УДК 681.3.016

ЭВОЛЮЦИОННАЯ ТЕХНОЛОГИЯ РАЗРАБОТКИ БАЗ ДАННЫХ

И.И. ПИЛЕЦКИЙ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки,6, Минск, 220013, Беларусь

СП ЗАО "Международный Деловой Альянс"
М. Богдановича, 155, Минск,220040, Беларусь

Поступила в редакцию 3 ноября 2008

На примере разработки и сопровождения конкретной базы данных (БД) нормативно-справочной информации (НСИ) для большой территориально-распределенной корпорации приводятся неостанавливаемые технологии разработки, развития, модернизации и совместного использования данных различными приложениями.

Ключевые слова: технология разработки баз данных, архитектура баз данных, жизненный цикл баз данных.

Введение

Предметом исследования является сложная территориально распределенная корпорация, например, такая как железная дорога (Белорусская, Казахская), которая представляет собой набор сложных динамических и статических территориально-распределенных объектов, и технология построения БД для систем управления грузоперевозками в таких корпорациях. В качестве объектов этих корпораций могут быть дорога и ее топология, станции и их топология, ремонтные заводы, депо, подвижной состав (локомотивы, вагоны, контейнеры) и их характеристики, груз и его характеристики, заказчик, расписание движения и т.д. Основу различных систем управления в таких корпорациях составляют БД, содержащие информацию как нормативно-справочного характера, так и о состоянии объектов слежения [1, 2].

Задачей исследования является разработка технологии построения и сопровождения больших БД для очень сложной и полностью неопределенной предметной области при отсутствии экспертов предметной области и при наличии экспертов узких и частных областей. В процессе промышленной эксплуатации таких баз данных происходит значительный рост объема хранимых данных. Известно, что производительность БД является решающим фактором эффективности управленческих и коммерческих приложений и, если поиск или запись данных выполняется медленно, способность к нормальной работе приложения падает. Поэтому задача построения эффективной и устойчивой к деградации БД является актуальной.

Проблему разработки БД можно решать традиционным способом — построением полной модели на основе сложной экспертной оценки, а можно строить общую модель, постоянно конкретизируя и дополняя ее на основе выявления новых сущностей в предметной области. БД постоянно корректируется и уточняется (модифицируются уже существующие и добавляются новые сущности, модифицируется модель базы данных), появляются различные новые источники и потребители информации. Такая концепция построения предполагает наличие жизненного цикла системы, который должен быть максимально автоматизированным, а само программное обеспечение и БД — минимально зависимыми от изменений друг в друге.

Решения, принятые при разработке архитектуры БД, являются определяющими, как по быстродействию, так и по дальнейшему жизнеобеспечению БД [2]. В процессе эксплуатации БД развивается так же, как и программное обеспечение (ПО). БД может легко деградировать по

причине изменения ее структурных элементов и/или по причине ее пополнения большими объемами данных.

БД НСИ большой корпорации является основой интеграции приложений и данных, но в силу специфики предметной области таких корпораций невозможно заранее определить все необходимые сущности и их атрибуты для построения БД НСИ. Сама БД НСИ должна развиваться и уточняться в процессе эксплуатации и постоянного расширения источников пополнения данных, а также параллельной разработки территориально распределенных приложений, использующих данные из БД НСИ.

Данная проблема усугубляется тем, что сама БД НСИ должна быть исторической, т.е. она должна обеспечить длительное хранение и доступ к информации с историей ее изменения на протяжении всего времени жизни объекта в БД.

Пути решения проблемы

Конечно, можно искать решение данной проблемы, используя уже существующие методологии разработки ПО. К настоящему времени наибольшее распространение получили следующие основные модели жизненного цикла (ЖЦ) ПО [3–7]:

– каскадная (водопадная) или последовательная модель (1970–1985 гг.);

– итеративная и инкрементальная или эволюционная (гибридная, смешанная) модель (1986–1990 гг.);

– спиральная (spiral) модель или модель Боэма (1988–1990 гг.).

Легко обнаружить, что в разное время и в разных источниках интерпретация этих моделей претерпела изменения. Например, ранее инкрементальная модель понималась как построение системы в виде последовательности сборок (релизов), определенной в соответствии с заранее подготовленным планом и заданными (уже сформулированными) и неизменными требованиями. Сегодня об инкрементальном подходе чаще всего говорят в контексте постепенного наращивания функциональности создаваемого продукта. Для каскадной модели допускаются повторные выполнения работ, выполненных на более ранних этапах.

Может показаться, что индустрия программной инженерии пришла, наконец, к правильной обобщенной модели. Однако классическая каскадная модель продолжает встречаться в реальной жизни и для небольших проектов с хорошо определенными требованиями является предпочтительной. Спиральная модель является ярким представителем эволюционного взгляда, но в то же время представляет собой единственную модель, которая уделяет явное внимание анализу и предупреждению рисков.

Взглядов на детализацию описания жизненного цикла разработки ПО может быть много — безусловно, методологии тоже различаются. Наиболее известные на сегодняшний день методологии разработки ПО [5, 6, 8, 9]: Rational Unified Process (RUP), Enterprise Unified Process (EUP), Microsoft Solutions Framework (MSF) в двух представлениях — MSF for Agile и MSF for CMMI (анонсированная изначально как "MSF Formal"), Agile-практики — eXtreme Programming (XP), Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), SCRUM и др.

Методология RUP основана на следующих основных принципах современной программной инженерии:

– итеративной разработке;

– управлении требованиями;

– компонентной архитектуре;

– визуальном моделировании;

– управлении изменениями;

– постоянном контроле качества.

Модель разработки и сопровождения БД

Основываясь на большом практическом опыте постоянной разработки, модификации и сопровождения больших БД, для определенной не полностью (полностью неизвестной) пред-

метной области можно определить жизненный цикл БД по аналогии с жизненным циклом программного обеспечения (ЖЦ ПО).

Создание модели данных — длительный, итерационный процесс по идентификации и документированию предметной области в той части общесистемных требований, которая касается структур данных и взаимосвязей между ними.

В общем случае разработка и сопровождение БД включает такие работы, как анализ предметной области, создание схемы БД, корректировку и поддержание схемы БД в актуальном состоянии, выпуск новой версии БД, выгрузку и загрузку данных. Жизненный цикл БД (по аналогии с жизненным циклом программного обеспечения) включает не только этап создания (проектирования) логической и физической моделей БД, этап генерации БД, этап загрузки данных и тестирования, но и этапы работ, которые выполняются при сопровождении БД:

– получение исходных данных для внесения изменений и подготовка документации для внесения изменений;

– выполнение подготовительных действий по анализу модели эксплуатируемой БД и планированию создания и запуска ее новой версии;

– выполнение редактирования старой схемы и получение новой схемы, получение задания на языке DDL для создания новой версии БД;

– запуск задания на создание новой пустой БД;

– выполнение выгрузки (экспорта) данных из старой версии БД;

– выполнение загрузки (импорта) данных из файлов-копий в новую версию БД;

– внесение или дополнение необходимых данных в новые объекты новой версии;

– выполнение процедуры проверки и опытной эксплуатации новой версии БД;

– ввод в промышленную эксплуатацию новой версии БД и объявление об ее выпуске.

Эволюционная технология разработки БД подразумевает циклический процесс, состоящий из этапов проектирования и сопровождения БД:

– анализа предметной области, построения модели данных, генерации кода для создания БД, генерации различных процедур, реализующих бизнес-процессы, генерации проектной и отчетной документации;

– автоматического построения по БД физической модели БД, сравнении моделей БД, слиянии различных моделей в одну другую, построении различных физических моделей по одной логической.

Известно, что принятые архитектурные решения при разработке ПО, являются определяющими как по быстродействию, так и по жизнеобеспечению, разрабатываемого ПО [5]. Именно архитектурные решения при проектировании ПО позволяют повысить эффективность приложений в разы. Но если изъяны архитектурных решений относительно ПО обнаруживаются, как правило, сразу во время промышленной эксплуатации, то неправильные решения, принятые при проектировании БД, имеют скрытый латентный период промышленной эксплуатации и проявляются со временем накопления больших объемов данных, БД деградирует, эффективность выполнения приложений падает. Архитектурные решения для БД данных малых объемов практически не имеют эффекта деградации.

Архитектура БД это в первую очередь информационная модель данных предметной области, модель бизнес процессов предметной области. Разработка модели данных — это не разовый, а циклический процесс. Решения, принятые при разработке архитектуры БД, являются определяющими как по быстродействию, так и по дальнейшему жизнеобеспечению БД. БД развивается в процессе эксплуатации так же, как и ПО. Поэтому сразу нужно проектировать ПО и БД с учетом их дальнейшей модернизации. Это значит, что приложение не должно напрямую работать с данными из БД, поскольку, код содержащийся в ПО для извлечения данных прямо из таблиц БД, не позволяет обеспечить дальнейшую жизнь информационной системы. ПО нельзя изменить потому, что оно содержит прямые ссылки к таблицам БД, а БД нельзя изменить потому, что оно напрямую используется в ПО.

При разработке данной БД НСИ учитывалось то, что данная система является центральной по отношению к другим системам и приложениям, разрабатываемым в рамках корпоративной системы Белорусской железной дороги. Сама система БД НСИ определена как единственный источник эталонной информации для приложений корпорации.

Практики эволюционной технологии разработки БД

По аналогии, с практиками используемыми в технологиях программирования, для ЖЦ ПО, выделим основные практики, которые наиболее важные для ЖЦ БД.

1. Структурные элементы и архитектура БД должны быть устойчивыми к постоянным изменениям. Модель базы данных должна строиться из слабозависимых подсхем (компонент) для конкретного класса решаемых задач предметной области. Каждая сущность должна отвечать общепринятым требованиям к проектированию БД (нормализации, ссылочной целостности, реляционности и т.д.). Для каждой подсхемы должны быть одна (или две) определяющих сущности, которые являются основным "строительным материалом" подсхемы. Некоторые сущности могут быть общими для некоторых подсхем или всей модели БД (не за счет дублирования, а за счет отношения). Сами подсхемы могут объединяться и ссылаться на структурные элементы в другой подсхеме.

Так, структурно и функционально очень сложная сущность "станция" состоит только из номера и названия станции и используется для построения новых подсхем и сущностей: дорога, отделение дороги, стыки отделений, пограничные переходы, участки дороги, участки движения; для построения подсхемы топология дорог; для построения подсхемы расписание движения.

Само множество значений сущности может быть разделено на различные подмножества в зависимости от характера работ или структурных элементов, составляющих эту сущность. Так, множество значений сущности "станция" логически разделено на: пассажирские и грузовые станции, выполняемые ими виды работ, пограничные, специализированные; по структуре подъездных путей и объектов расположенных на них.

Это значит, что множество значений сущностей может разделяться на подмножества, имеющие различную раскраску элементов в зависимости от их характеристик.

2. Внутренняя архитектура. Внутренняя архитектура БД (внутреннее представление данных) может существенно отличаться от внешнего представления данных и является определяющим для жизнеспособности БД [2, 10]. Внешнее представление должно динамически настраиваться под конечного пользователя. Пользователи привыкли видеть плоские и простые таблицы, но это не значит, что такие таблицы должны быть в БД.

3. Должна быть обеспечена независимость функционирования обслуживающего ПО от физической модели БД. Обслуживающее ПО должно динамически настраиваться на работу с физической схемой БД, т.е. должно быть ПО виртуализирующее физическую модель БД. Данное требование является определяющим для независимой модернизации самого ПО и БД.

4. Должны быть построены интерфейсы взаимодействия, как для загрузки данных, так и для предоставления их в другие системы. Данные, поступающие для загрузки в БД с разнообразных источников информации, должны быть реструктуризированы в соответствии с некоторым внутренним протоколом БД, на основании которого будут выполняться общепринятые операции вставки, замены, удаления, независимо от форматов и источников поступления данных.

Данные для выгрузки из БД и транспортировки в другие системы обычно предоставляются во внутреннем представлении и могут передаваться по общепризнанным протоколам: SOAP, HTTP, FTP или реплицироваться в различном формате. Конечно, в качестве инструмента доставки лучше всего использовать ETL-инструменты.

Залогом успешной промышленной эксплуатации БД НСИ являются грамотно спроектированные и реализованные интерфейсы взаимодействия ПО с другими системами.

5. Проектирование БД ведется на всех этапах разработки ПО. Как показывает практика, требования на обеспечение данными от различных приложений (уже работающих и разрабатывающихся) поступают спонтанно. Эти требования могут быть на предоставление дополнительных данных, которые уже имеются в БД, или которые отсутствуют в БД. В свою очередь, последние требования могут быть удовлетворены или незначительной реструктуризацией существующих сущностей, или серьезной модификацией модели БД, разработки новой подсхемы БД, требующей создания новой БД и ее перезагрузки. Как правило, в таких ситуациях

БД НСИ приобретает нового клиента, работающего в смежных и пересекающихся бизнес-процессах с уже существующими подсистемами.

Поэтому формализованное представление всего процесса постоянного реинжиниринга БД от определения требований до выпуска новой версии в терминах основных артефактов жизненного цикла БД - является важным технологическим решением.

6. Выпуск новых версий БД НСИ и документирование изменений. Динамическое развитие базы данных требует постоянного контроля изменений с фиксацией каждого завершенного состояния. Здесь под завершенным понимается такое состояние БД, при котором в базе присутствуют все необходимые на данный момент сущности (таблицы и группы таблиц) и тестовые данные, а также БД отвечает заявленным ранее требованиям (например, все данные для некоторого приложения присутствуют в БД и сама БД отвечает требованиям по быстродействию). В процессе разработки очень важно фиксировать такие состояния. Это позволяет быстро переключаться и переходить между версиями, а иногда вернуться назад к предыдущей версии, так как спроектированная текущая версия по ряду причин не удовлетворяет новым и уже существующим требованиям. Здесь важно обратить внимание на то, что версионному контролю подлежит не только структура БД (DDL описание базы) с описанием переменных среды, но и тестовые данные. Все изменения сопровождаются соответствующей подробной документацией.

7. Регулярное тестирование на соответствия базисным правилам системы на основе техники регрессионного тестирования. При первых шагах проектирования могут быть сформированы базисные правила, которым должна соответствовать БД. Например, в БД НСИ разрабатываемой для Белорусской железной дороги существуют следующие правила: фильтрации данных, наличия специальных (системных) полей в каждой таблице, именования полей, соответствия таблиц и view и др. Эти правила должны учитываться на всех этапах проектирования. Поэтому при формировании очередной версии БД должны быть выполнены тесты для всех базисных правил и затем выполнено тестирование на реально загруженных данных.

БД — это не только данные, но и алгоритмы работы с ними. Поэтому, как и тестирования любого ПО, необходима библиотека тестов для тестирования обслуживающего ПО БД по различным критериям белого и черного ящика. Регрессионное тестирование позволяет быть уверенным, что очередная версия отвечает существующим требованиям. В процессе работы над данным проектом не раз оказывалось, что при изменении БД в некоторых, весьма специфичных, ситуациях уже проверенные алгоритмы работали не корректно. Причем источником ошибки могла служить некорректная структура новых таблиц или недостаточная проверка алгоритма работы с данными. Подобного рода накопительные тесты существенно облегчают локализацию ошибки и дают большую уверенность в корректности данных в БД.

Изменение структуры БД влечет за собой изменение объемов хранимых данных, а так же временных параметров работы БД. Временные характеристики работы могут быть определены как требование заказчика. Именно поэтому необходим еще один набор тестов, который позволит определить соответствует ли данная версия БД необходимым временным характеристикам. Было отмечено, что для приведения БД в состояние, когда она удовлетворяет всем временным критериям, необходимо тесное взаимодействие между технологами, программистами и администраторами БД.

8. Синхронизация параллельных процессов разработки: администрирование, проектирование, кодирование, тестирование. Говоря об эволюционной разработке БД НСИ, необходимо отдельно отметить важность синхронизации этого процесса с остальными этапами жизненного цикла. На практике возникала ситуация когда архитекторами базы были произведены некоторые изменения, однако программисты продолжали работать с предыдущей структурой БД. Или же, согласно обновленной модели БД, программистами были произведены соответствующие изменения, но переменные окружения базы не были исправлены администратором.

Эти примеры наглядно демонстрируют важность синхронизации жизненных циклов ПО и БД. В процессе разработки программист работает с локальной копией базы, выполняет некоторые изменения, и синхронизирует эти изменения с эталонным вариантом модели БД и системы в целом. Такой же подход используется и архитекторами.

Синхронизация подразумевает теснейшее взаимодействие между участниками проекта, на основе логической, физической моделей данных и физической схемы самой БД. По опыту выполнения проекта для корпорации "Белорусская железная дорога", многие решения были приняты путем согласования различных схем и идей с участием архитекторов, программистов и администраторов.

9. Использование CASE средств. В настоящее время быстрое и грамотное проектирование и реализация баз данных (БД) без средств графического моделирования данных и ПО практически невозможна.

При реализации эволюционной технологии разработки БД необходимо регулярно выпускать новые версии БД НСИ и обслуживающего ПО. Регулярно и без ошибок выполнять такие работы без применения CASE средств практически невозможно. В качестве базового компонента проектирования и разработки БД используется CASE-средство AllFusion Erwin Data Modeler [11, 12], а в качестве базовых CASE-систем разработки и тестирования ПО — линейка продуктов IBM Rational Corp. [13, 14] и IBM WebSphere [15]. В качестве СУБД — IBM DB2 UDB [16].

10. Контроль над количественными изменениями. Необходимо отметить одну из проблем, возникающих при использовании технологии эволюционной разработки БД. Проблема эта, оказалось, имеет философские корни и может быть сформулирована как "переход количественных изменений в качественные различия". Действительно, на начало проекта база данных содержала небольшой набор таблиц с ограниченным набором данных. Разработанные на первых этапах алгоритмы работы с данными отвечали всем требованиям заказчика. Однако при изменении структуры БД, увеличении числа таблиц и связей между ними, количества записей в них, количества подключаемых пользовательских приложений ранее разработанные алгоритмы вышли из допустимых временных показателей. Причем изменились не только эти показатели, но и были обнаружены некоторые недостатки работы логики ПО. Потребовался детальный анализ проблемы, повлекший за собой дополнительную доработку архитектуры БД, усовершенствование и оптимизацию алгоритмов обслуживающего ПО и настройку параметров работы БД [10].

Основные результаты

В качестве основных результатов работы можно выделить разработку уточненной методологии эволюционной разработки ПО для определения модели жизненного цикла БД и применения эволюционной технологии разработки БД, определение и автоматизацию основных технологических операций по выпуску новой версии БД, применение методологии постоянной модификации БД.

Технология эволюционной разработки БД позволила создать динамически развивающуюся систему, которая даст возможность вносить изменения на любой стадии разработки. Выпуск новой версии БД может быть выполнен в течение недели или нескольких месяцев в зависимости от требований нового приложения. Правильно выбранные интерфейсы взаимодействия с другими системами, интерфейсы пополнения базы позволили практически исключить зависимость клиентских систем от изменений в БД.

На сегодняшний день база данных пополняется новыми сущностями, необходимыми для новых приложений, входящих в корпоративную систему "Белорусская железная дорога", причем новые подсхемы не являются оторванными в БД, а органически дополняют описание предметной области железной дороги.

Автор выражает благодарность сотрудникам, принимавшим участие в разработке эталонной НСИ Белорусской железной дороги.

EVOLUTIONAL TECHNOLOGY OF DATABASE'S DEVELOPMENT

I.I. PILETSKI

Abstract

The unstoppable technologies of a constant development and modernizing of a large corporation and the combined usage of databases by different applications are given by the example of the development and maintenance of the certain database (DB) of normative-reference information (Master Data).

Литература

1. *Пилецкий И.И.* // Аннотации докл. 11-й Междунар. науч.-практ. конф. "Инфотранс-2006". СПб., 2006. С. 58.
2. *Пилецкий И.И., Селицкий Д.В., Костиков В.А.* // Software Engineering Conference (Russia) SEC(R) 2005: Материалы междунар. конф. Moscow, October 26–28, 2005. Режим доступа: http://2005.secr.ru/program/schedule.html# — 17 с.
3. ГОСТ Р ИСО/МЭК 12207-99 "Информационная технология. Процессы жизненного цикла программных средств". М., 2000.
4. *Боэм Б.У.* Инженерное проектирование программного обеспечения / Пер. с англ.; Под ред. А.А. Красилова. М., 1985.
5. *Бобровский С.* Технологии Пентагона на службе российских программистов. Программная инженерия. СПб., 2003.
6. *Липаев В.В.* Программная инженерия. Методологические основы. М., 2006.
7. *Катков В.Л., Пилецкий И.И.* // Программное обеспечение ЭВМ / АН БССР. Ин-т математики. Минск, 1988. Вып. 79. С. 6–32.
8. *Кратчен Ф.* Введение в Rational Unified Process. 2–е изд. / Пер. с англ. М., 2002.
9. *Кролл П., Крачтен Ф.* Rational Unified Process — это легко. Руководство по RUP / Пер. с англ. М., 2004.
10. *Пилецкий И.И.* // Материалы 4-й Междунар. конф. специалистов в области обеспечения качества программного обеспечения, Минск, 17 октября 2008 г. — Режим доступа: http://it-conf.ru/ru/content/34.htm — 15 с.
11. *Маклаков С.В.* Создание информационных систем с AllFusion Modeling Suite. М., 2003.
12. AllFusion Erwin Data Modeler. Available: http://www.ca.com/us/products/product.aspx?id=260
13. *Трофимов С.А.* CASE-технологии: практическая работа в Rational Rose. М., 2002.
14. IBM Rational Software. Available: http://www-306.ibm.com/software/rational/
15. IBM WebSphere. Available: http://www-306.ibm.com/software/websphere/
16. IBM DB2. Available: http://www-01.ibm.com/software/data/db2/