электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Эволюция структур управления данными

77-30569/289266
11, ноябрь 2011
авторы: Ларионцева Е. А., Иванова Е. Ю.
УДК 004.655

МГТУ им. Н.Э. Баумана
spesial@mail.ru
kat.ivanova@hotmail.com

Эволюция структур управления данными

Введение

С самого начала развития вычислительной техники образовались два основных направления ее использования. Первое направление - применение вычислительной техники для выполнения численных расчетов, которые слишком долго или даже невозможно производить вручную. Становление этого направления способствовало интенсификации методов численного решения сложных математических задач, развитию класса языков программирования, ориентированных на удобную запись численных алгоритмов, становлению обратной связи с разработчиками новых архитектур ЭВМ. Второе направление - это использование средств вычислительной техники в автоматических или автоматизированных информационных системах.

В широком смысле *информационная система* представляет собой программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера, выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. Обычно объемы информации, с которыми приходится иметь дело таким системам, достаточно велики, а сама информация имеет достаточно сложную структуру. Классическими примерами информационных систем являются банковские системы, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т.д.

Второе направление появилось гораздо позже первого. Это связано с тем, что на начальном этапе развития вычислительной техники компьютеры обладали ограниченными запасами памяти. Вначале использовались два вида устройств внешней памяти: магнитные ленты и барабаны. При этом емкость магнитных лент была достаточно велика, но по своей физической природе они обеспечивали последовательный доступ к данным. Магнитные барабаны давали возможность

произвольного доступа к данным, но были ограниченного размера. Для численных расчетов эти ограничения несущественны - даже если программа должна обработать (или произвести) большой объем информации, при программировании можно продумать расположение этой информации во внешней памяти, чтобы программа работала как можно быстрее. Но для информационных систем, в которых потребность в текущих данных определяется пользователем, наличие только магнитных лент и барабанов неудовлетворительно – одним из основных требований, предъявляемых к таким системам, является быстрота выполнения операций. Именно требования к вычислительной технике со стороны различных приложений вызвали появление съемных магнитных дисков с подвижными головками, что явилось революцией в истории вычислительной техники. Эти устройства внешней памяти обладали существенно большей емкостью, чем магнитные барабаны, обеспечивали удовлетворительную скорость доступа к данным в режиме произвольной выборки, а возможность смены дискового пакета на устройстве позволяла иметь практически неограниченный архив данных.

Несомненно, и по сегодняшний день проблема хранения данных на различных запоминающих устройствах остается актуальной. Поэтому данная статья освещает различные структуры хранения и управления данными, выделяя все их преимущества и недостатки [1].

1. Файловые системы.

Важным шагом в развитии структур управления данными явился переход к файловым системам. С точки зрения прикладной программы *файл* - это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные. Правила именования файлов, способ доступа к данным, хранящимся в файле, и структура этих данных зависят от конкретной системы управления файлами и, возможно, от типа файла. Система управления файлами берет на себя распределение внешней памяти, отображение имен файлов в соответствующие адреса во внешней памяти и обеспечение доступа к данным.

Первая развитая файловая система была разработана фирмой *IBM* для ее серии 360. В этой системе поддерживались как чисто последовательные, так и индексно-последовательные файлы, а реализация во многом опиралась на возможности только появившихся к этому времени контроллеров управления дисковыми устройствами. Понятие файла в OS/360 было выбрано как основное абстрактное понятие, которому соответствовал любой внешний объект, включая внешние устройства, поэтому работать с файлами на уровне пользователя было очень неудобно. Требовался целый ряд громоздких и перегруженных деталями конструкций.

Файловые системы имеют следующие области применения. Во-первых, файлы применяются для хранения текстовых данных: документов, текстов программ и т.д. Такие файлы обычно образуются и модифицируются с помощью различных текстовых редакторов. Структура текстовых файлов обычно проста: это либо последовательность записей, содержащих строки текста, либо последовательность байтов, среди которых встречаются специальные символы (например, символы конца строки).

Файлы с текстами программ используются как входные тексты компиляторов, которые в свою очередь формируют файлы, содержащие объектные модули. С точки зрения файловой системы, объектные файлы также обладают очень простой структурой - последовательность записей или байтов. Система программирования накладывает на эту структуру более сложную и специфичную для этой системы структуру объектного модуля.

И последний тип файлов - это файлы, формируемые редакторами связей и содержащие образы выполняемых программ. Логическая структура таких файлов остается известной только редактору связей и загрузчику - программе операционной системы. Такую же структуру имеют файлы, содержащие графическую и звуковую информацию.

Таким образом, файловые системы обычно обеспечивают хранение слабо структурированной информации, оставляя дальнейшую структуризацию прикладным программам [2].

Однако для информационной системы, которая главным образом ориентирована на хранение, выбор и модификацию постоянно существующей информации, ее реализация на базе файловой системы, во-первых, требует создания достаточно сложной надстройки для многоключевого доступа к файлам, и, во-вторых, вызывает требование существенной избыточности хранения. Так, стремление выделить и обобщить общую часть информационных систем, ответственную за управление сложно структурированными данными, явилось главной побудительной причиной создания СУБД - систем управления базами данных. Очень скоро стало понятно, что невозможно обойтись общей библиотекой программ, реализующей над стандартной базовой файловой системой более сложные методы хранения данных. Традиционных возможностей файловых систем оказывается недостаточно для построения даже простых информационных систем. Был выявлен ряд потребностей, которые не покрываются возможностями систем управления файлами: поддержание логически согласованного набора файлов; обеспечение языка манипулирования данными; восстановление информации после разного рода сбоев; реально параллельная работа нескольких пользователей [1].

2. Базы данных, их структура и функции

Информационная система требует создания в памяти ЭВМ динамически обновляемой модели внешнего мира с использованием единого хранилища - **базы данных**. (словосочетание "динамически обновляемая" означает, что соответствие базы данных текущему состоянию предметной области обеспечивается не периодически, а в режиме реального времени. При этом одни и те же данные могут быть по-разному представлены в соответствии с потребностями различных групп пользователей.) Можно считать, что если прикладная информационная система опирается на некоторую систему управления данными, обладающую этими свойствами, то эта система управления данными является системой управления базами данных (СУБД).

База данных – совместно используемый набор логически связанных данных (и их описание), предназначенный для удовлетворения информационных потребностей организации. Это единое хранилище данных, которое однократно определяется, а затем используется одновременно многими пользователями из разных подразделений. Вместо разрозненных данных, здесь все данные собраны вместе с минимальной долей избыточности. База данных хранит не только рабочие данные той или иной организации, но и их описания – поэтому базу данных также называют набором интегрированных записей с самоописанием. Именно наличие самоописания данных в базе обеспечивает в ней независимость между программами и данными.

Отличительной чертой баз данных следует считать то, что данные хранятся совместно с их описанием, а в прикладных программах описание данных не содержится. Независимые от программ пользователя данные обычно называются **метаданными**. В ряде современных систем метаданные, содержащие также информацию о пользователях, форматы отображения, статистику обращения к данным и др. сведения, хранятся в словаре базы данных [3].

Система управления базами данных (СУБД) – это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ. Это программное обеспечение, которое взаимодействует с прикладными программами пользователя и базой данных и обладает следующими возможностями.

1) Позволяет определять базу данных, что осуществляется с помощью языка определения данных DLL (Data Definition Language)/ Язык DLL предоставляет пользователям средства указания типа данных и их структуры, а также средства задания ограничений для информации, хранимой в базе данных.

2) Позволяет вставлять, обновлять, удалять и извлекать информацию из базы данных, что осуществляется с помощью языка управления данными – DML (Data Manipulation Language). Наличие централизованного хранилища всех данных и их описаний позволяет использовать язык DML как общий инструмент организации запросов, иногда называемый языком запросов. Наличие языка запросов позволяет устранить присущие файловым системам ограничения, при которых пользователям приходится иметь дело только с фиксированным набором запросов или постоянно возрастающим количеством программ, что порождает проблемы управления программным обеспечением.

3) Предоставляет контролируемый доступ к базе данных с помощью

— системы обеспечения безопасности, предотвращающей несанкционированный доступ к базе данных со стороны пользователя;

— системы поддержки целостных данных, обеспечивающей непротиворечивое состояние хранимых данных;

— системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к базе данных;

— системы восстановления, позволяющей восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате сбоя аппаратного или программного обеспечения;

— доступного пользователям каталога, содержащего описание хранимой в базе данных информации.

—

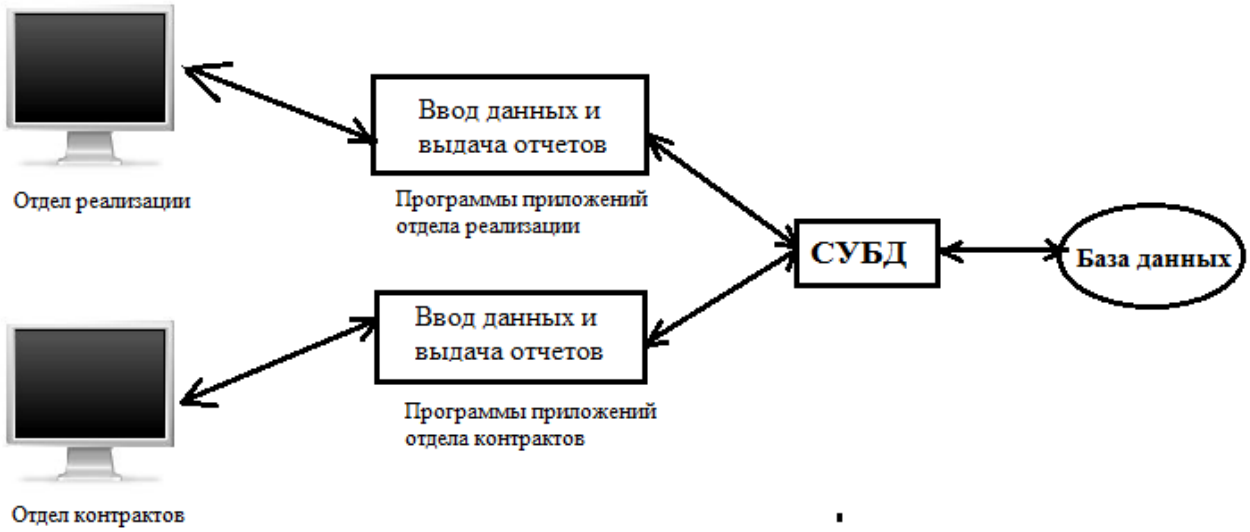

Рис. 1. Схема обработки данных с помощью СУБД.

Обычно современная СУБД содержит следующие компоненты:

- ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
- процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;
- подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;
- сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы [1].
-

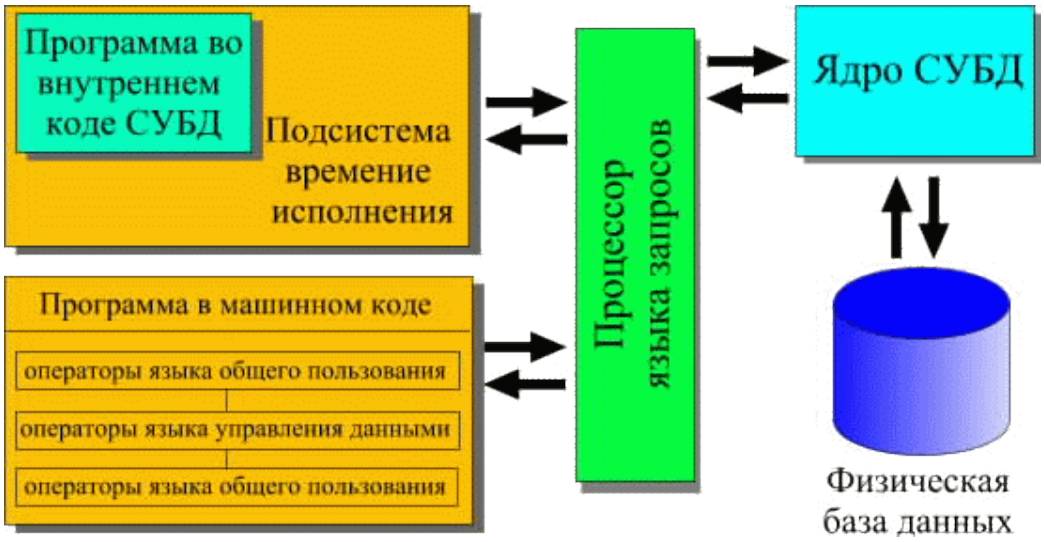

Рис. 2 Компоненты СУБД.

3. Основные функции СУБД

Выделяют следующие основные функции СУБД.

- *Непосредственное управление данными во внешней памяти*

Эта функция включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы). В некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти.

- *Управление буферами оперативной памяти*

СУБД обычно работают с БД значительного размера - этот размер обычно существенно больше доступного объема оперативной памяти. Если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти.

- *Управление транзакциями*

Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.

- *Журнализация*

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Вв любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД (журнал - это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, в которую поступают записи обо всех изменениях основной части БД).

- *Поддержка языков БД*

Для работы с базами данных используются специальные языки, называемые языками баз данных.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language) [2].

4. Обеспечение целостности баз данных

Язык SQL содержит специальные средства определения ограничений целостности БД. Обеспечение контроля целостности БД производится на языковом уровне, т.е. при компиляции операторов модификации БД компилятор SQL на основании имеющихся в БД ограничений целостности генерирует соответствующий программный код.

Специальные операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми в БД запросами (результатом любого запроса к реляционной БД является таблица) с именованными столбцами. Авторизация доступа к объектам БД производится также на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL разного вида пользователь должен обладать различными полномочиями. Пользователь, создавший таблицу БД, обладает полным набором полномочий для работы с этой таблицей. В число этих полномочий входит полномочие на передачу всех или части полномочий другим пользователям, включая полномочие на передачу полномочий. Полномочия пользователей описываются в специальных таблицах-каталогах, контроль полномочий поддерживается на языковом уровне.

Рассмотрим механизм функционирования на примере базы данных о клиентах. Можно выделить основные принципы построения системы.

- Наличие единого хранилища информации, откуда в любой момент времени доступны все сведения о предыдущем и планируемом взаимодействии с клиентами.
- Клиентские модули программы действуют по методике соответствия полномочий – то есть в идеальном варианте, каждый сотрудник компании видит в программе только ту часть информации, которая доступна ему по должностным полномочиям.
- Постоянный анализ собранной информации о клиентах и подготовка данных для принятия соответствующих организационных решений — например, сегментация клиентов на основе их значимости для компании.

На сервере компании находится некоторая «директория», в которой имеется база данных по всем существующим клиентам – телефоны, фамилии, договора, первичные документы, файлы и прочее. Вся эта информация строго структурирована, защищена и появляется на экране по первому клику мышки, но только в том случае, если у запрашивающего есть должностные полномочия на просмотр и изменения информации. Также из этой «директории» можно производить почтовую рассылку, формирование и печать первичных документов договоров, мгновенно получать разноуровневую аналитику по всем происходящим в компании бизнес-процессам и многое другое.

Одной из главных проблем при создании и поддержке подобной системы взаимоотношений является задача поддержания *целостности и безопасности информации о клиентах*. Компания, стремящаяся для более эффективного взаимодействия с клиентом собрать максимум информации о нем, должна позаботиться о *нераспространении этих данных* (Client Information Privacy).

Система должна не только проверять, не нарушаются ли ограничения в ходе выполнения различных операций, но и должным образом реагировать, если операция приводит к нарушению целостности. Имеется два типа реакции на попытку нарушения целостности:

- отказ выполнить "незаконную" операцию;
- выполнение компенсирующих действий.

Выполнение обработки целостности может быть произведено в случае добавления или удаления данных из таблиц базы, а также при изменении таблиц в результате добавления для нее ограничений

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

- декларативная поддержка ограничений целостности;
- процедурная поддержка ограничений целостности.

Декларативная поддержка ограничений целостности заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language).

Процедурная поддержка ограничений целостности заключается в использовании триггеров и хранимых процедур.

Наличие ограничения целостности (как декларативного, так и процедурного характера) *всегда* приводит к созданию или использованию некоторого программного кода, реализующего это ограничение.

Если используется декларативное ограничение целостности, то возможны два подхода.

- При декларировании (объявлении) ограничения текст ограничения хранится в виде некоторого объекта СУБД, а для реализации ограничения используются встроенные в СУБД функции, и тогда этот код представляет собой внутренние функции ядра СУБД.
- При декларировании ограничения СУБД автоматически генерирует триггеры, выполняющие необходимые действия по проверке ограничений [1].

Если ограничение целостности организовано в виде триггеров, то программный код является телом триггера. *Триггер* определяет набор действий, выполняемых вместе с операцией изменений значений данных в базе (к ним относятся типовые операции INSERT - вставить, UPDATE – обновить или DELETE – удалить данные) для заданной базовой или типизированной таблицы. Поэтому триггеры применяют с целью контроля целостности базы для:

- проверки входных данных;
- генерации значения для только что вставленной строки;
- чтения из других таблиц для задания перекрестных ссылок;
- записи в другие таблицы для контроля процесса.

С помощью триггеров можно выполнять операции удаления, вставки и обновления для производной таблицы, которая не унаследовала атрибут, разрешающий обновление.

Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде. Администратор имеет возможность просмотреть отчет, в котором выводится информация об используемых прикладной программой хранимых процедурах.

Если система не поддерживает ни декларативную поддержку ссылочной целостности, ни триггеры , то программный код, следящий за корректностью базы данных, приходится размещать в пользовательском приложении. Это сильно затрудняет разработку программ и не защищает от попыток пользователей напрямую внести некорректные данные в базу данных.

Рассмотрим создание триггера в базе данных и механизм его функционирования в базе данных на примере. Создадим в базе данных, целостность которой необходимо обеспечить, тестовую таблицу tab_test:

```
create table tab_test (ID int, name varchar (20))
```

Далее необходимо создать еще одну таблицу integrity_t для хранения значений маркера попыток нарушения целостности данных тестовой таблицы.

```
create table integrity_t (name varchar (20), flag int)
insert into integrity_t (name) values (‘tab_test)
insert into integrity_t (flag) values (0) where name = ‘tab_test’
```

После совершения подготовительных действий выполняется создание самого триггера для регистрации изменений (команда insert) в таблице tab_test.

```
create trigger trig after insert on tab_test for each row update
integrity_t set flag=1 where name=’tab_test’
```

Предположим, злоумышленник попытался нарушить целостность таблицы и внес некоторые данные в таблицу tab_test. В этом случае, мгновенно реагирует триггер и значение флага в таблице integrity_t с 0 меняется на 1, что является признаком несанкционированного изменения данных в базе.

Таким образом, с помощью определения соответствующих правил и процедур можно осуществлять контроль за целостностью данных в базе [4].

5. Заключение.

В последние десятилетия базы данных находят свое применение в различных областях науки, промышленности и бизнеса. Так, например, огромным шагом в структурировании данных на предприятии и автоматизации систем стало создание специальных систем по управлению взаимоотношений с клиентами - C*ustomer relationship management*, или сокращенно CRM системы - представляет собой набор определенного программного обеспечения (ПО), позволяющего автоматизировать и совершенствовать бизнес-процессы, связанные с управлением, продажами, маркетингом и сервисной поддержкой клиентов.

Основная цель создания таких баз данных - предоставить возможность оперативного поиска необходимой системной информации. Именно поэтому, в связи с все возрастающим интересом к базам данных, остро встает проблема обеспечения безопасности хранения информации и доступа к данным.

Список литературы

1) Зеленков Ю. А. Введение в базы данных, Я. : Центр ЯрГУ, 1998 г. с. 1-10, 70-85.

2) Пушников А. Ю. Введение в системы управления базами данных, Уфа.: изд-е Башкирского ун-та, 2000. с. 1-15, 45-47.

3) Кузнецов С. Д. Введение в стандарты языка баз данных SQL, М.Ж Центр Информационных Технологий, 1998 г. с. 20-30

4) http://www.ibm.com/, дата обращения 01.12.11