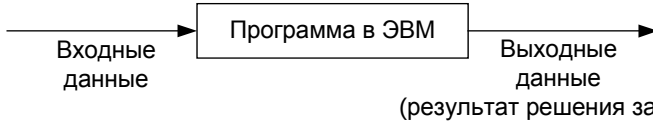
ГЛАВА 1. ВВЕДЕНИЕ В БАЗЫ ДАННЫХ. ОБЩАЯ ХАРАКТЕРИСТИКА ОСНОВНЫХ ПОНЯТИЙ ОБРАБОТКИ ДАННЫХ

1.1. Развитие основных понятий представления данных

Любой вычислительный процесс представляет собой отображение (по определенному алгоритму) некоторых входных данных в выходные.



Соотношение сложности представления обрабатываемых данных и алгоритма вычислений определяет два класса задач:

- вычислительные задачи – достаточно простое представление данных и сложный, многооперационный процесс вычислений;
- задачи обработки данных (невычислительные задачи) – простой алгоритм обработки данных и сложное представление обрабатываемых данных.

На начальной стадии обучения программированию основное внимание уделяется разработке алгоритма решения задачи. Однако часто оказывается, что возможность (или невозможность) решения конкретной задачи зависит не только от выбранного алгоритма, но и от того, какие понятия используются для представления обрабатываемых данных.

Рассмотрим простейший пример вычисления по формуле:

$$Y = X^2 + 5X,$$

где X и Y – определенные числа, которые являются здесь элементарными единицами данных (элементами данных).

При программировании алгоритма решения этой задачи (программирование формулы) используется простейший вид данных – простая переменная (X и Y представляются в программе простыми переменными). Заметим, что простая переменная в системах программирования характеризуется определенным типом ее значений, которые должны выбираться при программировании.

Рассмотрим другой пример:

$$S = a_1 + a_2 + ... + a_N .$$

Решение этой задачи в общем случае невозможно получить используя только простые переменные. Здесь обрабатывается не отдельное число, а последовательность чисел. В этом случае при программировании используется такой вид данных, как массив – совокупность элементов, с каждым из которых связан упорядоченный набор целых чисел, называемых индексами. Все элементы должны иметь одинаковый тип их значений, который и будет типом массива. В этом случае числа a_1, a_2,…, a_N представляются в программе массивом $A(1)$, $A(2)$,…, $A(N)$.

Приведенные примеры показывают, что изменение вида задач обусловливает необходимость использования других видов данных.

Ранние языки программирования (ФОРТРАН, АЛГОЛ-60) были предназначены для решения научно-технических вычислительных задач. В этих языках использовались только вышеуказанные виды данных (простые переменные и массивы) что было вполне достаточно.

Начиная с конца 60-х годов компьютеры начинают интенсивно использоваться для решения так называемых невычислительных задач, связанных с обработкой различного рода документов. Рассмотрим появление новых видов данных на примере упрощенных задач обработки данных.

Задача 1. Начисление заработной платы.

Рассматриваем задачу при двух упрощающих предположениях:

- сотруднику начисляется заработная плана на основе его оклада;
- никакие налоги и вычеты не учитываются.

Необходимые для решения этой задачи сведения о сотруднике представлены в следующей карточке НАЧИСЛЕНИЕ:

Фамилия, имя, отчество	Оклад	Количество отработанных дней в месяце	Начисленная сумма
FIO	*O*	K_o	*S*

Для каждого работника начисленная сумма за определенный месяц рассчитывается по следующей формуле:

$$S = K_o O / K_r ,$$

где K_r – количество рабочих дней в данном месяце.

Для каждого сотрудника соответствующие данные имеют конкретное значение, например:

Иванов Иван Иванович	1800	24	1800

Эти значения имеют смысл только во взаимосвязи друг с другом. Отдельно выбранное число 1800 теряет свой содержательный смысл, поэтому использовать такой вид данных, как простая переменная, здесь нельзя. В то же время набор соответствующих значений, характеризующих конкретного сотрудника, имеет разные типы (символьный и числовой), т.е. использовать для его представления такой вид данных, как массив, также нельзя. Таким образом, понятий «простая переменная» и «массив» недостаточно, чтобы представить соответствующую карточку.

Для описания аналогичных представлений данных в невычислительных задачах вводится ряд новых понятий.

Элемент данных (поле) – наименьшая единица поименованных данных.

Для данного примера элементами данных являются *FIO*, *O*, K_o, *S*.

Запись – поименованная совокупность элементов данных (полей).

Экземпляр записи – текущее значение элементов записи.

Файл – поименованная совокупность всех экземпляров записей заданного типа.

Пример файла НАЧИСЛЕНИЕ:

Иванов Иван Иванович	1800	24	1800
Петров Петр Петрович	2200	20	1830
Сидоров Сидор Сидорович	2500	24	2500

Таким образом, с помощью введенных понятий можно описывать соответствующие данные. Эти понятия нашли свое отражение в современных языках программирования, предназначенных как для вычислительных задач, так и для задач обработки данных.

В алгоритмическом языке Паскаль вводится такой вид данных, как запись (RECORD) – сложная переменная с несколькими компонентами, которые могут иметь разные типы. Кроме того, доступ к компонентам записи (полям) осуществляется не по индексу, а по имени. При программировании задачи 1 на языке Паскаль запись НАЧИСЛЕНИЕ представляется видом данных *RECORD*.

```
Salary = RECORD
        FIO:  string;
        O:    real;
        Ko:   real;
        S:    real;
END;
```

Решение задачи 1 состоит из двух этапов.

1. Ввод исходных данных и занесение их во внешнюю память.

```
type
Salary = RECORD
        FIO:  string;
        O:    real;
        Ko:   real;
        S:    real;
END;
FSalary = File of Salary;
var
  F: FSalary;
…
{ Ввод исходных данных }
repeat
   write('Введите количество сотрудников (не более',
         MaxN,' ): ');
   readln(N);
until (N>0) AND (N<=MaxN);
For I := 1 to N do
Begin
  Write('Введите фамилию сотрудника с номером ',I,': ');
  ReadLn(Sotr[i].FIO);
  Write('Введите оклад сотрудника с номером ', I, ': ');
  ReadLn(Sotr[i].O);
  Write('Введите кол-во отработанных дней сотрудника с
        номером ', I, ': ');
  ReadLn(Sotr[i].Ko);
End;

{ Занесение данных во внешнюю память }

Assign(F, 'MyFile.fsf');
```

```
Rewrite(F);
For I := 1 to N do
  Write(F, Sotr[i]);
Close(F);
…
```

2. Чтение исходных данных из внешней памяти, расчет начисленных сумм и вывод на печать.

```
…
{ Чтение данных из внешней памяти }
Assign(F, 'MyFile.fsf');
Reset(F);
For I := 1 to N do
  Read(F, Sotr[i]);
Close(F);
{ Расчет и печать начисленных сумм }
For I := 1 to N do
Begin
  Sotr[i].S := Sotr[i].O * Sotr[i].Ko / Kr;
  WriteLn(Sotr[i].FIO, ': ', Sotr[i].S);
End;
…
```

Представленные программы решают поставленную задачу при сделанных предположениях. Необходимые для этого данные хранятся в файле MyFile.fsf, предназначенном только для решения этой задачи. Такие программные системы носят название файловых систем.

Отметим, что в этом случае описание данных включено в прикладную программу. При изменении формата записей файла необходимо изменение прикладной программы.

Задача 2. Учет кадрового состава.

Здесь обрабатываются сведения о сотруднике, представленные в карточке СОТРУДНИК:

Фамилия, имя, отчество	Должность	Год рождения	Оклад	Место жительства
FIO	*D*	*G*	*O*	*M*

Решение задачи состоит из следующих этапов:

1. Ввод исходных данных и занесение их во внешнюю память.

2. Чтение исходных данных из внешней памяти с целью удаления, корректировки или добавления записи.

```
…
{ Чтение данных из внешней памяти }
Assign(F, ‘MyFile.fsf’);
Reset(F);
IsFound := False;
For I := 1 to N do
Begin
    Read(F, Sotr);
    If Sotr.FIO = KeyFio Then
    Begin
        IsFound := True;
        Sotr.D := ‘Начальник отдела’;
        Seek(F, FilePos(F)-1);
        Write(F, Sotr);
        Break;
    End;
If IsFound Then
WriteLn(‘Корректировка успешно произведена’)
Else WriteLn(‘Сотрудника ’, KeyFio, ‘не обнаружено’);
Close(F);
…
```

Разработанная программная система также является файловой системой. В рассматриваемом случае задача 2 решается независимо от задачи 1.

Задача 3. Учет экономии фонда оплаты труда (ФОТ) в связи с болезнью сотрудников.

Обрабатываются сведения, представленные записями ЭКОНОМИЯ ФОТ:

Фамилия, имя, отчество	Оклад	Количество дней на больничном листе	Невыплаченная сумма
FIO	*O*	$K_{дв}$	*SN*

$$SN = K_{дв} O / K_r .$$

Программа решения задачи 3 аналогична программе решения задачи 1. Получается еще одна файловая система.

Рассмотрим типичный случай, когда все три файловые системы функционируют в одной организации. Отметим следующие принципиальные эксплуатационные недостатки:

1. Информация дублируется. В трех файлах присутствуют поля *FIO*, *O*, что приводит к существенному перерасходу памяти.
2. При внесении изменений (например, изменении фамилии) приходится вносить одно и то же значение несколько раз в разные файлы, что приводит к увеличению затрат машинного времени.
3. Существует потенциальная возможность противоречивости данных (в один файл изменения внесены, в другой – нет).

Устранить перечисленные недостатки можно, объединив соответствующие записи и создав единую информационную базу для всех вышеназванных задач. На первый взгляд наиболее естественно объединить все записи в одну, убрав дублирующие поля. Получаем возможный вариант объединения:

FIO	*D*	*O*	*G*	K_o	*M*	$K_{ов}$	*S*	*SN*

Дублирование информации полностью убрано. Расход памяти минимален. Недостатки 1–3 устранены. Рассмотрим, как в этом случае изменится время решения задач 1–3. Время решения задачи прямо пропорционально объему считываемых из внешней памяти данных.

Обозначим T_i, l_i, N_i соответственно время решения, длину записи, число записей i-й задачи (i = 1, 2, 3) при использовании отдельных файлов для каждой задачи:

$$T_i \approx C\ \ l_i\ N_i\,,$$

где C – некоторый коэффициент пропорциональности.

Обозначим R_i, d, N соответственно время решения i-й задачи (i = 1, 2, 3) при использовании файла объединенных записей, длину записи, число записей:

$$R_i \approx C\ d\ N.$$

Заметим, что $N_1 = N_2 = N$, $N_3 \ll N$.

Тогда время решения i-й задачи (i = 1, 2) при использовании объединенного файла увеличится в $R_i / T_i \approx d / l_i$ раз. Для нашего примера время решения задач в зависимости от выбранной длины полей может изменяться в 2–3 раза. Таким образом, платой за исключение дублирования информации является увеличение времени решаемых задач. Заметим, что такое увеличение, как правило, допустимо.

Время решения задачи 3 увеличится в $R_3/T_3 \approx d\,N / l_3\,N_3$ раз. Так как для данного примера $N_3 \ll N$, то $R_3 \gg T_3$. Время решения задачи 3 может увеличиться на несколько порядков, что совершенно недопустимо.

Рассмотрим другой вариант построения единой информационной базы. Объединим записи задач 1 и 2, запись задачи 3 оставим отдельно. Получим два типа записей:

FIO	*D*	*O*	*G*	K_o	*S*	*M*

FIO	*O*	$K_{ов}$	*SN*

В этом случае дублирование остается (дублируются поля *FIO*, *O*). Но так как $N_3 \ll N$, то общий объем дублирования незначителен. Время решения задачи 1 и 2 в этом случае незначительно возрастет по сравнению с вариантом отдельных файловых систем, время решения задачи 3 такое же, как и в начальном варианте отдельного файла. Такое объединение позволяет значительно уменьшить влияние недостатков 1–3 и в то же время существенно увеличивает время решения всех задач. Все три задачи можно решать используя общую информационную базу из двух типов записей. Отметим, что два приведенных типа записей связаны друг с другом по полю *FIO* (находятся в некотором отношении).

Таким образом, в этом случае для решения вышеуказанных задач используется некоторый новый вид данных и появляется новое понятие «база данных».

База данных – совокупность экземпляров различных типов записей и отношений между записями и элементами.

Базу данных можно определить как совокупность взаимосвязанных хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

1.2. Системы управления базами данных

В файловых системах программы решения прикладной задачи работали с данными, предназначенными только для этой задачи. За сохранность и достоверность данных отвечал программист, работающий с этой задачей.

Использование базы данных предполагает работу с ней нескольких прикладных программ, решающих задачи разных пользователей.

Естественно, что за сохранность и достоверность интегрированных данных программист, решающий одну из прикладных задач, отвечать уже не может. Кроме того, расширение круга решаемых с использованием базы данных задач может приводить к появлению новых типов записей и отношений между ними. Такое изменение структуры базы данных не должно вести к изменению множества ранее разработанных и успешно функционирующих прикладных программных систем, работающих с базой данных. С другой стороны, возможное изменение любой из прикладных программ, в свою очередь, не должно приводить к изменению структуры данных.

Все вышесказанное обусловливает необходимость отделения данных от прикладных программ.

Роль интерфейса между прикладными программами и базой данных, обеспечивающего их независимость, играет программный комплекс – система управления базами данных (СУБД) (рис. 1). СУБД – программный комплекс поддержки интегрированной совокупности данных, предназначенный для создания, ведения и использования базы данных многими пользователями (прикладными программами).

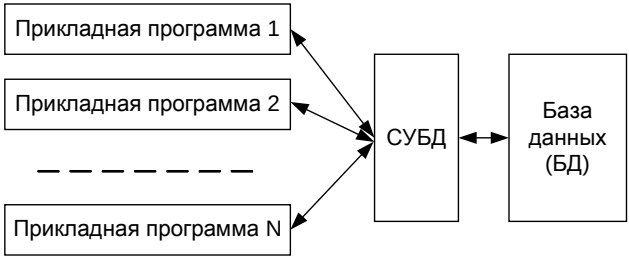


Рис.1. Обеспечение независимости прикладных программ и базы данных

Определим еще одно понятие.

Банк данных – система языковых, алгоритмических, программных, технических и организационных средств поддержки интегрированной совокупности данных, а также сами эти данные, представленные в виде баз данных.

Перечислим ряд наиболее распространенных СУБД для персональных ЭВМ.

Наиболее распространенными СУБД за последние годы были dBase-совместимые программные системы, разработанные разными фирмами. Первой широко распространенной системой такого рода была система dBase III – PLUS (фирма Achton-Tate). Развитый язык программирования, удобный интерфейс, доступный для массового пользователя, способствовали широкому распространению системы. В то же время работа системы в режиме интерпретации обусловливала низкую производительность на стадии выполнения. Это привело к появлению новых систем-компиляторов, близких к системе dBase III – PLUS: Clipper (фирма Nantucket Inc.), FoxPro (фирма Fox Software), FoxBase+ (фирма Fox Software), Visual FoxPro (фирма Microsoft). Система управления базами данных Access входит в пакет Microsoft Office 97 Professional. Одно время достаточно широко использовалась СУБД PARADOX (фирма Borland International). Фирма IBM представляет, в частности, СУБД DB2, которая в последние годы получает все большее распространение. Среди мощных СУБД, предназначенных для решения задач с использованием сети, необходимо отметить широко распространенные системы Oracle (Oracle Corp.), MS SQL-сервер (фирма Microsoft), Sybase (фирма Sybase Inc.). Кроме вышеуказанных зарубежных систем отметим и отечественную разработку – СУБД НИКА, преемницу широко распространенной в Советском Союзе СУБД ИНЕС для ЕС ЭВМ.

Перечислим основные функции системы управления базами данных.

1. Определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки.

В большинстве современных СУБД база данных представляется в виде совокупности таблиц. Рассматриваемая функция позволяет описать и создать в памяти структуру таблицы, провести начальную загрузку данных в таблицы. Примеры таких действий для СУБД MS Access и Sybase SQL Anywhere приведены на рисунках 2, 3.

Как правило, создание структуры базы данных происходит в режиме диалога. СУБД последовательно запрашивает у пользователя необходимые данные. Надо отметить, что для клиент-серверных СУБД данный диалог представляет собой графический интерфейс пользователя для формирования и выполнения соответствующих операторов языка SQL.

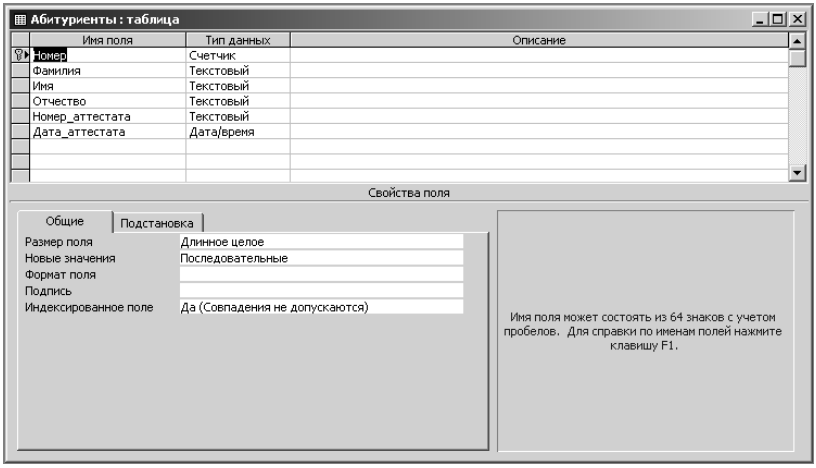
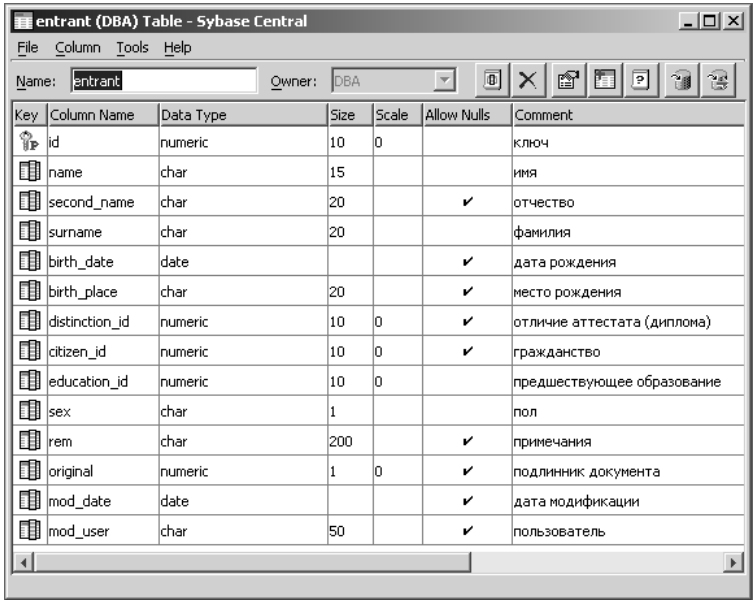


Рис. 2. Формирование структуры базы данных в СУБД Access



Key	Column Name	Data Type	Size	Scale	Allow Nulls	Comment
	id	numeric	10	0		ключ
	name	char	15			имя
	second_name	char	20		✔	отчество
	surname	char	20			фамилия
	birth_date	date			✔	дата рождения
	birth_place	char	20		✔	место рождения
	distinction_id	numeric	10	0	✔	отличие аттестата (диплома)
	citizen_id	numeric	10	0	✔	гражданство
	education_id	numeric	10	0		предшествующее образование
	sex	char	1			пол
	rem	char	200		✔	примечания
	original	numeric	1	0	✔	подлинник документа
	mod_date	date			✔	дата модификации
	mod_user	char	50		✔	пользователь

Рис. 3. Формирование структуры базы данных в СУБД Sybase

2. Предоставление пользователям возможности манипулирования данными (выполнение вычислений, разработка интерфейса ввода/вывода, визуализация).

В MS Access реализация данной функции сводится к созданию и выполнению запросов и форм ввода (рис. 4).

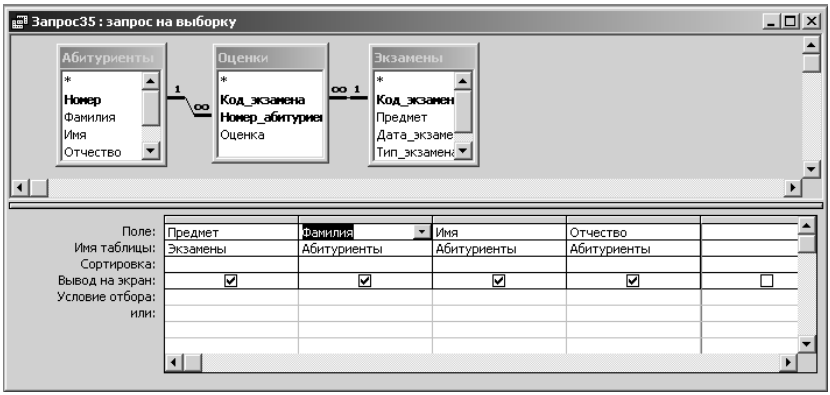


Рис. 4. Формирование запроса на выборку в СУБД Access

Для клиент-серверных СУБД существуют средства, позволяющие выполнять запросы, и программные средства, позволяющие создавать графический интерфейс пользователя.

Для Sybase SQL Anywhere средством для создания и выполнения запросов является программа Sybase ISQL, а средством создания – GUI-Sybase Power Designer. Пример работы ISQL приведен на рис. 5. Конечно, вовсе не обязательно использовать именно эти программные продукты. В настоящее время любой современный язык программирования имеет средства для доступа к базам данных.

3. Обеспечение логической и физической независимости данных.

Важнейшим свойством СУБД является возможность поддерживать два независимых взгляда на базу данных – взгляд пользователя, воплощаемый в «логическом» представлении данных, и «взгляд» системы – «физическое» представление данных в памяти ЭВМ. Обеспечение логической независимости данных предоставляет возможность изменения (в определенных пределах) «логического» представления базы данных без необходимости изменения физических структур хра-

нения данных. Таким образом, изменение «логического» представления данных в прикладных программах не приводит к изменению структур хранения данных. Обеспечение физической независимости данных предоставляет возможность изменять (в определенных пределах) способы организации базы данных в памяти ЭВМ не вызывая необходимости изменения «логического» представления данных. Таким образом, изменение способов организации базы данных не приводит к изменению прикладных программ.

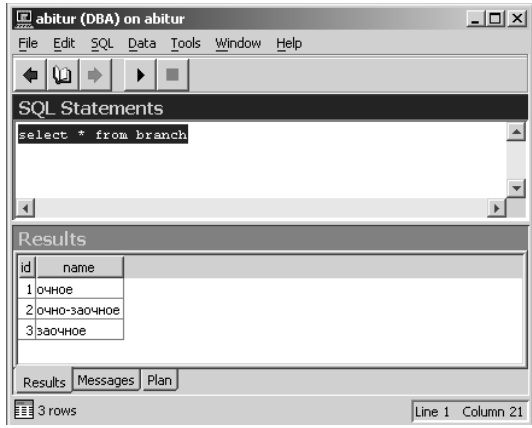


Рис. 5. Пример создания и выполнения запроса с помощью ISQL

4. Защита логической целостности базы данных.

Основной целью реализации этой функции является повышение достоверности данных в базе данных. Достоверность данных может быть нарушена при их вводе в БД или при неправомерных действиях процедур обработки данных, получающих и заносящих в БД неправильные данные. Для повышения достоверности данных в системе объявляются так называемые ограничения целостности, которые в определенных случаях «отлавливают» неверные данные. Так, во всех современных СУБД проверяется соответствие вводимых данных их типу, описанному при создании структуры. Система не позволит ввести символ в поле числового типа, не позволит ввести недопустимую дату и т.п. В развитых системах ограничения целостности описывает программист, исходя из содержательного смысла задачи, и их проверка осуществляется при каждом обновлении данных.

5. Защита физической целостности.

При работе ЭВМ возможны сбои в работе (например, из-за отключения электропитания), повреждение машинных носителей данных. При этом могут быть нарушены связи между данными, что приводит к невозможности дальнейшей работы. Развитые СУБД имеют средства восстановления базы данных. В таких системах в определенный момент БД копируется на резервные носители. Все обращения к БД записываются программно в журнал изменений. Если база данных разрушена, запускается процедура восстановления, в процессе которой в резервную копию из журнала изменений вносятся все произведенные изменения.

6. Управление полномочиями пользователей на доступ к базе данных.

Разные пользователи могут иметь разные полномочия по работе с данными (некоторые данные должны быть недоступны; определенным пользователям не разрешается обновлять данные и т.п.). В СУБД предусматриваются механизмы разграничения полномочий доступа, основанные либо на принципах паролей, либо на описании полномочий.

7. Синхронизация работы нескольких пользователей.

Достаточно часто может иметь место ситуация, когда несколько пользователей одновременно выполняют операцию обновления одних и тех же данных. Такие коллизии могут привести к нарушению логической целостности данных, поэтому система должна предусматривать меры, не допускающие обновление данных другим пользователям, пока работающий с этими данными пользователь полностью не закончит с ними работать.

8. Управление ресурсами среды хранения.

БД располагается во внешней памяти ЭВМ. При работе в БД заносятся новые данные (занимается память) и удаляются данные (освобождается память). СУБД выделяет ресурсы памяти для новых данных, перераспределяет освободившуюся память, организует ведение очереди запросов к внешней памяти и т.п.

9. Поддержка деятельности системного персонала.

При эксплуатации базы данных может возникать необходимость изменения параметров СУБД, выбора новых методов доступа, изменения (в определенных пределах) структуры хранимых данных, а также выполнения ряда других общесистемных действий. СУБД предостав-

ляет возможность выполнения этих и других действий для поддержки деятельности БД обслуживающему БД системному персоналу, называемому администратором БД.

1.3. Проблема целостности базы данных. Транзакции и блокировки

В функции современной СУБД кроме ведения собственно базы данных входит также ведение журнала транзакций. *Транзакция – это единица действий, производимых с базой данных*. В состав транзакции может входить несколько операторов изменения базы данных, но либо выполняются все эти операторы, либо не выполняется ни один.

Необходимость использования транзакций в базах данных проиллюстрируем на упрощенном примере.

Предположим, что база данных используется в некотором банке и один из клиентов желает перевести деньги на счет другого клиента банка. В базе данных хранится информация о количестве денег у каждого из клиентов. Нам нужно сделать два изменения в базе данных – уменьшить сумму денег на счете одного из клиентов и, соответственно, увеличить сумму денег на другом счете. Конечно, реальный перевод денег в банке представляет собой гораздо более сложный процесс, затрагивающий много таблиц, а возможно, и много баз данных. Однако суть остается та же – нужно либо совершить все действия (увеличить счет одного клиента и уменьшить счет другого), либо не выполнить ни одно из этих действий. Нельзя уменьшить сумму денег на одном счете, но не увеличить сумму денег на другом.

Предположим также, что после выполнения первого из действий (уменьшения суммы денег на счете первого клиента) произошел сбой. Например, могла прерваться связь клиентского компьютера с базой данных или на клиентском компьютере мог произойти системный сбой, что привело к перезагрузке операционной системы. Что в этом случае стало с базой данных? Команда на уменьшение денег на счете первого клиента была выполнена, а вторая команда – на увеличение денег на другом счете – нет. Без использования транзакций описанная выше ситуация привела бы к противоречивому, неактуальному состоянию базы данных.

Использование механизма транзакций позволяет находить решение в этом и подобных случаях. Перед выполнением первого действия выдается команда начала транзакции. Поскольку после выполнения первого действия транзакция не была завершена, изменения не будут

внесены в базу данных. Изменения вносятся (фиксируются) только после завершения транзакции. Оператор завершения транзакций обычно называется COMMIT. До выдачи данного оператора сохранения данных в базе не произойдет.

В нашем примере, поскольку оператор фиксации транзакции не был выдан, база данных «откатится» в первоначальное состояние – иными словами, суммы на счетах клиентов останутся те же, что и были до начала транзакции. Администратор базы данных может отслеживать состояние транзакций и в необходимых случаях вручную «откатывать» транзакции. Кроме того, в очевидных случаях СУБД самостоятельно принимает решение об «откате» транзакции.

Транзакции не обязательно могут быть короткими. Бывают транзакции, которые длятся несколько часов или даже несколько дней. Увеличение количества действий в рамках одной транзакции требует увеличения занимаемых системных ресурсов. Поэтому желательно делать транзакции по возможности короткими. В журнал транзакций заносятся все транзакции – и зафиксированные, и завершившиеся «откатом». Ведение журнала транзакций совместно с созданием резервных копий базы данных позволяет достичь высокой надежности базы данных.

Предположим, что база данных была испорчена в результате аппаратного сбоя компьютера, на котором был установлен сервер СУБД. В этом случае нужно использовать последнюю сделанную резервную копию базы данных и журнал транзакций. Причем применить к базе данных нужно только те транзакции, которые были зафиксированы после создания резервной копии. Большинство современных СУБД позволяют администратору воссоздать базу данных исходя из резервной копии и журнала транзакций.

С понятием транзакции тесно связано понятие блокировки. Блокировки необходимы для того, чтобы дать различным пользователям возможность одновременно работать с базой данных. При одновременной работе в некоторый момент времени разные пользователи могут обратиться к одной и той же записи или один пользователь может использовать данные, которые в данный момент меняет другой. СУБД должна запрещать подобные действия, поскольку они могут привести к ошибкам.

Для реализации этого запрета СУБД устанавливает блокировку на объекты, которые использует транзакция. Существуют разные типы блокировок – табличные, страничные, строчные и другие, которые

отличаются друг от друга количеством заблокированных записей. Чаще других используется строчная блокировка – при обращении транзакции к одной строке блокируется только эта строка, остальные строки остаются доступными для изменения.

Таким образом, процесс внесения изменений в базу данных состоит из следующей последовательности действий: выдается оператор начала транзакции, выдается оператор изменения данных, СУБД анализирует оператор и пытается установить блокировки, необходимые для его выполнения, в случае успешной блокировки оператор выполняется, затем процесс повторяется для следующего оператора транзакции. После успешного выполнения всех операторов внутри транзакции выполняется оператор фиксации транзакции. СУБД фиксирует изменения, сделанные транзакцией, и снимает блокировки. В случае неуспеха выполнения какого-либо из операторов транзакция «откатывается», данные получают прежние значения, блокировки снимаются.

1.4. Краткий обзор литературы и других доступных источников

Началом систематизированных публикаций в нашей стране, в которых рассматриваются концептуальные принципы построения баз данных, по-видимому, можно считать перевод технического отчета по анализу информационных систем общего назначения. Этот отчет был создан системным комитетом CODASYL на материалах сравнительного изучения десяти наиболее известных систем, разработанных в США [12]. Основная заслуга авторов состоит в попытке единого толкования понятий, реализуемых и обозначаемых в этих системах по-разному, а также в изложении (впервые) с единой позиции совокупности результатов, плохо совмещающихся, а иногда и противоречащих друг другу.

Классикой стала известная неоднократно переиздаваемая монография Дж. Мартина [23]. Эта монография, по сути, была первым шагом по превращению процесса разработки баз данных, основанного исключительно на интуиции и опыте разработчиков, в особую научную дисциплину.

Одним из первых фундаментальных учебников, где ясно и точно изложены основные принципы построения систем управления базами данных (книга входит в серию «Системное программирование», спонсируемую фирмой IBM; цель серии – накопление и широкое распространение принципов и технологий, имеющих большое значение для компьютерной индустрии), является книга К.Дж. Дейта [10].

Большой вклад в систематизацию вопросов разработки баз данных и их изложения внес Дж. Ульман [28, 29], долгие годы читавший курс лекций по системам баз данных. Главное отличие его книг в том, что автор большое внимание уделяет приложениям математического аппарата для дальнейшего развития теории баз данных, а также их практической разработке.

Развитию математического аппарата реляционных баз данных целиком посвящена монография Д. Мейера [24].

Вопросы автоматизированного проектирования баз данных рассматриваются Дж. Хаббардом [30].

Среди недавно изданных книг, ориентированных на специалистов, занимающихся проектированием, созданием и сопровождением баз данных, и рассчитанных также на использование в качестве учебных пособий, следует указать книги Г. Хансена и Дж. Хансена [31], Т. Конолли, К. Бэгг, А. Страчан [18]. Последняя фундаментальная монография концентрирует весь опыт разработки баз данных для промышленности, бизнеса и науки, а также обучения студентов. Она является беспрецедентно полным справочным руководством по проектированию, реализации и сопровождению баз данных. Ближайшие и долговременные перспективы развития массовых технологий баз данных излагаются в монографии А. Саймона [27].

Большое количество литературы посвящено работе в среде конкретных систем управления базами данных, например [1, 5–9, 14, 22, 25].

Детальный анализ различных аспектов технологий баз данных, включающий как методологические подходы, так и программный инструментарий, приводится в книге М. Когаловского [17] (к сожалению, быстро устаревающей).

В качестве одного из первых учебников по базам данных можно указать книгу В.Н. Четверикова и др. [35], в качестве учебника по одному, но очень важному аспекту баз данных – проектированию – книгу С.М. Диго [11]. К сожалению, оба вышеуказанных учебных пособия ориентированы на устаревшие программные средства и не могут быть использованы в полной мере.

Длительное время (1988–1999 гг.) учебные пособия по базам данных в центральных отечественных издательствах, к сожалению, не издавались. В последние годы эта ситуация стала меняться; отметим ряд учебников [15, 33], которые можно рекомендовать студентам для подготовки по отдельным аспектам баз данных.

Весьма доступным источником образовательной информации по базам данных является сеть Интернет. Так, например, на сайте FORUM www.citforum.ru выставлен учебный курс «Основы проектирования реляционных баз данных», подготовленный В.В. Кирилловым (Санкт-Петербургский государственный институт точной механики и оптики), [39] и курс С.Д. Кузнецова «Основы современных баз данных» (информационно-аналитические материалы Центра информационных технологий) [40].

Можно назвать также учебное пособие Ю.А. Зеленкова (Ярославский государственный университет) [38]. Перспективные направления баз данных изложены в информационно-аналитических материалах С.Д. Кузнецова [41, 42].

На сайте www.eManual.ru [48] представлен обзор «Серверы корпоративных баз данных» (В.З. Шнитман, С.Д. Кузнецов), посвященный аппаратно-программным платформам СУБД и конфигурации серверов баз данных. На том же сайте содержится большое количество разнообразной документации к различным конкретным СУБД, в том числе и к рассматриваемым в данном курсе. Интересные дополнительные материалы можно найти также на сайте «Открытые системы» [49] (http://www.osp.ru). Особый интерес представляет журнал «СУБД», который, к сожалению, перестал выходить как отдельное издание.

Основным недостатком Интернет-материалов является их недолговечность и неактуальность. Зачастую прекрасные учебные материалы становятся недоступными из-за ликвидации соответствующего сайта. Отмеченных недостатков лишены материалы фирм-производителей программного обеспечения – см., например, www.oracle.com [44]. Однако учебные курсы на этом сайте находятся в платном доступе, что лишает студентов и преподавателей возможности пользоваться ими. В списке литературы приводятся адреса сайтов ведущих компаний-производителей соответствующего программного обеспечения [44–47].

Приведенный выше обзор не является исчерпывающим. В Интернете появляются новые курсы и пропадают уже завоевавшие популярность. Несмотря на очевидную важность и полезность информации, размещенной в Сети, она не может быть использована как основа учебного курса, а лишь дополняет его.

По проблемам развития баз данных регулярно проводятся международные конференции, издаются ряд научных журналов. Информацию о конференциях и издаваемых научных журналах можно найти в сети Интернет. Некоторые из полезных ссылок приведены ниже.

http://www.cs.man.ac.uk/~franconi/kr.html
http://dbis-conference.informatik.tu-cottbus.de/adbis2003/topic/index.html
http://www.cs.washington.edu/homes/suciu/DBPL/
http://dke.cti.gr/SSTD03/CFP.htm
http://www.cs.man.ac.uk/~franconi/kr.html

1.5. Различные представления о данных в базах данных

Создание базы данных предполагает интеграцию данных, предназначенных для решения нескольких прикладных задач разных пользователей. Соответственно, при интеграции данных должны учитываться требования к данным каждого пользователя, основанные на его представлении о данных и связях между ними. Далее эти требования должны обобщаться в единое представление, которое и будет служить основой для построения единой базы данных (рис. 6).

Обобщение представлений всех пользователей о данных называется концептуальной моделью (схемой) БД. Концептуальная модель представляет информационное описание предметной области с учетом логических взаимосвязей, поэтому её еще называют инфологической (информационно-логической) моделью. В модели отсутствуют какие-либо понятия, связанные с ЭВМ, памятью ЭВМ, способами размещения данных в памяти ЭВМ, и, по сути, это модель только предметной области.

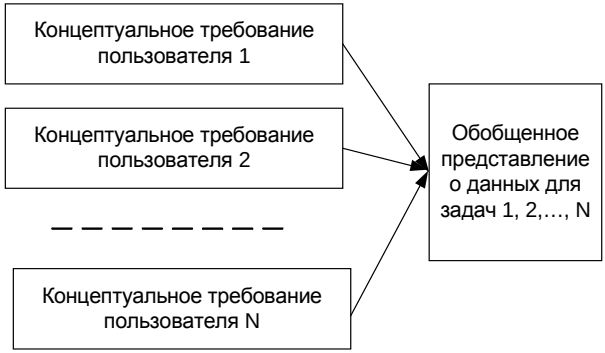


Рис. 6. Обобщение представления пользователей о данных

Как уже отмечалось, для создания базы данных и работы с ней используется система управления базами данных. Каждая конкретная

СУБД поддерживает определенный вид данных (форматов записей и отношений), называемый моделью данных.

Следующий этап разработки базы данных предполагает выбор представления концептуальной модели с помощью модели данных конкретной СУБД. Полученное таким образом представление концептуальной модели называется логической моделью БД. Или другими словами, логическая модель – это концептуальная схема, специфицированная в языке конкретной СУБД. Логическая модель представляет данные и элементы данных вне зависимости от их содержания и среды хранения. Далее разработчик системы средствами СУБД отображает полученную логическую модель БД в память ЭВМ и определяет методы доступа. Полученное представление данных в памяти ЭВМ называется внутренним представлением или структурой хранения. Прикладные программы работают с логической моделью, причем каждому пользователю представляется подмножество этой логической модели (подсхема), отражающее его представление о предметной области. Каждая прикладная программа «видит» и обрабатывает только те данные, которые необходимы именно ей.

Соответствующее «видение» данных прикладными программами (пользователями) представляет собой внешние представления. Взаимосвязь вышеуказанных моделей изображена на рис.7.

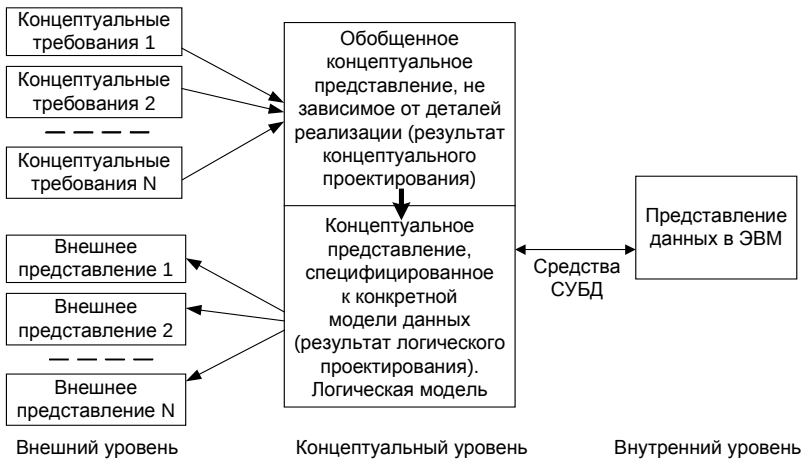


Рис. 7. Различные представления о данных в БД

На данной схеме выделены три различных уровня описания данных (внешний, концептуальный, внутренний). Эти уровни формируют так называемую трехуровневую архитектуру ANSI/SPARC, предложенную в 1975 г. Комитетом планирования стандартов и норм SPARC (Standards Planning and Requirements Committee) Национального института стандартизации США (American National Standards Institute – ANSI). Основная цель этой архитектуры состоит в отделении пользовательского представления о данных в базе данных от их физического представления.

Использование таких представлений о данных позволяет обеспечить выполнение основного требования к БД – независимости программ и данных. При изменении прикладных программ может измениться соответствующее внешнее представление, но логическая модель данных не изменяется и, соответственно, не будут изменяться другие прикладные программы. При изменении внутреннего представления (структур хранения) логическая модель не изменяется, соответственно, не изменяются прикладные программы.

Использование соответствующих представлений также позволяет четко разграничить полномочия различных лиц, работающих с базой данных.

Соответствующие представления позволяют описать «видение» базы данных разными лицами, работающими с ней:

- внешнее представление – представление специалиста предметной области (пользователя);
- внешнее представление и логическая модель – представление прикладного программиста, разрабатывающего конкретное приложение для пользователя;
- логическая модель и внутреннее представление – представление системного программиста, администрирующего базу данных.

1.6. Различные модели организации работы пользователей с базой данных

Постоянное изменение основных свойств вычислительной техники и развитие программного обеспечения обусловливало возникновение различных моделей взаимодействия пользователей с базой данных. Дадим краткую характеристику этим моделям в хронологическом порядке.

1.6.1. Модель с централизованной архитектурой

При использовании этой технологии база данных, СУБД и прикладная программа (приложение) располагаются на одном компьютере (мэйнфрейме или персональном компьютере). Для такого способа организации не требуется поддержки сети и все сводится к автономной работе. Работа построена следующим образом:

- База данных в виде набора файлов находится на жестком диске компьютера.
- На том же компьютере установлены СУБД и приложение для работы с БД.
- Пользователь запускает приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к БД на выборку/обновление информации.
- Все обращения к БД идут через СУБД, которая инкапсулирует внутри себя все сведения о физической структуре БД.
- СУБД инициирует обращения к данным, обеспечивая выполнение запросов пользователя (осуществляя необходимые операции над данными).
- Результат СУБД возвращает в приложение.
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Таким образом, в этой модели реализуется однопользовательский режим работы.

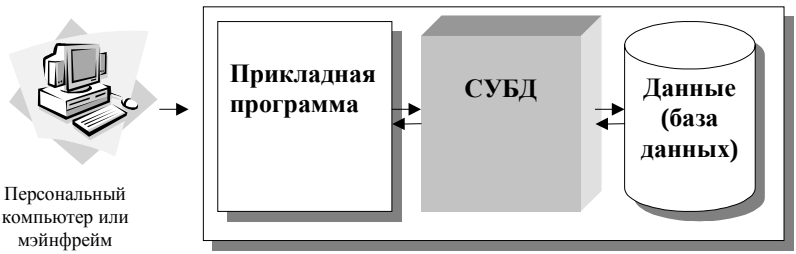


Рис. 8. Централизованная архитектура

Подобная архитектура использовалась в первых версиях СУБД DB2, Oracle, Ingres [8]. Исходная идея создания и работы с базами

данных предполагала многопользовательское использование данных. В данной модели с централизованной архитектурой реализуем и многопользовательский режим. С этой целью к мэйнфрейму подключалось несколько терминалов, но тогда приходилось обслуживать в рамках ресурсов одного компьютера весь комплекс возникающих задач – от собственно обработки и хранения данных до отображения информации и приема запросов пользователей. Модель была распространена в период «господства» больших ЭВМ (IBM-370, ЕС-1045, ЕС-1060). Все программы разных пользователей выполнялись одной ЭВМ в режиме разделения времени или мультипрограммирования. Одновременно с ростом сложности задач росло количество пользователей и объемы баз данных, вследствие чего подобная архитектура более не обеспечивала удовлетворительной производительности.

Основным недостатком этой модели является резкое снижение производительности при увеличении числа пользователей.

1.6.2. Модель с автономными персональными ЭВМ

Каждый пользователь имеет свою автономную персональную ЭВМ. База данных и СУБД копируются на компьютере каждого пользователя, и он работает с базой данных на своей ЭВМ.

Модель широко использовалась в начальный период появления персональных ЭВМ. Для нее характерна полная децентрализация данных. Основным недостатком модели является невозможность оперативного обновления данных на всех компьютерах при изменении их одним из пользователей.

1.6.3. Модель вычислений с сетью и файловым сервером (архитектура «файл-сервер»)

Увеличение сложности задач, появление персональных компьютеров и локальных вычислительных сетей явились предпосылками появления новой архитектуры «файл-сервер». Эта архитектура баз данных с сетевым доступом предполагает назначение одного из компьютеров сети в качестве выделенного сервера, на котором будут храниться файлы базы данных [36]. В соответствии с запросами пользователей файлы с файл-сервера передаются на рабочие станции пользователей, где и осуществляется основная часть обработки данных. Центральный сервер выполняет в основном только роль хранилища файлов, не участвуя в обработке самих данных [36].

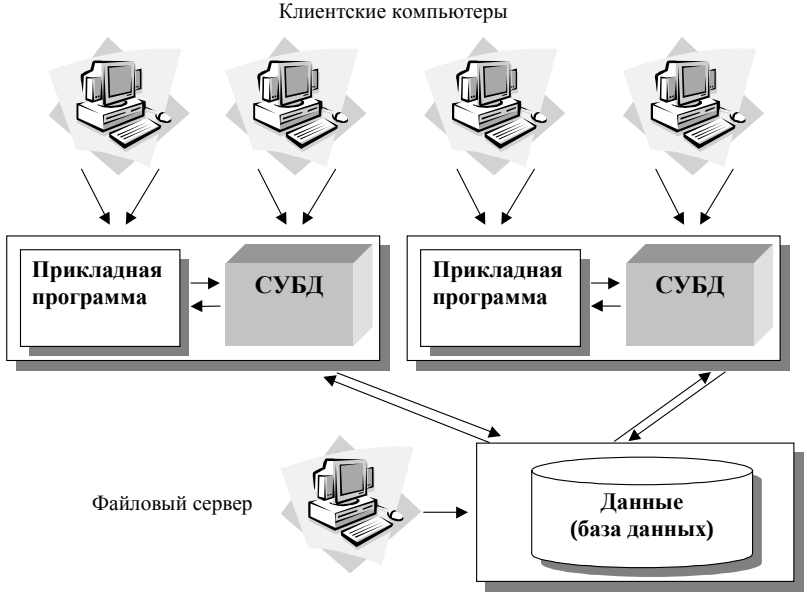


Рис. 9. Архитектура «файл-сервер»

Работа построена следующим образом:

- База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (файлового сервера).
- Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлены СУБД и приложение для работы с БД.
- На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к БД на выборку/обновление информации.
- Все обращения к БД идут через СУБД, которая инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на файловом сервере.
- СУБД инициирует обращения к данным, находящимся на файловом сервере, в результате которых часть файлов БД копируется на клиентский компьютер и обрабатывается, что обеспечивает вы-

полнение запросов пользователя (осуществляются необходимые операции над данными).

- При необходимости (в случае изменения данных) данные отправляются назад на файловый сервер с целью обновления БД.
- Результат СУБД возвращает в приложение.
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

В рамках архитектуры «файл-сервер» были выполнены первые версии популярных так называемых настольных СУБД, таких, как dBase и Microsoft Access.

В литературе [36] указываются следующие основные недостатки данной архитектуры:

- При одновременном обращении множества пользователей к одним и тем же данным производительность работы резко падает, т.к. необходимо дождаться пока пользователь, работающий с данными, завершит свою работу. В противном случае возможно затирание исправлений, сделанных одними пользователями, изменениями других пользователей.
- Вся тяжесть вычислительной нагрузки при доступе к БД ложится на приложение клиента, так как при выдаче запроса на выборку информации из таблицы вся таблица БД копируется на клиентскую машину и выборка осуществляется на клиенте. Таким образом, неоптимально расходуются ресурсы клиентского компьютера и сети. В результате возрастает сетевой трафик и увеличиваются требования к аппаратным мощностям пользовательского компьютера.
- Как правило, используется навигационный подход, ориентированный на работу с отдельными записями.
- В БД на файл-сервере гораздо проще вносить изменения в отдельные таблицы, минуя приложения, непосредственно из инструментальных средств (например, из утилиты Database Desktop фирмы Borland для файлов Paradox и dBase); подобная возможность облегчается тем обстоятельством, что фактически у таких СУБД база данных – понятие более логическое, чем физическое, поскольку под БД понимается набор отдельных таблиц, сосуществующих в отдельном каталоге на диске. Все это позволяет говорить о низком уровне безопасности – как с точки зрения хищения и нанесения вреда, так и с точки зрения внесения ошибочных изменений.

- Недостаточно развитый аппарат транзакций служит потенциальным источником ошибок в плане нарушения смысловой и ссылочной целостности информации при одновременном внесении изменений в одну и ту же запись.

1.6.4. Распределенная модель вычислений (архитектура «клиент – сервер»)

Использование архитектуры «клиент – сервер» предполагает наличие некоторого количества компьютеров, объединенных в сеть, один из которых выполняет особые управляющие функции (является сервером сети).

Так, архитектура «клиент – сервер» разделяет функции приложения пользователя (называемого клиентом) и сервера. Приложение-клиент формирует запрос к серверу, на котором расположена БД, на структурном языке запросов SQL (Structured Query Language), являющемся промышленным стандартом в мире реляционных БД. Удаленный сервер принимает запрос и переадресует его SQL-серверу БД [22, 36].

SQL-сервер – специальная программа, управляющая удаленной базой данных. SQL-сервер обеспечивает интерпретацию запроса, его выполнение в базе данных, формирование результата выполнения запроса и выдачу его приложению-клиенту. При этом ресурсы клиентского компьютера не участвуют в физическом выполнении запроса; клиентский компьютер лишь отсылает запрос к серверной БД и получает результат, после чего интерпретирует его необходимым образом и представляет пользователю. Так как клиентскому приложению посылается результат выполнения запроса, по сети «путешествуют» только те данные, которые необходимы клиенту. В итоге снижается нагрузка на сеть. Поскольку выполнение запроса происходит там же, где хранятся данные (на сервере), нет необходимости в пересылке больших пакетов данных. Кроме того, SQL-сервер, если это возможно, оптимизирует полученный запрос таким образом, чтобы он был выполнен в минимальное время с наименьшими накладными расходами [22, 36].

Все это повышает быстродействие системы и снижает время ожидания результата запроса. При выполнении запросов сервером существенно повышается степень безопасности данных, поскольку правила целостности данных определяются в базе данных на сервере и являются едиными для всех приложений, использующих эту БД. Таким образом, исключается возможность определения противоречивых правил

поддержания целостности. Мощный аппарат транзакций, поддерживаемый SQL-серверами, позволяет исключить одновременное изменение одних и тех же данных различными пользователями и предоставляет возможность откатов к первоначальным значениям при внесении в БД изменений, закончившихся аварийно [22, 36].

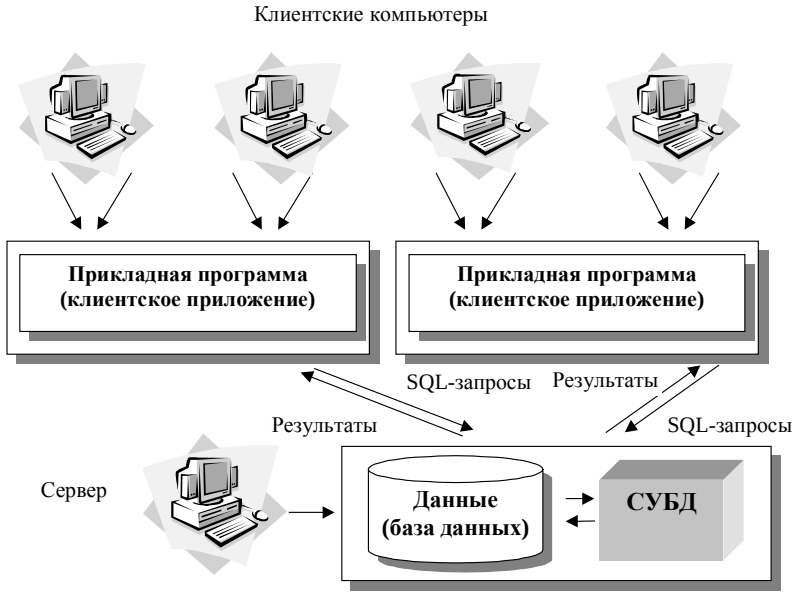


Рис. 10. Архитектура «клиент – сервер»

Итак, в результате работа построена следующим образом:

- База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (сервера сети).
- СУБД располагается также на сервере сети.
- Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлено клиентское приложение для работы с БД.
- На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к СУБД, расположенной на сервере, на выборку/обнов-

ление информации. Для общения используется специальный язык запросов SQL, т.е. по сети от клиента к серверу передается лишь текст запроса.

- СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере.
- СУБД инициирует обращения к данным, находящимся на сервере, в результате которых на сервере осуществляется вся обработка данных и лишь результат выполнения запроса копируется на клиентский компьютер. Таким образом СУБД возвращает результат в приложение.
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Рассмотрим, как выглядит разграничение функций между сервером и клиентом.

- Функции приложения-клиента:
  - Посылка запросов серверу.
  - Интерпретация результатов запросов, полученных от сервера.
  - Представление результатов пользователю в некоторой форме (интерфейс пользователя).
- Функции серверной части:
  - Прием запросов от приложений-клиентов.
  - Интерпретация запросов.
  - Оптимизация и выполнение запросов к БД.
  - Отправка результатов приложению-клиенту.
  - Обеспечение системы безопасности и разграничение доступа.
  - Управление целостностью БД.
  - Реализация стабильности многопользовательского режима работы.

В архитектуре «клиент – сервер» используются так называемые «удаленные» (или «промышленные») СУБД. Промышленными они называются из-за того, что именно СУБД этого класса могут обеспечить работу информационных систем масштаба среднего и крупного предприятия, организации, банка. Локальные СУБД предназначены для однопользовательской работы или для обеспечения работы информационных систем, рассчитанных на небольшие группы пользователей [36].

К разряду промышленных СУБД принадлежат Oracle, Gupta, Informix, Sybase, MS SQL Server, DB2, InterBase и ряд других [36].

Как правило, SQL-сервер обслуживается отдельным сотрудником или группой сотрудников (администраторы SQL-сервера). Они управляют физическими характеристиками баз данных, производят оптимизацию, настройку и переопределение различных компонентов БД, создают новые БД, изменяют существующие и т.д., а также выдают привилегии (разрешения на доступ определенного уровня к конкретным БД, SQL-серверу) различным пользователям [36].

Рассмотрим основные достоинства данной архитектуры по сравнению с архитектурой «файл-сервер»:

- Существенно уменьшается сетевой трафик.
- Уменьшается сложность клиентских приложений (большая часть нагрузки ложится на серверную часть), а следовательно, снижаются требования к аппаратным мощностям клиентских компьютеров.
- Наличие специального программного средства – SQL-сервера – приводит к тому, что существенная часть проектных и программистских задач становится уже решенной.
- Существенно повышается целостность и безопасность БД.

Однако нельзя не указать и некоторые недостатки и трудности, с которыми приходится сталкиваться при использовании данной архитектуры. Во-первых, это стоимость SQL-сервера. Внедрение архитектуры «клиент – сервер» требует существенных финансовых ресурсов. К числу недостатков можно отнести и то, что большое количество клиентских компьютеров, расположенных в разных местах, вызывает определенные трудности со своевременным обновлением клиентских приложений на всех компьютерах-клиентах. Однако эти недостатки не являются препятствием к использованию данных принципов построения корпоративных информационных систем. Так, архитектура «клиент – сервер» хорошо зарекомендовала себя на практике, в настоящий момент существует и функционирует большое количество БД, построенных в соответствии с данной архитектурой.

1.6.5. Распределенная модель вычислений (Клиент – сервер. Трехзвенная (многозвенная) архитектура)

Конец 90-х годов был ознаменован развитием сети Интернет. Сегодня мы смело можем сказать, что Интернет, как ранее вычислительная техника, проник или активно проникает во многие сферы человеческой жизни – экономику, науку, технику, образование. Развитие индустрии разработки программного обеспечения, с одной стороны, и сети Интернет как средства коммуникации, с другой стороны, привело

к появлению нового взгляда на проблему построения хранилищ информации и на принципы работы с ними. Так, в современных условиях является совершенно нормальным то, что человек, сидя дома, имеет возможность войти в глобальную сеть, посетить сайт компании, торгующей интересующими его товарами (например, книгами), и посмотреть, есть ли в наличии интересующая его книга, сколько она сейчас стоит, как обстоит дело с доставкой и, если все его устраивает, оформить заказ на приобретение книги.

Новое понимание действительности привело к возникновению новых технологий, новых подходов к хранению и обработке данных. А в чем, собственно, проблема, чем архитектура «клиент – сервер» плоха для организации такой деятельности, как описано выше в примере?

Один из основных моментов – снижение затрат на организацию рабочих мест пользователей. Так, для полноценной работы в Интернете не требуется каких-то запредельных мощностей. Необходима лишь возможность установки операционной системы и Web-браузера. Заметьте, что когда вы запрашиваете информацию о книге (выполняете запросы к базе данных Интернет-магазина), на ваш компьютер не ложится практически никакой нагрузки.

Второй момент – устранение бесконечных трудностей с обновлением клиентских приложений. Заметьте – когда Интернет-магазин меняет свое программное обеспечение, это никак не сказывается на вас, как обычном пользователе. Вы не скачиваете никаких дополнительных файлов.

Для разрешения описанных выше проблем появилось некоторое развитие архитектуры «клиент – сервер». На настоящий момент это развитие представляет собой так называемую трехзвенную (в некоторых случаях многозвенную) архитектуру (N-tier или multi-tier). Рассмотрев архитектуру «клиент – сервер», можно заключить, что она является 2-звенной: первое звено – клиентское приложение, второе звено – сервер БД + сама БД.

В трехзвенной архитектуре вся бизнес-логика (деловая логика), ранее входившая в клиентские приложения, выделяется в отдельное звено, называемое сервером приложений. При этом клиентским приложениям остается лишь пользовательский интерфейс. Так, в качестве клиентского приложения в описанном выше примере выступает Web-браузер.

Что улучшается при использовании трехзвенной архитектуры? Теперь при изменении бизнес-логики более нет необходимости изменять

клиентские приложения и обновлять их у всех пользователей. Кроме того, максимально снижаются требования к аппаратуре пользователей.

Итак, в результате работа построена следующим образом:

- База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (сервера сети).
- СУБД располагается также на сервере сети.
- Существует специально выделенный сервер приложений, на котором располагается программное обеспечение (ПО) делового анализа (бизнес-логика) [8].
- Существует множество клиентских компьютеров, на каждом из которых установлен так называемый «тонкий клиент» – клиентское приложение, реализующее интерфейс пользователя.
- На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение – тонкий клиент. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к ПО делового анализа, расположенному на сервере приложений.
- Сервер приложений анализирует требования пользователя и формирует запросы к БД. Для общения используется специальный язык запросов SQL, т.е. по сети от сервера приложений к серверу БД передается лишь текст запроса.
- СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере.
- СУБД инициирует обращения к данным, находящимся на сервере, в результате которых результат выполнения запроса копируется на сервер приложений.
- Сервер приложений возвращает результат в клиентское приложение (пользователю).
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

1.7. Краткий обзор СУБД

Первые специализированные программы для хранения и обработки данных стали появляться в конце 60-х – начале 70-х годов. СУБД помогала пользователям компьютеров организовывать и структурировать данные. Несмотря на то, что первые СУБД функционировали на больших ЭВМ (мэйнфреймах), с появлением персональных компьютеров активное использование БД и СУБД переместилось на них и стало доступным не только для индустриальных гигантов и исследователь-

ских центров, но и для обычных пользователей (малый и средний бизнес, образовательные учреждения, предприятия бытовой сферы, муниципальные организации).

СУБД сыграли важную роль в развитии компьютерных сетей и Интернета [8]. Так, вследствие развития технологий представления данных и обмена ими в сочетании с появлением современных сетевых протоколов произошел отход от представления вычислительной системы как отдельно стоящего персонального компьютера сначала в сторону многопользовательских систем, выполненных в рамках технологий компьютерных сетей, территориально ограниченных рамками одного здания, а далее в направлении распределенных баз данных, территориально расположенных на различных компьютерах, иногда в разных частях света. В настоящее время благодаря сети Интернет пользователю достаточно иметь лишь Web-браузер для успешной работы с базой данных, находящейся далеко за пределами здания, в котором расположен центральный офис его организации.

В литературе неоднократно можно встретить упоминания о том, что сегодня рынок СУБД – это большой бизнес. Независимые компании по производству программного обеспечения и крупные поставщики продают программы для управления базами данных на миллиарды долларов ежегодно. В большинстве корпоративных приложений, обеспечивающих ежедневную деятельность крупных компаний и организаций, используются базы данных. Эти приложения подразделяются на несколько быстро развивающихся категорий [8]:

- Планирование ресурсов предприятий (Enterprise Resource Planning – ERP).
- Регулирование отношений с клиентами (Customer Relationship Management – CRM).
- Управление системой поставок (Supply Chain Management – SCM).
- Автоматизация продаж (Sales Force Automation – SFA).
- Финансовые приложения.

Производители компьютерного оборудования разрабатывают и поставляют серверы, которые специально сконфигурированы для функционирования в качестве серверов баз данных. Коммерческое использование этих систем дает ежегодный оборот в миллиарды долларов. Базы данных обеспечивают информацией большинство Web-узлов, ориентированных на транзакции, и используются для отслеживания и анализа взаимодействия с Web-узлами. Таким образом, про-

блема управления базами данных затрагивает все сегменты рынка компьютерных технологий [8].

Многие авторы классифицируют СУБД на две большие категории: так называемые «настольные» и «серверные».

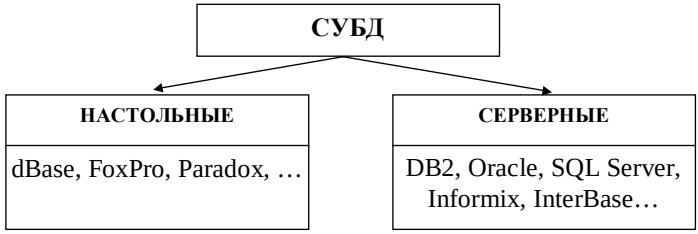


Рис. 11. Настольные и серверные СУБД

1.7.1. Настольные СУБД

Несмотря на высказывания многих авторов, что время этих СУБД прошло, они по-прежнему используются и некоторые из них достаточно активно. К числу подобных СУБД относятся dBase, FoxPro, Paradox, MS Access.

Конечно, настольные СУБД обладали, обладают и будут обладать всеми недостатками файл-серверной архитектуры. Не вызывают сомнения упреки в плохой защищенности данных, медленной работе, трудностях с поддержкой ограничений целостности, проблемах с дублированием данных при миграции и резервном копировании, трудностях администрирования, катастрофического снижения скорости обработки при возрастании объемов данных и т.д. и т.п.

Однако, на наш взгляд, используемые для решения проблемы средства должны соответствовать сложности самой проблемы. Так, вряд ли имеет смысл тратить на разработку и внедрение информационной системы средства, существенно большие, чем весь годовой оборот предприятия, а для многих предприятий сферы малого (а возможно, и среднего) бизнеса дело обстоит именно так. Следует понимать, что расходы на приобретение готового программного обеспечения (в частности, серверной СУБД), а также разработку соответствующей информационной системы, функционирующей под управлением этой СУБД, составят от нескольких десятков тысяч до нескольких миллионов долларов.

Итак, где же и как используются на сегодняшний день перечисленные выше СУБД? Прежде всего, это государственные (муниципальные) учреждения, сфера образования, сфера обслуживания, малый и средний бизнес. Специфика возникающих там задач заключается в том, что объемы данных не являются катастрофически большими, частота обновлений не бывает слишком высокой, организация территориально обычно расположена в одном небольшом здании, количество пользователей колеблется от одного до 10–15 человек. В подобных условиях использование настольных СУБД для управления информационными системами является вполне оправданным, и они с успехом применяются.

Более того, последние версии настольных СУБД приобрели такие качества, необходимые для нормальной работы, как поддержка ограничений целостности и механизма транзакций.

Некоторые настольные СУБД функционируют в среде Microsoft Windows, а также «обзавелись» средствами реализации оконного пользовательского интерфейса, например Microsoft Access (рис. 12) и Visual FoxPro.

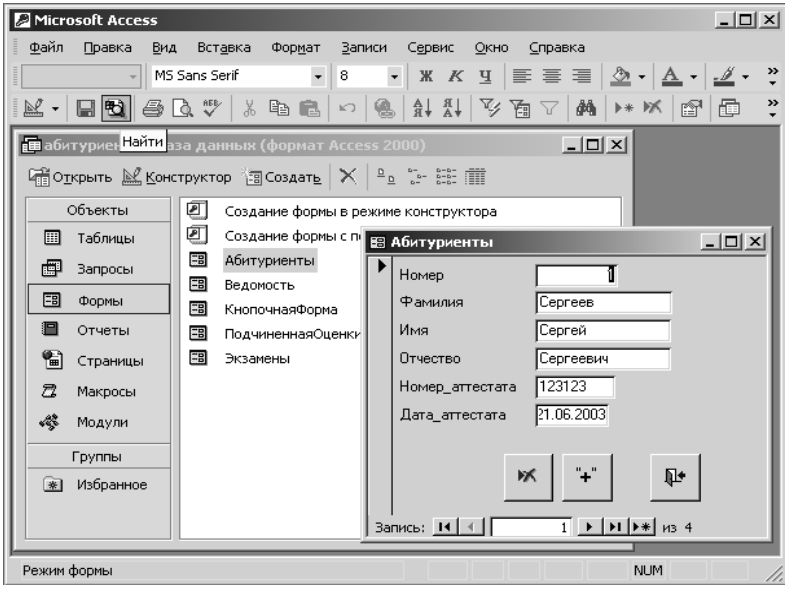


Рис. 12. Пользовательский интерфейс СУБД Access

Для тех СУБД, новые версии которых более не выпускаются, жизнь все равно не кончилась. Так, до настоящего времени используются базы данных, таблицы которых выполнены в формате dBase или Paradox. Совершенно понятно, что изначально эти СУБД были рассчитаны на работу в среде MS DOS, однако современные средства доступа к данным позволяют с успехом использовать их под Windows, не особенно различая, какой конкретно из форматов представления таблиц используется. В частности, разработанный фирмой Borland механизм BDE (Borland Database Engine) предоставляет средства для работы с таблицами dBase, FoxPro, Paradox, Access, которые инкапсулируют внутри себя всю информацию о структуре таблиц, делая для прикладного программиста «прозрачной» работу с данными. Так, текст программы, обрабатывающей базу данных, выполненную в формате одной из этих систем, вряд ли будет зависеть от того, с какой из систем действительно ведется работа.

1.7.2. Серверные СУБД

Однако для крупных организаций ситуация принципиально меняется. Там использование файл-серверных технологий является неудовлетворительным по описанным выше причинам. Поэтому на передний край борьбы за автоматизацию выходят так называемые серверные СУБД, разработкой которых активно занимаются компании IBM, Informix, Inprise, Microsoft, Oracle, Sybase. Так, в настоящий момент на рынке серверных СУБД видное место занимают DB2, Informix, MS SQL Server, Oracle, Sybase, Interbase. Вот некоторые выдержки из рекламных проспектов, взятые с сайтов www.interface.ru и www.ibm.com.ru:

«**Microsoft SQL Server 2000** – это законченное предложение в области баз данных и анализа данных для быстрого создания масштабируемых решений электронной коммерции, бизнес-приложений и хранилищ данных. Оно позволяет значительно сократить время выхода этих решений на рынок, одновременно обеспечивая масштабируемость, отвечающую самым высоким требованиям. В сервер SQL Server 2000 включена поддержка языка XML и протокола HTTP, средства повышения быстродействия и доступности, позволяющие распределить нагрузку и обеспечить бесперебойную работу, функции для улучшения управления и настройки, снижающие совокупную стоимость владения. Кроме того, SQL Server 2000 полностью использует все возможности операционной системы Windows, включая поддержку до 32 процессоров и 64 ГБ ОЗУ.

Пример пользовательского интерфейса СУБД SQL Server приводится на рис. 13.

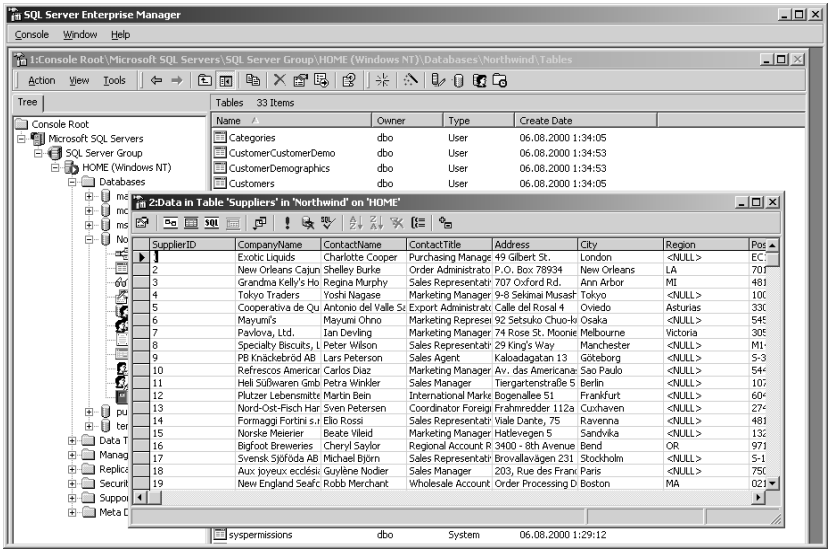


Рис. 13. Пользовательский интерфейс СУБД Microsoft SQL Server

Достоинства MS SQL Server:

- Широкая поддержка языка XML и стандартов Интернета.
- Удобный доступ к данным через Web.
- Эффективные средства анализа данных на базе Web.
- Платформа для безопасного размещения приложений.
- Масштабируемость для электронной коммерции.
- Масштабируемость для бизнес-приложений.
- Масштабируемость для хранилищ данных.
- Значительно увеличенные продолжительность бесперебойной работы и надежность.
- Интегрированные и расширяемые службы анализа.
- Упрощенное управление и настройка.
- Быстрые преобразование данных, разработка и отладка» [47].

«**Oracle9i Database** нацелена на недавно сложившийся рынок Интернет-приложений и отвечает самым строгим требованиям к качеству

обслуживания. Она обладает возможностями кластеризации, мощными и экономичными средствами безопасности, исключает потери данных и позволяет интерактивно обмениваться информацией. Oracle9i Database удовлетворит все потребности вашего электронного бизнеса в Интернете.

Основные качества:

- Масштабируемость (модуль Oracle Real Application Clusters обеспечивает масштабируемость, не зависит от используемых компонентов и позволяет масштабировать вашу систему своими силами).
- Высокая доступность (Oracle9i позволяет организовать непрерывный доступ к данным, практически исключая запланированные и аварийные задержки).
- Управление системами (встроенные в Oracle9i средства управления системами позволяют контролировать все жизненно важные компоненты, занятые в процессах электронного бизнеса).
- Безопасность в Oracle9i (Oracle9i – одна из самых безопасных платформ Интернета для защиты информационного актива вашей компании).

Новые возможности:

- Обмен деловой информацией и хранилища данных.
- Oracle9i Dynamic Services.
- Java и XML в Oracle9i.
- Любой масштаб СУБД.
- Любые компьютерные платформы и архитектуры.
- Любые типы приложений.
- Любые типы данных.
- Переносимость приложений на платформе Oracle» [44].

«**IBM DB2 Universal Database**:

- Поддерживает новейшие стандарты Java™, а также другие технологии Web.
- Предоставляет интегрированные инструменты для интеллектуального управления бизнесом.
- Расширяет объектно-реляционные возможности.
- Сохраняет лидерство в области систем управления данными с высокой доступностью и большой производительностью.
- Обеспечивает централизованное управление гетерогенными системами.

- Позволяет выполнять более глубокую обработку данных.
- Поддерживает широкий диапазон операционных систем, в том числе Linux®» [46].

«SQL-сервер баз данных **Borland InterBase 6** объединяет простоту использования, низкие затраты на сопровождение и мощность систем корпоративного уровня. Borland гарантирует, что InterBase 6 совмещает силу мощной апробированной архитектуры с развитыми технологиями, необходимыми для успеха прикладных систем.

Основные качества:

- Повышенная производительность за счет развитой архитектуры.
- Многопотоковая архитектура.
- Поддержка Java.
- Высокая надежность всех ваших приложений.
- Мощная поддержка различных типов данных.
- Сигнализаторы событий.
- Самонастройка и простота инсталляции.
- Реальная идентичность функциональных возможностей.
- Независимость от клиента и инструментария.
- Эффективность использования ресурсов.
- Строгое соблюдение индустриальных стандартов.
- Поддержка интернациональных требований бизнеса.
- InterBase: Embed.Deploy.Relax.
- Репликация в InterBase» [47].

Мы привели краткий обзор незначительной части существующих СУБД. При огромном разнообразии СУБД вполне естественны споры (которые возникли с момента появления СУБД и, по-видимому, не утихнут никогда) о том, какая СУБД лучше. Нам представляется, что однозначного ответа на этот вопрос не существует. Каждая СУБД имеет свои области применимости, у каждой из них существуют свои достоинства и недостатки. Любой читатель может иметь личные пристрастия, которые в его глазах уменьшают недостатки и увеличивают достоинства некоторой конкретной СУБД. Мы считаем, что это нормально. Общие рекомендации о том, какой из СУБД воспользоваться в каком-то конкретном случае, придумать сложно. Рекомендация может быть только одна: внимательно изучайте обзоры, читайте пресс-релизы, знакомьтесь с отзывами пользователей, сопоставляйте их наблюдения о плюсах и минусах систем с вашими потребностями и возможностями.

1.8. Основные этапы проектирования базы данных

Проектирование данных (базы данных) представляет собой процесс отображения исследуемых явлений реального мира в виде данных в памяти ЭВМ.

Конкретные явления реального мира, представляющие интерес для проводимого исследования, будем называть предметной областью.

Проектирование (моделирование) базы данных представляет собой многоэтапный процесс. Основные этапы этого процесса приведены на рис. 14).

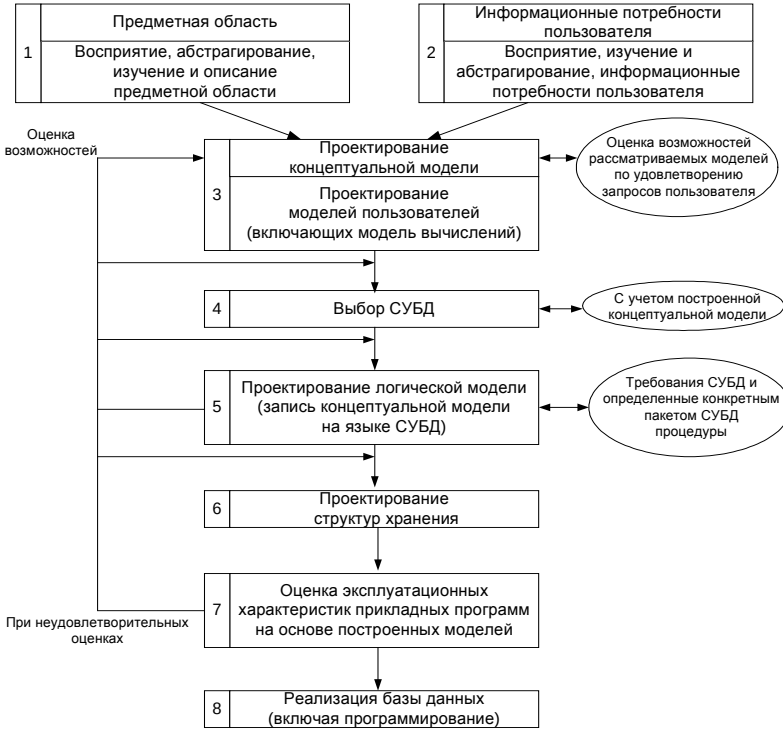


Рис. 14. Этапы проектирования базы данных

Заметим, что в представленном процессе проектирования достаточно часто возникает необходимость возврата на один или несколько шагов назад. Пусть, например, при проектировании логической модели (блок 5) не удается достичь адекватного представления концептуальной модели средствами модели данных СУБД. В этом случае необходимо либо вернуться на шаг назад и выбрать другую СУБД, либо вернуться к блоку 3 и изменить вид концептуальной модели. Так же, если полученные при реализации блока 7 оценки эксплуатационных характеристик не отвечают требованиям пользователя, возможны пересмотры всех ранее полученных решений (блоки 7, 6, 5, 4, 3).