Московский государственный университет экономики, статистики и информатики (МЭСИ)

Курс лекций «Базы данных»
Язык запросов QBE
Экранные формы
Отчетные формы

Начальник отдела НИЧ, к.э.н., доцент Д.Г. Корнеев

2009 год

Табличный язык запросов QBE

- В современных СУБД широко используются табличные языки запросов. Наиболее распространенным среди них является язык QBE (Query-By-Example - запрос по примеру).

- Язык QBE предназначен для работы в интерактивном режиме и ориентирован на конечного пользователя. Язык QBE реализован во многих современных СУБД, например в dBase IV и более старших версиях этой системы, Paradox, Access и др. Конкретные реализации этого языка несколько отличаются друг от друга, но все они построены по единому принципу.

- Суть подхода, воплощенного в языке QBE, заключается в следующем. В окне формирования запроса выделяются две зоны. В первой из них высвечивается «скелет» (образ, форма, структура) одной или нескольких таблиц, данные из которых будут участвовать в запросе. В качестве исходных для запроса могут указываться не только базовые таблицы, но и другие запросы.

Табличный язык запросов QBE

Во второй зоне («скелете» запроса табличной формы) пользователь задает условия запроса. В этой зоне пользователь определяет, какие поля участвуют в формировании запроса, а также условия отбора и некоторые другие характеристики запроса.

Например, если пользователю необходимо получить все записи с заданным значением конкретного атрибута, то в соответствующем столбце «скелета» указывается это значение.

На рис. (следующий слайд) представлен запрос к таблице, содержащей сведения о сотрудниках (Kadr) и включающей следующие атрибуты:

- FAM - фамилия;
- IMIA - имя;
- TABN - табельный номер;
- VOZR - возраст;
- POL - пол;
- ADR - адрес.

Табличный язык запросов QBE

Kadr	FAM	IMIA	TABN	VOZR	POL	ADR
				40		

Рис. 6.1. Простой запрос

Требуется выдать информацию обо всех сотрудниках в возрасте 40 лет.
В соответствующем столбце таблицы (VOZR) указывается цифра 40.
В столбце можно записывать не только значение атрибута, но и знак операции сравнения; по умолчанию принимается знак равенства («=»).

Табличный язык запросов QBE

Задание сложных запросов. Допускается задание и простых запросов, включающих только один аргумент поиска, и сложных запросов, компоненты которых связаны операторами AND (И) или OR (ИЛИ). Операторы AND и OR в явном виде не указываются при формулировании запроса на QBE. При отображении запросов на экране используется следующее правило: *если в сложном запросе его компоненты представляют разные атрибуты, которые должны быть связаны оператором AND, то они записываются в одной строке (рис. 6.2).*

На рис. 6.2 изображен запрос: «Выдать информацию о сотруднике с фамилией Диго и именем Светлана»

Kadr	FAM	IMIA	VOZR	POL	ADR
	"Диго"	"Светлана"			

На одной строке

Рис. 6.2. Сложный запрос с оператором AND

Табличный язык запросов QBE

Если компоненты запроса должны быть связаны операторами OR, то они записываются на разных строках (рис. 6.3).

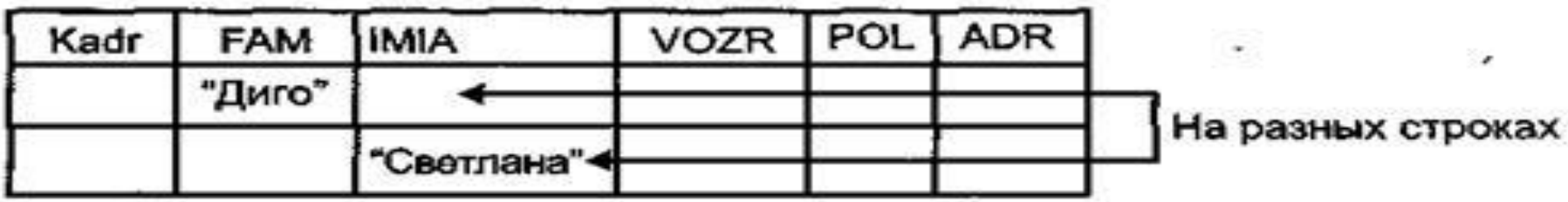

Рис. 6.3. Сложный запрос с оператором OR

На рис. 6.3 изображен запрос - «Выдать информацию о сотрудниках, имеющих либо фамилию Диго, либо имя Светлана».

В связи с тем, что интерпретация запроса зависит от взаимного расположения элементов сложного запроса на строках экрана, такого рода языки запросов называются *табличными двухмерными*.

Табличный язык запросов QBE

Как указывалось выше, при задании запроса в QBE экран обычно делится на две зоны: зона, в которой указываются данные, исходные для запроса, и зона, в которой описывается ответ. В некоторых реализациях языка при описании отдельных видов запросов появляются дополнительные зоны (например, в dBase IV при задании вычисляемого поля).

Вид, в котором представляются структуры исходных таблиц, а также то, где фиксируются условия поиска, могут различаться в конкретных системах.

Так, в dBase IV таблицы как в зоне «запроса», так в зоне «ответа» представляются в табличном виде, а условия отбора записей указываются в таблицах зоны «запроса». В Access, FoxPro исходные таблицы представлены в анкетной форме (поля таблицы перечисляются один под другим), а в зоне «ответа» в табличной форме отображаются те атрибуты (поля), которые будут выдаваться в ответе. Условия отбора записей задаются в зоне «ответа».

Табличный язык запросов QBE

Переменные для примера. В некоторых случаях при формулировке запроса необходимо использовать так называемые переменные для примера (или «наполнители»).

Переменные для примера (example variables) также записываются в определенных графах таблицы, но они обозначают не какое-либо определенное значение, а любое. Конкретное значение наполнителя несущественно. Переменные для примера используются для установления связей между атрибутами в одной или нескольких таблицах.

Переменные, применяемые для задания значений ключей поиска, и переменные, указываемые для примера, должны при записи запроса отличаться друг от друга.

В разных СУБД «наполнители» и обычные значения атрибутов поиска различаются по-разному: в некоторых системах «наполнители» подчеркиваются, в других - используются специальные ограничители при указании переменных в запросе, в третьих - такое понятие вообще не вводится и т.п.

Табличный язык запросов QBE

Совместная обработка нескольких таблиц. В некоторых запросах могут потребоваться данные из нескольких таблиц. Например, в базе данных, кроме таблицы «Кадры» (KADR),

- FAM - фамилия;
- IMIA - имя;
- TABN - табельный номер;
- VOZR - возраст;
- POL - пол;
- ADR - адрес.

имеется таблица «Выработка» (VRBT) с полями:

- TABN - табельный номер;
- DAT - дата;
- KODDET - код детали;
- KOLV - количество.

В запросе «Выдать информацию о выработке рабочего Евгения Петрова» *необходима совместная обработка таблиц VRBT и KADR*, так как в таблице «Выработка» нет сведений о фамилиях и именах рабочих.

Табличный язык запросов QBE

«Скелеты» всех таблиц, которые нужны для реализации запроса (в нашем примере - двух таблиц), должны быть вызваны на экран.
Дальнейшие действия, которые необходимо выполнить, чтобы осуществить связывание таблиц, будут зависеть от используемой СУБД. Так, в некоторых системах для связывания таблиц используются «наполнители». Их значения могут быть любыми, но они должны быть одинаковыми в обеих связываемых таблицах.

В примере, представленном на рис. 6.4, в качестве наполнителя используется буква А, и она подчеркивается.
подчеркивается.

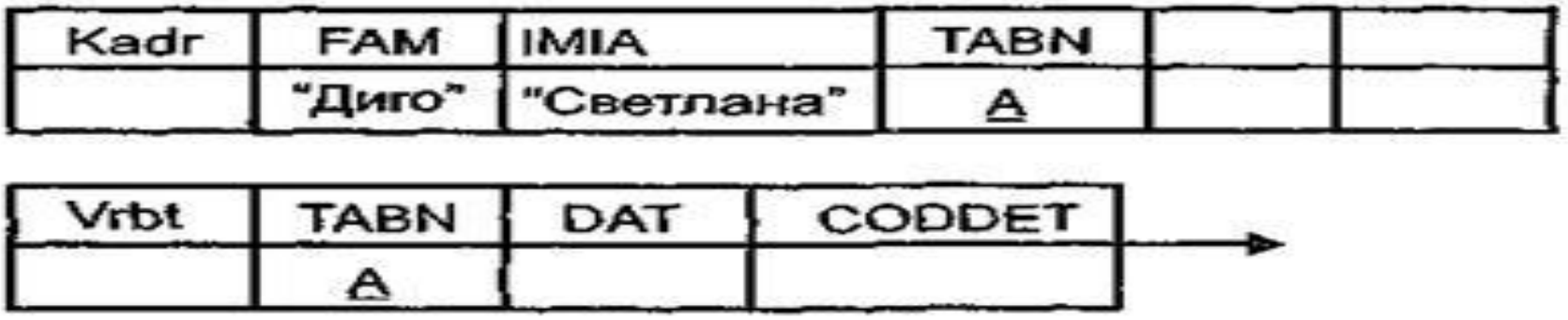

Рис. 6.4. Запрос со связыванием таблиц

Табличный язык запросов QBE

В более поздних версиях СУБД используются визуальные способы установления связей между таблицами: для связывания таблиц нужно мышью позиционироваться на нужном поле в основной таблице и, не отпуская кнопки мыши, переместиться к полю в зависимой таблице. На экране появится линия, связывающая таблицы.

Существуют и другие способы установления связей.

Теоретически возможны разные типы соединений таблиц. *Наиболее распространенным является соединение, при котором в результатную таблицу помещаются те соединенные записи, для которых значение поля связи основной таблицы совпадает с соответствующим полем в зависимой таблице.* В описанных выше случаях устанавливается именно такое соединение.

Табличный язык запросов QBE

В настоящее время широко используются такие понятия, как «левое» и «правое» соединение, когда в результатную таблицу помещаются все записи из основной или зависимой таблицы соответственно, даже если для них нет связанных записей в другой таблице.

Но не все системы позволяют в QBE реализовывать такие соединения.

В случаях, когда возможно задание разных типов соединений, конкретный способ реализации отличается в разных СУБД. Так, в Access «левое» и «правое» соединения можно определить, задав для связи «параметры объединения» или перейдя в SQL.

В dBase IV никаких специфических терминов для обозначения такого типа соединений нет, но включение слова Every в запрос на QBE выполняет ту же роль.

Табличный язык запросов QBE

Работа с несколькими таблицами в конкретных СУБД различается не только тем, каким способом можно определить связь между таблицами.

Так, например, некоторые системы обязывают пользователя связать те таблицы/файлы, которые указываются как исходные для запроса; другие автоматически связывают открытые файлы по тем полям, которые система воспринимает как поля связи (чаще всего это поля, имеющие одинаковые имена, тип и длину); третьи - оставляют эти таблицы изолированными, если пользователь не указал, как они должны быть связаны, четвертые - выполняют декартово произведение открытых таблиц.

Например, в dBase IV вызвать несколько файлов БД на панель запросов и не связать их было нельзя.

В MS Query, Access если таблицы не связаны, то при выполнении запроса это приводит к связыванию каждой записи одной таблицы с каждой записью другой (декартово произведение).

Табличный язык запросов QBE

Описание ответа. Кроме задания условия отбора данных, при описании запроса должна быть возможность указать, какие атрибуты и в какой последовательности входят в ответ. В ответ могут выдаваться не только реальные поля, которые хранятся в одной из базовых таблиц, но и вычисляемые поля.

Можно выделить два вида вычислений, которые могут выполняться в запросах, формах, отчетах: *это агрегирующие операторы, которые выполняют операции над группой записей, и обычные вычисления, затрагивающие отдельные поля одной или нескольких связанных записей*.

Агрегирующие показатели могут быть включены не только в «Запросы», но и в «Отчеты». Возможности включения агрегирующих показателей в запросы и отчеты различаются между собой.

Результатом запроса всегда является плоская таблица. Поэтому в запросах могут быть получены только одноуровневые итоги. В отчетах же может быть получено несколько степеней итогов.

Табличный язык запросов QBE

Набор агрегирующих функций может быть различным в разных системах. Обычно во всех реализациях СУБД включены следующие функции: Sum (сумма), Min (минимум), Max (максимум), Avg (среднее), Count (подсчет).

Некоторые системы включают дополнительные статистические функции, такие, как отклонение, стандартное отклонение, дисперсия и др.

Результаты вычислений, выводящиеся в поле, не запоминаются в базовой таблице. Вместо этого вычисления снова проводятся всякий раз, когда выполняется запрос, поэтому результаты всегда представляют текущее содержимое базы данных. Обновить вычисленные результаты вручную невозможно (таблица, содержащая вычисляемое поле, имеет статус «только для чтения»).

Табличный язык запросов QBE

Для удобства восприятия ответа часто требуется определить упорядоченность данных в ответе. Язык QBE обеспечивает такую возможность.

Опять-таки возможности задания упорядочения ответа различаются в разных СУБД:

- некоторые системы разрешают проводить упорядочение по произвольным полям,
- другие требуют, чтобы поле упорядочения стояло в ответе обязательно первым, а если упорядочение ведется по нескольким полям, то чтобы эти поля следовали в ответе друг за другом в порядке их старшинства;
- некоторые СУБД различают обычное и словарное упорядочение (когда учитывается и не учитывается регистр соответственно), другие - нет;
- в некоторых системах, даже если не задано никакое упорядочение, ответ всегда выдается упорядоченным по первому полю таблицы ответа и т.п.

Табличный язык запросов QBE

Дополнительные возможности. Кроме собственно поисковых запросов язык QBE позволяет выполнять и другие операции, например корректировку данных. Набор допустимых операций, а также способы их задания несколько различаются в разных системах.

Кроме того, некоторые СУБД позволяют формировать запросы специальных видов: параметрические, перекрестные и некоторые другие (не все из них, наверное, могут быть отнесены к QBE, но они реализованы одними и теми же компонентами СУБД).

Запросы, сформулированные на QBE, могут быть запомнены для их последующего многократного использования.

Понятие, классификация и роль экранных форм

- Ввод и просмотр данных в режиме таблицы имеет много очевидных недостатков. Поэтому, как правило, для удобства пользователей создаются экранные формы, которые позволяют рационально расположить данные на экране, использовать разнообразные элементы оформления, обеспечивать возможности специфических проверок при вводе данных. Во многих СУБД имеются возможности использования Мастеров, автоматизирующих процесс создания экранных форм.

- Генераторы экранных форм являются компонентом языков 4-го поколения.

- Использование генераторов экранных форм позволяет практически без программирования создавать довольно сложные системы обработки данных с хорошим пользовательским интерфейсом.

- Если первоначально формы определялись только как способ отображения одной записи из БД, то сейчас в формах допускается наличие табличной (многострочной) части; более того, допускается наличие множества «динамических» частей, возможно, из разных БД.

Понятие, классификация и роль экранных форм

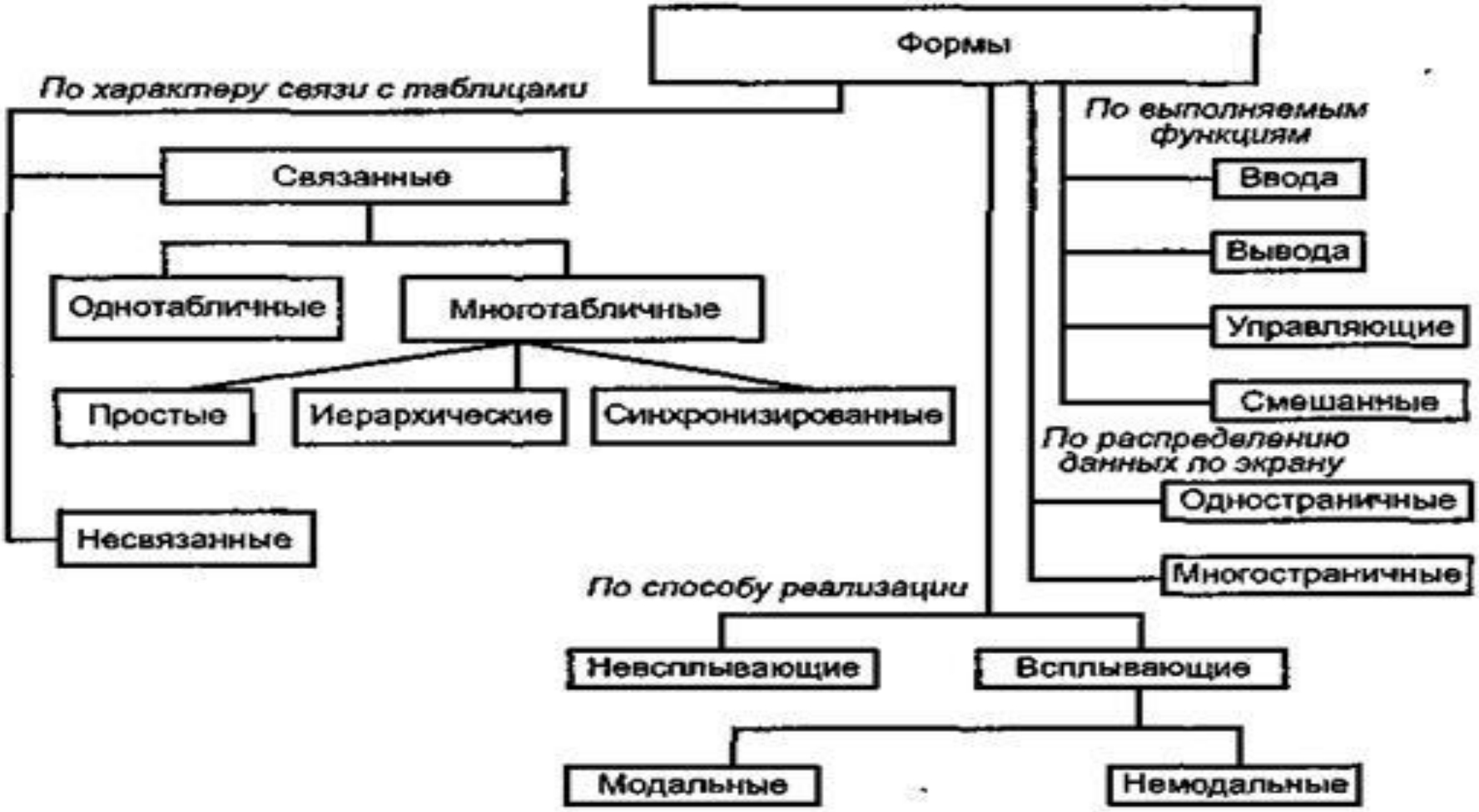

Рис. 8.1. Классификация экранных форм

Понятие, классификация и роль экранных форм

1. По **характеру связи с таблицами** различают связанные и не связанные экранные формы. Если форма отражает какие-либо данные из таблиц баз данных, она называется связанной (или присоединенной), в противном случае - *несвязанной*.

По *числу используемых таблиц* выделяют *однотабличные* и *многотабличные* формы.

По характеру соподчинения отдельных частей многотабличные формы классифицируются как *простые*, *иерархические* и *синхронизированные*.

Простые многотабличные формы хотя и содержат данные из разных таблиц, но не имеют в своем составе соподчиненных частей.

Такие формы могут возникнуть, например, когда они базируются на таблицах, связанных друг с другом отношением 1:1, или когда в форму выводятся данные из таблиц, связанных друг с другом отношением 1 :М, но в форму в основном выводятся данные, находящиеся со стороны «М», а из таблицы, находящейся со стороны «1», берутся какие-то, обычно справочные, данные, *т.е. ведущим здесь как бы является таблица, находящаяся со стороны «М».*

Понятие, классификация и роль экранных форм

Но наиболее естественной для многотабличных форм все-таки является ситуация, когда ведущая таблица находится на стороне «1».

В этом случае создаются иерархические формы, когда в форму в ее общей части выводятся данные из одной записи ведущей таблицы, а в табличной части - множество связанных с ней записей ведомого (зависимого) файла.

Иногда (по разным причинам) бывает нецелесообразно выводить в одну иерархическую форму данные и из основного, и из зависимого файла, и данные из зависимого файла выводятся в отдельной «зоне», которая открывается «при нажатии» соответствующей управляющей кнопки.

Такие формы называются синхронизированными.

Понятие, классификация и роль экранных форм

2. По **выполняемым функциям** различают формы *ввода, вывода, управляющие, смешанные*. Назначение каждого вида этих форм ясно из их названия. Следует обратить внимание на то, что даже в случае, если формы для ввода и вывода полностью совпадают по своему внешнему виду, иногда целесообразно их выполнить как самостоятельные формы в целях обеспечения безопасности данных.

3. По **распределению данных по экранам** (страницам) формы делятся на *одностраничные* и *многостраничные*; одной из разновидностей многостраничных форм можно считать *формы с вкладками*.

Понятие, классификация и роль экранных форм

4. По **способу реализации** экранные формы могут быть:

- *всплывающими;*
- *невсплывающими*.

Всплывающая форма располагается поверх других открытых форм, даже если активной является другая форма.
Всплывающая форма может быть:

- *немодальной;*
- *модальной.*

Если всплывающая форма - *модальная,* пользователь имеет возможность получить доступ к другим объектам и командам меню, пока форма открыта. Если всплывающая форма является *немодальной*, нельзя получить доступ к любым другим объектам или командам меню, пока форма открыта. Пользователь должен выполнить какое-либо действие, чтобы фокус был переключен на другую форму (или окно).

Понятие, классификация и роль экранных форм

5. По **форме представления информации** экранные формы могут содержать символьную информацию, деловую графику, информацию, представленную в мультимедийной форме.
Например, в БД, хранящей информацию о животных, наряду с описанием каждого вида может выводиться его изображение.

Несмотря на такое широкое применение экранных форм для реализации разных целей, основное внимание далее уделим *их использованию для организации ввода данных в БД*, потому что, во-первых, это является одним из основных назначений экранных форм; во-вторых, именно этот аспект наиболее значим для процессов создания и ведения БД; в-третьих, такие функции, как создание меню, вывод информации из БД и т.п., могут выполняться с использованием и других средств СУБД.

Рекомендации по созданию экранных форм

Рассмотрим основные рекомендации по созданию форм.

1. Порядок размещения элементов в форме будет зависеть от типа и назначения формы.

Порядок расположения полей на экране для форм, использующихся *для вывода информации*, определяется в основном смысловой группировкой информации, удобством для восприятия. Так как информационные потребности разных пользователей могут различаться, то в принципе на основе одного и того же источника (таблицы, запроса) может быть создано несколько разных экранных форм, ориентированных на разные запросы пользователей, с соответствующим составом и порядком следования полей.

Лучше, чтобы для форм, использующихся *для ввода данных,* порядок расположения полей в форме *совпадал с порядком их расположения во входном документе.*

Рекомендации по созданию экранных форм

2. Если форма предназначена для ввода данных, то из нее могут быть исключены поля, которые автоматически вводятся в БД и не могут быть изменены пользователем (например, поле счетчика, вычисляемые поля).

3. Средства современных СУБД обладают разнообразными возможностями по оформлению экрана. При выборе стиля оформления экрана желательно выполнять следующие рекомендации:

а. не стоит злоупотреблять использованием цветов, шрифтов и других оформительских эффектов; оформление экрана не должно отвлекать от выполнения основных функций;

б. яркие цвета (например, красный) лучше использовать только для целей привлечения внимания (например, при сигнале о существенной ошибке);

Рекомендации по созданию экранных форм

Возможности задания ограничений целостности при описании таблицы могут при «несистемном» проектировании привести к нежелательным последствиям. Предположим, вы создали несколько форм для ввода данных в одну таблицу (например, первая форма -для ввода данных с одного документа, вторая форма - с другого), а в таблице определено несколько обязательных полей, часть из которых должна вводиться посредством первой формы, а другая часть - посредством второй формы. В этом случае частичный ввод данных из одной формы, а потом добавление данных из другой будет невозможно выполнить.

В силу имеющихся ограничений целостности часть полей является обязательной для ввода, другая - нет. *Для полей, обязательных для заполнения, можно использовать специальное цветовое выделение.*

Общая характеристика отчетов

Термин «отчет» понимается в ИС шире, чем это традиционно принято. Под *отчетом* здесь понимается любой выходной документ: список (например, сотрудников), письмо, адрес, печатающийся на конверте (почтовая этикетка), отчет в традиционном понимании этого слова. Создание отчетов (выходных документов) является одной из наиболее важных функций информационных систем.

Для создания отчетов используются высокоуровневые средства автоматизации - генераторы отчетов. Генераторы отчетов, так же как и генераторы форм ввода-вывода, являются компонентами языков 4-го поколения. Они включены в состав большинства СУБД. Кроме того, генераторы отчетов представлены и как самостоятельный класс программного обеспечения. Существует даже англоязычный термин «reporting», объединяющий все вопросы, относящиеся к процессу получения отчетов.

Общая характеристика отчетов

Источниками для получения отчетов могут быть не только таблицы баз данных, но и запросы, а также записи, отобранные с помощью фильтров. Некоторые генераторы отчетов позволяют проводить отбор данных, включаемых в отчет, непосредственно пользуясь средствами самого генератора отчетов.

Отчеты позволяют выполнять следующие действия:

- *проводить группировку данных;*
- *вычислять многоуровневые промежуточные и общие итоги по отдельным полям;*
- *вводить в отчеты вычисляемые поля;*
- *выводить в отчеты данные из разных источников;*
- *включать в отчеты данные, отобранные по заданным критериям;*
- *использовать различные формы представления информации;*
- *качественно оформлять выводимые данные.*

Общая характеристика отчетов

Отчеты имеют много общего с формами. Однако отчеты в отличие от форм не предназначены для ввода и правки данных в таблицах. *Они позволяют только выводить данные в различном виде.* Вывод отчета может быть осуществлен на экран, на печать, а также в файл.

Чаще всего отчеты используются для вывода информации на печать (т.е. для получения так называемых твердых копий). Для документов, даже одинаковых по содержанию, могут использоваться разные приемы их оформления в зависимости от того, куда осуществляется вывод информации. Например, при выдаче информации на экран могут использоваться специальные эффекты (мигание, динамические изображения, полосы прокрутки и т. п.).

Общая характеристика отчетов

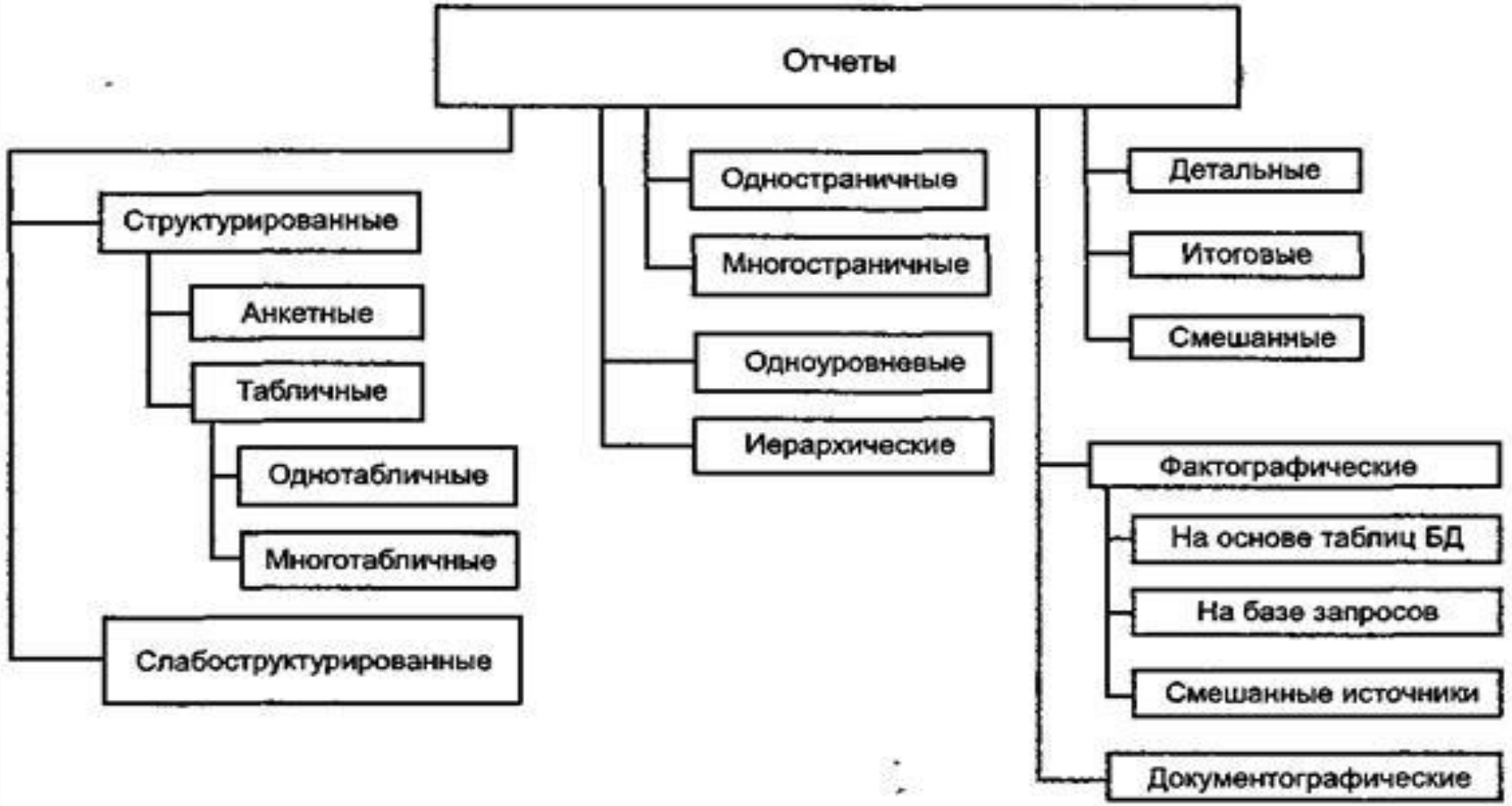

Рис. 9.1. Классификация отчетов

Общая характеристика отчетов

Различают отчеты *анкетной* и *табличной* формы. При анкетной форме данные об одном объекте (сотруднике, товаре и т.п.) обычно размещаются один под другим, причем слева указывается название атрибута (поля), а справа - его значение. После вывода информации об одном объекте выводится информация о следующем объекте (рис. 9.2). Иногда такой тип документа называют документом в виде формы (действительно, это очень напоминает позаписный вывод информации на экран при использовании экранных форм).

Документы табличной формы включают в себя привычные таблицы с названиями атрибутов в заголовках столбцов; данные о каждом объекте представляются в одной строке (рис. 9.3).

Табличные документы могут включать либо одну таблицу - *однотабличные* документы, либо несколько таблиц (обычно разной структуры) - *многотабличные* документы.

Общая характеристика отчетов

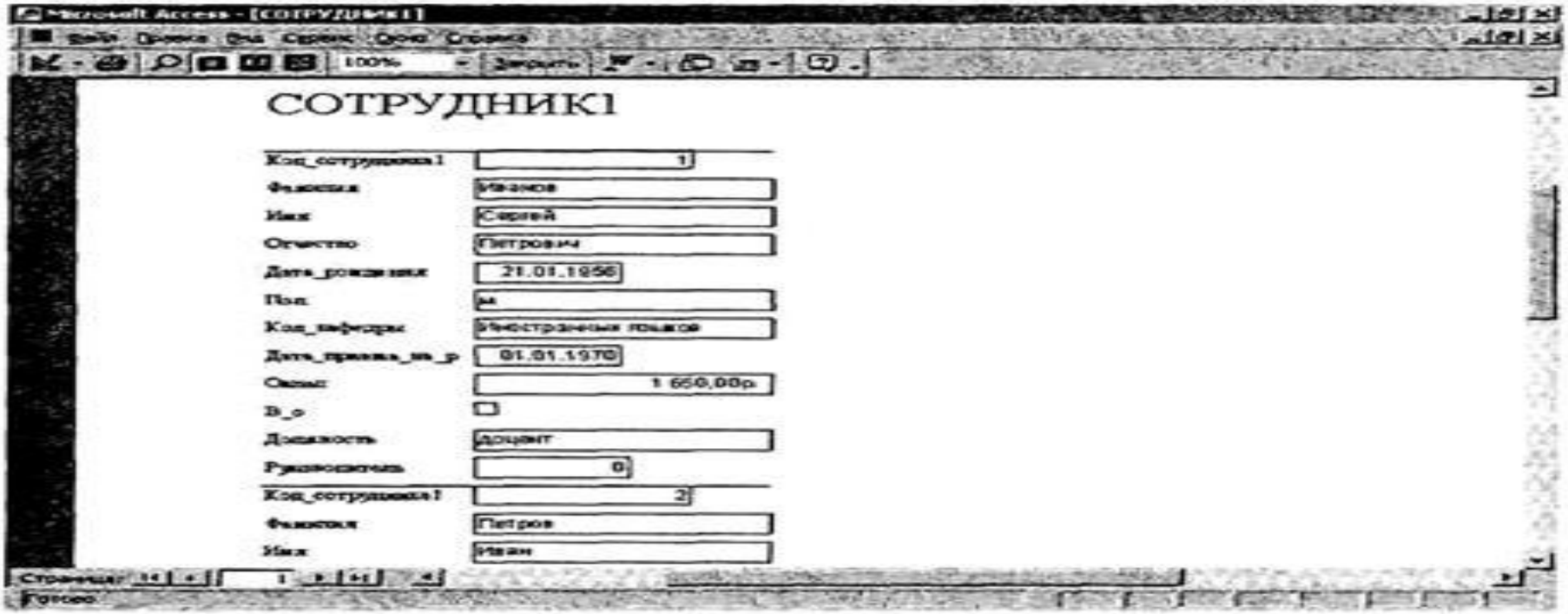

Рис. 9.2. Фрагмент документа анкетной формы

Ведомость на выплату зарплаты
Отдел_____
Месяц_____

№ п/п	ФИО	Сумма на руки	Подпись получателя

Рис. 9.3. Пример документа табличной формы

Общая характеристика отчетов

Для каждой страницы отчета выделяют *верхний* и *нижний колонтитулы* (заголовок и «подножие» страницы). Ну и, естественно, главное место принадлежит *области данных*. В этой области размещаются данные из БД.

Кроме того, в документе может быть обеспечена группировка данных, причем в отличие от запросов группировка может быть одноуровневой и многоуровневой (иерархической). В последнем случае для каждого уровня группировки могут быть созданы *зоны заголовка*.

Группировка обычно используется в целях подсчета каких-либо итоговых показателей для каждой группы (суммы, количества элементов в группе и т.п.). При этом возможно получение *итоговых* документов, включающих только итоговые значения, *детальных* документов, имеющих только детальные строки, и *смешанных*, содержащих как детальные строки, так и итоговые.

Общая характеристика отчетов

Источниками информации для отчетов могут быть либо реальные таблицы базы данных, либо предварительно созданные запросы (представления - VIEW), отбирающие информацию, выводимую в отчет. Кроме того, в отчет могут включаться вычисляемые поля. Вычисляемые поля, как, впрочем, и реальные поля БД, могут входить в любую зону документа.

В последнее время в отчеты, наряду с *символьной* информацией, часто включается *деловая графика*.

Кроме документов, содержащих главным образом *фактографическую* информацию из баз данных, можно создавать и документы, которые в основном, напротив, включают какой-то текст (*документографические*), в который вкраплены данные из БД (*документы типа письма*).

В документах фактографического типа можно различать просто какой-то текст, не имеющий жесткой связи с элементами БД (например, название документа, поясняющий текст), названия элементов из БД (например, «Фамилия») и значения этих элементов (например, Иванов, Петров), элементы оформления (линия, рисунки).

Понятие курсора. Работа с курсорами.

Курсор – используемая в рамках SQL, встроенного в процедурный язык, возможность для позаписного доступа к таблицам из БД.

Работу с курсором можно разделить на несколько четко выраженных стадий.

- Прежде чем курсор может быть использован, его следует объявить (определить). В ходе этого процесса выборка данных не производится, просто определяется оператор SELECT, который будет использован, и некоторые опции курсора.
- После объявления курсор может быть открыт для использования. В ходе этого процесса уже производится выборка данных согласно предварительно определенному оператору SELECT.
- После того как курсор заполнен данными, могут быть извлечены (выбраны) отдельные необходимые строки.
- После того как это сделано, курсор должен быть закрыт и, возможно, должны быть освобождены ресурсы, которые он занимал (в зависимости от СУБД).

После того как курсор объявлен, его можно открывать и закрывать столь часто, сколько необходимо. Если курсор открыт, операция выборки может выполняться так часто, как необходимо.

Понятие курсора. Работа с курсорами.

Курсоры создаются с помощью оператора DECLARE, синтаксис которого различен для разных СУБД.
Оператор DECLARE дает курсору имя и принимает оператор SELECT, дополненный при необходимости предложением WHERE и другими.
Чтобы показать, как это работает, мы создадим курсор, который будет делать выборку всех клиентов, не имеющих адресов электронной почты, в виде части приложения, позволяющего служащему вводить недостающие адреса.
Версия для Oracle:

```
DECLARE CURSOR CustCursor
IS
SELECT * FROM Customers
WHERE cust_email IS NULL;
```

Понятие курсора. Работа с курсорами.

Теперь, после того как курсор определен, его можно открыть. Курсоры открываются с помощью оператора OPEN CURSOR, синтаксис которого настолько прост, что его поддерживают большинство СУБД:

OPEN CURSOR CustCursor

При обработке оператора **OPEN CURSOR** выполняется запрос, и выборка данных сохраняется для последующих просмотра и прокрутки.

Теперь доступ к данным этого курсора может быть получен с помощью оператора **FETCH.**

Оператор **FETCH** указывает строки, которые должны быть выбраны, откуда они должны быть выбраны и где их следует сохранить (имя переменной, например).

Понятие курсора. Работа с курсорами.

В первом примере используется синтаксис Oracle для выборки одной строки курсора (первой).

```
DECLARE TYPE CustCursor IS
REF CURSOR RETURN Customers%ROWTYPE;
DECLARE CustRecord Customers%ROWTYPE;
BEGIN;
OPEN CustCursor;
FETCH CustCursor INTO CustRecord;
CLOSE CustCursor;
END;
```

В данном примере оператор **FETCH** используется для выборки текущей строки (автоматически он начнет с первой строки) в переменную, объявленную с именем ***CustRecord***. С выбранными данными ничего не делается.

Понятие курсора. Работа с курсорами.

В следующем примере (в нем вновь используется синтаксис Oracle) выбранные данные подвергаются циклической обработке от первой строки до последней:

```
DECLARE TYPE CustCursor IS
REF CURSOR  RETURN Customers%ROWTYPE;
DECLARE CustRecord Customers%ROWTYPE;
BEGIN;
OPEN CustCursor;
LOOP
FETCH CustCursor INTO CustRecord;
EXIT WHEN CustCursor%NOTFOUND;
END LOOP;
CLOSE CustCursor;
END;
```

Аналогично предыдущему примеру, здесь используется оператор **FETCH** для выборки текущей строки в переменную, объявленную с именем ***CustRecord.*** Однако в отличие от предыдущего примера, здесь оператор **FETCH** находится внутри цикла **LOOP**, так что он выполняется снова и снова.

Понятие курсора. Работа с курсорами.

Код **EXIT WHEN CustCursor%NOTFOUND** указывает, что этот процесс должен быть завершен (выход из цикла), когда больше не останется строк для выборки. В этом примере для простоты также не выполняется никакой обработки, тогда как в реальный программный код следовало бы включить операторы анализа и обработки данных.

Как следует из предыдущих примеров, после использования курсоров их нужно закрывать.

Вот соответствующий синтаксис для СУБД Oracle:

CLOSE CustCursor;

Для закрытия курсора используется оператор **CLOSE**;

После того как курсор закрыт, его нельзя использовать, не открыв перед этим вновь. Однако его не нужно объявлять заново при повторном использовании, достаточно оператора **OPEN**.