozważań udało się opracować podstawy języka PySBQL, który jest jednocześnie językiem programowania i językiem zapytań. Język ten łączy zalety podejścia stosowego w językach zapytań, z łatwością programowania języka Python. Wyniki te zostały zaprezentowane na konferencji IBIZA w 2007 roku. Następnie ukazały się drukiem w [SB1].

W tej tematyce z panią Burzańską opublikowaliśmy jeszcze jedną pracę [SB2] podejmujący polemikę z pewnymi założeniami przyjętymi w [13]. W języku SBQL przyjęto założenie, że napis niesie za sobą jedną wartość, gdy tymczasem językach programowania używamy lewej i prawej wartości.

Do tematyki tej powróciliśmy na krótko ucząc się zagadnień związanych z deforestacją [17, 18]. Metody deforestacji udało się zastosować w języku SBQL. Wyniki zostały zaprezentowane w [SB3].

Zainspirowani pracami Roberta Ghrista [19, 20, 21] z dr. hab. Aleksandrem Ćwiszewskim opracowaliśmy prosty czytelny algorytm redukcji kompleksu simplicjalnego zachowujący równoważność homotopijną. Redukcja ta ma zastosowanie w badaniu pokrycia siecią sensorów których pozycja nie jest

znana, przy czym musi być znana informacja, jakie sensory znajdują się w pobliżu danego sensora. Wyniki te zostały zaprezentowane w pracy [SN].

Prace [TU1] i [TU2] są rozszerzeniem prac [T1, T2 i T3]. Tematyką wpisują się w dorobek przedstawiany w rozdziale 2, jednakże mój udział autorski w nich jest mniejszy niż 50 %, w związku z czym zostały pominięte w prezentacji dorobku naukowego.

9 Informacje o autorze

Piotr Konrad Wiśniewski

Data i miejsce urodzenia: 1 maja 1972 r. Białogard, Polska
Stan cywilny: żonaty, dwoje dzieci.
Zatrudnienie:

- od maja 2008r. adiunkt Zakładzie Baz Danych, Wydział Matematyki i Informatyki, Uniwersytet Mikołaja Kopernika w Toruniu.
- od stycznia 2003r. do kwietnia 2008. adiunkt w Katedrze Algebry i Geometrii WMIiI UMK w Toruniu,
- od października 2002r. do grudnia 2003r. adiunkt w Zakładzie Algebry i Topologii WMIiI UMK w Toruniu,
- od października 1996r. do września 2002r. asystent a Zakładzie Algebry i Topologii WMIiI UMK w Toruniu.

Edukacja

2001 Uzyskanie stopnia doktora nauk matematycznych, na podstawie rozprawy *Twierdzenie Laskera - Noether dla przemiennych i noetherowskich algebr modułowych nad punktową algebrą Hopfa*, WMiI UMK w Toruniu.

1996 Uzyskanie stopnia magistra matematyki, na podstawie pracy *Wielkie Twierdzenie Fermata dla pierścieni cyklotomicznych*, WMiI UMK w Toruniu.

Konferencje naukowe matematyczne

Bruksela 2002 Na konferencji *Hopf algebras in Noncommutative Geometry and Physics* wygłosiłem referat pt *The Lasker-Noether theorem for commutative and noetherian module algebras over a pointed Hopf algebra*, na którym przedstawiłem główne wyniki mojej rozprawy doktorskiej.

Getynga 2003 Na konferencji *Invariant Theory and its Interactions with Related Fields INGO 2003*, prezentowałem plakat *Invariance of minimal primes of an H-module Algebras over a connected Hopf algebra over a field of characteristic 0* z wynikami, które zostały następnie ujęte w [M2].

Lubbock 2004 Na konferencji *Red Riders Symposium* przedstawiłem referat *The Lasker Noether theorem for H module algebra over Hopf algebra H.*

Konferencje naukowe informatyczne

IBIZA 2007 Na konferencji *Informatyka badania i zastosowania*, która odbyła się w Kazimierzu Dolnym przedstawiłem referat pt *Pysbql-python-like query language constructed using stack base approach* omawiający wyniki pracy [SB1]

IBIZA 2008 Kazimierz Dolny. Na konferencji tej przedstawiłem plakat prezentujący idee *Model obiektowy w hurtowniach danych.*

FIT 2009 Na konferencji *Forum Informatyki Teoretycznej*, Zakopane, prezentowałem referat *Problem automatycznej dereferencji w językach rodziny SBQL*, pokazujący problemy w językach rodziny SBQL. Problemy te stały się inspiracją do pracy [SB2].

SMI 2009 Na Konferencji *Sejmik Młodych Informatyków*, Międzyzdroje, prezentowałem plakat *L-Value and R-Value Concept-Proposition to Solve Ref & Deref Chaos in SBQL Languages Family* informujący o wynikach pracy [SB2].

FIT 2010 Zakopane. Na konferencji tej przedstawiłem referat *Kilka słów o niezgodności impedancji*, prezentujący początkowe wyniki opublikowane później w pracy [T1].

FGIT 2010 Future Generation Information Technology, December 13 - 15, 2010, International Convention Center Jeju, Jeju Island, South Korea. Na konferencji tej prezentowałem trzy referaty:

- *Recursive query facilities in relational databases: a survey* prezentujący wyniki prac naszych doktorantów opublikowanych w [5].

- *Recursive Queries Using Object Relational Mapping* prezentujący wyniki pracy [R2]
- *Intermediate Structure Reduction Algorithms for Stack Based Query Languages* prezentujący wyniki opublikowane w [SB3]

ADBIS 2011 15th East-European Conference on Advances in Databases and Information Systems, 19- 24 września 2011, Wiedeń, Austria. Na konferencji tej prezentowałem referat *Hibernate the Recursive Queries - Defining the Recursive Queries using Hibernate ORM*, przedstawiający wyniki pracy [R3]

FGIT 2011 Jeju Island, South Korea. Na konferencji tej brałem udział w pracach komitetu programowego podkonferencji **DTA 2011**, oraz wygłaszałem referaty:

- *Application of wavelets and kernel methods to detection and extraction of behaviours of freshwater mussels*, w referacie tym przedstawiałem wyniki pracy [22] w imieniu autorów. Praca ta powstawała w dużej mierze na moim rodzimym wydziale i na bieżąco śledziłem jej postępy. W tej systuacji mogłem zastąpić autorów, z których żaden nie był w stanie udać się na tę konferencję.
- *Efficient Implementation of Recursive Queries in Major Object Relational Mapping Systems* przezntujący wyniki pracy [R4]
- *Partial Aggregation Using Hibernate* prezentujący wyniki pracy [P1]

ADBIS 2012 16th East-European Conference on Advances in Databases and Information Systems, 17 - 20 Września, Poznań, Polska. Występowałem z referatem *Extending HQL with Plain Recursive Facilities* prezentujący wyniki z [R5]

CS&P 2012 Concurrency, Specification, and Programming, 26 - 28 wrzesnia 2012, Berlin, Niemcy. Na konferencji tej prezentowałem referat *Universal query language* omawiający wyniki pracy [T2].

FGIT 2012 Kangwondo, South Korea, na konferencji tej brałem udział w pracach komitetu programowego podkonferencji **DTA 2012**, oraz wygłaszałem referaty:

- *Unrolling SQL:1999 recursive queries* zawierający wyniki pracy [R6]
- *A Multi-set Relational Algebra in View of Universal Query Language* prezentujący wyniki pracy [T3]
- *Coverage verification algorithm for sensor networks* prezentujący wyniki pracy [SN]

ADBIS 2013 17th East-European Conference on Advances in Databases and Information Systems, 1 - 4 Września, Genua, Włochy. Wygłosiłem tam referat *On Materializing Paths for Faster Recursive Querying* prezentujący wyniki pracy [R7]

FedCSIS 2013 Kraków, Polska. Na konferencji byłem organizatorem workshopu PBDA2013. Wygłosiłem również referat *On Redundant Data for Faster Recursive Querying Via ORM Systems* prezentujący wyniki pracy [R8].

BCI 2013 6th Balkan Conference in Informatics, 19 - 21 września, Saloniki, Grecja, Na konferencji tej prezentowałem referat *Enhanced segment trees in object-relational mapping* zawierający wyniki z [P2].

CS&P Concurrency, Specification, and Programming, 25-27 września 2013, Warszawa. Prezentowany referat *Automatic Selection of Functional Indexes for Object Relational Mapping System*, obejmował wyniki [P3].

Inne wystąpienia

- W lipcu 2002 na seminarium prowadzonym przez prof. Hansa Jürgena Schneidera na Uniwersytecie Monachijskim prezentowałem referat *Connected Hopf Algebras and Minimal Primes in H-module Algebras.*
- Podczas Stażu na Uniwersytecie w Monachium w 2003r prezentowałem wyniki badań w referacie *Invariants of Minimal Prime Ideals Under Action of Skew Derivations.*
- Texas Tech University Lubbock, Texas, USA, listopad 2004. Na seminarium prof. Mary Neusell prezentowałem referat *Coalgebras with a Dixmier basis and primary decomposition.* Referat ten był punktem

wyjścia do badań nad istnieniem bazą Dixmiera w strukturze coalgebrowej algebry Stenroda w charakterystyce dodatniej. Rozważania te były celem mojego dwutygodniowego pobytu w Lubbock.

- W maju 2012r. wygłaszałem referat *Enhanced Counting Trees In Hibernate View* na seminarium prof. Andrzeja Skowrona na Wydziale Matematyki Informatyki i Mechaniki Uniwersytetu Warszawskiego. Referat ten przedstawiał wstępne założenia prac badaeczych, których zwieńczeniem jest praca [P2].

- W lutym 2013 przedstawiałem referat *Uniwersalny język zapytań oparty o Uniwersalny model stanu* na seminarium prowadzonym przez prof. Kazimierza Subietę w Polsko Japońskiej Wyższej Szkole Technik Komputerowych. Na wystąpieniu tym omawiałem bieżący stan prac nad modelem USM oraz językiem UQL.

- W sierpniu 2013 wygłosiłem na zaproszenie prof. Agostino Cortesiego wygłosiłem referat *Object Relational Mapping as Optimization Layer* na Universita Ca'Foscari, Wenecja. W referacie tym przedstawiłem bieżący stan prac nad tematyką prezentowaną w niniejszym autoreferacie, z pominięciem prac [T1 - T4]

Udział w projektach badawczych i rozwojowych

Kardionet *Rozwój zaawansowanych metod obrazowania medycznego, rozproszonej akwizycji, archiwizacji i teletransmisji w zintegrowanym systemie opieki kardiologicznej ostrych zespołów wieńcowych*, projekt finansowany przez Fundusze Norweske, nr PL-0262. W projekcie tym zajmowałem się architekturą rozwiązań bazodanowych ze szczególnym uwzględnieniem składowania obrazowań medycznych w formacie DICOM.

SYNAT *Utworzenie uniwersalnej, otwartej, repozytoryjnej platformy hostingowej i komunikacyjnej dla sieciowych zasobów wiedzy dla nauki, edukacji i otwartego społeczeństwa wiedzy*, projekt sponsorowany przez NCBiR, nr SP/I/1/77065/10. W projekcie zajmowałem się aspektami dopasowania obiektów, oraz opiekowałem się architekturą bazy SONCA realizowanej przez zespół prof. Sona Nguyena na Wydziale Matematuki Informatyki i Mechaniki Uniwersytetu Warszawskiego.

PLGRID PLUS *Dziedzinowo zorientowane usługi i zasoby infrastruktury PL-Grid dla wspomagania Polskiej Nauki w Europejskiej Przestrzeni Badawczej.* Projekt uzyskał finansowanie ze środków Europejskiego Funduszu Rozwoju Regionalnego w ramach Programu Operacyjnego Innowacyjna Gospodarka, nr POIG.02.03.00-00-096/10. Projekt jest realizowany przez Konsorcjum PL-Grid, a koordynowany przez ACK CYFRONET AGH. W projekcie tym badałem aspekty optymalizacji komunikacji z bazą MySQL w ramach modułu `accounting`.

Doświadczenie dydaktyczne

W latach 2007 - 2013 wypromowałem ponad 50 prac magisterskich i szereg prac dyplomowych w szeroko rozumianej tematyce baz danych.

Kształcenie młodych kadr

Aleksandra Boniewicz jest doktorantką, którą prowadzę jako promotor pomocniczy. W ramach wspólnie prowadzonych badań nad integracją metod odpytywania struktur rekurencyjnych z systemami odwzorowań obiektowo-relacyjnych, jest współautorką prac [R3], [R6], [R7], [R8]. Przewód doktorski został otwarty na wydziale Matematyki Informatyki i Mechaniki Uniwersytetu Warszawskiego w lutym bieżącego roku. Pani Boniewicz planuje złożyć rozprawę doktorską na przełomie listopada i grudnia 2013 roku. Tytułem rozprawy jest *Optymalizacja warstwy dostępu do danych w aplikacjach korzystających z odwzorowań obiektowo-relacyjnych.*

Michał Gawarkiewicz jest doktorantem z którym prowadzę badania nad integracją technik wspierania zapytań analitycznych z systemami odwzorowań obiektowo-relacyjnych, poprzez wykorzystanie danych nadmiarowych. Do tej pory udało się opublikować następujące prace [P1], [P2] i [F]. Kolejne praca jest w przygotowaniu. Przewód doktorski planuje się otworzyć do końca roku 2013. Planowany termin złożenia rozprawy doktorskiej to czerwiec 2014.

Doświadczenie organizatorskie

W latach 2011 i 2012 brałem udział w pracach komitetu programowego konferencji DTA (Database Theory and Applications), która odbywała się w ramach multikonferencji FGIT (Future Generation in Information Technologies) w Korei Południowej.

W ramach konferencji FedCSIS 2013 zorganizowałem warsztaty PBDA na temat optymalizacji przetwarzania danych w aplikacjach biznesowych.

Staże

Monachium 2003 W okresie luty - lipiec 2003r. byłem na stażu naukowym jako postdoc na Uniwersytecie Monachijskim. Wyjazd był finansowany przez Europejską Fundację Nauki.

Lubbock 2004 W listopadzie 2004r. spędziłem dwa tygodnie na stażu w Texas Tex University, na zaproszenie prof. Mary Neusel.

NEUCA 2011 W czerwcu i lipcu 2011r. byłem na stażu w firmie NEUCA S.A. jako ekspert w zakresie baz danych oraz nowoczesnych technik inżynierii programowania.

Cytowania moich prac

Wyniki doktoratu opublikowane w [M1] zostały zacytowane w pracach [23], [24].

Praca [R1] o przenoszeniu predykatów była cytowana przez trzy publikacje: [5], [11], [25]. Pierwsza praca o integracji zapytań rekurencyjnych z systemami odwzorowań obiektowo-relacyjnych została dostrzeżona w [26], [25], [27]. Tekst [R4] został zacytowany przez [4].

Wyniki uzyskane dla ujednoliconego modelu składu zaprezentowane w [T1] zostały zacytowane przez [4].

W sumie moje prace były cytowane 9 razy. Indeks Hirsha wyliczony z tych cytowań wynosi 2.

Baza Web Of Knowledge wykazuje 4 cytowania zewnętrzne i wylicza indeks Hirsha 2.

Literatura

[1] Bauer, C., King, G.: Java Persistence with Hibernate. Manning Publications Co., Greenwich, CT, USA (2006)

[2] O'Neil, E.J.: Object/relational mapping 2008: Hibernate and the Entity Data Model (EDM). In Wang, J.T.L., ed.: SIGMOD Conference, ACM (2008) 1351–1356

[3] Melnik, S., Adya, A., Bernstein, P.A.: Compiling mappings to bridge applications and databases. ACM Trans. Database Syst. **33** (2008)

[4] Wegrzynowicz, P.: Performance antipatterns of one to many association in hibernate. In: Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on. (2013) 1475–1481

[5] Przymus, P., Boniewicz, A., Burzańska, M., Stencel, K.: Recursive query facilities in relational databases: A survey. In Zhang, Y., Cuzzocrea, A., Ma, J., Chung, K.I., Arslan, T., Song, X., eds.: FGIT-DTA/BSBT. Volume 118 of Communications in Computer and Information Science., Springer (2010) 89–99

[6] Valduriez, P., Boral, H.: Evaluation of recursive queries using join indices. In: Expert Database Conf. (1986) 271–293

[7] Ioannidis, Y.E.: On the computation of the transitive closure of relational operators. In: Proceedings of the 12th International Conference on Very Large Data Bases. VLDB '86, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1986) 403–411

[8] Ioannidis, Y.E., Ramakrishnan, R.: Efficient transitive closure algorithms. In: Proceedings of the 14th International Conference on Very Large Data Bases. VLDB '88, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1988) 382–394

[9] Brandon, D.: Recursive database structures. J. Comput. Sci. Coll. **21** (2005) 295–304

[10] Ordonez, C.: Optimization of linear recursive queries in sql. IEEE Trans. Knowl. Data Eng. **22** (2010) 264–277

[11] Cortesi, A., Halder, R.: Abstract interpretation of recursive queries. In Hota, C., Srimani, P.K., eds.: ICDCIT. Volume 7753 of Lecture Notes in Computer Science., Springer (2013) 157–170

[12] Subieta, K.: Theory and Construction of Object Query Languages [in Polish]. Publishers of the Polish-Japanese Institute of Information Technology (2004)

[13] Subieta, K., Beeri, C., Matthes, F., Schmidt, J.W.: A stack-based approach to query languages. In: East/West Database Workshop. (1994) 159–180

[14] Grefen, P.W.P.J., de By, R.A.: A multi-set extended relational algebra - a formal approach to a practical issue. In: ICDE, IEEE Computer Society (1994) 80–88

[15] Shaw, G.M., Zdonik, S.B.: A query algebra for object-oriented databases. In: ICDE, IEEE Computer Society (1990) 154–162

[16] Subieta, K., Kambayashi, Y., Leszczyłowski, J.: Procedures in object-oriented query languages. In Dayal, U., Gray, P.M.D., Nishio, S., eds.: VLDB, Morgan Kaufmann (1995) 182–193

[17] Wadler, P.: Deforestation: transforming programs to eliminate trees. Theoretical Computer Science **73** (1990) 231 – 248

[18] Johann, P.: Short cut fusion: proved and improved. In: Proceedings of the 2nd international conference on Semantics, applications, and implementation of program generation. SAIG'01, Berlin, Heidelberg, Springer-Verlag (2001) 47–71

[19] Ghrist, R., Muhammad, A.: Coverage and hole-detection in sensor networks via homology. In: Proceedings of the 4th international symposium on Information processing in sensor networks. IPSN '05, Piscataway, NJ, USA, IEEE Press (2005)

[20] De Silva, V., Ghrist, R.: Coordinate-free coverage in sensor networks with controlled boundaries via homology. Int. J. Rob. Res. **25** (2006) 1205–1222

[21] Silva, V.D., Ghrist, R.: Homological sensor networks. Notices of the American Mathematical Society **54** (2007) 2007

[22] Przymus, P., Rykaczewski, K., Wisniewski, R.: Application of wavelets and kernel methods to detection and extraction of behaviours of freshwater mussels. In Kim, T.H., Adeli, H., Slezak, D., Sandnes, F.E., Song, X., Chung, K.I., Arnett, K.P., eds.: FGIT. Volume 7105 of Lecture Notes in Computer Science., Springer (2011) 43–54

[23] Amano, K., Masuoka, A.: Picard–vessiot extensions of artinian simple module algebras. Journal of Algebra **285** (2005) 743 – 767

[24] Amano, K.: Relative invariants, difference equations, and the picard-vessiot theory. (2005)

[25] Burzańska, M.J.: New query rewriting methods for structured and semi-structured databases. (2011)

[26] Grzegorowski, M., Pardel, P.W., Stawicki, S., Stencel, K.: Sonca: Scalable semantic processing of rapidly growing document stores. In Pechenizkiy, M., Wojciechowski, M., eds.: ADBIS Workshops. Volume 185 of Advances in Intelligent Systems and Computing., Springer (2012) 89–98

[27] de la Rosa Perez, J., Suárez, J.L., Sancho-Caparrini, F.: Sylvadb: A polyglot and multi-backend graph database management system. In Helfert, M., Francalanci, C., Filipe, J., eds.: DATA, SciTePress (2013) 285–292

 Piotr Wiśniewski