УДК 681.3:002.651.028(083.73)

В.В. Фальфушинский

Миграция устаревших программных платформ

Предложен обзор метода миграции устаревших программных платформ и их улучшений на примере баз данных. Приведены основные аспекты, которые следует учитывать при повторном использовании метода миграции для других программных платформ. Исследованы скорости выполнения запросов на программной платформе после миграции.

A brief overview of the migration of old program platforms and their improvements are described on the example of data bases The main aspects are highlighted, that can be used in re-using such a method of migration of old program platforms. Also, the research of execution speed of queries is made on a program platform after the migration.

Запропоновано огляд методу міграції застарілих програмних платформ та їх покращення на прикладі баз даних. Наведено основні аспекти, які слід враховувати при повторному застосуванні методу міграції для інших програмних платформ. Досліджено швидкості виконання запитів на програмній платформі після міграції.

Введение. С развитием современных сетевых технологий и технологий баз данных (БД) возникла задача миграции устаревших программных платформ на более современные, которые дают возможность обрабатывать и хранить большие объемы данных. Многие компании имели свои внутренние разработки для хранения данных и БД.

Такие разработки характеризовались большой скоростью считывания данных, но не имели возможности работы по сети, а также обладали некоторыми ограничениями на объем файла или количество объектов для хранения. Вскоре такие внутренние разработки были дополнены поддержкой современных БД, таких как *MS SQL Server*, *Oracle*, *Mysql*, *MS Access* и *IBM DB*2. Для более специфичных задач были использованы *CouchDB*, *Memcached* и *Membase* БД – представители направления *NoSQL*.

Общеизвестно, что направление *SQL* БД существует еще с 60-х годов и представлено несколькими стандартами, а также уровнями, отличающимися количеством инструкций, а соответственно и сложностью языка. К тому же существует язык программирования *SQL*, дающий возможность управлять базой данных [1].

В отличие от *SQL*, подходы *NoSQL* несколько различны. В основу такой концепции заложены такие термины:

– не реляционная модель данных;

– открытый исходный код;

– хорошая горизонтальная масштабируемость.

В качестве одного из методологических обоснований подхода *NoSQL* используется эвристический принцип, известный как теорема системы автоматического регулирования (CAP), утверждающая, что в распределенной системе невозможно одновременно обеспечить согласованность данных, доступность и устойчивость к расщеплению распределенной системы на изолированные части. Таким образом, при необходимости достижения высокой доступности и устойчивости к разделению предполагается не фокусироваться на средствах обеспечения согласованности данных, обеспечиваемых традиционными *SQL*-ориентированными СУБД с транзакционными механизмами на принципах *ACID* (атомарность, согласованность, изолированность, долговечность) [2–4].

Существует несколько направлений использования *NoSQL*:

- документ-ориентированные СУБД;
- хранилища «ключ–значение», кортежные хранилища;
- граф базы данных;
- базы данных в ОЗУ;
- табличные БД.

Совсем недавно многие из *NoSQL* БД получили большую популярность в области задач поиска (*Memcached*, *BigTable*, *Apache Hadoop*), обработки больших объемов документов и финансовой аналитике (*CouchDB*).

BigTable БД – высокопроизводительная база данных, построенная на основе *Google File System* (*GFS*), *Chubby Lock Service* и некоторых

других продуктах *Google*, сегодня не распространяется и не используется за пределами *Google*, хотя *Google* предлагает использовать ее как часть *Google App Engine*. Основное направление – задачи поиска данных в больших массивах. Работа над *BigTable* начата в 2004 году, и сейчас СУБД используется в различного рода приложениях *Google*, таких как *MapReduce*, часто используемое для создания и модификации данных, хранящихся в *BigTable*, *Google Reader*, *Google Maps*, *Google Book Search*, *Search_History*, *Google Earth*, *Blogger.com*, *Google Code hosting*, *Orkut* и *YouTube*. Причины, побудившие *Google* к созданию собственной базы данных, – масштабируемость и контроль над производительностью [5].

Apache Hadoop – проект фонда *Apache Software Foundation*, свободно распространяемый набор утилит, библиотек и программный каркас для разработки и выполнения распределенных программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе для *Yahoo*! и *Facebook*. Разработан в рамках вычислительной парадигмы *MapReduce*, согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат.

Разработка была инициирована в начале 2005 года Дугом Каттингом с целью построения программной инфраструктуры распределенных вычислений для проекта *Nutch* – свободной программной поисковой машины на *Java*, ее идейной основой стала публикация сотрудников *Google* Джеффри Дина и Санжая Гемавата о вычислительной концепции *MapReduce* [6,7].

CouchDB – документо-ориентированная система управления базами данных, не требующая описания схемы данных. Эта программа есть свободной, открытой и написана на языке *Erlang*. *CouchDB* можно рассматривать как сервер веб-приложений; для реализации этой идеи в *CouchDB* встроен производительный веб-сервер, а программный код, как и данные, сохраняется в той же базе данных. Для автоматизации работы с приложениями *CouchDB* используется утилита *CouchApp*. В отличие от реляционных СУБД, *CouchDB* предназначена для работы с полуструктурированной информацией [8–11].

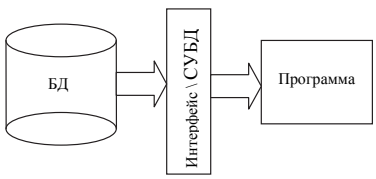
CouchDB имеет следующие особенности:

- данные сохраняются не в строках и колонках, а в виде *JSON*-подобных документов, моделью которых являются не таблицы, а деревья;
- типизация элементов данных, т.е. сопоставление отдельным полям документов типа *INTEGER*, *DATE* и прочих, не поддерживается – вместо этого пользователь может написать функцию-валидатор;
- целостность базы данных обеспечивается исключительно на уровне отдельных записей (но не на уровне связей между ними);
- связи между таблицами или записями принципиально не поддерживаются, соответственно, операция объединения (*JOIN*) между таблицами не определена;
- для построения индексов и выполнения запросов используются функции-представления (*view*) [1];
- функции-валидаторы, функции-представления, функции-фильтры сохраняются в текстовом виде в самой базе данных;
- эти функции, как правило, написаны на языках *JavaScript* или *Erlang*, а для их выполнения запускается отдельный сервер запросов, взаимодействие с которым происходит посредством сокетов и текстового *JSON*-протокола;
- каждой базе данных в системе *CouchDB* соответствует единственное *B*-дерево (не путать с двоичным деревом);
- каждое *B*-дерево хранится в виде отдельного файла на диске;
- одновременно может быть запущено несколько потоков для чтения базы данных и только один – для записи.

В украинской практике задачами миграции программного обеспечения (ПО) на разные альтернативные платформы занимаются многие. Одна из последних разработок – направления оптимизации запросов при конвертации *DL*/1 (*IMS*) в *SQL* [12].

Архитектура взаимодействия программы и БД

На рисунке представлена типичная схема взаимодействия программы с БД. В такой схеме существует несколько сегментов, связанных между собой – БД, где хранятся данные, интерфейс или СУБД, куда может входить как программный интерфейс, так и специальный язык и программа. Так как каждый из продуктов, которые надо было адаптировать для новой БД, имел собственный интерфейс или же язык для доступа к БД, то это и являлось основной проблемой при переходе на новую платформу.



Взаимодействие программы и БД

В процессе такой миграции программы с БД на более новые БД возникло два подхода:

- реализация типичных запросов к БД на аналогичном языке новой БД;
- реализация отдельных операторов и конструкций языка взаимодействия с БД как определенных запросов на аналогичном языке новой БД.

Первый подход ограничивает круг выполнения запросов с помощью ограничений, которые есть в БД:

- запросы могут генерироваться динамически;
- в зависимости от условия критерии выполнения запроса могут меняться;
- может меняться порядок/приоритеты выполнения запросов.

Если к БД генерируется большое количество динамических запросов, то такой подход невозможен.

Второй подход более гибких запросов, но и более трудоемкий, так как интерфейс содержит множество функций, которые следует переписать, а некоторые даже переделать и дополнить в зависимости от ситуации, возникающей при выполнении запроса к БД.

Так как в ГеоПоиск вызывается 1077 запросов к БД, то решено использовать второй метод.

Постановка задачи

Существует ПО, имеющее свою собственную разработку для хранения данных, а также множество компаний, имеющих такое ПО в области геологии. Например:

– ГеоПоиск;
– ГЕО Прайм;
– *GeoOffice Solver*;
– КАМЕРТОН;
– ПАНГЕЯ.

Каждая из разработок хранения данных отличается достаточно быстрой системой ввода–вывода, но параллельно имеет ряд недостатков в сравнении с существующими современными БД, например, ограничения на объем файла БД и отсутствие поддержки работы по сети. Для этого многие разработчики решаются на миграцию на более новые и быстрые БД.

Для правильной миграции стоит провести анализ структуры такого ПО (количества библиотек, функций, основных операторов, условных обозначений, спецификации данных, системы ввода-вывода данных). Для решения такой проблемы была взята БД «МикроПоиск». Эта БД автоматизирует построения проблемно-ориентированных систем обработки данных для научных и инженерных приложений. Инструментальный характер «МикроПоиска» означает наличие в нем штатных средств проектирования и ведения реляционно-сетевых БД, поддержки программирования и генерации приложений БД, ориентированных на различные категории пользователей.

Инструментарий базового языка программирования основан на Си и включает две группы штатных средств СУБД «МикроПоиск»: сервер языка запросов *MPSQL* и средства пользовательского интерфейса *MPSQL* (редактор запросов и генератор запросов). *API* сервера языка запросов реализует интерфейс с прикладными БД для программ на разных языках программирования. Он обеспечивает обмен данными с СУБД через массивы структур.

Модель данных реляционно-сетевой СУБД «МикроПоиск» основана на модели Чена «сущ-

ность-связь» (*ER*-модели). В СУБД «МикроПоиск» интенсионал (схема БД) представляется графом, вершинам которого соответствуют классы объектов (сущностей), а ребрам – классы бинарных связей между объектами. Элементарные атрибуты объектов соответствуют основным типам данных языка Си. Поименованная совокупность элементарных атрибутов может составлять повторяющуюся группу. Кроме того, допускаются атрибуты содержащие тексты произвольной длины.

Библиотека *MPSQL* работает с разными запросами. Запрос является непустой последовательностью операторов, разделенных символом **';'**. Каждый оператор формирует множество однотипных сущностей. Допускаются вложенные операторы.

Пользователю разрешается присваивать множествам имена и использовать их в последующих операторах запроса, что упрощает его понимание, а также в большинстве случаев ускоряет обработку.

Допустимы строковые константы, указываемые в одиночных или двойных кавычках. Допустимо включение в них кавычек другого типа. В любом месте запроса, кроме строковых констант, можно вставить комментарий, который будет проигнорирован при обработке запроса. Вложенные комментарии недопустимы. Пробелы, символы новой строки, перевода каретки и табуляции (а также другие символы в *ANSI*-кодировке от 1 до 32) также игнорируются, если они не находятся внутри строковых констант.

Язык запросов к СУБД «МикроПоиск» имеет более нижний уровень в сравнении с *SQL*. Таким образом, не всегда один оператор языка запросов соответствует только одному оператору с языка *SQL*, потому возникает сразу несколько задач, связанных с переходом от одного языка запросов к другому. Распознавание конструкций языка запросов к СУБД «МикроПоиск» осуществляется встроенными функциями синтаксического анализатора, в результате строится абстрактное синтаксическое дерево (*AST*). Объекты такого дерева – объекты процесса перехода от одного языка запросов к другому.

После построения синтаксического дерева любого *MPSQL*-запроса выполняется последовательный вызов соответствующих функций нижнего уровня. Каждая из этих функций обращается к МР-базе, читая или записывая данные в МР-файл [13–14].

Для решения задачи миграции приложений «МикроПоиска» на современные СУБД функции нижнего уровня переделаны так, что каждая из них транслируется в запрос на *SQL*. Таким образом, каждый *mpSQL*-запрос СУБД «МикроПоиск» встроенным парсетром языка запросов преобразуется в набор последовательных запросов на стандартном *SQL*. Рассмотрим простейший запрос по критерию «_Таблица (Скважина, Номер);». Данный запрос выбирает все значения атрибута «Номер» таблицы «Скважина». В преобразовании в *SQL* он проходит в несколько этапов:

- «*select count*(*) *as OnesCount from* [Скважина]». Вызывается функция, которая получает количество записей в таблице «Скважина»;
- «*select * from* [Скважина] *where* [*id_well*] = = 334». Вызывается функция, формирующая определенное количество запросов указаного типа, которые выбирают все данные из таблицы «Скважина», где «*id_well*» = идентификатору скважины. В процессе выполнения такой функции накапливается определенное количество запросов, которые потом выполняются вместе.

Спецификация данных в БД «МикроПоиск»

Структуры данных специфицируются на языке описания данных (*DataBase Description*, *DBD*) с двумя основными конструкциями: объект и атрибут. Каждый атрибут может состоять из элементов, групп или массивов, но обязательно имеет тип, формат и длину.

Объекты связываются между собой с помощью связи. Связи бывают разных типов: 1:1, 1:*N*, *N*:1 и *N*:*M*. Например,

1:1 – связь «один с одним», например, связь между объектами Человек с ролями Муж и Жена;

1:*N* – связь «один с несколькими», например, связь между объектами Человек с ролями Мать и Ребенок;

N:*M* – связь «несколько с несколькими», например, связь между объектами Человек с ролями Продавец и Покупатель [13].

Схема БД «МикроПоиск»

Схема прикладной БД «МикроПоиск» хранится в одном файле с расширением MP. Схема БД хранится в файле с расширением *dbd*. К БД положен еще словарь в файле с расширением *WRD*. Словарь – это текстовый файл в кодировке *DOS*, где хранятся словарные статьи, связанные с числами БД. Например, в РБД насыщение – это числовой атрибут *NOB* таблицы ГИС. При запросе к РБД значения *NOB* (1, 2, 3, …) подменяются на соответствующие строки из словаря (газ, вода, нефть, …) [13].

Схема *SQL* БД поставляется в файл *GP*8, который содержит строку подключения, логин, время последнего входа, строка статуса. К файлу *gp*8 прилагается словарь в файле с расширением *WRD*. Схема *SQL* БД есть частью БД и представлена в виде таблиц.

В табл. 1 приведены служебные таблицы схемы БД «МикроПоиск» в *SQL*. В списке атрибутов выделены внешние и внутренние ключи таблиц. В процессе разработки схемы БД в *SQL* учтены такие особенности:

• Синонимы. Добавления синонимов атрибутов и объектов, так как некоторые атрибуты или объекты могут быть названы по-разному в зависимости от используемого контекста.

• Связи. В БД «МикроПоиск» есть объект «СВЯЗЬ», который никак не может быть заменен на оператор или запрос в *SQL*. Для этого предложено использовать отдельную таблицу для реализации последнего.

• Форматы данных. В БД «МикроПоиск», каждый атрибут может отображаться по-разному, для этого задействована система форматов. В *SQL* такой возможности нет. Потому для решения такой проблемы было предложено добавить таблицу форматов, которая будет связана с атрибутом из таблицы *Gpattributes*.

• Типы атрибутов. В БД «МикроПоиск» существуют такие типы данных – *TGROUP*, *TARRAY*. Для этих типов добавлена таблица с их описанием. В процессе разработки схемы БД в *SQL* было реализовано тип данных *THIDDEN*, которые дают возможность прятать определенные атрибуты. Например, индексные поля таблиц.

Таблица 1. Служебные таблицы схемы БД «МикороПоиск» в *SQL*

Название таблицы	Атрибуты	Назначение таблицы
Gpobjects	***id***, *name*	таблица имен объектов
Gpattributes	***id***, ***objid***, *name*, *type*, *format*, *data_type*, *length*	таблица атрибутов
Gplinks	***id***, *name*, *ObjFrom*, *ObjTo*, *type*, *key*	таблица связей
Gpattributetypes	***id***, *name*	таблица типов атрибутов (группа, массив, скрытый и т.д.).
Gpdatatypes	***id***, *name*	таблица типов данных
Gpformat	***id***, *name*	таблица форматов представления данных
Gpindexes	***id***, ***objid***, ***attid***, *PrimaryKey*, *ForeignKey*	таблица индексов
Gpsynonyms	***id***, ***objid***, ***attid***, ***relid***, *name*	таблица синонимов
Gpaliases	***id***, ***objid***, ***attid***, *tableName*	таблица синонимов для таблиц

В первой версии *SQL* БД схема хранится отдельно от базы, но в процессе разработки переработано несколько типов решений:

• Хранение БД в формате *XML* отдельно от схемы. Такой вариант не был принят, так как обновление схемы БД занимало много времени.

• Частичное хранение БД в *XML*. Часть объектов хранилось в *XML*, а часть было записано в свойства атрибутов таблиц. Проблема перезагрузки базы была решена, но появилась проблема с обновлением атрибутов таблиц и их свойств, что вело за собой обновление данных системных таблиц, что в какой-то мере могло повлиять на общую работоспособность базы.

• Полное хранение схемы БД в базе *SQL*. Проблемы с перезагрузкой схемы были решены. Не нужно использовать системные таблицы, а значит нет зависимости от производителя СУБД (*MS SQL*, *MS ACCESS*, *MySQL*). Появилась возможность внедрять новые возможности и масштабировать схему без влияния на работоспособность базы.

Последний вариант хранения схемы БД был наиболее удачным, и было решено его оставить.

Пример разработки БД в «МикроПоиск» и *SQL*

В табл. 2 описан пример схемы БД «МикроПоиск». В схеме представлено несколько объектов – Месторождение, Скважина, Пропластки. Каждый объект имеет несколько атрибутов разных типов. Обьект Скважина имеет групповой атрибут Параметры. Некоторые атрибуты имеют синонимы, например, Месторождение – Площадь. Объекты имеют связи: МестСкважина между Месторождение и Скважина; СквПропластки между Скважина и Пропластки.

Таблица 2. Схема БД «МикороПоиск»

Название объекта	Атрибуты
Месторождение	Код, Шифр, Название, *Xmin*, *Xmax*, *Ymin*, *Ymax*, Комментарий
Скважина	Номер, Ствол, Куст, Назначение, Состояние, Категория, Заказчик, Подрядчик, *X*, *Y*, Альтитуда, Забой, Магнитное_склонение, Проектный_азимут, Проектное_смещение, Азимут_профиля, Дата_обработки, Интервал, Начало_бурения, Конец_бурения, Имя*DB*Мфайла, *TMP*1, *TMP*2, Комментарий, Параметры
	Параметры: Код, Название, Значение
Пропластки	Скважина, Куст, Имя*DBM*файла, Фрагмент, *ZK*, *ZP*, к_ппл

В табл. 3 показан пример реализации аналогичной схемы БД в *SQL*. Аналогичная схема БД в *SQL* представлена несколькими таблицами, описывающими отдельные объекты, атрибуты. В *SQL* БД добавлены внутренние и внешние ключи, выделенные жирным. Групповые атрибуты представлены в виде отдельных таблиц с установленными необходимыми атрибутами в схеме БД. Добавлены скрытые атрибуты, которыми являются внешние и внутренние ключи таблиц и не видны пользователю. Форматы и типы атрибутов, а также связи были добавлены в таблицы по умолчанию и не показаны в табл. 3.

Таблица 3. Схема БД «МикроПоиск» в *SQL*

Название объекта	Атрибуты	Описание
1	2	3
Месторождение	***id_mest***, Код, Шифр, Название, ДопКод, Недропользователь, Начало_разработки, Подсчет_запасов, Тип, ТипКоорд, *Xmin*, *Xmax*, *Ymin*, *Ymax*, Пласты, Исследования, Комментарий, Файлы	Обьект «Месторождение» с атрибутами

Продолжение табл.

1	2	3
Скважина	***id_well***, ***id_mest***, Номер, Ствол, Куст, Назначение, Состояние, Категория, Заказчик, Подрядчик, *X*, *Y*, Альтитуда, Забой, Магнитное_склонение, Проектный_ азимут, Проектное_смещение, Азимут_профиля, Дата_обработки, Интервал, Начало_бурения, Конец_ бурения, Имя*DBM*файла, *TMP*1, *TMP*2, Комментарий	Обьект «Скважина» с атрибутами
Пропластки	***id_proplast***, ***id_well***, Скважина, Куст, Имя*DBM*файла, Фрагмент, *ZK*, *ZP*, к_ппл	Обьект «Пропластки» с атрибутами
Параметры	***id***, ***id_well***, Код, Название, Значение	Обьект «Параметры» с атрибутами, которые принадлежат обьекту «Скважина»

Типы данных

Для правильного представления данных в базе используется система приведения типов из типов данных СУБД «МикроПоиск» в *SQL* [1]. В табл. 4 показано сравнение разных типов в БД «МикроПоиск» и *SQL*, их размер в байтах, формат данных.

Для каждого объекта и атрибута БД может существовать синоним, потому что в некоторых предметных областях один и тот же объект может называться по-разному. Так же в схеме БД появилась возможность создавать синонимы для таблиц, потому что в *SQL* базе данных групповые атрибуты могут иметь только уникальные имена, а в БД «МикроПоиск» они могут быть одинаковыми, но принадлежать разным объектам.

Таблица 4. Сопоставление типов в БД «МикроПоиск» и *SQL*

Тип данных СУБД «МикроПоиск»	Тип данных *SQL*	Размер, байт	Формат
TLONG	*int*	4	%8l*d*
TINT	*smallint*	2	%3*d*
TDOUBLE	*float*	8	%2*d*
TFLOAT	*real*	4	%10.6*f*
TCHARF	*nvarchar*	–	%–5*s*

В *SQL* БД добавлена возможность присвоения каждому атрибуту отдельного формата. Так,

в табл. 4 приведены примеры форматов, которые могут быть назначены для типов данных.

Примеры выполнения запросов

Для проверки основных операторов языка запросов составим несколько примеров типичных запросов. В табл. 5 и 6 показаны примеры запросов к БД «МикроПоиск» и их аналоги в *SQL*. Каждая таблица содержит описание запроса и время выполнения в миллисекундах.

Таблица 5

Дать пользователю указать атрибут Название из таблицы Месторождение и потом присвоить атрибуту Шифр из таблицы Месторождение значение «666»	
МикроПоиск	*m*=_Указать (Месторождение , Название) ; _Присвоить (*m*, Шифр =«666») ; _Таблица (m)
SQL	*select count*(*) as *OnesCount from* [Месторождение] *select* [*id_mest*] *from* [Месторождение] *select * from* [Месторождение] *where* [*id_mest*] =31 *update* [Месторождение] *set* [Шифр]= '666' *where* [*id_mest*] =31 *select * from* [Месторождение] *where* [*id_mest*] =31 *select count*(*) *as OnesCount from* [Месторождение] *select* [*id_mest*] *from* [Месторождение] *select* * from [Месторождение] where [id_mest] =31 select * from [Пласты] where [id_mest] = 31 order by [id] ASC select * *from* [Месторождение] *where* [*id_mest*] =31 *select * from* [Исследования] *where* [*id_mest*] = 31 *order by* [*id*] *ASC* *select * from* [Месторождение] *where* [*id_mest*] =31
Время выполнения, миллисекунд	10

Просмотрев трассировку выполнения запросов к БД «МикроПоиск» становится ясно, что один запрос представлен несколькими запросами *SQL*, что существенно влияет на производительность. Эта проблема говорит о том, что нельзя в полной мере представить соотношение один в один операторов БД «МикроПоиск» в *SQL*. Потому были реализованы такие методы:

- Загрузка индексов отдельно от данных. Использование таблицы индексов и/или *Recordset*.
- Буферизированный ввод данных.

Исследования разных методов записи данных в *SQL*

В процессе разработки была исследована скорость выполнения отдельных *SQL* операторов и их комбинаций для записи данных.

Таблица 6

Дать пользователю указать атрибут Название из таблицы Месторождение, перейти по связи МестСкважина. Дать пользователю указать атрибут Номер из таблицы Скважина. Перейти по связи СквПропластки. Показать атрибут Скважина таблицы Пропластки.	
МикроПоиск	*s*=_Указать (Месторождение , Название) /МестСкважина ; *s*=_Указать (*s* , Номер); *g*=*s*/СквПропластки ; _Таблица (*g*,Скважина)
SQL	*select count*(*) as *OnesCount from* [Месторождение] *select* [*id_mest*] *from* [Месторождение] *select count*([*id_well*]) *as id_count from* [Скважина] *where* [*id_mest*]=31 *select* [*id_well*] *from* [Скважина] *where* [*id_mest*]=31 select * from [Скважина] where [id_well] =334 … *select * from* [Скважина] *where* [*id_well*] =363 *select count*([*id_proplast*]) *as id_count from* [Пропластки] *where* [*id_well*]=337 *select* [*id_proplast*] *from* [Пропластки] *where* [*id_well*]=337 select count(*) as OnesCount from [Пропластки] *select* [*id_proplast*] *from* [Пропластки] *select * from* [Пропластки] *where* [*id_proplast*] =231 *select * from* [Пропластки] *where* [*id_proplast*] =232
Время выполнения, миллисекунд	10

Такая проблема была вызвана малой скоростью записи данных при большом количестве запросов. Для исследования взят файл размером 324 Кб (949 строк), который был записан в базу разными методами. Результаты исследования изложены в табл. 7.

- *DELETE*/*INSERT* – сначала данные удаляются из таблицы с помощью *DELETE* и последовательно вставляются с помощью *INSERT*;

• *Batch DELETE/INSERT* – сначала данные удаляются из таблицы с помощью *DELETE*, далее запросы *INSERT* накапливаются и выполняются в *Batch* режиме;

• Последовательное выполнение *UPDATE* – последовательно выполняются запросы типа *UPDATE*;

• *Batch UPDATE* – запросы типа *UPDATE* накапливаются и выполняются в *Batch* режиме.

Таблица 7. Результаты разных методов записи данных в *SQL*

Метод записи данных	Время выполнения, секунды
DELETE/INSERT	4
Batch DELETE/INSERT	12
Последовательное выполнение *UPDATE*	8
Batch UPDATE	14

На основании этого исследования был выбран метод *DELETE/INSERT* и реализован метод буферизированного ввода данных.

Заключение. В статье описана технология миграции устаревших программных платформ на более современные на примере реализации БД «МикроПоиск» в *SQL*. В процессе работы реализовано схему базы данных в *SQL*; переработаны некоторые операторы языка в соответствии с возможностями базы данных *SQL*; проведено исследование по записи данных в базу данных *SQL*; реализован буферизированный метод ввод данных; реализована поддержка разных типов БД – старая БД «МикроПоиск» и новая *SQL* БД «МикроПоиск»; для нахождения разорванных связей и восстановления БД предусмотрена система восстановления после сбоев; для удобного администрирования БД реализован набор функций по созданию точек восстановления; реализована встроенная система аутентификации; проведен сравнительный анализ выполнения запросов в БД «МикроПоиск» и *SQL*.

1. *ДСТУ* ISO/IEC 9075-1:2008 Інформаційні технології. Мови баз даних – SQL. Ч. 1. – К.: Держспоживстандарт України, 2003 – Нац. стандарт України. – Введ. 2003-07-01.
2. *Cloud* analytics: Do we really need to reinvent the storage stack? / A. Rajagopal, G. Karan, P. Prashant et al. // HotCloud '09 the 2009 USENIX Annual Techn. Conf. (USENIX '09), June 14–19, 2009.
3. *Database*-Agnostic Transaction Support for Cloud Infrastructures / C. Navraj, C. Bunch, C. Krintz et al. // USENIX Annual Techn. Conf. (USENIX '09), June 14–19, 2009.
4. *Megastore*: Providing Scalable, Highly Available Storage for Interactive Services / J. Baker, C. Bondc, J. Corbett et al. // Conf. on Innov. Data Syst. Res. (CIDR), Jan. 2011. – P. 223–234.
5. *Bigtable*: A Distributed Storage System for Structured Data. / F. Chang, D. Jeffrey, G. Sanjay et al. // Google. OSDI'06: Seventh Symp. on Operating Syst. Design and Implementation, Seattle, WA, Nov. 2006.
6. *Kellerman J*. HBase: structured storage of sparse data for Hadoop. – http://blog.rapleaf.com/wp-content/uploads/ 2007/12/hbase.pdf 22, Dec. 2011.
7. *Jerey D*., *Sanjay G*. Mapreduce: simplied data processing on large clusters. Commun. ACM. – 2008. – P. 107–113.
8. *Lakshman A*., *Malik P*. Cassandra – A Decentralized Structured Storage System // The 3rd ACM SIGOPS Intern. Workshop on Large Scale Distributed Systems and Middleware. – New York: Cornell University. 11.10.2009.
9. *Sanjay G*., *Howard G*., *Shun-Tak L*. The google file system. SOSP '03: Proc. of the 19 th ACM symp. on Operating systems principles. – New York, NY, USA, 2003. – P. 29–43.
10. *Lennon J*. Exploring CouchDB. IBM. 31 Mar. 2009. – http://www.ibm.com/developerworks/opensource/library/os-couchdb/index.html
11. *Brown MC*. Getting Started with CouchDB. – New York: O'Reilly Media, – Febr. 2012. – P. 50.
12. *Оптимізація* запитів при конвертації *DL*/1 (*IMS*) в *SQL* / А.В. Анісімов, О.П. Кулябко, П.П. Кулябко та ін. // УкрПРОГ, 25–27 трав. 2010 р., м. Київ, Україна.
13. *Информационные* хранилища в промысловой геофизике / В.О. Гречко , В.Г. Тульчинский, П.Г. Тульчинский и др. // Проблемы программирования. – 2000. – № 1–2. – С. 42–48.
14. *Развитие* технологии «ГеоПоиск» для изучения нефтегазовых и рудных месторождений. НТВ «Каротажник» / В.Д. Косаченко, М.Д. Красножон, В.Г. Тульчинский и др. // – Тверь: АИС, 2007. – **155**. – С. 50–67.

Поступила 05.04.2012

Тел. для справок: (044) 526-3603 (Киев)
E-mail: *pgt@ukr.net*