БАЗИ ДАНИХ, БАЗИ ЗНАНЬ ТА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

О. В.Тегельман

Національний авіаційний університет

кафедра комп'ютерних інформаційних технологій

Науковий керівник – Харченко О.Г. (д.т.н., професор)

Рефакторінг баз даних – еволюційний підхід до розробки ERP-систем

Вступ

У своїй книзі «Рефакторінг»[1]Мартін Фаулер зазначав, що якщо є можливість піддати зміні вихідний код ПЗ, то повинна існувати можливість провести рефакторінг над схемою бази даних(БД). Але він відмітив, що рефакторінг БД – значно складніша задача, ніж рефакторінг коду, оскільки базам даних властива висока ступінь зв'язаності.

Рефакторінг БД є одним з методів еволюційної розробки(ExtremeProgramming - XP, RationalUnifiedProcess – RUP, Scrum тощо), що дуже широко застосовується на даному етапі розвитку індустрії інформаційних технологій(IT). Він передбачає розвиток існуючої схеми БД шляхом внесення невеликих змін з метою підвищення якості проекту БД без змін її семантики.

Головною ціллю даної статті є дослідження поняття рефакторінгу БД, операцій рефакторінгу, які найчастіше застосовуються, а також показати, що рефакторінг БД потрібен так само, як і рефакторінг програмного коду. Також в даній роботі зроблено акцент на еволюційній розробці, яка застосовується при проектуванні, реалізації та впровадженні схеми БД.

1. Рефакторінг БД як основа еволюційного підходу до проектування інформаційних систем

На даному етапі розвитку IT-індустріїв розробці БДдосить широко розпочинають використовуватися еволюційні методи. Це означає, що схема БД не повинна відразу проектуватись на початкових етапах проекту,а повинна поступово нарощуватись протягом всього періоду здійснення проекту. Це дає можливість оперативно реагувати на зміни, що вносяться в вимоги замовника, та вносити ці зміни в проект БД.Такий еволюційний підхід зараз широко використовується при розробці БД.

Переваги еволюційного підходу полягають в наступному[2]:

- Мінімізація невиправданих затрат
- Попередження необхідності в суттєвих доробках
- Постійна впевненість в наявності робочої системи
- Постійна впевненість в тому, що існуючий на даний момент проект БД володіє найвищою якістю
- Зменшення загальної трудомісткості

До методів еволюційної розробки відносять :

- Рефакторінг БД
- Еволюційне моделювання даних
- Регресійне тестування БД

Рефакторінг БД – це зміна в схемі БД, яка покращує її проект, зберігаючи незмінною поведінкову та інформаційну семантику БД. Рефакторінгу можуть бути піддані або структурні аспекти схеми БД, такі як визначення таблиць або представлень, або функціональні аспекти, такі як процедури, що зберігаються, та тригери. При проведені рефакторінгу схеми БД доводиться не лише перевизначити саму схему, але вносити зміни й у зовнішню систему, яка«прив'язана» до схеми БД.

Для того, щоб можна було легко вносити зміни в існуючу схему БД, які не будутьтягнути за собою порушення роботи БД або програм, які використовують її потрібно мати можливість перевірити чи всі програми продовжують функціонувати після внесення необхідних змін. Іншими словами, потрібно передбачити мож-

ливість проведення повного регресійного тесутвання. Якщо при цьому буде виявлено, що відбулось порушення роботи якогось компоненту, то повинна бути можливість або усунути помилку, що виникла, або здійснити відкочування системи.

Модель даних, тести БД, дані, що передбачені для випробувань, та інші об'єкти є важливими артефактами проекту, якими потрібно керувати точно так само, як і будь-якими іншими. Все повинно використовуватись для того, щоб при виникненні помилок в роботі після внесення змін, можна було відмінити всі внесені зміни. Тому для забезпечення успішного проведення рефакторінгу БД потрібно поставити наступні об'єкти під контроль засобами керування конфігураціями : сценарії на мові визначення даних(МВД), сценарії завантаження/вибірки даних, файли з визначеннями моделей даних, метадані об'єктно-реляційного відображення, довідкові дані, визначення процедур, що зберігаються, та тригерів, визначення представлень, обмеження цілісності посилань, тестові дані, сценарії створення тестових даних, тестові сценарії.

2. ТЕХНОЛОГІЯ РЕФАКТОРІНГУ БД

2.1. Операції рефакторінгу

Операції рефакторінгу потрібно проводити, коли наявні наступні аномалії: багатоцільові стовпчики, багатоцільові таблиці, надлишкові дані, таблиці з дуже великою кількістю стовпчиків, таблиці з дуже великою кількістю рядків, багатозначні таблиці, наявність нереалізованих змін.

Деталізована характеристика цих аномалій приведена в таблиці 2.1.

Таблиця 2.1. Характеристики аномалій БД

Недолік	**Опис**
Багатоцільові стовпчики	Якщо стовпчик використовується в деяких цілях, то велика ймовірність того, що існує додатковий код, який слугує для забезпечення використання вихідних даних по призначенню
Багатоцільові таблиці	Якщо яка-небудь таблиця використовується для збереження даних о сутностях декількох різних типів, то її поява в базі даних по всій ймовірності є наслідком упущення в проекті
Надлишкові дані	Наявність надлишкових даних – це дуже важлива проблема в повсякденно базах даних, що перебуваються в експлуатації, оскільки при збереженні одних і тих самих даних в декількох місцях виникає ймовірність порушення сумісності або цілісності
Таблиці з дуже великою кількістю стовпців	Якщо в таблиці є дуже велика кількість стовпчиків, то це вказує на те, що в таблиці будуть зберігатись або зберігаються дані, які відносять до різних сутностей
Таблиці з дуже великою кількістю рядків	Наявність таблиць з великою кількістю записів слугує ознакою сповільнення швидкості роботи БД, виконання запитів, обробки даних тощо
Багато значні стовпчики	Це стовпчики, в яких в різних позиціях представлено декілька різних фрагментів інформації.
Наявність нереалізованих змін	Якщо зміни в базі даних довго не проводились, то це може слугувати наглядною ознакою того, що схему БД потрібно піддати рефакторінгу

Рефакторінг БД –сприяє покращенню проектуБД(під проектом розуміється все те, що може бути включеним в схему БД: таблиці, представлення, тригери, хранимі функції, послідовності тощо) і разом з цим забезпечується збереження функціональної та інформаційної семантики БД. Іншими словами, проведення операцій рефакторінгу не повинно приводити до додавання нової функціональності або порушення роботи уже існуючої, а також не повинно мати своїм наслідком додавання нових даних або змінювати сенс вже існуючих.

Операції рефакторінгу БД концептуально є іншими, ніж операції рефакторінгу програмного коду, оскільки при проведені рефакторінгу програмного коду необхідно турбуватись лише про збереження функціональної семантики, а при здійсненні операцій рефакторінгу БД пот-

рібно також забезпечити і збереження інформаційної семантики. Ускладнення операцій рефакторінгу БД може бути обумовлене наявністю великої кількості зв'язків, що підтримуються архітектурою даних. Зв'язаність – це міра залежності між двома об'єктами. Чим тісніше два об'єктазв'язані, тим більша ймовірність того, що при внесенні зміни в одному з них прийдеться вносити зміни й в іншому. Найпростіше рефакторінг БД проводити тоді, коли зі схемою працює лише одна зовнішня програма. Більш складними є ситуації, коли до БД вже звертається не одна, а декілька зовнішніх програм.

Проводячи операції рефакторінгу БД , потрібно добиватись того, щоб зберегти інформаційну та функціональну семантику. Іншими словами, в схему не повинно бути щось додане, або видалене.

Під **інформаційною семантикою** розуміється сенс інформації, що знаходиться в БД, з точки зору користувача цієї інформації. Виконання умови по збереженню інформаційної семантики означає, що в при будь-якому зміненні значень даних що зберігаються в деякому стовпчику, клієнти, що використовують цю інформацію, не повинні відчувати які-небудь негативні наслідки таких змін.

Що стосується **функціональної семантики**, ціль її збереження полягає в тому, щоб всі функціональні засоби(тригери, хранимі функції тощо) залишились незміненими. Це означає, що весь вихідний код, що використовується дляроботи зі частинами схеми БД, що змінились, повинен бути перероблений в ціляхнадання тих же функціональних можливостей, що і до внесення змін. При цьому необхідно враховувати те, що операції рефакторінгу БД представляють собою підмножину перетворень БД. Перетворення БД може передбачати або не передбачати зміни в семантиці, а проведення операцій рефакотрінуБД не повинно тягнути за собою зміну семантики.

Розрізняють шість категорій операцій рефакторінг БД. Це :

- **Операції рефакторінгу структури**– зміна у визначенні однієї або декількох таблиць або представлень
- **Операції рефакторінгу якості даних** – зміни, що дозволяють підвищити якість даних, що зберігаються в таблиці
- **Операції рефакторінгу цілісності посилань** - зміни, які гарантують, що вказані в посиланні рядки існують в іншій таблиці, і дозволяють забезпечити видалення належним чином рядків, які стають більше не потрібними
- **Операції рефакторінгу архітектури** – зміни, що сприяють в цілому покращення взаємодії зовнішніх програм з базою даних
- **Операції рефакторінгу** методів – внесення змін в храниму процедуру, функцію або тригер, що сприяють покращення їх якості
- **Перетворення, що відрізняються від рефакторінгу** – зміни в схемі БД, що приводять до зміни її семантики

2.2. Процес рефакторінгу БД

На рис. 2.1 зображено блок-схему, що ілюструє процес рефакторінгу БД. Зазвичай цей процес включає наступні дії, що виконуються по ітеративному принципу: перевірка прийнятності розглядуваної зміни операції рефакторінгу БД; вибір найбільш підходящої операції рефакторінгуБД; позначення як застарілої вихідної схеми БД; тестування до виконання, під час виконання та після виконання операцій рефакторінгу;внесення змін в схему БД; переміщення вихідних даних; внесення змін в програми, що мають доступ до БД; виконання регресійних тестів; керування роботами, які виконуються згідно версій; публікація відомостей про проведення рефакторінгу.

Перш за все потрібно визначити , чи повинна бути проведена операція рефакторінгу. Для цього потрібно розглянути три наступні проблеми:

1. Чи має сенс операція рефакторінгу, що пропонується – дуже часто буває, що розробники системи, яка взаємодіє з БД, вважають, що потрібно щось в схемі БД змінити. В таких випадках, вони повинні звертатись до адміністраторів або спеціалістів в області БД, які повинні приймати рішення, чи потрібно проводити рефакторінг.
2. Чи повинні бути виконані зміни негайно–якщо рефакторінг не є критичним, то тоді його проведення можна буде відкласти на пізніший час і виконати ті зміни, які в даний момент часу є критичними для створюваної системи

Чи виправдовує результат затрачені зусилля – адміністратор БД або спеціаліст в цій сфері повиненоцінити наслідки проведення рефакторінгу, щоб не виявилось, що виконана процедура не ефективна, а в результаті змін була отримана некоректна схема БД.

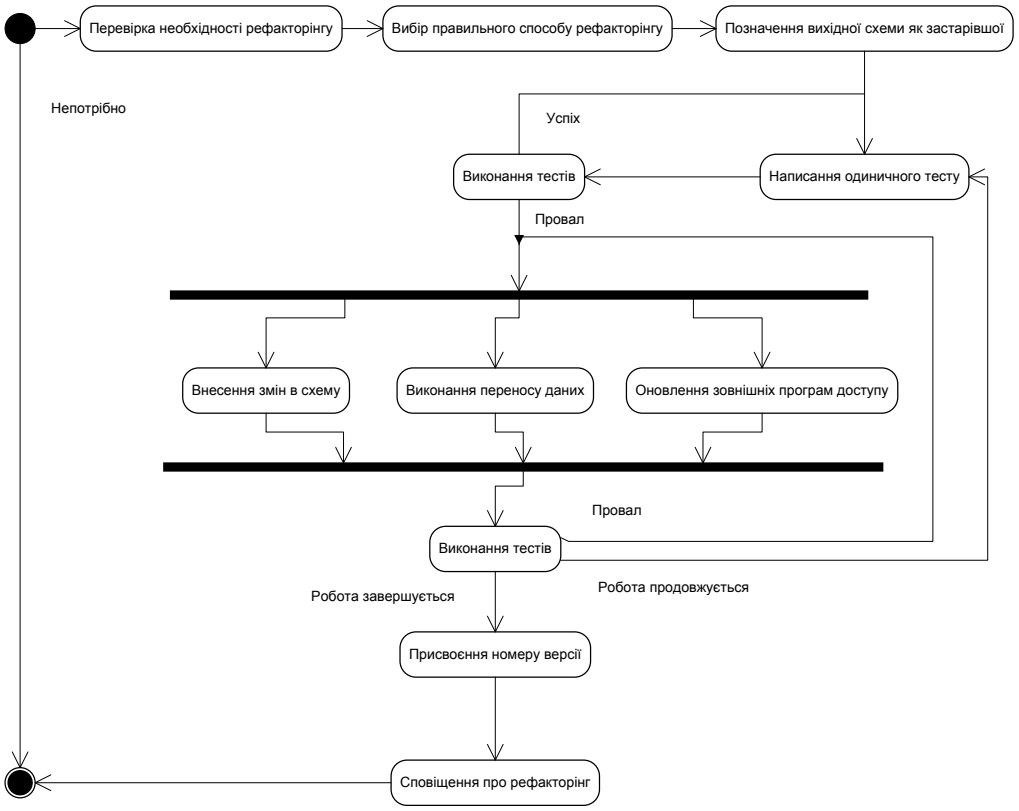

Рис. 2.1. Процес рефакторінгу БД

Після цього необхідно вибрати необхідні операції рефакторінгу.

Якщо доступ до БД мають багато програм, то може виникнути потреба передбачити перехідний період, на якому паралельно підтримуються як вихідна, так і нова схеми, що дозволить групам розробників поступово проводити рефакторінг як БД та і коду системи до її повторного розгортання. Як правило перехідний період продовжується на протязі декількох кварталів, або років.

Для перевірки того, що зміни , які були внесені в схему БД правильні, необхідно впевнитися в тому, що база даних як і в минулому коректно працює із зовнішніми програмами. Для цього потрібно провести тестування до проведення, під час проведення та після проведення рефакторінгу, в процесі якого вирішити наступні задачі : перевірку схеми БД; перевірку способів використання схеми БДу зовнішніх програмах; перевірка правильності переносу даних; перевірка коду зовнішніх програм.

Тестування до проведення рефакторінгу необхідне для того, щоб впевнитись, що в схемі БД ніяких більше змін не потрібно вносити, окрім запланованих.

Тестування під час виконання рефакторінгу дає можливість скоротити час, який може бути потрібним для внесення додаткових змін в схему БД. Це тестування дає можливість перевірити відповідність схеми БД до вимог під час процесу рефакторінгу й ефективно реагувати на незаплановані зміни в інформаційній або функціональній семантиці схеми БД.

Тестування після проведення рефакторінгу дає можливість впевнитись в тому, що схема БД функціонує коректно, зміни, що були на неї накладені, не вплинули на роботу з БД зовнішніх програм.

Для модифікації схеми БД необхідно написатискрипти для зміни БД у відповідності до операції рефакторінгу, що повинна бути проведена. В основному – це сценарії. Для того, щоб вдало виконати модифікацію БД потрібно використовувати невеликі сценарії. Це дає змогу простіше супроводжувати сценарії, дасть можливість виконувати рефакторінг у відповідності до певного порядку по відношенню до БД. Також це дає змогу контролювати наявність різних версій схем БД і по відношенню до них застосовувати в разі потреби різні операції рефакторінгу, що подані в сценарії.

2.3. Особливості процесу рефакторінгу БД

Під час проведення рефакторінгу зазвичай виникає потреба застосувати до БД декілька змін одночасно. Однакна практиці проведення подібних перетворень є досить складним процесом, який більш ризикований порівняно з поетапним виконанням окремих вказаних операцій. Якщо в результаті проведення невеликих змін буде виявлено яке-небудь порушення в роботі, то досить велика ймовірність того, що причину цього порушення можна буде легко знайти.

Під час розробки програмного забезпечення доводиться застосовувати до схеми БД сотні операцій рефакторінгу чи перетворень. Багато цих операцій рефакторінгу базуються на інших операціях рефакторінгу. Тому повинен бути передбачений певний спосіб позначення всіх операцій рефакторінгу, а також позначення будь-яких залежностей між ними.

Найчастіше для ідентифікації використовуються три підходи : номер версії, дата та час, унікальний ідентифікатор.

Номер версії представляє собою як правило ціле число, яке присвоюється за допомогою інструментального засобу видачі версій програмного забезпечення після кожної компіляції програми та успішного виконання всіх модульних тестів. Ця стратегія є однією з найпростіших для рефакторінгу БД. Операції при цьому можуть розглядатись як послідовна черга змін, що застосовуються в порядку, який визначається номером версії. Версія БД безпосередньо при цьому пов’язана з версією програмного продукту.

Відмітка дати та часу представляє собою присвоєння операції рефакторінгу поточної дати та/або часу. Це теж досить проста стратегія, а керування операціями рефакторінгу організовується у вигляді послідовної черги.

Унікальний ідентифікатор представляє собою послідовно прирощуване значення, яке присвоюється операції рефакторінгу. Але при застосуванні даної стратегії повинен існувати певний механізм синхронізації операцій рефакторінгу БД з відповідними до них релізами програмного продукту.

Великі зміни в БД, такі як реалізація загальної стратегії переходу до застосування сурогатних ключів у всіх таблицях або впровадження однозначної стратегії іменування у всій БД, повинні здійснюватись у виді ряду невеликих операцій рефакторінгу.

Адміністратор БД повинен виділити для проведення операцій рефакторінгу перехідний період, який повинен бути достатнім й для всіх інших груп розробників програмного продукту. Найбільш простим є такий підхід, який передбачає сумісне вироблення загального рішення про застосування однакового перехідного періоду для різних категорій операцій рефакторінгу, а в подальшому –неухильне застосування такого перехідного періоду.

Основним недоліком такого підходу є те, що він потребує прийняття найбільш довготривалих перехідних періодів з усіх можливих. Гостроту цієї проблеми можна усунути, видаляючи ті частини схеми БД, які більше не використовуються, навіть не дивлячись на те, що перехідний період міг ще й не закінчитись, при цьому потрібно узгоджувати рішення про застосування більш короткого перехідного періоду з групою контролю над внесенням змін в БД або проводячи безпосереднє узгодження з іншими групами розробників.

2.4. Особливості рефакторінгу БД для ERP-систем

Проведення рефакторінгу БД для ERP-систем не є одноманітною задачею. Зазвичай процес рефакторінгу триває від декількох тижнів до декількох місяців. Під час нього всі роботи ведуться не над робочою схемою БД, а над її копією. Це дає можливість зажди бути впевненим в наявності робочої схеми.

На початковому етапі проведення рефакторінгу створюється резервна копія робочої схеми БД. Це дає можливість після завершення рефакторінгу проводити тестування написаних скриптів, які будуть накладатись на схему БД, що знаходиться в експлуатації. В основному це використовується для перевірки коректності накладення змін на структуру схеми БД, тобто на таблиці, тригери, представлення тощо.

Після створення резервної копії БД відбувається розгортання схеми на стороні розробників. На цю схему розробники будуть накладати зміни. Під час накладання змін ведеться певна таблиця, в якій записується вся інформація про зміни, які були внесенні до БД. Приклад таблиці приведено нижче. Зазвичай вона включає в себе наступну інформацію : порядковий номер внесеної зміни; загальний опис зміни, що була зроблена; інформація про розробника, що вніс зміну; скрипт накладеної зміни; дата.

ID	Description	Script	Developer	Date
1	Зміна назви стовпчика …	Alter table t1 alter column c1	Petro Petrov	10.10.2010
2	Видалення таблиці	Drop table	Ivan Ivanov	11.10.2010
…	…	…	…	…

На завершальному етапі проведення рефакторінгу БД для ERP-систем формується загальний файл зі скриптами, які використовуються для накладення змін на схемі БД, що перебуває в експлуатації.

Після формування файлу зі скриптами, відбувається його тестування. Для цього роблять резервну копію структури БД, на яку накладались зміни(наприклад, на виході отримуємо файл_1 зі SQL-конструкціями для створення БД). Після цього розгортається схема БД, яка перебуває в експлуатації. На цю схему накладають всі зміни, що знаходять в файлі зі скриптами. Далі створюється резервна копія структури цієї БД(на виході отримуємо файл_2). Файл_1 та файл_2 порівнюються за допомогою спеціального програмного забезпечення. Якщо знаходяться відмінності у скриптах для створення схеми БД, то знаходяться критичні місця в файлі з скриптами, вносяться зміни в нього. Після цього відбувається повторне тестування файлу скриптів. Так відбувається до тих пір, доки всі помилки не будуть виправленні.

Після того, як було від тестовано файл зі скриптами, відбувається тестування ERP-систем на коректність її взаємодії з БД, на яку накладались зміни. Якщо під час тестування не було виявлено помилок, то тоді всі зміни накладаються на схему, що перебуває в експлуатації. Якщо ж помилки були виявленні, то тоді вносяться відповідні зміни до вихідного коду ERP-систем. Після внесення цих змін відбувається повторне тестування взаємодії системи з БД. Так триває до тих пір, доки не будуть усуненні всі помилки або залишаться ті, які не є критичним для роботи.

Список використаної літератури

1. Fowler M.(1999) Refactoring: ImprovingtheDesignofExistingCode. MenloPark,CA: AddisonWesleyLongman.
Scott W. Ambler, Pramodkumar J. Sadalage (2007)RefactoringDatabases: EvolutionaryDatabaseDesign