**ДИПЛОМНАЯ РАБОТА**

**Создание информационно-справочной подсистемы САПР конструкторско-технологического назначения. Интегральные микросхемы**

**СОДЕРЖАНИЕ**

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ, СОКРАЩЕНИЙ И ТЕРМИНОВ

ВВЕДЕНИЕ8

1. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

1.1 Наименование, область применения, цель создания информационно-справочной подсистем САПР конструкторско-технологического назначения

1.2 Задачи, решаемые подсистемой

1.3 Функциональные требования к подсистеме

2. ОБЗОР ОБЬЕКТОВ ПОДЛЕЖАЩИХ ОБРАБОТКЕ ИНФОРМАЦИОННО-СПРАВОЧНОЙ СИСТЕМЫ

2.1 Классификация интегральных микросхем

2.2 Условно-графическое изображение цифровых микросхем

2.3 Диодно-транзисторная логика (ДТЛ)

2.4 Транзисторно-транзисторная логика (ТТЛ)

2.4.1 Логические уровни ТТЛ микросхем

2.4.2 Семейства ТТЛ микросхем

2.5 Логика на комплементарных МОП транзисторах (КМДП)

2.5.1 Особенности применения КМОП микросхем

2.5.2 Логические уровни КМОП микросхем

2.5.3 Семейства КМОП микросхем

3. ИНФОРМАЦИОННО-СПРАВОЧНАЯ СИСТЕМА

3.1 Определение и классификация БД

3.2 Компилятор Visual C++, версия 6

ВЫВОДЫ

ПЕРЕЧЕНЬ ССЫЛОК

ПРИЛОЖЕНИЕ А

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ, СОКРАЩЕНИЙ И ТЕРМИНОВ**

# БД – База данных

# ЭВМ – Электронно-вычислительная машина

САПР – Система автоматизации производства

МДП – метал диэлектрик полупроводник

МОП – метал окись полупроводник

КМОП - комплиментарный метал окись полупроводник

РТЛ – резистивно–транзисторная логика

ДТЛ – диодно–транзисторная логика

ТТЛ – транзисторно–транзисторная логика

ЭСЛ – эмиттерно–связанная логика

И2Л – интегральная инжекционная логика

СУБД – система управления БД

# **ВВЕДЕНИЕ**

Различные возможности и границы применения вычислительной техники для автоматизации проектирования определяются уровнем формализации научно-технических знаний в конкретной отрасли. Чем глубже разработана теория того или иного класса технических систем, тем больше объективных возможностей существуют для автоматизации процесса их проектирования. Применение ЭВМ при проектно-конструкторских работах в своем развитии прошло несколько стадий и претерпело значительные изменения.

С появлением вычислительной техники был сделан акцент на автоматизацию проектных задач, имеющих четко выраженный расчетный характер, когда реализовывались методики, ориентированные на ручное проектирование. Затем, по мере накопления опыта, стали создавать программы автоматизированных расчетов на основе методов вычислительной математики (параметрическая оптимизация, метод конечных элементов и т. п.).

С внедрением специализированных терминальных устройств появляются универсальные программы для ЭВМ для решения как расчетных, так и некоторых рутинных проектных задач (изготовление чертежей, спецификаций, текстовых документов и т. п.). В последние годы большое внимание уделяется автоматизации расчетно-конструкторских работ при проектировании типовых узлов и агрегатов, когда синтез конструкции проводится эвристически, а основные параметры выбираются и оптимизируются в интерактивном режиме диалога проектировщика и ЭВМ.

Современное производство достигло той стадии развития, на которой технологические процессы представляют собой сложные последовательности операций, требующие точного представления, учета и распределения. Ручное или автоматизированное исполнение этих операций на уровне представления затруднительно. Решение этих задач возможно при использовании информационно-справочной подсистем САПР конструкторско-технологического назначения.

Внедрение информационно-справочных подсистем САПР конструкторско-технологического назначения на первых этапах развития промышленности повысило производительность труда, так как дало возможность в автоматизированном режиме производить управление информацией на разных стадиях производства, а также в различных его отделах.

Для обработки и хранения информации на всех стадиях производства применяются информационно-справочные системы. Которые содержат информацию об объектах используемых на производстве, их характеристику, назначение, маркировку и т.д.

# **1. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

**1.1 Наименование, область применения, цель создания информационно-справочной подсистем САПР конструкторско-технологического назначения**

Разрабатываемая подсистема, называемая «Информационно-справочная подсистема САПР конструкторско-технологического назначения. «Интегральные микросхемы» предназначена для учета, сбора, содержания справочной информации для конструкторско-технологических целей. Цель разработки – обеспечить информационно-справочную подсистему, которая обеспечит автоматизированный доступ и учет информации с помощью разработанной специально для этих целей базой данных (БД). Информационно-справочная подсистема должна использоваться в целях предоставления информации сразу по мере ее необходимости, должна быть также функция добавления информации, так как жесткая БД быстро устареет и в ней пропадет необходимость.

На этапах создания такой БД возникают некоторые проблемы:

1. нужно учесть доступ к БД различных категорий рабочих;
2. отображаемая информация должна быть наиболее полной;
3. БД не должна содержать данные, которые не принадлежат данной подсистеме.

**1.2 Задачи решаемые подсистемой**

Задачи, решаемые подсистемой, заключаются в обеспечении информацией об объектах, подлежащих для разработки систем или их технологического обеспечения. Из всех задач, которые должны решаться подобными системами следует выделить следующие:

* Обеспечение распределения микросхем по типу, корпусу, параметрам и т.д.
* Возможность добавления и корректировки информации в данной подсистеме
* Обеспечение легкости работы и поиска информации
* Предоставление наиболее полной информации об объектах
* Учет имеющихся в наличии микросхем

## 1.3 Функциональные требования к подсистеме

Подсистема должна обеспечивать следующие функциональные возможности автоматизированная система учёта интегральных микросхем:

1) добавление новых объектов в базу данных (БД);

2) изменение характеристик объектов базы данных;

3) удаление объектов из базы данных.

При добавлении базы данных выбор общих характеристик (наименование, код, и т.п.) объекта возлагается на оператора. Параметры учета о изменении количества и наличие микросхем на производстве оператор использующий данную подсистему не имеет право менять эти параметры вручную.

Подсистема обеспечивает обновление существующих микросхем за счёт объектно-ориентированной структуры программного комплекса – перекомпоновка без изменения базовых объектов программного комплекса.

**2. ОБЗОР ОБЬЕКТОВ ПОДЛЕЖАЩИХ ОБРАБОТКЕ ИНФОРМАЦИОННО-СПРАВОЧНОЙ СИСТЕМЫ**

**2.1 Классификация и обозначения цифровых микросхем**

Интегральная микросхема это микроэлектронное изделие, состоящее из активных (транзисторов) и пассивных (диодов, резисторов, конденсаторов) элементов, а также из соединяющих их проводников, которое изготавливается в едином технологическом процессе в объеме полупроводника или на поверхности диэлектрического основания, заключено в корпус и представляет собой неразделимое целое. Иногда ее называют интегральной схема, иногда микросхемой, соответственно, возможны сокращенные обозначения ИМС, ИС, МС.

По технологии изготовления микросхемы делятся на три разновидности: полупроводниковые (самые распространенные), пленочные (почти не выпускаются) и гибридные (выпускают немного и выпуск сокращают).

В полупроводниковых микросхемах все элементы и их соединения изготавливаются в объеме (внутри) и частично на поверхности полупроводника. Иногда полупроводниковую микросхему называют твердотельной схемой, что является буквальным переводом с английского языка (solid state).

В пленочной микросхеме все элементы и их соединения выполнены в виде пленок из проводящих и диэлектрических материалов на диэлектрическом основании. В этих микросхемах нет транзисторов и диодов.

В гибридных микросхемах пассивные элементы и соединительные проводники изготавливают по пленочной технологии, а бескорпусные транзисторы и диоды, изготовленные отдельно по полупроводниковой технологии, соединяют тонкими проводами диаметром 0,04 мм с контактными площадками.

По функциональному назначению микросхемы делятся на две категории:

– аналоговые, обрабатывающие сигналы, изменяющиеся по закону непрерывной функции;

– цифровые, обрабатывающие цифровые сигналы.

Транзисторы, применяющиеся в цифровых микросхемах, бывают двух типов:

– обычные (n–p–n или p–n–p) биполярные транзисторы;

– полевые (униполярные) транзисторы.

В цифровых микросхемах применяются полевые транзисторы только с изолированным затвором, имеющие структуру: металл (затвор), диэлектрик (изоляция затвора), полупроводник (канал, сток–исток), сокращенно МДП, а так как в качестве диэлектрика обычно используется окись кремния, то обычно эти транзисторы, а также микросхемы на них сокращенно называют МОП. Чаще всего в цифровых микросхемах используют пары МОП транзисторов, дополняющие друг друга по проводимости канала, такие микросхемы называют КМОП от слова комплиментарный, что означает дополняющий.

В зависимости от элементов, на которых собраны входные и выходные каскады микросхем, от схемных особенностей этих каскадов цифровые микросхемы делятся на несколько групп или, так называемых "логик" (здесь под словом "логика" подразумевается логический элемент или электронный ключ):

1. РТЛ, – резистивно–транзисторная логика, в которой на входах стоит резистивный сумматор токов, реализующий для положительной логики функцию ИЛИ; выходной каскад собран на транзисторном инверторе;

2. ДТЛ, – диодно–транзисторная логика, в которой на входах стоит несколько диодов, реализующих функцию И или ИЛИ; выходной каскад на транзисторах;

3. ТТЛ, – транзисторно–транзисторная логика, в логических элементах которой ко входам подключены эмиттеры многоэмиттерного транзистора; с помощью этого многоэмиттерного транзистора реализуется функция И; выходной каскад собран на транзисторах;

4. ЭСЛ, – эмиттерно–связанная логика, в которой на входах стоят транзисторы, эмиттеры которых связаны друг с другом;

5. nМОП, pМОП, – МОП логика, все элементы которой выполнены на МОП транзисторах с проводимостью канала n–типа (n–МОП) или p–типа (p–МОП);

6. КМОП, – логика, все элементы которой выполнены на двух типах МОП транзисторов nМОП и pМОП, дополняющих друг друга, т.е. комплиментарных;

7. И2Л, – интегральная инжекционная логика, в которой отсутствуют резисторы; инжекция носителей в область базы транзистора осуществляется с помощью активных генераторов тока, выполненных на p–n–p транзисторах, тогда как сам базовый инвертор, – на n–p–n транзисторах.

По принятой у нас системе обозначение микросхемы должно состоять из четырех основных элементов:

1) цифра, соответствующая конструктивно–технологической группе (1, 5, 6, 7, – полупроводниковые микросхемы, из них 7, – бескорпусные; 2, 4, 8, – гибридные микросхемы; 3, – прочие, в том числе пленочные, вакуумные, керамические и т.д.);

2) две, а в последнее время три цифры, обозначающие порядковый номер разработки серии микросхем;

3) две буквы, обозначающие функциональное назначение микросхемы; первая буква соответствует подгруппе (сейчас девятнадцать подгрупп), вторая, – виду (от трех до семнадцати видов в подгруппе);

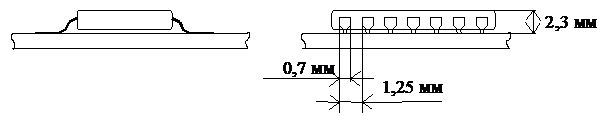
4) порядковый номер разработки данной микросхемы внутри своего вида в данной серии.

Номером серии микросхемы считают первые три или четыре цифры. Для микросхем, используемых в устройствах широкого применения, перед номером серии ставится буква К. Для характеристики материала и типа корпуса микросхемы после буквы К могут быть добавлены следующие буквы: Р, – для пластмассового корпуса второго вида, М, – для керамического, металлического и стеклокерамического корпуса второго типа. В конце обозначения микросхемы может быть добавлена буква, конкретизирующая один из основных ее параметров.

Например: КМ155ЛА3, К561ИЕ33, 564ЛА7, КР565РУ8Г.

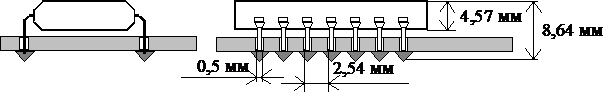
Корпуса цифровых микросхем бывают в основном двух видов:

1. Планарные (плоские), у этих микросхем условное обозначение корпуса начинается с цифры 4; выводы числом от четырнадцати до сорока двух расположены с двух сторон микросхемы с шагом 1.25 мм, прямые, припаиваются, как правило, к дорожкам печатной платы на стороне установки микросхем; такие корпуса часто называют SOIC (small outline integrated cirquit, – микросхема в малом корпусе с выводами, не лежащими в одну линию). Иногда такой тип корпуса называют сокращенно, – SO.



**Рисунок 2.1 - Планарный корпус микросхемы**

2. Корпус dip – dual in line package, – в две линии расположенные выводы (иногда этот тип корпуса называют DIL, иногда, чтобы указать, что корпус изготовлен из пластмассы – PDIP, plastic DIP), – корпус микросхемы, у которой обозначение корпуса начинается с цифры 2; выводы числом от четырнадцати до сорока двух с двух сторон микросхемы с шагом обычно 2,5 мм, изогнутые под углом 900 , припаиваются только в отверстиях печатных плат.

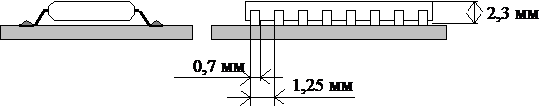


**Рисунок 2.2 - DIP корпус микросхемы**

Отечественные ТТЛ микросхемы в планарных корпусах часто имеют в обозначении серии вторую цифру 3 (133, 136), они обычно выпускаются для специального применения при температуре от – 60 0C до 125 0C, а в dip–корпусах имеют вторую цифру 5 (155,1531), выпускаются для широкого применения при температуре от – 10 0C до 70 0C.

Среди миниатюризированных современных корпусов микросхем, предназначенных для припаивания только на стороне установки микросхем, можно в качестве примера привести следующие:

– SOIC – small outline integrated circuit, при обозначении SN…DW



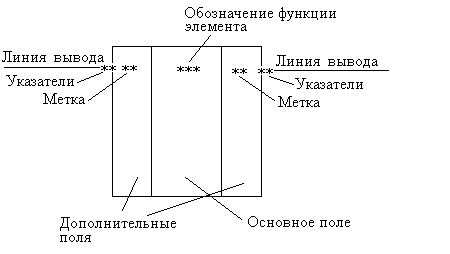
**Рисунок 2.3 - SOIC – small outline integrated circuit, при обозначении SN…DW**

В Европе в обозначении ТТЛ микросхем имеются числа 54 для микросхем специального (военного) применения, и 74, – для широкого (гражданского) применения. Буквы в конце зарубежных обозначений означают: L, – низкое потребление мощности, но низкое быстродействие; H, – высокое быстродействие, но и большое потребление мощности; S, – с диодами Шоттки (Sсhottky); A, – улучшенные, перспективные от слова Advance (вольный перевод "аванс"); F, – быстрые от слова Fast – быстрый.

В обозначение зарубежных КМОП (CMOS) микросхем обычно входит число 40 (CD4011B).

2.2 Условно-графическое изображение цифровых микросхем

Цифровая или микропроцессорная микросхема, ее элемент или компонент; цифровая микросборка, ее элемент или компонент обозначаются на принципиальных схемах условно-графическим обозначением в соответствии с ГОСТ2.743-91. Условно-графическое обозначение микросхемы имеет форму прямоугольника, к которому подводят линии выводов. Условно-графическое обозначение микросхемы может содержать три поля: основное и два дополнительных, которые располагают слева и справа от основного (рисунок 1). В первой строке основного поля условно-графическое обозначение помещают обозначение функции, выполняемой элементом. В последующих строках основного поля располагают информацию по ГОСТ 2.708.

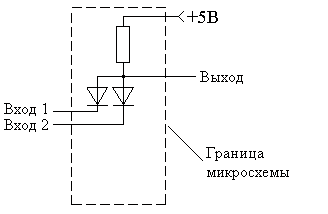


**Рисунок 2.4. Условно-графическое изображение цифровых микросхем**

В дополнительных полях помещают информацию о назначениях выводов (метки выводов, указатели). Дополнительные поля на условно-графическом изображении цифровых микросхем могут отсутствовать. Входы на условно-графическом изображении цифровых микросхем располагают слева, а выходы - справа. Номера выводов микросхем помещают над линией вывода ближе к изображению микросхемы.

2.3 Диодно-транзисторная логика (ДТЛ)

Наиболее простой логический элемент получается при помощи диодов. Схема такого элемента приведена на рисунке 1.

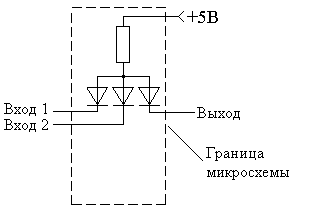


**Рисунок 2.5. Принципиальная схема логического элемента "2И", выполненного на диодах**

В этой схеме при подаче нулевого потенциала на любой из входов (или на оба сразу) через резистор будет протекать ток и на его сопротивлении возникнет падение напряжения. В результате на выходе схемы единичный потенциал будет только если подать единичный потенциал сразу на оба входа микросхемы. То есть схема реализует функцию "2И".

Количество входов элемента "И" зависит от количества диодов. Если использовать два диода, то получится элемент "2И", если три диода - то "3И", если четыре диода, то "4И", и так далее. В микросхемах выпускается максимальный элемент "8И".

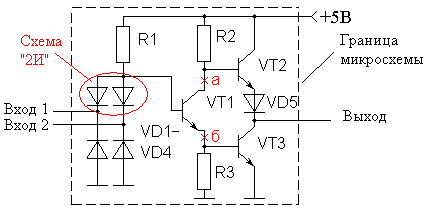
Приведенная схема обладает таким недостатком, как смещение логических уровней на выходе микросхемы. Напряжение нуля и напряжение единицы на выходе схемы выше входных уровней на 0.7 В. Это вызвано падением напряжения на входных диодах. Скомпенсировать это смещение уровней можно диодом, включенном на выходе схемы, как это показано на рисунке 2.



**Рисунок 2.6 Принципиальная схема усовершенствованного логического элемента "2И", выполненного на диодах**

В этой схеме логические уровни на входе и выходе схемы одинаковы. Более того, схема на рисунке 2 будет нечувствительна не только к входным напряжениям, большим напряжения питания схемы, но и к отрицательным входным напряжениям. Диоды выдерживают напряжение до сотен вольт. Поэтому такая схема до сих пор используется для защиты цифровых устройств от перегрузок по напряжению, возникающих, например, в цепях, выходящих за пределы устройства. Естественно, что для защиты одного входа достаточно одного диода на входе элемента. В результате получается только схема защиты без функции "И".

Однако новая схема не может каскадироваться, так как вырабатывает только вытекающий ток, а для следующего каскада требуется втекающий выходной ток схемы. Поэтому к схеме диодного логического элемента "И" обычно подключается двухтактный усилитель на биполярных транзисторах. Схема такого элемента приведена на рисунке 2.7.



**Рисунок 2.7. Принципиальная схема базового элемента ДТЛ микросхемы**

Усилитель, использованный в схеме на рисунке 3, позволяет вырабатывать как втекающий, так и вытекающий выходной ток. Тем не менее следует помнить, что это источник напряжения, и если не ограничить выходной ток микросхемы, то можно вывести ее из строя.

Приведенный на рисунке 3 логический элемент используется в таких современных сериях микросхем как 555, 533, 1531, 1533. В этих сериях микросхем для повышения быстродействия применяются транзисторы и диоды Шоттки.

Обратите внимание, что транзистор VT1 инвертирует сигнал на выходе элемента "И". То есть вместо логической 1 на выходе присутствует логический 0. И наоборот вместо логического нуля на выходе присутствует логическая единица, а схема в целом реализует функцию "2И-НЕ":



Условно-графическое изображение такого логического элемента показано на рисунке 4, а таблица истинности приведена в таблице 1.

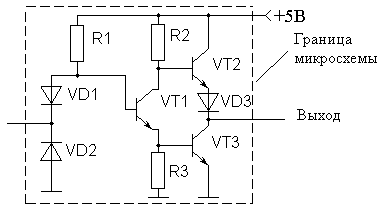


**Рисунок 2.8. Условно-графическое изображение элемента "2И-НЕ".**

**Таблица 2.1. Таблица истинности схемы, выполняющей логическую функцию "2И-НЕ"**

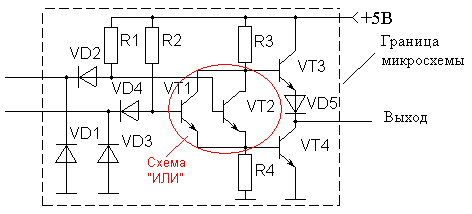
|  |  |  |
| --- | --- | --- |
| x1 | x2 | F |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

На основе базового элемента строится и инвертор. В этом случае на входе используется только один диод. Схема ДТЛ инвертора приведена на рисунке 5.



**Рисунок 2.9. Принципиальная схема инвертора ДТЛ микросхемы**

В состав современных серий микросхем кроме элементов "И" входят логические элементы "ИЛИ" транзисторы VT2 соединяются параллельно в точках "а" и "б", показанных на рисунке 3, а выходной каскад используется один. Схема логического элемента "2ИЛИ-НЕ" приведена на рисунке 6.



**Рисунок 2.10. Принципиальная схема логического элемента "2ИЛИ-НЕ" ДТЛ микросхемы**

Схемы "ИЛИ-НЕ" в этих сериях микросхем имеет обозначение ЛЕ. Например схема К555ЛЕ1 содержит в одном корпусе четыре элемента "2ИЛИ-НЕ". Таблица истинности, реализуемая этой схемой, приведена в таблице 2, а условно-графическое изображение такого элемента показано на рисунке 7.



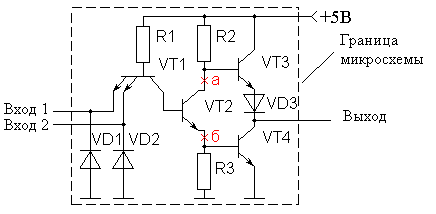
**Рисунок 2.11. Условно-графическое изображение элемента "2ИЛИ-НЕ"**

**Таблица 2.2 Таблица истинности схемы, выполняющей логическую функцию "2ИЛИ-НЕ"**

|  |  |  |
| --- | --- | --- |
| x1 | x2 | F |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

2.4 Транзисторно-транзисторная логика (ТТЛ)

В ТТЛ схемах вместо параллельного соединения диодов используется многоэмиттерный транзистор. Физика работы этого элемента не отличается от работы диодного элемента "2И". Высокий потенциал на выходе многоэмиттерного транзистора получается только в том случае, когда на обоих входах элемента (эмиттерах транзистора) присутствует высокий потенциал (то есть нет эмиттерного тока). Принципиальная схема типового элемента ТТЛ микросхемы приведена на рисунке 1.



**Рисунок 2.12. Принципиальная схема типового элемента ТТЛ микросхемы**

Умощняющий усилитель, как и в диодно-транзисторном элементе, инвертирует сигнал на выходе схемы. По такой схеме выполнены базовые элементы микросхем серий 155, 131, 155 и 531. Схемы "И-НЕ" в этих сериях микросхем обычно имеет обозначение ЛА. Например, схема К531ЛА3 содержит в одном корпусе четыре элемента "2И-НЕ". Таблица истинности, реализуемая этой схемой, приведена в таблице 1, а условно-графическое обозначение этих элементов приведено на рисунке 2.

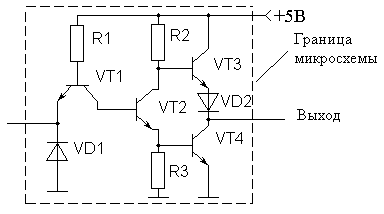


**Рисунок 2.13 Условно-графическое изображение элемента "2И-НЕ"**

**Таблица 2.3. Таблица истинности схемы, выполняющей логическую функцию "2И-НЕ"**

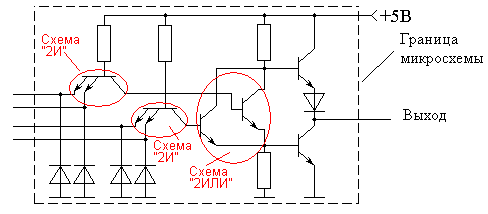
|  |  |  |
| --- | --- | --- |
| x1 | x2 | F |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

На основе базового элемента строится и инвертор. В этом случае на входе используется только один диод. Схема ТТЛ инвертора приведена на рисунке 3.



**Рисунок 2.14. Принципиальная схема инвертора ТТЛ микросхемы**

При необходимости объединения нескольких логических элементов "И" по схеме "ИЛИ" (или при реализации логических элементов "ИЛИ") транзисторы VT2 соединяются параллельно в точках "а" и "б", показанных на рисунке 8, а выходной каскад используется один. В результате быстродействие такого достаточно сложного элемента получается точно таким же как и у одиночного элемента "2И-НЕ". Принципиальная схема логического элемента "2И-2ИЛИ-НЕ" приведена на рисунке 4.



**Рисунок 2.15. Принципиальная схема ТТЛ микросхемы "2И-2ИЛИ-НЕ"**

Использование таких элементов будет показано позднее, а условно-графическое изображение элемента "2И-2ИЛИ-НЕ" приведено на рисунке 5. Такие элементы содержатся в цифровых микросхемах с обозначением ЛР.



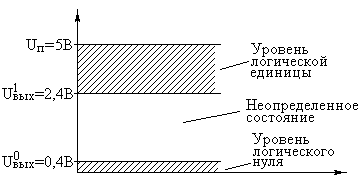
**Рисунок 2.16. Условно-графическое обозначение элемента "2И-2ИЛИ-НЕ" ТТЛ микросхем**

Схемы "ИЛИ-НЕ" в ТТЛ сериях микросхем обычно имеет обозначение ЛЕ. Например схема К1531ЛЕ5 содержит в одном корпусе четыре элемента "2ИЛИ-НЕ".

Так как и в схемах ТТЛ и в схемах ДТЛ используется одинаковый выходной усилитель, то и уровни логических сигналов в этих схемах одинаковы. Поэтому часто говорят, что это ТТЛ микросхемы, не уточняя по какой схеме выполнен входной каскад этих микросхем. Более того! Появились КМОП микросхемы, совместимые с ТТЛ микросхемами по логическим уровням, например К1564 (иностранный аналог SN74HCT) или К1594 (иностранный аналог SN74АСT).

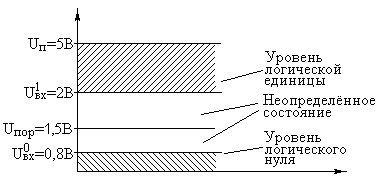
2.4.1 Логические уровни ТТЛ микросхем

В настоящее время применяются два вида ТТЛ микросхем - с пяти и и с трёхвольтовым питанием, но, независимо от напряжения питания микросхем, логические уровни нуля и единицы на выходе этих микросхем совпадают. Поэтому дополнительного согласования между ТТЛ микросхемами обычно не требуется. Допустимый уровень напряжения на выходе цифровой ТТЛ микросхемы показан на рисунке 6.



**Рисунок 2.17. Уровни логических сигналов на выходе цифровых ТТЛ микросхем**

Как уже говорилось ранее, напряжение на входе цифровой микросхемы по сравнению с выходом обычно допускается в больших пределах. Границы уровней логического нуля и единицы для ТТЛ микросхем приведена на рисунке 7.



**Рисунок 2.18. Уровни логических сигналов на входе цифровых ТТЛ микросхем**

2.4.2 Семейства ТТЛ микросхем

Первые ТТЛ микросхемы оказались на редкость удачным решением, поэтому их можно встретить в аппаратуре, работающей до сих пор. Это семейство микросхем серии К155. Стандартные ТТЛ микросхемы - это микросхемы, питающиеся от источника напряжения +5В. Зарубежные ТТЛ микросхемы получили название SN74. Конкретные микросхемы этой серии обозначаются цифровым номером микросхемы, следующим за названием серии. Например, в микросхеме SN74S00 содержится четыре логических элемента "2И-НЕ". Аналогичные микросхемы с расширенным температурным диапазоном получили название SN54 (отечественный вариант - серия микросхем К133).

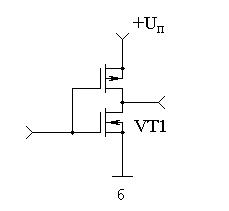
Отечественные микросхемы, совместимые с SN74 выпускались в составе серий К134 (низкое быстродействие низкое потребление - SN74L), К155 (среднее быстродействие среднее потребление - SN74) и К131 (высокое быстродействие и большое потребление). Затем были выпущены микросхемы повышенного быстродействия с диодами Шоттки. В названии зарубежных микросхем в обозначении серии появилась буква S. Отечественные серии микросхем сменили цифру 1 на цифру 5. Выпускаются микросхемы серий К555 (низкое быстродействие низкое потребление - SN74LS) и К531 (высокое быстродействие и большое потребление - SN74S).

В настоящее время отечественная промышленность производит микросхемы серий К1533 (низкое быстродействие низкое потребление - SN74ALS) и К1531 (высокое быстродействие и большое потребление - SN74F).

В Европе производится трехвольтовый вариант ТТЛ микросхем - SN74ALB

2.5 Логика на комплементарных МОП транзисторах (КМДП)

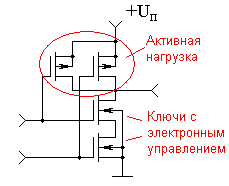
Микросхемы на комплементарных транзисторах строятся на основе МОП транзисторов с n- и p-каналами. Один и тот же потенциал открывает транзистор с n-каналом и закрывает транзистор с p-каналом. При формировании логической единицы открыт верхний транзистор, а нижний закрыт. В результате ток через микросхему не протекает. При формировании логического нуля открыт нижний транзистор, а верхний закрыт. И в этом случае ток через микросхему не протекает. Простейший логический элемент - это инвертор. Его схема приведена на рисунке 1.



**Рисунок 2.19. Принципиальная схема инвертора, выполненного на комплементарных МОП транзисторах**

На этой схеме для упрощения понимания принципов работы микросхемы не показаны защитные и паразитные диоды. Особенностью микросхем на комплементарных МОП транзисторах является то, что в этих микросхемах в статическом режиме ток практически не потребляется. Потребление тока происходит только в момент переключения микросхемы из единичного состояния в нулевое и наоборот. Этим током производится перезаряд паразитной ёмкости нагрузки.

Схема логического элемента "И-НЕ" на КМОП микросхемах практически совпадает с упрощенной схемой "И" на ключах с электронным управлением, которую мы рассматривали ранее. Отличие заключается в том, что нагрузка подключается не к общему проводу схемы, а к источнику питания. Принципиальная схема элемента "2И-НЕ", выполненного на комплементарных МОП транзисторах приведена на рисунке 2.



**Рисунок 2.20 Принципиальная схема элемента "2И-НЕ", выполненного на комплементарных МОП транзисторах**

В этой схеме можно было бы применить в верхнем плече обыкновенный резистор, однако при формировании низкого уровня схема постоянно потребляла бы ток. Вместо этого, в качестве нагрузки используются p-МОП транзисторы. Эти транзисторы образуют активную нагрузку. Если на выходе требуется сформировать высокий потенциал, то транзисторы открываются, а если низкий - то закрываются.

В приведённой на рисунке 2 схеме ток от источника питания на выход микросхемы будет поступать через один из транзисторов, если хотя бы на одном из входов (или на обоих сразу) будет присутствовать низкий потенциал (уровень логического нуля). Если же на обоих входах будет присутствовать уровень логической единицы, то оба p-МОП транзистора будут закрыты и на выходе микросхемы сформируется низкий потенциал. В этой схеме, так же как и в схеме на рисунке 1, если транзисторы верхнего плеча будут открыты, то транзисторы нижнего плеча будут закрыты, поэтому в статическом состоянии ток микросхемой от источника питания потребляться не будет.

Условно-графическое изображение такого логического элемента показано на рисунке 3, а таблица истинности приведена в таблице 1. В таблице 1 входы обозначены как x1 и x2, а выход - F.

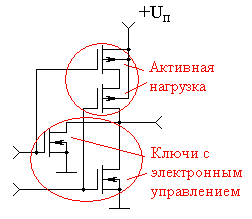


**Рисунок 3. Условно-графическое изображение элемента "2И-НЕ"**

**Таблица 2.5 Таблица истинности схемы, выполняющей логическую функцию "2И-НЕ"**

|  |  |  |
| --- | --- | --- |
| x1 | x2 | F |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Логический элемент "ИЛИ-НЕ", выполненный на КМОП транзисторах, представляет собой параллельное соединение ключей с электронным управлением. Отличие от схемы "2ИЛИ", рассмотренной ранее, заключается в том, что нагрузка подключается не к общему проводу схемы, а к источнику питания. Вместо резистора в качестве нагрузки используются p-МОП транзисторы. Принципиальная схема элемента "2ИЛИ-НЕ", выполненного на комплементарных МОП транзисторах приведена на рисунке 4.



**Рисунок 2.21. Принципиальная схема элемента "2ИЛИ-НЕ", выполненного на комплементарных МОП транзисторах**

В схеме логического элемента "2ИЛИ-НЕ" в качестве нагрузки используются последовательно включенные p-МОП транзисторы. В ней ток от источника питания на выход микросхемы будет поступать только если все транзистора в верхнем плече будут открыты, т.е. если сразу на всех входах будет присутствовать низкий потенциал (уровень логического нуля). Если же хотя бы на одном из входов будет присутствовать уровень логической единицы, то верхнее плечо будет закрыто и ток от источника питания поступать на выход микросхемы не будет.

Таблица истинности, реализуемая этой схемой, приведена в таблице 2, а условно-графическое обозначение этих элементов приведено на рисунке 5.



**Рисунок 2.22. Условно-графическое изображение элемента "2ИЛИ-НЕ"**

**Таблица 2.6 Таблица истинности схемы, выполняющей логическую функцию "2ИЛИ-НЕ"**

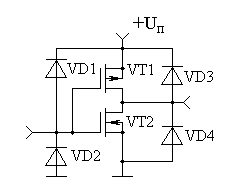
|  |  |  |
| --- | --- | --- |
| x1 | x2 | F |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

В настоящее время именно КМОП микросхемы получили наибольшее развитие. Причём наблюдается постоянная тенденция к снижению напряжения питания. Первые серии микросхем такие как К1561 (иностранный аналог C4000В) обладали достаточно широким диапазоном изменения напряжения питания (3..18В). При этом при понижении напряжения питания у конкретной микросхемы понижается её предельная частота работы. В дальнейшем, по мере совершенствования технологии производства, появились улучшенные микросхемы с лучшими частотными свойствами и меньшим напряжением питания.

2.5.1 Особенности применения КМОП микросхем

Первой и основной особенностью КМОП микросхем является большое входное сопротивление этих микросхем. В результате на вход этой цифровой микросхемы может наводиться любое напряжение, в том числе и равное половине напряжения питания, и храниться на нём достаточно долго. При подаче на вход КМОП микросхемы половины питания открываются транзисторы как в верхнем, так и в нижнем плече выходного каскада микросхемы, в результате микросхема начинает потреблять недопустимо большой ток и может выйти из строя. Вывод: входы цифровых микросхем ни в коем случае нельзя оставлять неподключенными.

Второй особенностью КМОП микросхем является то, что они могут работать при отключенном питании. Однако работают они чаще всего неправильно. Эта особенность связана с конструкцией входного каскада КМОП микросхем. Полная схема КМОП инвертора приведена на рисунке 6.



**Рисунок 2.23. Полная схема КМОП инвертора**

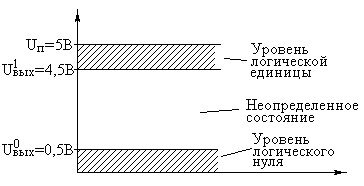
Диоды VD1 и VD2 были введены для защиты входного каскада от пробоя статическим электричеством. В то же самое время при подаче на вход микросхемы высокого потенциала он через диод VD1 попадёт на шину питания микросхемы, и так как она потребляет достаточно малый ток, то микросхема начнёт работать. Однако в ряде случаев тока может не хватить. В результате микросхема может работать неправильно. Вывод: при неправильной работе микросхемы тщательно проверьте питание микросхемы, особенно выводы корпуса. При плохо пропаянном выводе отрицательного питания его потенциал будет отличаться от потенциала общего провода схемы.

Третья особенность КМОП микросхем связана с паразитными диодами VD3 и VD4, которые могут быть пробиты при неправильно подключенном источнике питания (микросхемы ТТЛ выдерживают кратковременную переполюсовку питания). Для защиты микросхем от переполюсовки питания следует в цепи питания предусмотреть защитный диод.

Четвёртая особенность КМОП микросхем - это протекание импульсного тока по цепи питания при переключении микросхемы из нулевого состояния в единичное и наоборот. В результате при переходе с ТТЛ микросхем на КМОП резко увеличивается уровень помех. В ряде случаев это важно и приходится отказываться от применения КМОП микросхем в пользу ТТЛ или BICMOS.

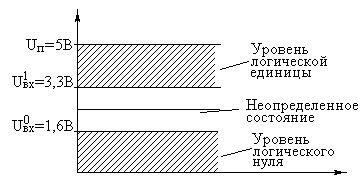
2.5.2 Логические уровни КМОП микросхем

Логические уровни КМОП микросхем существенно отличаются от логических уровней ТТЛ микросхем. При отсутствии тока нагрузки напряжение на выходе КМОП микросхемы совпадает с напряжением питания (логический уровень единицы) или с потенциалом общего провода (логический уровень нуля). При увеличении тока нагрузки напряжение логической единицы может уменьшается до 2,8В (Uп=15В) от напряжения питания. Допустимый уровень напряжения на выходе цифровой КМОП микросхемы (серия микросхем К561) при пятивольтовом питании показан на рисунке 7.



**Рисунок 2.24 Уровни логических сигналов на выходе цифровых КМОП микросхем**

Как уже говорилось ранее, напряжение на входе цифровой микросхемы по сравнению с выходом обычно допускается в больших пределах. Для КМОП микросхем договорились о 30% запасе. Границы уровней логического нуля и единицы для КМОП микросхем при пятивольтовом питании приведена на рисунке 8.



**Рисунок 2.25 Уровни логических сигналов на входе цифровых КМОП микросхем**

При уменьшении напряжения питания границы логического нуля и логической единицы можно определить точно так же (разделить напряжение питания на 3).

2.5.3 Семейства КМОП микросхем

Первые КМОП микросхемы не имели защитных диодов на входе, поэтому их монтаж представлял значительные трудности. Это семейство микросхем серии К172. Следующее улучшенное семейство микросхем серии К176 получило эти защитные диоды. Оно достаточно распространено и в настоящее время. Серия К1561 завершает развитие первого поколения КМОП микросхем. В этом семействе было достигнуто быстродействие на уровне 90нс и диапазон изменения напряжения питания 3..15В. Так как в настоящее время распространена иностранная аппаратура, то приведу иностранный аналог этих микросхем - C4000В.

Дальнейшим развитием КМОП микросхем стала серия SN74HC. Эти микросхемы отечественного аналога не имеют. Они обладают быстродействием 27нс и могут работать в диапазоне напряжений 2..6В. Они совпадают по цоколёвке и функциональному ряду с ТТЛ микросхемами, но не совместимы с ними по логическим уровням, поэтому одновременно были разработаны микросхемы серии SN74HCT (отечественный аналог - К1564), совместимые с ТТЛ микросхемами и по логическим уровням.

В это время наметился переход на трёхвольтовое питание. Для него были разработаны микросхемы SN74ALVC с временем задержки сигнала 5,5нс и диапазоном питания 1,65..3,6В. Эти же микросхемы способны работать и при 2,5 вольтовом питании. Время задержки сигнала при этом увеличивается до 9нс.

Наиболее перспективным семейством КМОП микросхем считается семейство SN74AUC с временем задержки сигнала 1,9нс и диапазоном питания 0,8..2,7В.

**3. ИНФОРМАЦИОННО-СПРАВОЧНАЯ СИСТЕМА**

**3.1 Определение и классификация БД**

База данных – это информационная модель предметной области, совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений. Данные (файлы) хранятся во внешней памяти и используются в качестве входной информации для решения задач.

СУБД - это программа, с помощью которой реализуется централизованное управление данными, хранимыми в базе, доступ к ним, поддержка их в актуальном состоянии.

Системы управления базами данных можно классифицировать по способу установления связей между данными, характеру выполняемых ими функций, сфере применения, числу поддерживаемых моделей данных, характеру используемого языка общения с базой данных и другим параметрам.

Классификация СУБД:

* по выполняемым функциям СУБД подразделяются на операционные и информационные;
* по сфере применения СУБД подразделяются на универсальные и проблемно-ориентированные;
* по используемому языку общения СУБД подразделяются на замкнутые, имеющие собственные самостоятельные языки общения пользователей с базами данных, и открытые, в которых для общения с базой данных используется язык программирования, расширенный операторами языка манипулирования данными;
* по числу поддерживаемых уровней моделей данных СУБД подразделяются на одно-, двух-, трехуровневые системы;
* по способу установления связей между данными различают реляционные, иерархические и сетевые базы данных;
* по способу организации хранения данных и выполнения функций обработки базы данных подразделяются на централизованные и распределенные.

Системы централизованных баз данных с сетевым доступом предполагают две основные архитектуры – файл-сервер или клиент-сервер.

Архитектура файл-сервер. Предполагает выделение одной из машин сети в качестве центральной (главный сервер файлов), где хранится совместно используемая централизованная база данных. Все другие машины исполняют роль рабочих станций. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится их обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает.

Архитектура клиент-сервер. Эта модель взаимодействия компьютеров в сети для современных СУБД фактически стала стандартом. Каждый из подключенных к сети и составляющих эту архитектуру компьютеров играет свою роль: сервер владеет и распоряжается информационными ресурсами системы, клиент имеет возможность пользоваться ими. Помимо хранения централизованной базы данных сервер базы данных обеспечивает выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запроса SQL.

Сервер базы данных представляет собой СУБД, параллельно обрабатывающую запросы, поступившие со всех рабочих станций. Как правило, клиент и сервер территориально отделены друг от друга, и в этом случае они образуют систему распределенной обработки данных.

Характеристиками СУБД являются:

* производительность;
* обеспечение целостности данных на уровне баз данных;
* обеспечение безопасности данных;
* возможность работы в многопользовательских средах;
* возможность импорта и экспорта данных;
* обеспечение доступа к данным с помощью языка SQL;
* возможность составления запросов;
* наличие инструментальных средств разработки прикладных программ.

Производительность СУБД оценивается:

* временем выполнения запросов;
* скоростью поиска информации;
* временем импортирования баз данных из других форматов;
* скоростью выполнения операций (таких как обновление, вставка, удаление);
* временем генерации отчета и другими показателями.

Безопасность данных достигается:

* шифрованием прикладных программ;
* шифрованием данных;
* защитой данных паролем;
* ограничением доступа к базе данных (к таблице, к словарю и т.д.).

Обеспечение целостности данных подразумевает наличие средств, позволяющих удостовериться, что информация в базе данных всегда остается корректной и полной. Целостность данных должна обеспечиваться независимо от того, каким образом данные заносятся в память (в интерактивном режиме, посредством импорта или с помощью специальной программы). Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности.

Система управления базами данных управляет данными во внешней памяти, обеспечивает надежное хранение данных и поддержку соответствующих языков базы данных. Важной функцией СУБД является функция управления буферами оперативной памяти. Обычно СУБД работают с базами данных больших размеров, часто превышающими размеры оперативной памяти ЭВМ. В развитых СУБД поддерживается свой набор буферов оперативной памяти с собственной дисциплиной их замены.

Наибольшее распространение в настоящее время получили системы управления базами данных Microsoft Access и Oracle.

Этапами работы в СУБД являются:

* создание структуры базы данных, т.е. определение перечня полей, из которых состоит каждая запись таблицы, типов и размеров полей (числовой, текстовый, логический и т.д.), определение ключевых полей для обеспечения необходимых связей между данными и таблицами;
* ввод и редактирование данных в таблицах баз данных с помощью представляемой по умолчанию стандартной формы в виде таблицы и с помощью экранных форм, специально создаваемых пользователем;
* обработка данных, содержащихся в таблицах, на основе запросов и на основе программы;
* вывод информации из ЭВМ с использованием отчетов и без использования отчетов.

Реализуются названные этапы работы с помощью различных команд.

Централизованная база данных обеспечивает простоту управления, улучшенное использование данных на местах при выполнении дистанционных запросов, более высокую степень одновременности обработки, меньшие затраты на обработку.

Распределенная база данных предполагает хранение и выполнение функций управления данными в нескольких узлах и передачу данных между этими узлами в процессе выполнения запросов. В такой базе данных не только различные ее таблицы могут храниться на разных компьютерах, но и разные фрагменты одной таблицы. При этом для пользователя не имеет значения как организовано хранение данных, он работает с такой базой, как с централизованной.

Известны три типа моделей описания баз данных – иерархическая, сетевая и реляционная, основное различие между которыми состоит в характере описания взаимосвязей и взаимодействия между объектами и атрибутами базы данных.

Иерархическая модель предполагает использование для описания базы данных древовидных структур, состоящих из определенного числа уровней. «Дерево» представляет собой иерархию элементов, называемых узлами. Под элементами понимается список, совокупность, набор атрибутов, элементов, описывающих объекты.

В качестве примера простой иерархической структуры можно привести административную структуру высшего учебного заведения, элементами которой являются: «Университет – Факультет – Группа». На каждом уровне иерархии данной структуры могут быть использованы различные атрибуты. Например, атрибутами третьего уровня могут быть: специализация группы, численный состав, фамилия старосты группы и другие.

В данной модели имеется корневой узел или просто корень – «Университет», который находится на самом верхнем уровне иерархии, а потому не имеет узлов, стоящих выше его. Каждый узел модели имеет только один исходный, находящийся по отношению к нему на более высоком уровне, а на последующих уровнях классификации он может иметь один, два или большее количество узлов, либо не иметь их вообще.

Принципы иерархии:

* иерархия всегда начинается с корневой вершины (или главного узла);
* исходный узел, из которого строится дерево, называется корневым узлом или просто корнем, причем одно дерево может иметь только один корень;
* узел может содержать один или несколько атрибутов, описывающих находящийся в нем объект;
* порожденные узлы могут встраиваться в «дерево» как в горизонтальном направлении, так и в вертикальном;
* доступ к порожденным узлам возможен только через исходный узел, поэтому существует только один путь доступа к каждому узлу.

Достоинством модели является простота ее построения, легкость понимания сути принципа иерархии, наличие промышленных СУБД, поддерживающих данную модель. Недостатком является сложность операций по включению в иерархию информации о новых объектах базы данных и удалению устаревшей информации.

Сетевая модель описывает элементарные данные и отношения между ними в виде ориентированной сети. Это такие отношения между объектами, когда каждый порожденный элемент имеет более одного исходного и может быть связан с любым другим элементом структуры. Например, в структуре управления учебным заведением порожденный элемент «Студент» может иметь не один, а два исходных элемента: «Студент – Учебная группа», и «Студент – Комната в общежитии».

Сетевые структуры могут быть многоуровневыми и иметь разную степень сложности. Схема, в которой присутствует хотя бы одна связь «многие ко многим» и которая требует для своей реализации использования сложных методов, является сложной схемой.

База данных, описываемая сетевой моделью, состоит из областей, каждая из которых состоит из записей, а последние, в свою очередь, состоят из полей. Недостатком сетевой модели является ее сложность, возможность потери независимости данных при реорганизации базы данных. При появлении новых пользователей, новых приложений и новых видов запросов происходит рост базы данных, что может привести к нарушению логического представления данных.

Реляционная модель имеет в своей основе понятие «отношения», и ее данные формируются в виде таблиц. Отношение – это двумерная таблица, имеющая сове название, в которой минимальным объектом действий, сохраняющим ее структуру, является строка таблицы (кортеж), состоящая из ячеек таблицы – полей.

Каждый столбец таблицы соответствует только одной компоненте этого отношения. С логической точки зрения реляционная база данных представляется множеством двумерных таблиц различного предметного наполнения.

В зависимости от содержания отношения реляционной базы данных бывают объективными и связными. Объективные отношения хранят данные о каком-либо одном объекте, экземпляре сущности. В них один из атрибутов однозначно определяет объект и называется ключом отношения или первичным атрибутом (для удобства он записывается в первом столбце таблицы). Остальные атрибуты функционально зависят от этого ключа. В объективном отношении не может быть дублирующих объектов и в этом – основное ограничения реляционной базы данных. Связное отношение хранит ключи нескольких объектных отношений, по которым между ними устанавливаются связи.

Если набор атрибутов базы данных заранее не фиксирован, то возможны различные варианты их группировки, однако, независимо от выбранного способа, должны соблюдаться единые требования. В частности, если база данных содержит множество отношений, то они должны иметь минимальную избыточность представления информации; атрибуты, включаемые в базу данных, должны обеспечивать выполнение массовых расчетов; при добавлении в базу данных новых атрибутов перестройка наборов отношений должна быть минимальной.

К числу достоинств реляционной модели относятся: простота построения, доступность понимания, возможность эксплуатации базы данных без знания методов и способов ее построения, независимость данных, гибкость структуры и другие. Недостатками модели являются: низкая производительность по сравнению с иерархической и сетевой моделями, сложность программного обеспечения, избыточность.

**3.2 Компилятор Visual C++, версия 6**

* Стандартный вариант
* Профессиональный вариант
* Корпоративный вариант
* Инструменты разработчика
  + Интегрированный отладчик
  + Встроенные редакторы ресурсов
  + Дополнительные утилиты
* Возможности компилятора
  + Средства автоматизации и макросы
  + ClassView
  + Настраиваемые панели инструментов и меню
  + Рабочие пространства и файлы проектов
  + Предварительно скомпилированные файлы заголовков
  + MFC
  + Макроподстановка функций
* Опции компиляции
  + General
  + Debug
  + C/C++
  + Link
  + Resources
  + MIDL
  + Browse Info
  + Custom Build

Новая версия VisualC++ позволет создавать любые приложения для Windows95, 98 и NT с использованием новейших программных технологий и методик. Пакет программ MicrosoftVisualC++ версии 6 поставляется в трех различных вариантах: стандартном, профессиональном и корпоративном.

Стандартный вариант VisualC++ (ранее он назывался учебным) содержит почти все те же средства, что и профессиональный, но в отличие от последнего в нем отсутствуют модуль Profiler, несколько мастеров, возможности по оптимизации кода и статической компоновке библиотеки MFC , некоторые менее важные функции. Этот вариант в наибольшей мере подходит для студентов и иных категорий индивидуальных пользователей, чему, в частности, способствует и сравнительно низкая его цена. Лицензионное соглашение стандартного варианта программы в новой версии, в отличие от старой, разрешает ее использование для создания коммерческих программных продуктов.

Профессиональный вариант В этом варианте программы можно создавать полнофункциональные графические и консольные приложения для всех платформ Win32, включая Windows95, 98 и NT. Перечислим новые возможности профессионального варианта:

* • поддержка автоматического дополнения выражений (технология IntelHSense);
* • шаблоны OLEDB; • средства визуального проектирования приложений, работающих с базами данных
* • (без возможности модификации данных).

Корпоративный вариант С помощью корпоративного варианта VisualC++ вы можете создавать приложения типа клиент/сервер для работы в Internet или в корпоративной среде (intranet). Приобретая корпоративный вариант программы, вы получаете в свое распоряжение не только все возможности профессионального варианта, но и ряд дополнительных средств, включая:

• Microsoft Transaction Server;

• средства визуального проектирования приложений, работающих с базами данных;

• модуль SQL Editor;

• модуль SQL Debugger;

• классы библиотеки MFC для доступа к базам данных;

• ADO Data-Bound Dialog Wizard;

• Поддержка технологии Remote Automation;

• Visual SourceSafe.

Компилятор MicrosoftVisualC++ содержит также средства, позволяющие в среде Windows разрабатывать приложения для других платформ, в том числе для AppleMacintosh. Кроме того, программа снабжена редакторами растровых изображений, значков, указателей мыши, меню и диалоговых окон, позволяющими работать с перечисленными ресурсами непосредственно в интегрированной среде.

Коснувшись темы интеграции, следует упомянуть о мастере ClassWizard, помогающем в кратчайшие сроки создавать приложения OLE с использованием библиотеки MFC . Инструменты разработчика Новая версия компилятора MicrosoftVisualC++ содержит множество интегрированных средств визуального программирования. Ниже перечислены утилиты, которые вы можете использовать непосредственно из VisualC++. Интегрированный отладчик Разработчики компании Microsoft встроили первоначальный отладчик CodeView непосредственно в среду VisualC++. Команды отладки вызываются из меню Debug. Встроенный отладчик позволяет пошагово выполнять программу, просматривать и изменять значения переменных и многое другое.

Встроенные редакторы ресурсов позволяют создавать и модифицировать ресурсы Windows, такие как растровые изображения, указатели мыши, значки, меню, диалоговые окна и т.д. Редактор диалоговых окон — это достаточно удобное средство, позволяющее легко и быстро создавать сложные диалоговые окна. С помощью этого редактора в разрабатываемое диалоговое окно можно включить любые элементы управления (например, надписи, кнопки, флажки, списки и т.д.). При этом вы можете изменять как внешний вид элементов управления, так и их свойства (наряду со свойствами самого диалогового окна). Редактор изображений позволяет быстро создавать и редактировать собственные растровые изображения, значки и указатели мыши. Пользовательские ресурсы данного типа сохраняются в файле с расширением RC и включаются в файлы сценариев ресурсов. Редактор двоичных кодов позволяет вносить изменения непосредственно в двоичный код ресурса. Редактор двоичных кодов следует использовать только для просмотра ресурсов или внесения мелких изменений в те из них, тип которых не поддерживается в VisualC++. Редактор строк представляет собой ресурс, содержащий список идентификаторов и значений всех строковых надписей, используемых в приложении. К примеру, в этой таблице могут храниться сообщения, отображаемые в строке состояния. Каждое приложение содержит единственную таблицу строк. Наличие единой таблицы позволяет легко менять язык интерфейса программы — для этого достаточно перевести на другой язык строки таблицы, не затрагивая код программы.

Дополнительные утилиты ActiveX Control Test Container С помощью этой утилиты, разработанной специалистами Microsoft, вы можете быстро протестировать созданные вами элементы управления. При этом можно изменять их свойства и характеристики. APITextViewer Данная утилита позволяет просматривать объявления констант, функций и типов данных Win32 API, а также копировать эти объявления в приложения VisualBasic или в буфер обмена. AVIEditor Эта утилита позволяет просматривать, редактировать и объединять AVI-файлы. DataObjectViewer Утилита DataObjectViewer отображает список форматов данных, предлагаемых объектами ActiveX и OLE, которые находятся в буфере обмена или участвуют в операции перетаскивания (drag-and-drop). DDESpy эта утилита предназначена для отслеживания всех DDE-сообщений. DocFileViewer эта утилита отображает содержимое составных OLE-документов. ErrorLookup ErrorLookup эта утилита позволяет просматривать и анализировать всевозможные сообщения об ошибках. HeapWalkUtility Эта утилита выводит список блоков памяти, размещенных в указанной динамической области (куче). HelpWorkshop Эта утилита позволяет создавать и редактировать файлы справки. OLE Client/Server, Tools иView Утилита OLEViewer отображает информацию об объектах ActiveX и OLE, инсталлированных на вашем компьютере. Эта утилита также позволяет редактировать реестр и просматривать библиотеки типов. Утилиты OLE Client и OLE Server предназначены для тестирования OLEклиентов и серверов. TheProcessViewer Эта утилита позволяет следить за состоянием выполняющихся процессов и потоков. ROTViewer Эта утилита отображает информацию об объектах ActiveX и OLE, в данный момент загруженных в память. Spy++ Эта утилита выводит сведения о выполняющихся процессах, потоках, существующих окнах и оконных сообщениях. StressUtility Эта утилита позволяет захватывать системные ресурсы и используется для тестирования системы в ситуациях, связанных с недостатком системных ресурсов. В число захватываемых ресурсов входят глобальная и пользовательская динамические области (кучи), динамическая область GDI, свободные области дисков и дескрипторы файлов. Утилита Stress может выделять фиксированное количество ресурсов, а также производить выделение в ответ на получение различных сообщений. Кроме того, утилита способна вести журнал событий, что помогает обнаруживать и воспроизводить аварийные ситуации в работе программы. MFC Tracer Эта утилита позволяет устанавливать флаги трассировки в файле AFX.INI. С помощью данных флагов можно выбрать типы сообщений, которые будут посылаться приложением в окно отладки. Таким образом, утилита Tracer является средством отладки. UUIDGenerator Эта утилита предназначена для генерации универсального уникального идентификатора (UUID), который позволяет клиентским и серверным приложениям распознавать друг друга. WinDiff Эта утилита дает возможность сравнивать содержимое файлов и папок. Zooming Эту утилиту можно использовать для захвата и просмотра в увеличенном виде выбранной области на рабочем столе.

Компилятор VisualC++ содержит много новых инструментальных средств и улучшенных возможностей. В следующих параграфах дается их краткий обзор. Средства автоматизации и макросы С помощью сценариев VisualBasic вы можете автоматизировать выполнение рутинных и повторяющихся задач. VisualC++ позволяет записывать в макрокомандах самые разные операции со своими компонентами, включая открытие, редактирование и закрытие документов, изменение размеров окон. Можно также создавать надстроечные модули, интегрируя их в среду с использованием объектной модели VisualC++. ClassView Вкладка ClassView теперь позволяет работать с классами Java так же, как с классами C++.

Вы можете просматривать и редактировать интерфейсы СОМ-объектов, созданных на базе MFC или ALT, а также разбивать классы по папкам удобным для вас образом. Стало легче настраивать панели инструментов и меню в соответствии с вашими предпочтениями. В частности, вы можете выполнять следующие действия:

* добавлять меню в панель инструментов;
* добавлять и удалять команды меню и кнопки панели инструментов;
* заменять кнопки панели инструментов соответствующими командами меню;
* создавать копии команд меню или кнопок панелей инструментов на разных панелях, с тем чтобы облегчить доступ к ним в разных ситуациях;
* создавать новые панели инструментов и меню;
* настраивать внешний вид существующих панелей инструментов и меню;
* назначать команды меню новым кнопкам панелей инструментов.

Файлы рабочего пространства теперь имеют расширение DSW(раньше использовалось расширение MDP). Создаваемые проекты записываются в файлы двух типов: внутренние (DSP) и внешние (МАК). Файлы с расширением DSP создаются при выборе нового проекта или при открытии файла проекта, созданного в ранней версии программы. Чтобы сохранить проект во внешнем файле с расширением МАК, используйте команду Export Makefile из меню Project. Проекты теперь могут содержать активные документы, например электронные таблицы или текстовые документы Word. Вы можете редактировать их, даже не покидая VisualStudio. Когда создается новое рабочее пространство, VisualC++ создает файл имя\_рабочегo\_npocmpaнcтвa.DSW. Эти файлы больше не содержат данных, специфичных для вашего компьютера. Предварительно скомпилированные файлы заголовков VisualC++ помещает описания типов данных, прототипы функций, внешние ссылки и объявления функций-членов в специальные файлы, называемые файлами заголовков. Эти файлы содержат важные определения, необходимые во многих местах программы. Части файлов заголовков обычно повторно компилируются при компиляции каждого из включающих их модулей. К сожалению, повторная компиляция значительно замедляет работу компилятора.

VisualC++ позволяет существенно ускорить этот процесс за счет возможности предварительной компиляции файлов заголовков. Хотя идея не нова, для ее реализации специалисты Microsoft использовали принципиально новый подход. Предварительной компиляции может подвергнуться только "стабильная" часть файла; оставшаяся же часть, которая впоследствии может модифицироваться, будет компилироваться вместе с приложением.

Это средство удобно применять, например, в том случае, когда в процессе разработки приложения приходится часто изменять программный код, сохраняя описание классов. Предварительная компиляция файлов заголовков приведет к значительному ускорению работы с программой и в тех случаях, когда заголовки заключают в себе больше программного кода, чем основной модуль.

Компилятор VisualC++ предполагает, что текущее состояние рабочей среды идентично тому, которое было при компиляции заголовков. В случае обнаружения каких-либо конфликтов будет выдано предупреждающее сообщение. Такие ситуации могут возникать при изменении модели использования памяти, значений предопределенных констант или опций отладки/компиляции.

В отличие от многих других компиляторов C++, VisualC++ не ограничивается предварительной компиляцией только файлов заголовков. Благодаря возможности проводить компиляцию до заданной точки программы вы можете предварительно скомпилировать даже какую-нибудь часть основного модуля. В целом, процедура предварительной компиляции используется для тех частей программного кода, которые вы считаете стабильными; таким образом значительно сокращается время компиляции частей программы, изменяемых в процессе работы над ней.

Приложения Windows просты в использовании, но создавать их довольно сложно. Программистам приходится изучать сотни различных API-функций. Чтобы облегчить их труд, специалисты Microsoft разработали библиотеку MicrosoftFoundationClasses— MFC . Используя готовые классы C++, можно гораздо быстрее и проще решать многие задачи. Библиотека MFC существенно облегчает программирование в среде Windows. Те, кто обладает достаточным опытом программирования на C++, могут дорабатывать классы или создавать новые, производные от существующих.

Классы библиотеки MFC используются как для управления объектами Windows, так и для решения определенных общесистемных задач. Например, в библиотеке имеются классы для управления файлами, строками, временем, обработкой исключений и другие.

По сути, в MFC представлены практически все функции WindowsAPI. В библиотеке имеются средства обработки сообщений, диагностики ошибок и другие средства, обычные для приложений Windows. MFC обладает следующими преимуществами.

* Представленный набор функций и классов отличается логичностью и полнотой. Библиотека MFC открывает доступ ко всем часто используемым функциям WindowsAPI, включая функции управления окнами приложений, сообщениями, элементами управления, меню, диалоговыми окнами, объектами GDI (GraphicsDeviceInterface— интерфейс графических устройств), такими как шрифты, кисти, перья и растровые изображения, функции работы с документами и многое другое.
* Функции MFC легко изучать. Специалисты Microsoft приложили все усилия для того, чтобы имена функций MFC и связанных с ними параметров были максимально близки к их эквивалентам из WindowsAPI. Благодаря этому программисты легко смогут разобраться в их назначении.
* Программный код библиотеки достаточно эффективен. Скорость выполнения приложений, основанных на MFC , будет примерно такой же, как и скорость выполнения приложений, написанных на С с использованием стандартных функций WindowsAPI, а дополнительные затраты оперативной памяти будут весьма незначительными.
* • MFC содержит средства автоматического управления сообщениями. Библиотека MFC устраняет необходимость в организации цикла обработки сообщений — распространенного источника ошибок в Windows-приложениях. В MFC предусмотрен автоматический контроль за появлением каждого сообщения. Вместо использования стандартного блока switch/case все сообщения Windows связываются с функциямичленами, выполняющими соответствующую обработку.
* MFC позволяет организовать автоматический контроль за выполнением функций. Эта возможность реализуется за счет того, что вы можете записывать в отдельный файл информацию о различных объектах и контролировать значения переменных-членов объекта в удобном для понимания формате.
* MFC имеет четкий механизм обработки исключительных ситуаций. Библиотека MFC была разработана таким образом, чтобы держать под контролем появление таких ситуаций. Это позволяет объектам MFC восстанавливать работу после появления ошибок типа "outofmemory" (нехватка памяти), неправильного выбора команд меню или проблем с загрузкой файлов либо ресурсов.
* MFC обеспечивает динамическое определение типов объектов. Это чрезвычайно мощное программное средство, позволяющее отложить проверку типа динамически созданного объекта до момента выполнения программы. Благодаря этому вы можете свободно манипулировать объектами, не заботясь о предварительном описании типа данных. Поскольку информация о типе объекта возвращается во время выполнения программы, программист освобождается от целого этапа работы, связанного с типизацией объектов.
* MFC может использоваться совместно с подпрограммами, написанными на языке С. Важной особенностью библиотеки MFC является то, что она может "сосуществовать" с приложениями, основанными на WindowsAPI. В одной и той же программе программист может использовать классы MFC и вызывать функции WindowsAPI. Такая прозрачность среды достигается за счет согласованности программных обозначений в обеих архитектурах. Другими словами, файлы заголовков, типы и глобальные константы MFC не конфликтуют с именами из WindowsAPI. Еще одним ключевым моментом, обеспечивающим такое взаимодействие, является согласованность механизмов управления памятью.
* MFC может быть использована для создания программ, работающих в среде MSDOS. Библиотека MFC была создана специально для разработки приложений в среде Windows. В то же время многие классы предоставляют объекты, часто используемые для ввода/вывода файлов и манипулирования строковыми данными. Такие классы общего назначения могут применяться в приложениях как Windows, так и MS-DOS.

Макроподстановка функций Компилятор MicrosoftVisualC++ поддерживает возможность макроподстановки функций. Это означает, что вызов любой функции с любым набором инструкций может быть заменен непосредственной подстановкой тела функции. Многие компиляторы C++ разрешают производить макроподстановку только для функций, содержащих определенные операторы и выражения. Например, иногда оказывается невозможной макроподстановка функций, содержащих операторы switch, while и for. VisualC++ не накладывает ограничений на содержимое функций. Чтобы задать параметры макроподстановки, выберите в меню Project команду Settings, затем активизируйте вкладку C/C++ и, наконец, выберите элемент Optimizations из списка Category.

Компилятор MicrosoftVisualC++ предоставляет огромные возможности в плане оптимизации приложений, в результате чего вы можете получить выигрыш как в отношении размера программы, так и в отношении скорости ее выполнения, независимо от того, что представляет собой ваше приложение. Перечисленные ниже опции компиляции позволяют оптимизировать программный код, сокращая его размер, время выполнения и время компиляции. Чтобы получить к этим опциям доступ, нужно в меню Project выбрать команду Settings.

На вкладке General можно включить или отключить возможность использования библиотеки MFC (список MicrosoftFoundationClasses). Здесь также можно указать папки, в которые компилятор будет помещать промежуточные (поле Intermediatefiles) и выходные (поле Outputfiles) файлы.

Debug а вкладке Debug можно указать местонахождение исполняемого файла и рабочей папки, задать аргументы командной строки, а также путь и имя удаленного исполняемого файла на сетевом диске. Кроме того, в списке Category можно выбрать элемент AdditionalDLLs, предназначенный для задания дополнительных библиотек динамической компоновки (DLL).

Вкладка C/C++ содержитследующиекатегорииопций: General, C++ Language, Code Generation, Customize, Listing Files, Optimizations, Precompiled Headers и Preprocessor. В поле ProjectOptions отображается командная строка проекта.

Опции категории General позволяют установить уровень контроля за ошибками (список Warninglevel), указать, какую отладочную информацию следует включать (список Debuginfo), выбрать тип оптимизации при компиляции (список Optimizations) и задать директивы препроцессора (поле Preprocessordefinitions).

Опции категории C++ Languageпозволяют выбрать способ представления указателей на члены классов (группа Pointer-to-member representation), включить обработку исключительных ситуаций (Enable exception handling), разрешить проверку типов объектов на этапе выполнения (EnableRun-timeTypeInformation) и запретить замещение конструкторов при вызове виртуальных функций (Disableconstructiondisplacements).

Опции категории CodeGeneration позволяют задать тип процессора, на который должен ориентироваться компилятор (список Processor), выбрать тип соглашения о вызовах функций (список Callingconvention), указать тип компоновки динамических библиотек (список Useruntimelibrary) и установить порядок выравнивания полей структурированных переменных (список Structmemberalignment).

В категории Customize можно задать следующие опции:

* Disablelanguageextensions (компиляция производится в соответствии с правилами ANSIС, а не MicrosoftС);
* Enablefunction-levellinking (функции при компиляции обрабатываются особым образом, что позволяет компоновщику упорядочивать их и исключать неиспользуемые);
* Eliminateduplicatestrings(в таблицу строк модуля не включаются повторяющиеся строки);
* Enableminimalrebuild(позволяет компилятору обнаруживать изменения в объявлениях классов C++ и выявлять необходимость повторной компиляции исходных файлов);
* Enableincrementalcompilation(дает возможность компилировать только те функции, код которых изменился с момента последней компиляции);
* Suppressstartupbannerandinformationmessages(в процессе компиляции запрещается вывод сообщения с информацией об авторских правах и номере версии компилятора).

ListingFiles Опции категории ListingFiles позволяют задавать сведения, необходимые для создания SBRфайла (группа Generatebrowseinfo), который используется при построении специальной базы данных с информацией о всех классах, функциях, переменных и константах программы. Кроме того, в этой категории можно указать, следует ли создавать файл с ассемблерным кодом программы, какого он должен быть типа и где располагаться (список Listingfiletype и поле Listingfilename).

Опции категории Optimizationsпозволяют устанавливать различные параметры оптимизации программного кода (список Optimizations). Также можно указать, каким образом следует выполнять макроподстановку функций (список Inlinefunctionexpansion).

Опции категории PrecompiledHeaders определяют, следует ли использовать файлы предварительно скомпилированных заголовков (файлы с расширением РСН). Наличие таких файлов ускоряет процесс компиляции и компоновки. После компиляции всего приложения эти файлы следует удалить из папки проекта, поскольку они занимают очень много места.

Опции категории Preprocessor позволяют задавать параметры работы препроцессора. Здесь же можно указать дополнительные папки для включаемых файлов заголовков (поле Additional #include directories), а также установить опцию Ignore standard include paths, которая служит указанием игнорировать папки, перечисленные в переменных среды PATHили INCLUDE.

Вкладка Link содержит опции пяти категорий: General, Customize, Debug, Inputи Output.

В категории General в поле Outputfilename можно задать имя и расширение выходного файла. Как правило, для файла проекта используется расширение ЕХЕ. В поле Object/librarymodules указываются объектные и библиотечные файлы, компонуемые вместе с проектом. Также могут быть установлены следующие опции:

* Generatedebuginfo(в исполняемый файл включается отладочная информация);
* Linkincrementally(частичная компоновка; эта опция доступна, если в категории Customize установлен флажок Useprogramdatabase);
* Enableprofiling(в исполняемый файл включается информация для профилировщика);
* Ignorealldefaultlibraries(удаляются все стандартные библиотеки из списка библиотек, который просматривается компоновщиком при разрешении внешних ссылок);
* Generatemapfile(создается МАР-файл проекта).
* Customize

В категории Customize можно установить такие опции:

* Linkincrementally(аналогична одноименной опции из категории General);
* Useprogramdatabase(в служебную базу данных программы помещается отладочная информация);
* Outputfilename(задает имя выходного файла);
* Printingprogressmessages(в процессе компиляций выводятся сообщения о ходе компоновки);
* Suppressstartupbanner(аналогична подобной опции категории Customize вкладки C/C++).

Категории Debug позволяют указать, следует ли генерировать МАР-файл проекта, а также задают различные параметры отладки.

Посредством опций категории Input приводится различная информация об объектных и библиотечных файлах, компонуемых вместе с проектом.

Опции категории Output позволяют задать базовый адрес программы (Baseaddress), точку входа (Entry-pointsymbol), объем виртуальной и физической памяти, выделяемой для стека (группа Stackallocations), и номер версии проекта (группа Versioninformation).

Вкладка Resources позволяет указать имя файла ресурсов (обычно это файл с расширением RES) и задать некоторые дополнительные параметры, такие как язык представления ресурсов, папки включаемых файлов и макросы препроцессора.

Вкладка MIDL предназначена для задания различных параметров генерации библиотеки типов. BrowseInfo На вкладке BrowseInfo можно указать имя файла базы данных, содержащей описания классов, функций, констант и переменных программы.

Вкладка CustomBuild предназначена для задания дополнительных команд компиляции, которые будут выполняться над выходным файлом.

**ВЫВОДЫ**

В бакалаврской работе было рассмотрено методы создание информационно-справочной подсистем САПР конструкторско-технологического назначения. Были рассмотрены всевозможные комбинации технических средств для выполнения и исполнения данной работы. Для данной системы была выбрана БД реляционной модели.

Реляционная модель имеет в своей основе понятие «отношения», и ее данные формируются в виде таблиц. Отношение – это двумерная таблица, имеющая свое название, в которой минимальным объектом действий, сохраняющим ее структуру, является строка таблицы (кортеж), состоящая из ячеек таблицы – полей.

Каждый столбец таблицы соответствует только одной компоненте этого отношения. С логической точки зрения реляционная база данных представляется множеством двумерных таблиц различного предметного наполнения.

Для реализации данной БД можно было выбрать любой язык программирования, но был выбран VisualC++ стандартный вариант. Стандартный вариант VisualC++ (ранее он назывался учебным) содержит почти все те же средства, что и профессиональный, но в отличие от последнего в нем отсутствуют модуль Profiler, несколько мастеров, возможности по оптимизации кода и статической компоновке библиотеки MFC , некоторые менее важные функции.

**ПЕРЕЧЕНЬ ССЫЛОК**

1. "Микроэлектроника: Учеб. пособие для втузов. В 9 кн. /под ред. Л.А. Коледова. Кн. 4. Гибридные интегральные микросхемы/ Л.А. Коледов, Э.М. Ильина. – М.: Высш. шк., 1987.

2. Малышев И.А. "Технология производства интегральных микросхем". – М.: Радио и связь, 1991.

3. Курносов А.И. "Технология производства полупроводниковых приборов и интегральных микросхем". – М., 1979.

4. С.А. Бирюков. Цифровые устройства на МОП-интегральных микросхемах. "Радио и связь", 1996 г., 192 с.

5.Н.А. Елагин, А.В. Ростов. Конструкции и технологии в помощь любителям электроники. "СОЛОН-Р", М., 2001 г., 106 с.

6. В.Л. Шило. Популярные цифровые микросхемы. "Радио и связь", 1989 г., 352 с.

7. http://www.chertegey.net.ru

8. http://www.cosmos.rcnet.ru

9. http://www.lanit.ru

10. http://www.sapr.km.ru

1. Веретенникова Е.И., Патрушина С.М., Савельева Н.Г. Информатика: Учебное пособие. Серия «Учебный курс». – Ростов н/Д: Издательский центр «МарТ», 2002.
2. Могилев А.В. Информатика: Учеб. пособие для студ. пед. вузов / А.В. Моглиев, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера.- 2-е изд., стер. – М.: Издательский центр «Академия», 2003.