УДК 004.656

ЕФЕКТИВНА РЕАЛІЗАЦІЯ ЕКОНОМІЧНИХ РОЗРАХУНКІВ В СИСТЕМАХ МАСШТАБУ ПІДПРИЄМСТВА

В.Г. Романенко

Національний Університет «Києво-Могилянська Академія»
Київ, Гарматна, 16, кв. 3
тел. (044) 456 5183; e-mail: Vladam.Romanenko@gmail.com

Робота присвячена дослідженню ефективної реалізації підсистеми фінансово-аналітичних розрахунків у системах масштабу підприємства. Головна мета дослідження – комбінація ефективності та гнучкості. Гнучкість обчислень досягається використанням метаданих для представлення формул. Дані предметної області також моделюються на основі метаданих. Ефективність досягається кешуванням та адаптивним виконанням обчислень. Залежно від результатів дослідження продуктивності обчислення можуть бути виконані на рівні бізнесу або бази даних.

The paper concerns the problem of building effective economical calculations engine for enterprise systems. Two main purposes concerned are effectiveness in terms of performance and flexibility. Flexibility of calculations is achieved by use of metadata to represent formulas. Domain data is modeled using metadata too. Performance is achieved by caching and adaptable calculations execution. Based on performance research calculations can be executed on business or database level.

Вступ

Упровадження систем керування підприємством нині актуальне для України. Чимало підприємств виходять нині на новий рівень управління, на якому їм необхідно мати змогу більш ефективно та оперативно контролювати фінансові та виробничі показники функціонування підприємства.

Важлива складова система керування підприємством є підсистема фінансово-аналітичних розрахунків. Одними з основних вимог, що висуваються до цієї підсистеми, є:

- висока ефективність розрахунків, оскільки доводиться мати справу з великими обсягами даних, що моделюються;
- можливість програмного забезпечення адаптуватися під вимоги користувача, змінювати алгоритми розрахунків відповідно до вимог законодавства та особливостей функціонування бізнес-процесів на конкретному підприємстві.

Традиційні підходи реалізації систем фінансово-аналітичних розрахунків передбачають вбудовану модель розрахунків, яка не може змінюватися користувачем. Для внесення змін до таких систем необхідно залучати програміста, який модифікує алгоритми функціонування та розрахунки фінансово-аналітичної підсистеми під вимоги конкретного замовника. Подібне негнучке рішення призводить до того, що користувачі відмовляються від використання такого негнучкого рішення на користь менш спеціалізованого і можливо не такого зручного, але більш гнучкого і універсального засобу такого як Excel.

У даній роботі досліджується реалізація високоефективного модуля розрахунків, який є базовою складовою частиною фінансово-аналітичної підсистеми. Умови реалізації передбачають побудову такого механізму, який би давав змогу вносити прості зміни до алгоритму розрахунків кінцевому користувачеві без необхідності втручання програміста. Таким чином гнучкість є однією з базових вимог. Іншою вимогою є прийнятні показники швидкості обчислень розрахунків для великих масивів інформації, які є характерними для великих підприємств.

Основні напрямки даного дослідження були зосереджені на таких питаннях:

- аналіз підходів до проектування схеми реляційної бази даних;
- архітектура виконання обчислень, можливість винесення обчислень на рівень СКБД;
- вплив механізмів кешування на підвищення ефективності обчислень.

Вимоги гнучкості та ефективності багато в чому є протилежними, тому головна проблема полягає у необхідності сумістити **ефективність виконання розрахунків** та **гнучкість адаптації** під вимоги замовника. Гнучкість досягається за рахунок того, що всі розрахунки описуються у вигляді формул, які можуть змінюватися користувачем. Формули оперують поняттями предметної області, які також задаються та змінюються користувачем за допомогою блока розрахунків.

Завдання блока розрахунків полягає в описі моделі предметної області бізнес розрахунків та побудови на їх основі змінних формул. Блок розрахунків також забезпечує створення сховища даних, роботу з даними та виконання розрахунків за описаними формулами. Блок розрахунків включає інтерфейс користувача («Редактор метаданих»), який забезпечує редагування моделі предметної області та формул. Для тестування формул редактор також надає можливість вводу даних та відображення результатів розрахунків.

Блок розрахунків виконує наступні функції:

- опис користувачем моделі предметної області;
- формування сховища даних на основі моделі;

- наповнення користувачем формул розрахунків;
- опис користувачем формул розрахунків;
- оптимізоване виконання розрахунків за формулами на базі вхідних даних;
- збереження результатів розрахунків у сховищі даних.

Модель предметної області

Модель обчислень предметної області включає ряд основних сутностей (рис. 1).

Словники – містять переліки основних об'єктів, стосовно яких здійснюється фінансовий аналіз, та їх складових. Приклади словників: продукція, контрагенти, співробітники, ринки збуту тощо.

Вхідні статті – представляють собою переважно векторні дані в розрізі часу, які відображають різноманітні показники виробництва. Приклади вхідних статей: продаж, ціна, кредити тощо.

Розрахункові статті – векторні дані в розрізі часу, що відображають показники, розраховані на основі вхідних даних.

Діаграма сутність-зв'язок:

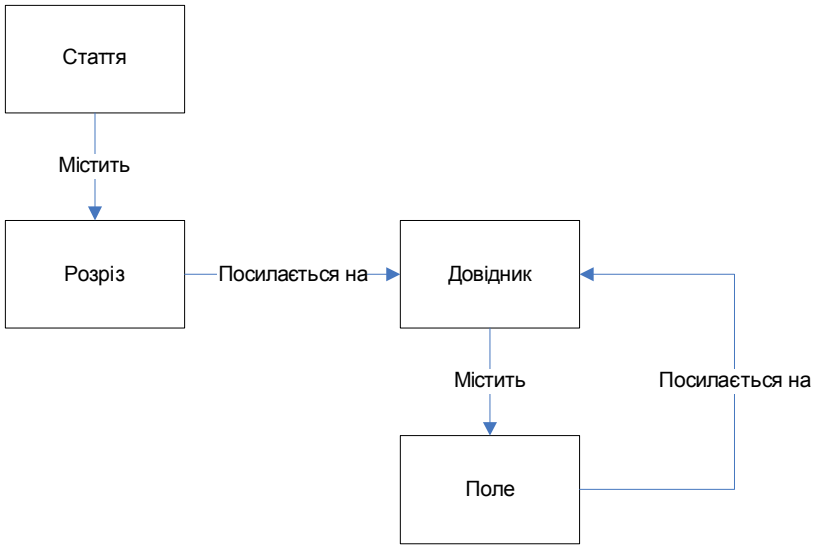

Рис. 1

Модель предметної області, яка використовується у блоці розрахунків, досить проста і за рахунок цього на її основі можуть бути ефективно реалізовані налагоджувані розрахунки. Основним поняттям, яким оперує блок розрахунків є ***стаття,*** яка є вектором значень, розподілених у часі, де на кожен день визначено одне значення статті. Наприклад, стаття «Ставка податку» (табл. 1).

Таблиця 1.

Дата	1.01.2005	2.01.2005	3.01.2005	4.01.2005	5.01.2005	…	31.12.2006
Значення	0,20	0,20	0,20	0,20	0,20		0,20

Значення статті може також характеризуватися ***розрізами***. Наприклад: стаття «Прогноз продажу» має два розрізи – «товар» і «ринок» (табл. 2).

Таблиця 2.

Товар	Ринок	1.01.2005	2.01.2005	3.01.2005	4.01.2005	…	31.12.2006
Товар 1	Ринок 1	10	20	12	15		30
Товар 1	Ринок 2	15	16	17	13		20
Товар 2	Ринок 1	12	18	14	15		15
Товар 2	Ринок 2	16	22	20	18		22

Значення розрізів беруться із ***довідників***. Довідник – це масив записів. Кожний запис має ряд ***полів***. Наприклад: довідник «Товар» має поля «Код», «Найменування», «Тип», «Одиниця виміру» (табл. 3).

Таблиця 3.

Код	Найменування	Тип	Одиниця виміру
100	Товар 1	Продукти споживання	Кг
101	Товар 2	Побутова техніка	Шт

Поля довідника можуть посилатися на поля інших довідників. Наприклад, поле «Тип» у довіднику «Товар» посилається на довідник «Тип товару», а поле «Одиниця виміру» посилається на довідник «Одиниці виміру».

Загальна архітектура та схема бази даних

Блок розрахунків побудований за принципом трьохшарової архітектури (рис. 2).

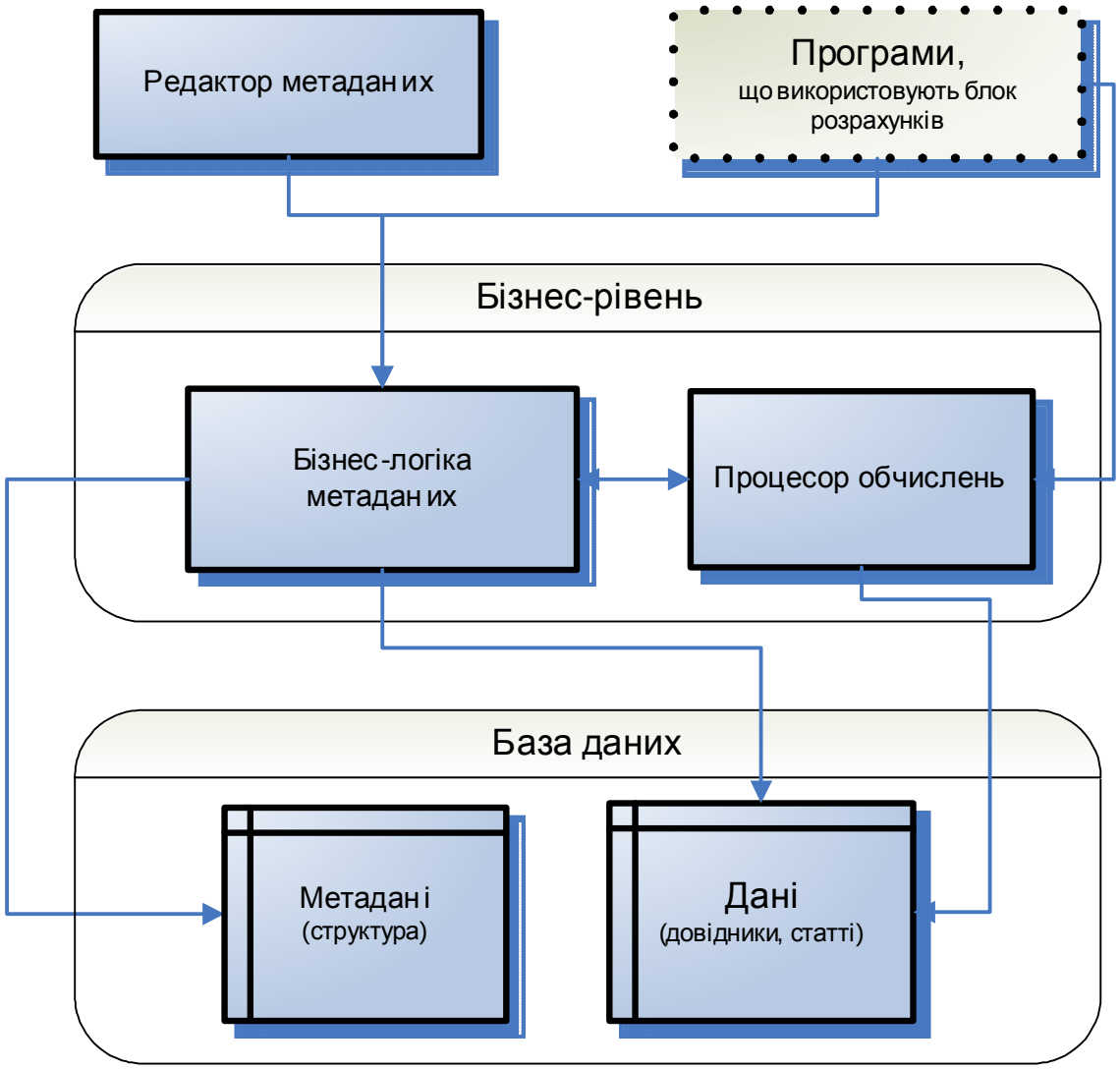

Рис. 2

Для отримання прийнятної гнучкості інструментального засобу, кінцевий користувач повинен мати змогу змінювати структуру даних відповідно до своїх вимог. Для отримання високих показників ефективності, структура даних має бути оптимізована на рівні бази даних.

Досліджено два альтернативні підходи до реалізації бази даних. Перший підхід передбачає створення окремої таблиці для кожної статті та словника. Таким чином можна легко створювати схеми даних оптимізовані під конкретні моделі користувачів. Недоліками такого підходу є велика кількість таблиць у базі даних, необхідна для реалізації усіх статей. Здійснене дослідження показало, що цільова база даних ефективно працює з кількістю таблиць порядку декількох тисяч, що є цілком прийнятним для системи масштабу підприємства, яка потребує порядку 2 – 10 тисяч статей. Даний підхід є зручним для вибірок даних з бази, оскільки для отримання даних за однією статтею досить зробити вибірку з однієї таблиці. Це значною мірою полегшує розробку блоку розрахунків. Недоліком такої реалізації є складність модифікації схеми даних після наповнення бази, оскільки повторна генерація схеми таблиці значно ускладнює можливість міграції уже внесених даних.

Другий підхід полягав у збереженні даних усіх статей в одній таблиці універсальної структури. При такому підході кількість таблиць у базі є значно меншою, але через те що всі операції, як вибірки даних, так і вставки, пов'язані з однією таблицею, це приводить до значних спадів продуктивності через затримки виконання операцій. Одна таблиця з усіма даними стає слабким місцем і унеможливлює розпаралелення і одночасний доступ багатьох користувачів.

У результаті досліджень обрано перший підхід.
За цим підходом спроектовано логічну схему таблиць для збереження метаданих (рис. 3).

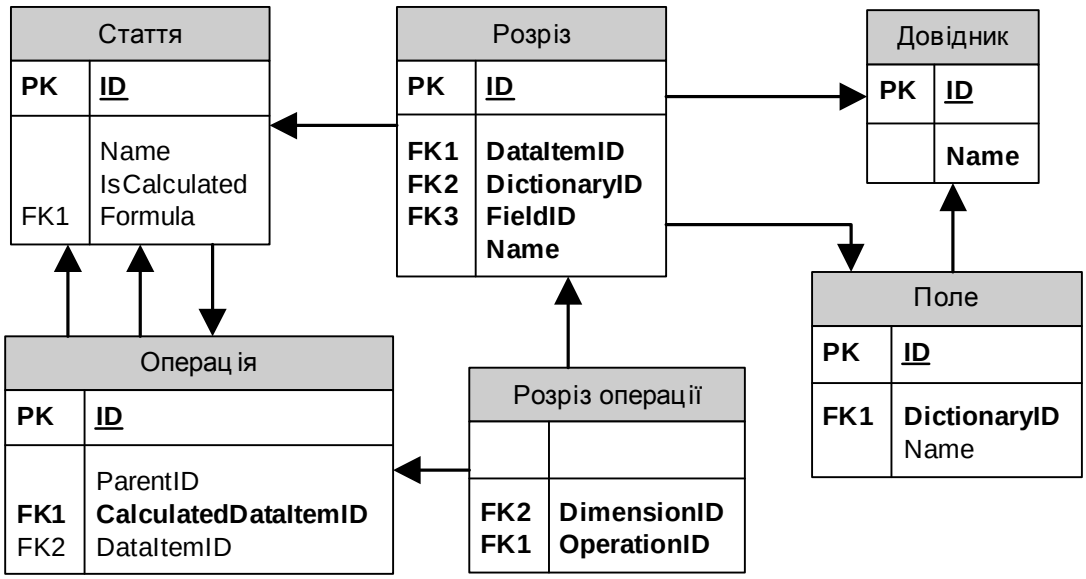

Рис. 3

Реалізація розрахунків

В алгоритмі розрахунку можна виділити наступні складові частини:

- алгоритм генерації дерева розрахунку за описом XML;
- алгоритм ініціалізації контексту параметрів для вузлів графа;
- алгоритм обробки циклічних посилань;
- алгоритм оптимізації графа розрахунку для використання попередньо розрахованих даних;
- алгоритм оптимізації графа розрахунку з використанням пакетів SQL;
- алгоритм виконання розрахунків.

Схема алгоритму розрахунку показано на рис. 4.

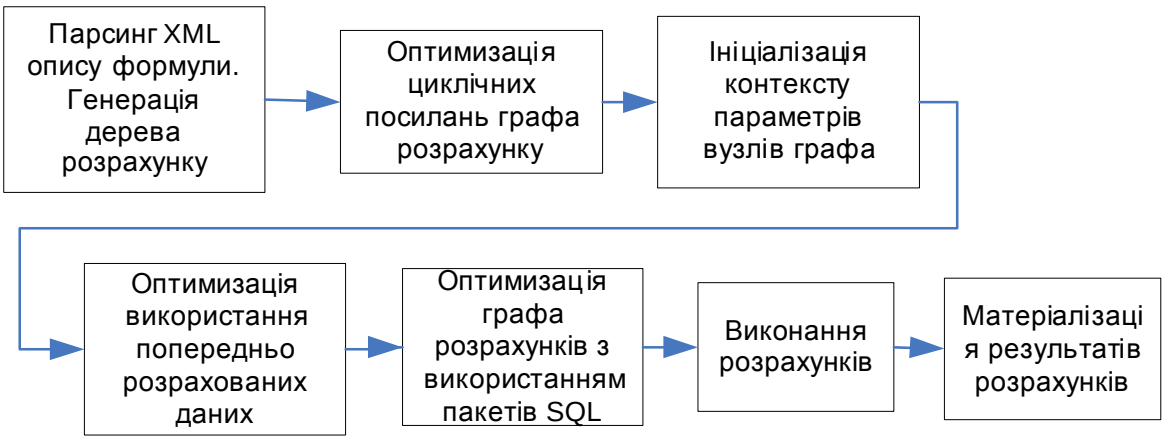

Рис. 4

У реалізації блоку розрахунків досліджено два підходи: виконання обчислень на рівні бізнес-логіки та на рівні бази даних. Реалізація обчислень на рівні бізнесу є дещо простішою порівняно з написанням складних SQL-запитів. Крім того, при реалізації обчислень на рівні бази даних значною проблемою є збереження тимчасових даних та передача результатів за деревом обчислень для використання в якості вхідних даних наступними операторами. З іншого боку, реалізація обчислень на рівні бізнес-логіки вимагає завантаження великих обсягів даних, що негативно впливає на швидкодію.

На рис. 5 показано приклад графа розрахунку статті, яка у якості одного з операндів операції використовує іншу розрахункову статтю. Сірим кольором виділений вузол, який відповідає за інфраструктуру і не несе ніякої логічної функції.

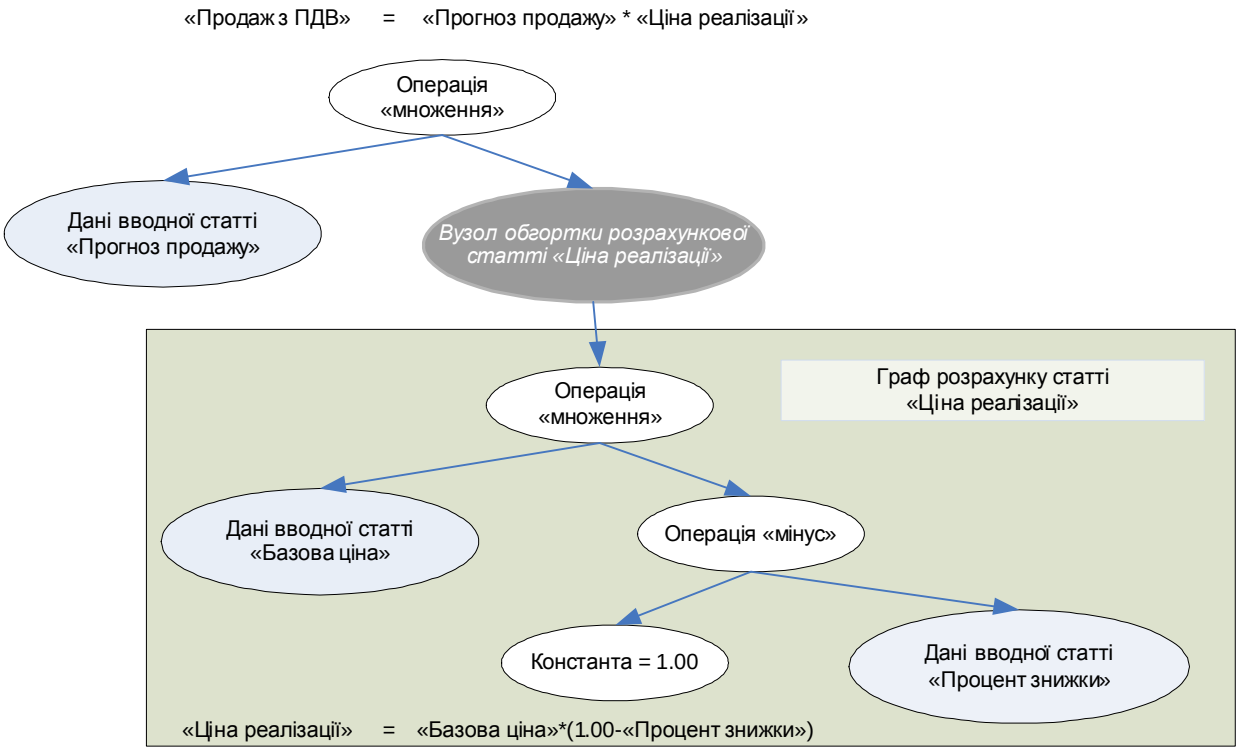

Рис. 5

Під калькуляцією графа розрахунку розуміємо процес виконання математичних операцій над вузлами графа, а також виконання операцій пов'язаних з забезпеченням необхідної інфраструктури. У реалізованому підході результатом виконання операцій буде відповідний SQL запит, який виконує безпосередньо математичні операції. Для відображення даних у програмі використовується матеріалізація, тобто виконання запиту, та отримання даних. Таким чином, у результаті виконання калькуляцій над деревом розрахунку отримаємо SQL запит. Наприклад, у результаті калькуляцій над деревом, що представлене на рис. 5 побудуємо наступний SQL запит:

```
SELECT t0.Value * t1.Value AS Value, t0.Date as Date, t0.TimeStepID as TimeStepID,
        t0.[Field107], t0.[Field138], t0.[Field206], t0.[Field15], t0.[Field143], t0.[Field157], t1.[Field94]
FROM
(
        SELECT Value, TimeStepID, Date, DataItem34.Field157, DataItem34.Field206,
                DataItem34.Field143, DataItem34.Field15, DataItem34.Field138, DataItem34.Field107
        FROM DataItem34
        WHERE Date between '2005-01-01 00:00:00' and '2006-12-31 00:00:00' AND TimeStepID = 5
) AS t0,
(
        SELECT t0.Value, t0.Date, t0.TimeStepID,
                t0.[Field157], t0.[Field138], t0.[Field94], t0.[Field107], t0.[Field143], t0.[Field15]
        FROM
        (
                SELECT t0.Value * t1.Value AS Value, t0.Date as Date, t0.TimeStepID as TimeStepID,
                        t0.[Field157], t0.[Field94], t0.[Field143], t0.[Field15], t1.[Field138], t1.[Field107]
                 FROM
                (
                        SELECT TimeStepID, Value, Date, DataItem12.Field15,
                                DataItem12.Field94, DataItem12.Field157, DataItem12.Field143
                        FROM DataItem12
                        WHERE Date between '2005-01-01 00:00:00' and '2006-12-31 00:00:00'
                                AND TimeStepID = 5
                ) AS t0,
                (
                        SELECT t0.Value - t1.Value AS Value, t1.Date as Date, t1.TimeStepID as TimeStepID,
                                t1.[Field138], t1.[Field107], t1.[Field157]
                        FROM
                        (
                                SELECT 1.00 AS Value
                        ) AS t0,
```

```
                    (
                        SELECT TimeStepID, Value, Date,
                            DataItem36.Field107, DataItem36.Field138, DataItem36.Field157
                        FROM DataItem36
                        WHERE Date between '2005-01-01 00:00:00' and '2006-12-31 00:00:00'
                            AND TimeStepID = 5
                    ) AS t1
                    WHERE 1=1
                ) AS t1
                WHERE 1=1  AND t0.Date = t1.Date
                    AND ((t1.[Field157] IS NULL AND t0.[Field157] IS NULL) OR (t1.[Field157] = t0.[Field157]))
        ) AS t0
        WHERE  t0.Date BETWEEN '2005-01-01 00:00:00' AND '2006-12-31 00:00:00'
) AS t1
WHERE 1=1  AND t0.Date = t1.Date
AND ((t1.[Field157] IS NULL AND t0.[Field157] IS NULL) OR (t1.[Field157] = t0.[Field157]) )
AND ((t1.[Field138] IS NULL AND t0.[Field138] IS NULL) OR (t1.[Field138] = t0.[Field138]) )
AND ((t1.[Field107] IS NULL AND t0.[Field107] IS NULL) OR (t1.[Field107] = t0.[Field107]) )
AND ((t1.[Field143] IS NULL AND t0.[Field143] IS NULL) OR (t1.[Field143] = t0.[Field143]) )
AND ((t1.[Field15] IS NULL AND t0.[Field15] IS NULL) OR (t1.[Field15] = t0.[Field15]) )
```

Моделювання

Для дослідження різних підходів до реалізації обчислень створено спрощену модель фінансово-аналітичної підсистеми, яка містить 3000 статей вхідних даних та 5142 обчислюваних статей. Побудовано граф розрахунків, який об'єднує усі статті в розрахункову статтю верхнього рівня, так званий „баланс підприємства". Підготовлено два набори вхідних даних – для моделювання обчислень протягом 4 місяців роботи підприємства, та моделювання протягом 3 років.

Модель обчислень включає складні проміжні обчислення, що включають у себе окрім звичайних операцій +, -, *, / також зсув значень даних відносно часу. Зразок частини графа обчислень показано на рис. 6. Вузли цього графа зазначають операції, а листові вузли CIN – вибірки вхідних даних.

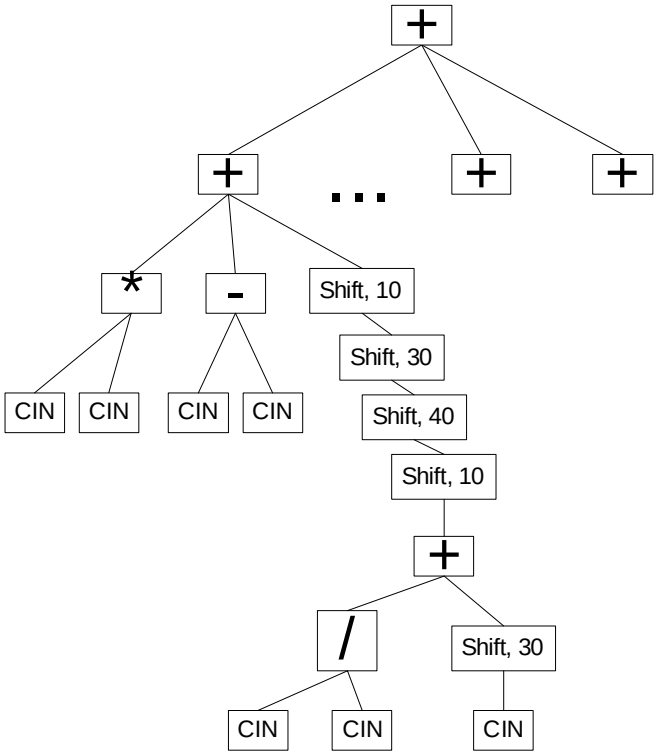

Рис. 6

Чисельні результати здійснених експериментів наведено у табл. 4.

Таблиця 4.

Параметр	Скорочене моделювання		Повне моделювання	
	Бізнес	База даних	Бізнес	База даних
Кількість вхідних статей	3000	3000	3000	3000
Кількість розрахункових статей	5142	5142	5142	5142
Проміжок моделювання	4 місяці	4 місяці	3 роки	3 роки
Час виконання	**23.38**97268	**18.21**80504	**39.50**00000	**49.07**81250
Обсяг пам'яті	27 272 068	14 127 412	161 809 832	87 511 908

Графічне представлення результатів показано на рис. 7.

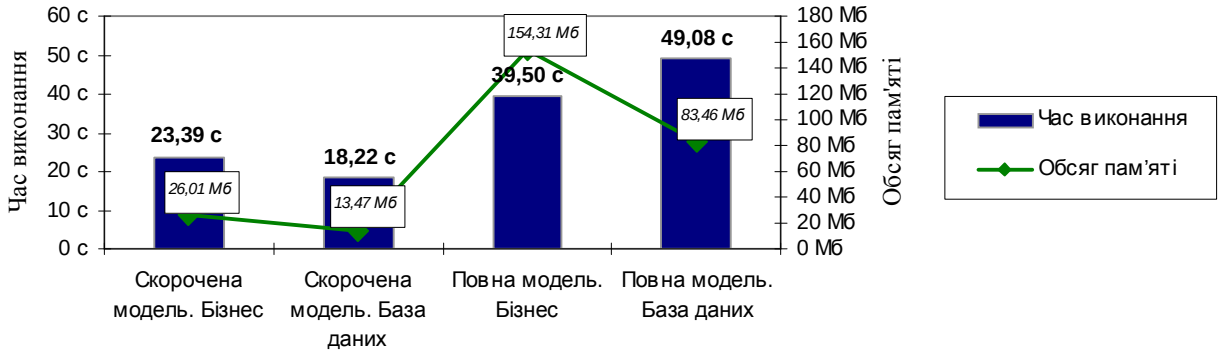

Рис. 7

Висновки

Дослідження підходів до реалізації розрахунків на великих обсягах показало, що доцільним є виконання низькорівневих операцій на рівні бази даних. Слід зауважити, що різкої різниці між часом виконання в обох підходах немає. У випадку великих обсягів даних виконання на рівні СКБД дало навіть гірші результати за часом виконання. Але цей підхід має незаперечну перевагу за ефективним використанням пам'яті.

Тому доцільним вбачається побудова гібридного підходу, коли прості та низько рівневі розрахунки виконуються на базі даних, а складні алгоритми виконуються на рівні бізнес-логіки. Крім спрощення реалізації та відлагодження алгоритмів виконання розрахунків на бізнесі дає також змогу абстрагуватися від конкретної СКБД, що дозволить створювати більш гнучкі системи, які можливо буде переносити на інші платформи.

Подальшим розвитком досліджень є більш глибоке вивчення оптимізації структури бази даних, які можуть вплинути на швидкодію виконання обчислень. Також необхідно дослідити підходи до реалізації кешування обчислених даних. Передбачається також створення складніших моделей розрахунків, які б оперували багатопараметричними даними, які більш повно відображають реальні економічні показники, що застосовуються у розрахунках.

1. *Александров А.* Оперативный бизнес-анализ. – "Открытые системы", 2005, № 12 (http://www.osp.ru/os/2005/12/018.htm)
2. *Черняк Л.* Мониторинг бизнес-процессов. – "Открытые системы", 2005, № 10 (http://www.osp.ru/os/2005/10/024.htm)
3. *Шишков В.* Развитие бизнеса компании на основе анализа создания стоимости. – Финансовая газета, 2003, № 39 (615) сентябрь.
4. *Фаулер М.* Архітектура корпоративних программных приложений. – М.: Вільямс, 2004. – 544 с.
5. *Гамильтон Б.* - ADO.NET Сборник рецептов. Для профессионалов. – СПб: "Питер", 2005. – 576 с.
6. *Goodson J.* Optimized ADO.NET, http://www.theserverside.net/articles/showarticle.tss?id=OptimizingADONET.