СТРУКТУРА И СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ
«ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ И ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ ДАННЫХ»

1. Титульный лист, подписанный автором.
2. Номер варианта и задание на курсовое проектирование.
3. Реферат объемом до одной страницы, в котором приводятся:
   3.1 тема индивидуального задания курсовой работы,
   3.2 объем пояснительной записки, количество иллюстраций, таблиц, приложений, количество использованных источников;
   3.3 перечень ключевых слов, отражающих существо проекта;
   3.4 текст реферата, включающий:
   – объект разработки;
   – цель работы;
   – использовавшиеся методы и средства разработки;
   – полученные результаты;
   – основные технико-эксплуатационные характеристики;
   – рекомендации по внедрению;
   – область применения;
   – экономическую эффективность или значимость работы;
   – предложения по развитию программы и базы данных.
4. Содержание (оглавление) пояснительной записки.
5. Перечень сокращений, условных обозначений, символов, единиц и терминов.
6. Введение, представляющее постановку задачи с кратким перечнем известных аналогов, их характеристики и опыта эксплуатации.
7. Основная часть пояснительной записки, содержащая:
   7.1 краткий обзор методов и средств проектирования баз данных и приложений с обоснованием используемых методов и средств;
   7.2 системный анализ предметной области с перечислением специалистов в ПО, их прав и ролей, схем и описаний процессов с участвующими в них объектами ПО, обоснованием выбора автоматизируемых процессов и функций;
   7.3 Назначение проектируемой базы данных
   При формулировке назначения проектируемой базы данных должны быть определены:
   - выполняемые функции;
   - задачи, решаемые системой;
   - требования к информации;
   - требования к надёжности;
   - требования к составу и параметрам технических средств;
   - требования к информационной и программной совместимости.
   7.4 концептуальную схему в виде диаграммы Чена и текстовое описание данных для автоматизируемых функций ПО;

7.5 логическую структуру БД в виде системы связанных таблиц с обоснованием третьей нормальной формы (в виде ER диаграмм), описание структуры таблиц и представлений, контролируемых ограничений на значения данных и целостности базы;
7.6 обоснование выбора средств для хранения и обработки базы данных (СУБД и среда программирования);
7.7 определение набора учетных записей пользователей, их групп (ролей) и прав доступа к данным;
7.8 физическую организацию данных, включающую оценки объемов данных, использование средств ускорения доступа (индексов) для типовых запросов, размещение данных в файлах операционной системы;
7.9 описание приложений для пользователя: задачи, реализуемые процедурами сервера и клиента, логическая структура (схема) вызовов функций пользователя в программе клиента, блок-схемы алгоритмов и программ для сложных функций клиента и хранимых процедур, модули данных и функций;
7.10 руководство для пользователей и руководство для сопровождающего программиста.

8. Заключение, в котором содержатся выводы по результатам работы, дается оценка соответствия результатов требованиям задания, представлены сведения по результатам отладки программы и БД, рекомендации по внедрению и развитию системы.

9. Список литературы, содержащий перечень источников, на которые имеется ссылка в тексте пояснительной записки.

10. Приложение, содержащее вспомогательные материалы:

− примеры отчетов и образцы форм, создаваемых программой;
− распечатки результатов выполнения функций при отладке;
− тексты основных методов, функций и процедур;
− сообщения программы;
− скрипт для создания базы и всех ее статических объектов, включая хранимые процедуры, триггеры.

Результатом курсового проектирования является проектная документация (пояснительная записка, выполненная в строгом соответствии со стандартами СТО 701-2005 и СТО 702-2005), база данных и приложения для пользователей.

К защите проекта представляются:

1) пояснительная записка к курсовому проекту в напечатанном и сброшюрованном виде;

2) перемещаемый носитель информации (компакт-диск), содержащий:
− полный скрипт для создания базы, всех ее объектов;
− проект (исходные модули) приложения в использовавшейся среде разработки программного обеспечения, например в Delphi;
− установочную версию (дистрибутив) программы, содержащую все необходимые для установки приложения файлы;
− копию базы данных;
− файл с пояснительной запиской в формате редактора Word 97/2000/2003.

ПРИМЕРЫ СОДЕРЖАНИЯ ОСНОВНЫХ РАЗДЕЛОВ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

НАИМЕНОВАНИЕ РАБОТЫ: «Автоматизация учета на складе магазина хозяйственных товаров».

ВВЕДЕНИЕ

Большой ассортимент товаров, предлагаемых к продаже, возросший товарооборот, необходимость хранения данных о большом количестве производителей, поставщиков усложняют задачу учета поставок, продаж и остатков на складе магазина. Ручная обработка этих данных не позволяет получить оперативную информацию об уменьшении запасов пользующихся спросом товаров и, следовательно, необходимости пополнения этих запасов. Расчет прибыли, затрат, рентабельности, поиск решений, приводящих к повышению эффективности работы магазина, становятся трудоемкими.

В любой организации, как большой, так и маленькой, возникает проблема такой организации управления данными, которая бы обеспечила наиболее эффективную работу. Некоторые организации используют шкафы для хранения папок с документами и ручной поиск и обработку необходимой информации. Но большинство предпочитают компьютеризированные базы данных и программы, позволяющие эффективно хранить, быстро извлекать нужную информацию и управлять большими объёмами данных.

Большое количество информации, высокие требования к точности, многочисленные вычисления, потребность в постоянном обновлении данных делают необходимым применение баз данных для учета товаров на складе магазина.

ОБЪЕКТ АВТОМАТИЗАЦИИ – организация заказа товаров для продажи через магазин с целью получения прибыли.

ПРЕДМЕТ АВТОМАТИЗАЦИИ – учет остатков товаров на складе для обеспечения своевременного заказа товаров, запас которых может закончиться в ближайшее время.

ЦЕЛЬ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ – увеличение прибыли за счет своевременности заказов на поставку товаров в склад магазина и снижения трудоемкости учетных операций.

СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ (ПО).

При анализе ПО необходимо:

- собрать и обобщить материал, всесторонне характеризующий деятельность объекта автоматизации;
- ознакомиться с перспективами развития объекта автоматизации;
- обосновать необходимость применения автоматизированных средств обработки информации и/или управления;
- выявить возможности автоматизации информационных процессов для повышения эффективности, надежности и снижения трудоемкости работ.

- установить множество лиц, участвующих в процессах, выполняемых в ПО, использующих и обрабатывающих информацию.
- определить цели и критерии деятельности лиц, принимающих решение в ПО.

ПРИМЕР АНАЛИЗА ПРЕДМЕТНОЙ ОБЛАСТИ

Задача автоматизации складского учета является частью общей задачи бухгалтерского учета. Она не является отдельной областью, а тесно связана с другими областями. Поэтому необходимо четко определить границы поставленной задачи, чтобы в последствии избежать необходимости изменения структуры программы в связи с появившимися новыми подзадачами с одной стороны, и не повторять обработку данных из других областей, для работы с которыми уже разработаны другие программные средства.

Ассортимент товаров, продаваемых в хозяйственном магазине, очень велик и невозможно хранить все товары в помещении магазина. Для этих целей существует склад. В магазине остается небольшая часть товара, которую возможно реализовать в течение дня, и которая не занимает большого пространства. Если остаток в помещении магазина уменьшается, то менеджер магазина обращается на склад, предъявляя список необходимых на данный момент товаров. Если на складе оставшееся количество определенного товара уменьшается (допустим до 20 единиц), то формируется заказ на поставку этого товара от поставщиков.

Основные понятия, используемые при разработке информационной системы:

Товар – продукт труда, предназначенный для продажи.

Заказ – документально оформленная и оплаченная заявка на доставку в склад магазина определенного товара.

Поставщик – организация, поставляющая товары какой-либо группы.

Прибыль от торговой деятельности – это чистый доход магазина.

Доход – деньги, получаемые от какого-либо рода деятельности (в рассматриваемой предметной области – от торговли хозяйственными товарами).

Прибыль определяется как разность между выручкой и всеми затратами на приобретение и продажу товаров.

Выручка – вырученные от продажи всех товаров деньги. Определяется как сумма произведений количества проданных товаров на их отпускную цену.

Отпускная цена – цена, по которой товар продается в магазине.

Затраты включают оплату поставщикам (производителям) стоимости и доставки всех заказанных товаров, заработную плату сотрудникам, налоговые отчисления, арендную плату.

Затраты на оплату стоимости и доставки определяются как сумма произведений количества проданных товаров на их закупочную цену.

Закупочная цена - цена, по которой товар закуплен у производителя (поставщика) включая стоимость доставки.

Для упрощения учебной задачи примем следующие допущения:

- стоимость всех находившихся на складе товаров до внедрения информационной системы не изменяется и не влияет на прибыль (стоимость проданных товаров равна стоимости вновь заказанных товаров);

- расходы на налоги, заработную плату, аренду постоянны во все время функционирования информационной системы и не учитываются при определении прибыли;
- отпускная цена любого товара устанавливается как закупочная цена товара, умноженная на 1.15.

При расчете прибыли используются данные из следующих областей: сведения о заказанных товарах и сведения о проданных товарах.

Для формирования заказов необходимы сведения о заканчивающихся на складе товарах, сведения о принадлежности товаров к какой-либо группе и поставщиках товаров этой группы.

Для получения прибыли от торговли хозяйственными товарами необходимо выяснить, какой товар заканчивается на складе, заказать его, оплатить поставку, разместить образцы в торговых залах и оформить продажу.

НАЗНАЧЕНИЕ ПРОЕКТИРУЕМОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Функции, решаемые информационной системой

Автоматизированная информационная система учета на складе магазина хозяйственных товаров должна обеспечивать выполнение следующих функций:

- ввод и хранение необходимой для работы информации;
- исключение избыточности данных;
- обеспечение целостности данных;
- обновление информации в базе данных (БД);
- резервное архивирование данных.

Задачи, решаемые информационной системой:

- хранение информации об имеющихся на складе товарах;
- хранение информации о проданных товарах;
- выдачу информации о заканчивающихся товарах;
- хранение информации о принадлежности товара группе товаров;
- хранение информации о поставщиках определенных групп товаров;
- хранение информации о поставленных на склад товарах;
- расчет отпускной цены товара;
- расчет выручки от продажи и затрат на заказ товаров;
- печать заказов на поставку товаров;
- хранение данных о заказанных товарах;
- расчет прибыли;
- формирование и печать отчетных документов.

Требования к представлению информации

Информация должна представляться в удобной для пользователя форме и быть доступной для просмотра и модификации.

Требования к надёжности.

Система должна обеспечивать:

- целостность данных;

- непротиворечивость данных;
- возможность хранения архивов на внешних носителях.

Требования к составу и параметрам технических средств.

Указать минимальные требования к серверу и рабочей станции (процессор, оперативная память, жёсткий диск, и др.).

КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ.

Концептуальная схема должна отражать состав и взаимодействие объектов будущей БД. Для этой цели разработано несколько систем соглашений о представлении информации, содержащейся в БД. Например, диаграммы потоков данных (Data Flow Diagrams), фиксирующих пути перемещения информации с указанием мест ее хранения и обработки, или универсальный язык моделирования (UML) - промышленный стандарт создания моделей процессов и данных для объектно-ориентированных разработок информационных систем (ИС). Подобные системы предназначены для автоматизации всего процесса разработки ИС.

Одним из простых, наглядным, а значит, удобным для обсуждения со специалистами ПО, средством концептуального моделирования данных является диаграмма Чена. Диаграмма Чена представляет собой граф с двумя типами вершин. Вершины первого типа - сущности, представляющие объекты ПО, изображаются прямоугольниками. Сущность (как понятие) образуется типизацией множества объектов, похожих по составу информации, требуемой для выполнения автоматизируемых функций. Объекты, использованные на входах, выходах, условиях выполнения и требуемых ресурсах функций в схемах процессов являются основой для образования сущностей. Таким образом, каждой сущности соответствует множество похожих (однотипных) объектов, которым в базе данных будут соответствовать экземпляры сущности. Так, для процесса выполнения заказов на изготовление окон появятся сущности «Заявка», «Заказчик», «Конструкция окна» и т.д. Каждая из этих сущностей является обобщенным представителем множества реально существующих заявок, заказчиков, конструкций. Важно лишь то, что все объекты, образующие сущность, описываются одинаковыми наборами свойств – атрибутов, различающихся значениями у конкретных объектов.

Атрибуты используются для хранения набора данных об объектах ПО, включаемых в БД. При определении атрибутов сущности учитываются не все возможные характеристики объектов, а только те из них, которые нужны для выполнения используемых и перспективных автоматизируемых функций. Для каждой сущности определяется ключ. Для каждого атрибута кроме имени необходимо определить тип (формат) и ограничения на возможные значения данного. Кроме того, в сущностях определяются ограничения, в которых участвуют несколько атрибутов. Например, дата заключения договора должна быть больше даты оформления заявки. Для последующей реализации в базе, в сущности важно выделить атрибуты, допускающие множество значений в одном экземпляре сущности – множественные атрибуты, указав их максимальное и среднее количество в объекте. Примерами множественных атрибутов могут служить профессии и/или места работы сотрудника.

Выделенные из схем процессов сущности являются основой для построения концептуальной схемы данных в виде диаграммы Чена.

Второй тип вершин в диаграммах Чена, называемый «связи» и изображаемый ромбами, предназначен для представления взаимодействий объектов, образовавших сущности. Связи, соединяя сущности, образуют концептуальную схему предметной области. Связи могут отражать взаимодействия любого числа сущностей. Например, взаимодействие «Заявка» и «Заказчик» связывает две сущности, а значит, образует бинарную связь. Связь сущностей «Студент», «Учебная дисциплина» и «Преподаватель» следует рассматривать как связь трех сущностей (тернарную). Связь, как и сущность, имеет имя и может иметь атрибуты, которые не могут быть отнесены ни к одной из связанных сущностей. Например, атрибут «Оценка» принадлежит связи «Успеваемость», соединяющей сущности «Студент», «Учебная дисциплина» и «Преподаватель».

Другой важной характеристикой связи является ее интерпретация на множестве объектов, образовавших сущность. Каждая сущность схемы является представителем множества объектов, существующих в ПО. Поэтому на схеме важно показать, как соотносятся связанные объекты, образовавшие сущности, то есть определить тип связи, показывающий, какое количество объектов одной сущности может быть связано с одним объектом другой сущности. Например, одному объекту, соответствующему сущности «Заказчик», может соответствовать (быть связанными) ноль, один или более объектов сущности «Заявка», если мы допускаем возможность работы с потенциальными заказчиками и возможность неоднократного обращения одного заказчика. При этом появление объекта «Заявка» при отсутствии ее заказчика оказывается бессмысленным. Таким образом, можно утверждать, что объекты «Заявка» появляются обязательно в связи с определенным заказчиком и не могут существовать самостоятельно. Такие сущности называют обязательно участвующими (существующими только) в связи с другими сущностями или «слабыми». Напротив, объект сущности «Заказчик» может появиться и без связи с заявкой как возможный будущий заказчик. Такие сущности называют не обязательно участвующими в связи, или «сильные» сущности.

Для представления типа связи в схемах Чена используется специальная разметка линий, соединяющих сущности и связи. На концах линий, присоединенных к сущностям, применяются четыре символа:

|| - две вертикальные линии, означают обязательное участие в связи одного и только одного объекта,

0| - ноль и вертикальная линия, означают участие в связи не более одного объекта,

>| - один или более объектов могут участвовать в связи,

>0 – ноль или более, то есть любое число объектов может участвовать в связи.

Например, связь между заказчиком и его заявками можно представить следующей схемой.

Установленный тип связи утверждает, что экземпляры сущности «Заказчик» могут появиться в БД без связанных экземпляров сущности «Заявка», но каждый экземпляр

сущности «Заявка» должен быть связан точно с одним экземпляром сущности «Заказчик». Это связь типа «один ко многим» (1:M) со слабой сущностью «Заявка».

Другой пример взаимодействия это связь между договором и сметой.

Каждому договору должна соответствовать своя единственная смета, являющаяся обоснованием цены.

Таким образом, связь между экземплярами этих сущностей является «один к одному» (1:1) с обязательным участием обоих объектов в связи

В третьем примере рассматривается взаимодействие между сущностью «Договор» и «Типовая конструкция».

В одном договоре может не быть типовых конструкций, а в другом использоваться их любое число, типовая конструкция может не применяться, если она только что разработана, или применяться во многих договорах. Эту связь следует представить бинарным отношением типа «многие ко многим» с «сильными» сущностями, необязательно участвующими в связи.

ВЫБОР СУБД

Выполняется выбор системы управления базами данных, с помощью которой будет реализована информационная модель ПО и инструмента для разработки приложения.

При выборе СУБД учитывается тип используемой вычислительной сети и операционной системы, нагрузка на сеть и базу данных, создаваемая автоматизируемыми функциями пользователей. Влияние нагрузки, создаваемой рабочими станциями пользователей, оценивается частотой выполнения различных автоматизируемых функций, требуемым для них объемом хранимых и передаваемых данных и ограничениями на время выполнения типовых запросов.

Обоснование выбора должно содержать характеристики конкурентоспособных СУБД и их сравнительный анализ и сопоставление с требованиями ПО, а также сопоставление требований приложения с инструментами разработки программ, совместимыми с выбранной СУБД. При выборе СУБД в курсовом проектировании необходимо учитывать предстоящую реализацию проекта базы и приложения и ориентироваться на доступные компьютеры и инструменты разработки.

ЛОГИЧЕСКАЯ МОДЕЛЬ БД

Логическая схема получается преобразованием концептуальной схемы в ER диаграмму и далее в структуру данных, поддерживаемую выбранной СУБД.

Рассмотрим процесс построения структуры БД применительно к наиболее распространенной реляционной модели данных.

Основой для построения логической структуры данных является концептуальная схема.

Общий подход преобразования концептуальной схемы в логическую состоит в том, что каждую сущность, являющуюся представителем множества однотипных объектов, задают схемой отдельного отношения (нормализованной таблицы). Атрибуты сущности образуют столбцы таблицы.

Для поддержания соответствия между экземпляром сущности в главной таблице и его атрибутами во вспомогательной таблице создается столбец внешнего ключа, в который заносятся значения первичного ключа сущности из главной таблицы. Таким образом, вынесенные в отдельную таблицу атрибуты поддерживают связь «многие к одному» через соответствие внешнего и первичного ключей (FK – PK). Причем наличие экземпляра сущности – записи в главной таблице является обязательным для связанных записей ее атрибутов. Поэтому создаваемая связь должна обладать свойством ссылочной целостности и не допускать появления записей с атрибутом у несуществующей записи сущности. Создание отдельной таблицы для множественного атрибута решает проблему их хранения, но при этом несколько усложняет экранные формы для управления данными в приложениях. В формах для ввода и редактирования экземпляра сущности необходимо предусмотреть элементы управления для добавления, удаления, корректировки атрибута и отображения всех значений атрибута у экземпляра сущности.

Еще одним приемом, улучшающим структуру БД, является создание справочников и классификаторов данных. Для столбцов, значения которых принадлежат фиксированному ограниченному множеству или носят условно постоянный характер, создаются справочники и классификаторы данных в виде отдельных таблиц. При создании базы необходимо использовать общероссийские классификаторы. Например, общероссийский классификатор форм собственности (ОКФС), Общероссийский классификатор административно-территориального деления (ОКАТО), Общероссийский классификатор специальностей по образованию (ОКСО). Перечень общероссийских классификаторов можно найти по ссылке http://goskomstat.karelia.ru/uslugi/klassif.php3.

После создания справочника или выбора готового классификатора вносятся изменения в структуры информационных таблиц путем замены имен и доменов полей, вошедших в справочники, их кодами.

Далее каждая таблица анализируется на принадлежность к нормальным формам и при необходимости выполняется ее нормализация.

Декомпозицию отношения преобразованием к третьей нормальной форме можно выполнить следующим образом. Проверяется наличие прямой и однозначной (функциональной) зависимости данных в каждом неключевом атрибуте от всех значений первичного ключа. Если часть неключевых атрибутов отношения зависит не от всех атрибутов первичного ключа, то эти атрибуты и определяющие их атрибуты первичного ключа выносятся из структуры данного отношения в новое отношение. Если какое-то подмножество неключевых атрибутов функционально зависит от других неключевых атрибутов, и при этом транзитивно зависит от ключа таблицы, то определяющие и зависимые атрибу-

ты образуют новое отношение. При этом зависимые атрибуты удаляются из структуры исходного отношения.

Во вновь построенных и преобразованных таблицах проверяются первичные ключи. Если первичный ключ таблицы оказывается слишком длинным, неудобным для идентификации строк и для поддержания связей в БД, то в таблицу вводится дополнительное поле – искусственный первичный ключ. Изменение первичного ключа в одной таблице распространяется на соответствующие внешние ключи связанных таблиц.

Затем в логическую структуру БД вносятся связи, заданные между сущностями концептуальной схемы. Способ представления связи в реляционной модели данных зависит от ее типа.

Следующей важной задачей логического проектирования является определение групп (ролей) пользователей и их прав в БД. Для этого автоматизируемые функции разбиваются на группы в соответствии с задачами пользователей в предметной области. Группам даются понятные имена, ставятся в соответствие данные (таблицы и столбцы таблиц), для которых устанавливаются необходимые права доступа (включение, удаление, чтение, изменение). Для пользователей определяются учетные записи, которые распределяются по группам.

Для повышения надежности данных и уменьшения времени восстановления из-за сбоев аппаратуры или стихийных бедствий необходимо ежедневно производить резервное копирование данных.

ФИЗИЧЕСКАЯ МОДЕЛЬ БД

На этапе физического проектирования решаются вопросы эффективного размещения данных на машинных носителях и использования средств ускорения доступа к данным.

В процессе физического проектирования базы должно быть определено:

- количество и типы используемых носителей информации;
- количество и размеры файлов операционной системы, в которых размещается база данных, их расположение на носителях информации.
- типы, количество и режимы обновления индексов для пользовательских таблиц и представлений;
- резервирование свободных областей памяти при загрузке базы, частота и способы реорганизации (уплотнения) базы данных;
- способы и средства обеспечения надежности (резервирования и восстановления) данных.

Разбивка данных по таблицам осуществляется на основании логической модели применительно к выбранной системе управления базами данных.

Пример проектирования таблиц СУБД MS SQL Server 2005

Таблица Б.1– Group_Products (тблГруппы_товаров)

Наименование	Описание	Размер	Ограничения	Комментарии
Id_gr	**Код группы**	uniqueidentifer	Primary key	Уникальный ключ группы товаров
Name_gr	Имя группы	varchar(25)	Not null	Категория товаров
Supplier	Поставщик	varchar(25)	Not null	Наименование
Treatment	ОбращатьсяК	varchar(25)		Представительное лицо
Phone	Телефон	varchar(11)		Телефон для связи
Bank_details	Банк.реквизиты	varchar(50)		Реквизиты для оплаты

Таблица Б.2 – Products(тблТовар)

Наименование	Описание	Размер	Ограничения	Комментарии
Id_tov	**Код товара**	uniqueidentifer	Primary key	Уникальный ключ товара
Id_gr	*Код группы*	int	Foreign key	Внешний ключ (Код группы из таблицы тблГруппы_товаров)
Name_tov	Наименование	varchar(35)	Not null	Название товара
Ed_izm	ЕдцаИзмерения	varchar(10)		Единица измерения
Prize	ЦенаЗакупочная	smallmoney		Закупочная цена

МЕТОДИКА РАЗРАБОТКИ ПРИЛОЖЕНИЯ ДЛЯ БАЗЫ ДАННЫХ

На второй стадии курсового проектирования выполняется разработка программы, реализующей автоматизированные функции пользователей и дополнительные функции администратора для обслуживания БД.

Разработка программы предполагает последовательное выполнение этапов технического и рабочего проектирования.

На этапе технического проектирования разрабатываются алгоритмы и принимаются решения по функциям, структуре, интерфейсам и защите программного обеспечения. В процессе технического проектирования программы, предназначенной для работы с базой данных, решаются следующие задачи.

1. Производится классификация и распределение автоматизируемых функций по пунктам меню главного диалогового окна пользователя. В один класс следует включать функции, объединяемые общей логикой применения или общими используемыми объектами.

2. Выделяются наиболее употребительные функции, которые целесообразно представить кнопками в общих для многих экранных форм панелях инструментов.

3. Выбираются функции, для эффективной реализации которых целесообразно использовать хранимые процедуры. Для сложных функций разрабатываются алгоритмы, в необходимых случаях представляемые блок-схемами и текстовыми описаниями.

4. Вводятся дополнительные функции, обеспечивающие защиту данных от разрушения при программных и аппаратных отказах, путем автоматического или вызываемого пользователем копирования информации на резервный носитель, а также восстановления БД с копии.

5. В системах для многих пользователей определяются полномочия (права) пользователей по работе с функциями и данными. Устанавливается необходимый уровень изоляции транзакций, выполняющих функции и хранимые процедуры. В необходимых случаях предусматриваются индивидуальные блокировки данных в операторах транзакций.

6. Определяется тип диалога в виде одно- или много-документального интерфейса для функций пользователя. Разрабатываются экранные формы для организации диалога при выполнении функций и структура создаваемых программой документов (отчетов, справок, писем и т.д.). Создание макетов экранных форм и отчетов следует выполнять средствами визуального программирования, если они имеются в выбранном инструменте разработки, совмещая этим этапы технического и рабочего программирования.

7. Разрабатываются описания экранных форм и контекстных подсказок, подключаемых в качестве оперативной помощи (Help) пользователю.

8. Продумывается способ установки программной системы на компьютер пользователя и ее переустановки в случае отказа компьютера.

На этапе рабочего проектирования создается структура программной системы. Структура программы должна строиться по модульному принципу так, чтобы самостоятельные функции были реализованы отдельными модулями или методами объектно-ориентированного программирования и допускали автономные изменения. Функции распределяются по программным модулям.

Определяется способ вызова (запуска) головной программы пользователем. При этом необходимо, чтобы приложение, изменяющее среду операционной системы и СУБД, восстанавливало ее после окончания работы.

Программирование начинается с создания пользовательского меню и продолжается последовательной разработкой, подключением и отладкой отдельных функций и хранимых процедур, начиная с функций ввода и редактирования данных.

Программы, реализующие функции пользователя, должны в начале контролировать выполнимость функции (наличие всех данных, готовность устройств и т.д.) и сообщать о недостающих ресурсах, а в процессе выполнения перехватывать исключения среды программирования и СУБД, заменяя их понятными пользователю сообщениями.