ПРЕДИСЛОВИЕ

Цель данного учебного пособия — дать студентам необходимые знания по принципам построения систем автоматизированного проектирования технологических процессов, ознакомить их с методами и алгоритмами решения задач технологического назначения.

Пособие содержит разделы, посвященные алгоритмизации описания объектов проектирования в САПР и элементов технологических процессов, созданию автоматизированных справочников. Основное внимание уделяется алгоритмизации решения типовых задач автоматизированного проектирования технологических процессов.

Пособие может быть использовано для выполнения как лабораторных и практических работ, так и для самостоятельных занятий в процессе изучения разделов дисциплин.

ВВЕДЕНИЕ

Любое изделие, объект проектирования, проходит последовательно ряд этапов жизненного цикла [1, 2]. И на каждом этапе используется заданная совокупность взаимосвязанных автоматизированных систем конструкторского, технологического и производственного назначений. Такая технология носит название CALS-технология — технология информационной поддержки производства изделия на всех этапах его жизненного цикла.

CALS-технология предполагает, что задачи автоматизации технологического проектирования не могут качественно решаться без тесной связи с задачами автоматизированного конструкторского проектирования изделий.

Важность такого подхода подчеркивается тем, что современные разработчики САПР создают интегрированные, комплексные системы автоматизации конструкторских и технологических задач в едином информационном пространстве. Применение таких систем позволит повысить качество изделий за счет более полного учета имеющейся накопленной информации об объекте проектирования.

Одним из компонентов программной среды интегрированных систем является программный комплекс системы автоматизированного проектирования технологических процессов (САРР-системы). К таким системам относятся, например, Компас-Вертикаль, СПРУТ ТП, TechCard, T-FLex Технология. Перечисленные программные продукты позволяют разрабатывать полый комплект технологической документации в соответствии с ЕСТД и сократить время технологической подготовки производства и трудоемкость проектирования технологических процессов.

В этой работе мы попытаемся разобраться в некоторых аспектах автоматизации решения задач технологического проектирования.

1. ОПИСАНИЕ СТРУКТУРЫ ОБЪЕКТА ПРОЕКТИРОВАНИЯ В САПР

1.1. Электронная структура изделий

В САПР структура объектов проектирования описывается на основе принципов иерархичности и декомпозиции [3]. Принцип иерархичности означает структурированное представление объекта проектирования в виде взаимосвязанных иерархических уровней, а принцип декомпозиции (блочности) — разбиение объекта каждого уровня на ряд составных частей (блоков). На основе этих принципов создается электронная структура изделия (ЭСИ).

Существующий в настоящее время ГОСТ 2.053-2013 устанавливает общие требования к электронной структуре изделий всех отраслей промышленности. В соответствии с этим электронная структура изделия — это совокупность составных частей изделия и связей между ними, определяющих иерархию составных частей, предназначена для организации информационного взаимодействия между автоматизированными системами. Она представляется в виде ориентированного ациклического графа, вершины которого соответствуют компонентам, а ребра, соединяющие вершины, — отношениям (связям) между компонентами.

ЭСИ используют:

- для представления вариантов состава и структуры изделия;
- для структурирования проектной и рабочей конструкторской документации на изделие;
- для представления информации о применяемости, правилах использования составных частей при различных условиях (в том числе исполнениях) и заменяемости (в том числе взаимозаменяемости) составных частей;
- для представления технических данных об изделии на стадиях жизненного цикла изделия (ЖЦИ).

В соответствии с ГОСТ различают следующие основные разновидности ЭСИ: функциональную, конструктивную, производственно-технологическую, физическую, эксплуатационную и совмещенную.

Функциональная ЭСИ предназначена для определения назначения изделия и его составных частей и предъявляемых к ним функциональных требований. Как правило, функциональная ЭСИ выполняется на стадии разработки технического предложения на изделие и уточняется на стадии технического проекта.

Конструктивная ЭСИ предназначена для отображения конкретных технических решений, определяющих конструкцию комплексов, сборочных единиц и комплектов. Как правило, конструктивная ЭСИ выполняется на стадиях разработки эскизного проекта, технического проекта и рабочей конструкторской документации (КД). Конструктивная ЭСИ — основной конструкторский документ.

Производственно-технологическая ЭСИ предназначена для отображения особенностей технологии изготовления и (преимущественно) сборки изделия. Производственно-технологическую ЭСИ выполняют на стадиях технологической подготовки производства и в процессе производства изделия.

Физическая ЭСИ предназначена для отображения информации о конкретном экземпляре изделия. Физическая ЭСИ выполняется на стадии производства изделия и, как правило, корректируется в течение всего срока эксплуатации (например, отражаются изменения в комплектности данного экземпляра изделия).

Эксплуатационная ЭСИ предназначена для группирования и отображения информации о тех СЧ изделия, которые подлежат обслуживанию и/или замене в ходе использования изделия по назначению. Эксплуатационная ЭСИ выполняется на стадиях разработки эскизного проекта, технического проекта и рабочей КД.

Совмещенная ЭСИ предназначена для группирования и отображения комплексной информации об изделии и включает в себя отдельные разновидности ЭСИ (например, конструктивную ЭСИ и эксплуатационную ЭСИ)*.

В САПР изделий и технологических процессов электронная структура изделий реализуется при разработке и эксплуатации систем управления инженерными данными. Примером может служить «ЛОЦМАН» — система управления инженерными данными и жизненным циклом изделия, разработанная компанией АСКОН. Основные функциональные возможности таких систем охватывают следующие направления:

- управление хранением данных и документами;
- управление процессами и потоками работ;
- управление структурой изделий;
- автоматизация генерации выборок и отчетов;
- механизм авторизации.

1.2. Описание структуры объекта

1.2.1. Описание предметной области

В соответствии с ГОСТ 2.108-68 на каждую сборочную единицу проектируемого изделия составляется конструкторская спецификация — конструкторский документ, который описывает состав сборочной единицы и совокупность документов, которые необходимы для их изготовления.

Спецификация как текстовый документ состоит из ряда разделов: документация; комплексы; сборочные единицы; детали; стандартные изделия; прочие изделия; материалы; комплекты.

В раздел «Документация» вносится информация о документах, разработанных в проекте на специфицируемый узел. В разделы «Комплексы», «Сборочные единицы», «Детали» и в другие разделы вносится информация об элементах, непосредственно входящих в данный узел.

К конструкторским документам в соответствии с ГОСТ 2.102-2013 относятся: электронные модели деталей и сборочных единиц, чертежи деталей, сбо-

рочные чертежи, чертежи общего вида, монтажные чертежи и т. д. К текстовым электронным документам относятся: пояснительная записка, технические условия, расчеты, инструкции и т. д.

В ГОСТ 2.053-2013 даны понятия и определение информационному объекту — это именованная совокупность данных, обладающая набором атрибутов и предполагающая определенные методы обработки.

Учитывая это, к информационным объектам можно отнести все элементы, описываемые в конструкторской спецификации. Их можно квалифицировать на простые и составные. К составным элементам относятся: изделия и сборочные единицы. Простыми элементами являются детали, стандартные изделия, материалы, комплекты, комплексы.

В процессе проектирования могут быть использованы унифицированные узлы, типовые или заимствующие детали, то есть те элементы, которые могут входить в состав нескольких составных элементов.

Учитывая это, модель данных предметной области, описывающей изделие, может быть представлена в виде взаимосвязанных объектов. Рассмотрим их.

Элемент — это информационный объект, описываемый в конструкторской спецификации: деталь, сборочная единица, стандартное изделие, материал, комплект, комплекс.

Связь — это информационный объект, описывающий входимость одного элемента в состав другого элемента по ключевой фразе «что входит — куда входит». Например, «станок — шпиндельный узел»: шпиндельный узел входит в состав станка.

Тип элемента — это информационный объект, определяющий тип элемента: изделие, деталь, сборочная единица, стандартное изделие и т. д.

Документ — это информационный объект, описывающий конструкторский или технологический документ, разработанный для данного элемента: файлы чертежей, 3D-моделей, технических расчетов, карты технологических документов и т. д.

В соответствии со структурой и принципами нормализации структура базы данных может быть представлена в виде взаимосвязанных таблиц, а ее программная реализация — в виде дерева (рис. 1.1–1.3). Структура каждой таблицы базы данных подробно описывается в последующих разделах.

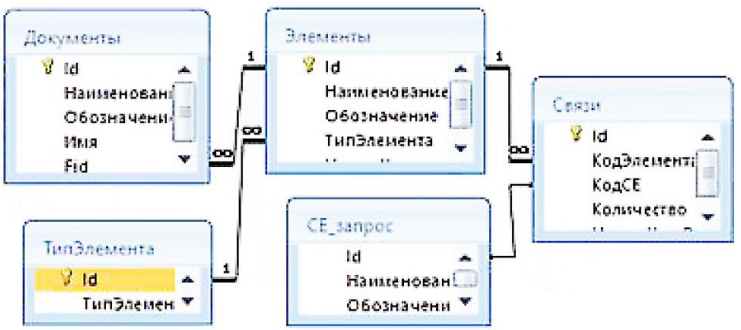

Рис. 1.1

Структура базы данных

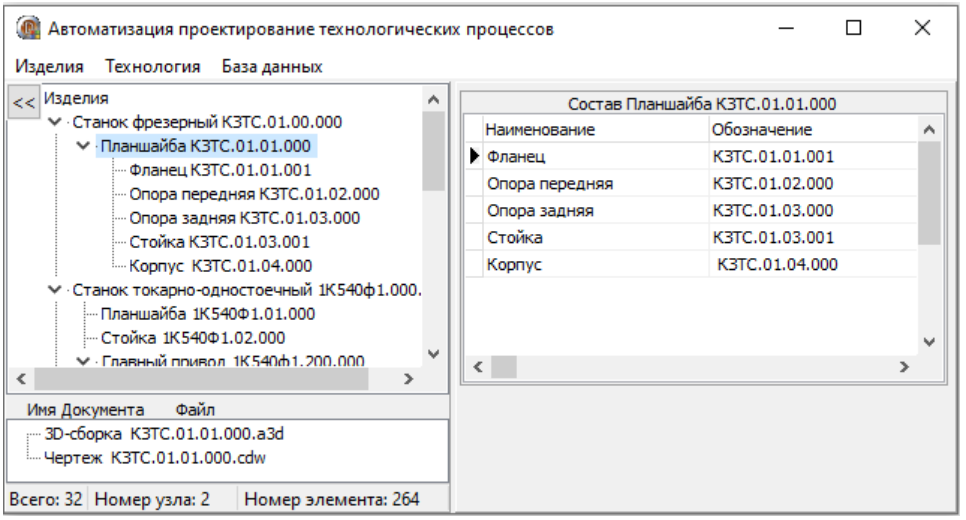

Рис. 1.2

Структура и состав сборочной единицы Планшайба и ее файлы

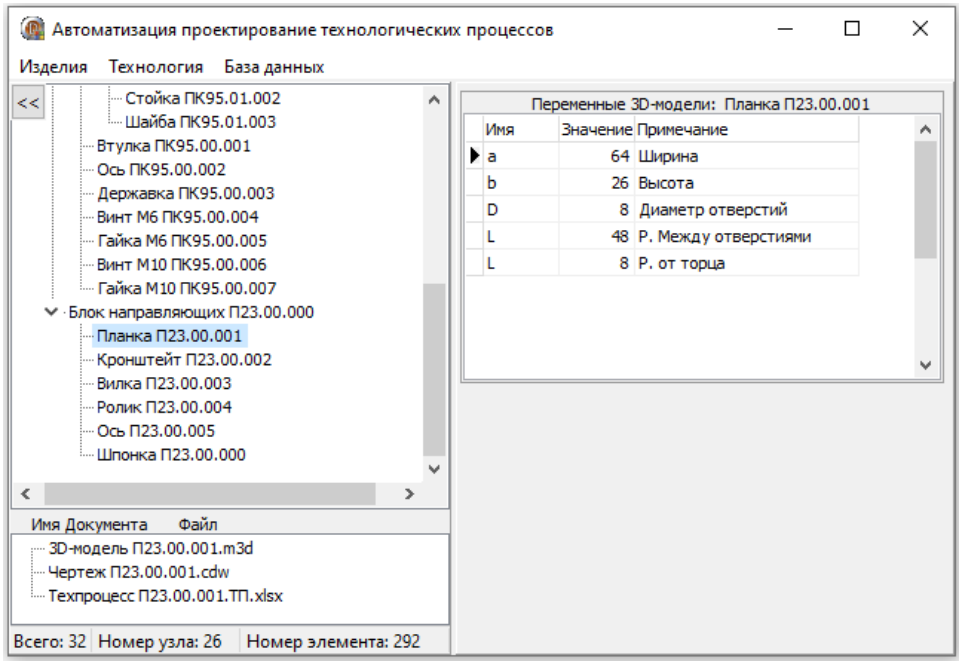

Рис. 1.3

Переменные 3D-модели планки П23.00.001 и ее файлы

1.2.2. Процедуры доступа к данным

Рассмотрим технологию связывания базы данных с приложением [6].

Доступ к данным может быть осуществлен через механизм *ADO (ActiveX Data Objects)*, являющийся стандартом фирмы Microsoft.

Компонентом соединения с базой данных является *TADOConnection*. Наборами данных, через которые осуществляется доступ к таблице базы данных, являются компоненты *TADOTable* (таблицы) и *TADOQuery* (запросы).

TADOTable обеспечивает прямой доступ к каждой записи и каждому полю таблицы через свойство *TableName*. Компонент *TADOQuery* позволяет осуществить доступ к отдельным записям таблицы или к записям нескольких связанных таблиц.

Главным свойством компонента *TADOQuery* является свойство SQL. В нем разработчик создает необходимую инструкцию-запрос, написанную с помощью структурированного языка запросов. С ее помощью из таблицы или таблиц выбираются те записи, которые необходимы для решения поставленной задачи, или выполняется их модификация: изменение, удаление и добавление.

Выбранные из базы данных записи могут быть модифицированы или удалены из набора *TADOQuery* и TADOTable. Для этого используются навигационные методы, такие как редактирование (*Edit*), удаление (*Delete*) и вставка новой записи (*Insert*).

Данные, полученные компонентами *TADOTable* и *TADOQuery* из базы данных, отражаются на форме в визуальных компонентах, которые связаны с наборами данных через специальный невизуальный компонент *TDataSourse*. Он называется источником записи.

Связь между компонентами можно схематично отобразить следующим образом:

- *TADOConnection* ⇨ *TADOTable* ⇨ *TDataSourse* ⇨ «визуальный компонент» (например, *TDBGrid*, *TEdit* и т. д.);
- *TADOConnection* ⇨ *TADOQuery* ⇨ *TDataSourse* ⇨ «визуальный компонент» (например, *TDBGrid*, *TEdit* и т. д.).

Перечислим некоторые важные свойства *TADOTable* и *TADOQuery*:

- методы *First*, *Next*, *Prior*, *Last* — установка указателя на первую запись, на следующую запись, на предыдущую запись, на последнюю запись, соответственно;
- *свойство RecNo* — переход на запись;
- *свойство RecordCount* — количество записей в наборе;
- *метод Locate* — поиск записи по критерию;
- *свойство Filter* — фильтрации записей;
- *метод Insert* разрешает добавить новую запись в набор данных;
- *метод Edit* разрешает редактировать запись;
- *метод Delete* удаляет текущую запись из набора данных;
- *метод Open* открывает набор данных;
- *метод Cancel* отменяет ввод новой записи или изменения, внесенные в текущую запись набора данных.

Свойство *SQL* набора данных *TADOQuery* может быть сформировано двумя способами: в программном коде какого-то события в виде текстовой строки или в инспекторе в свойстве *SQL* компонента *TADOQuery* .

Рассмотрим формирование фрагмента кода для различных задач. В качестве примера возьмем виртуальную таблицу «Деталь».

1. Выбор записи по критерию:

```
str:='SELECT Деталь.* FROM Деталь Where ((id=:par));';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.id;//критерий выбора
ADOQuery1.Open;
```

2. Добавление записи в таблицу:

```
str:='INSERT INTO Деталь (Наименование, Обозначение, Тип) Values (:par1,:par2,:par3);';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.Name; //наименование элемента
ADOQuery1.Parameters[1].Value:=Tr.Oboz;//обозначение элемента
ADOQuery1.Parameters[2].Value:=Tr.Tip;//тип элемента
ADOQuery1.ExecSQL;
```

3. Редактирование поля записи:

```
str:='UPDATE Деталь SET Наименование =:par1 Where (id=:par2);';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.F;//новое наименование
ADOQuery1.Parameters[1].Value:=Tr.id;//критерий выбора записи
ADOQuery3.ExecSQL;
```

4. Удаление записи:

```
ADOQuery3.Close;
ADOQuery3.SQL.Clear;
str:= 'Delete FROM Деталь Where (id=:par);'
ADOQuery3.SQL. Add(str);
ADOQuery3.Parameters[0].Value:=Tr.id;
ADOQuery3.ExecSQL;
```

1.2.3. Процедуры описания изделий

Для описания структуры изделий необходимо создать ряд программных процедур. Перечислим их:

- добавить элемент первого уровня: изделие, узел;
- добавить дочерний элемент: элемент спецификации;
- редактировать элемент;
- удалить элемент;

- ввести элемент в состав другого элемента;
- подключить ранее созданный файл к элементу;
- показать состав объекта;
- показать атрибуты объекта;
- показать параметры модели.

Элементы изделий и сборочных единиц в соответствии со структурой БД хранятся в двух связанных таблицах (рис. 1.1 и 1.4): Элементы и Связи.

Алгоритм заполнения дерева и таблиц БД элементами может быть представлен в следующем виде:

- выбор записи из таблицы Элементы по выделенному узлу дерева, в состав которого добавляется новый элемент;
- ввод нового элемента спецификации в таблицу Элементы;
- добавление записи в таблицу Связи, т. е. ввод элемента в состав выделенного элемента.

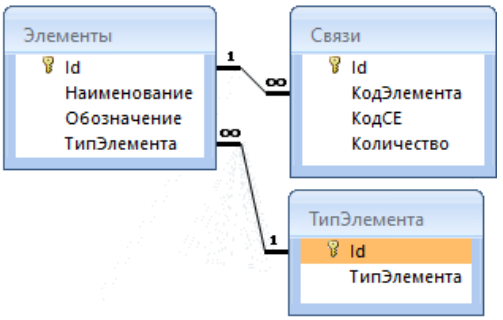

Рис. 1.4

Фрагмент структуры базы данных

Процедура добавления узла в дерево элементов первого уровня:

TreeView1.Items.Add(nil,Tr.Ftxt).

Процедура добавления узла в дерево подчиненных элементов:

TreeView1.Items.AddChild(TreeView1.Selected,Tr.Ftxt).

Процедура добавления элементов в таблицу Элементы и таблицу Состав состоит из четырех частей:

- поиск составного элемента в таблице Элементы по выделенному узлу дерева:

```
str:='SELECT Id FROM Элементы
Where (((Наименование+'+'" "'+'+Обозначение)=:par));';
Tr.Ftxt:=TreeView1.Selected.Text;
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:= Tr.Ftxt;
```

```
ADOQuery1.Open;
Tr.FFid:=ADOQuery1.Fields[0].AsInteger; //Ключ составного элемента.
```

- добавление подчиненных элементов в таблицу Элементы и дерево:

```
str:='INSERT INTO Элементы (Наименование, Обозначение, ТипЭлемента,НомерУзла) Values (:par1,:par2,:par3);';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.Fnaimen; //наименование
ADOQuery1.Parameters[1].Value:=Tr.Foboz;//обозначение
ADOQuery1.Parameters[2].Value:=Tr.FTipEl;//тип элемента
ADOQuery1.ExecSQL;
```

- поиск подчиненного элемента в таблице Элементы:

```
str:='SELECT Id FROM Элементы
Where (((Наименование+'+'" "'+'+Обозначение)=:par));';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
txt:=TreeView1.Selected.Text;
ADOQuery1.Parameters[0].Value:=Tr.Ftxt;
ADOQuery1.Open;
Tr.Fid:=ADOQuery1.Fields[0].AsInteger; //Ключ потомка
```

- ввод элемента в состав составного элемента:

```
str:='INSERT INTO Состав (КодЭлемента, КодСЕ, Количество)
Values (:par1,:par2,:par3);';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.Fid;//ключ потомка
ADOQuery1.Parameters[1].Value:=Tr.FFid;//ключ родителя
ADOQuery1.Parameters[2].Value:=Tr.FCol;//количество потомков
ADOQuery1.ExecSQL;
```

Запрос на редактирования выбранных данных в БД и в дереве состоит из трех частей:

- выбрать запись в таблице Элементы по выделенному узлу дерева;
- редактировать выбранную запись (обновление данных) в таблице Элементы;
- редактировать выбранную запись в таблице Связи.

Ниже представлен код редактирования выбранной записи в таблицах Элементы и Связи:

```
Tr.Fnaimen:=edit1.Text;// поле наименование
Tr.Foboz:= edit2.Text;// поле обозначение
Tr.FTipEl:=DBLookupComboBox1.KeyValue;// поле со списком из таблицы Типы элементов
Tr.FCol:= strToInt(edit3.Text);// поле количество
ADOQuery1.Edit;
ADOQuery1.Fields[4].AsString:=Tr.Fnaimen;
ADOQuery1.Fields[5].AsString:=Tr.Foboz;
ADOQuery1.Fields[2].AsInteger:=Tr.FTipEl;
ADOQuery1.Post;
Tr.Ftxt:=Tr.Fnaimen+' '+Tr.Foboz;
TreeView1.Selected.Text:= Tr.Ftxt;
//Редактировать таблицу связи
 str:='UPDATE Связи SET Количество =:par1 Where (КодЭлемента=:par2) and (КодСЕ=:par3);';
ADOQuery2.Close;
ADOQuery2.SQL.Clear;
ADOQuery2.SQL.Add(str);
ADOQuery2.Parameters[0].Value:=Tr.FCol;//новое количество
ADOQuery2.Parameters[1].Value:=Tr.FidELem;//id элемента
ADOQuery2.Parameters[2].Value:=Tr.FidRod;//id род узла
ADOQuery2.ExecSQL;
```

Алгоритм удаления элемента из дерева и из БД:

- выбор в дереве удаляемого элемента;
- запуск запроса на удаление выделенного элемента, если он не имеет дочерних элементов.

После удаления элемента из таблицы Элементы все связанные записи таблицы Связи удаляются автоматически.

Программный код удаления записи:

```
if TreeView1.Selected.Count=0 then
begin
    ADOQuery3.Close;
    ADOQuery3.SQL.Clear;
    ADOQuery3.SQL. Add('Delete FROM Элементы Where (((Наименование+'+'" "'+'+Обозначение)=:par));');
    ADOQuery3.Parameters[0].Value:=Tr.Ftxt;
    ADOQuery3.ExecSQL;
    TreeView1.Items.Delete(TreeView1.Selected);
  end
  Else ShowMessage ('Есть подчиненные элементы!');
```

Код выбора элементов, которые входят в состав выделенной в дереве сборочной единицы (рис. 1.5):

str:='SELECT Элементы.Наименование, Элементы.Обозначение '+
'FROM Элементы INNER JOIN Связи ON Элементы.Id'+
'=Связи.КодЭлемента WHERE (((Связи.КодСЕ)=:par));';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.id;//критерий выбора.
ADOQuery1.Open;

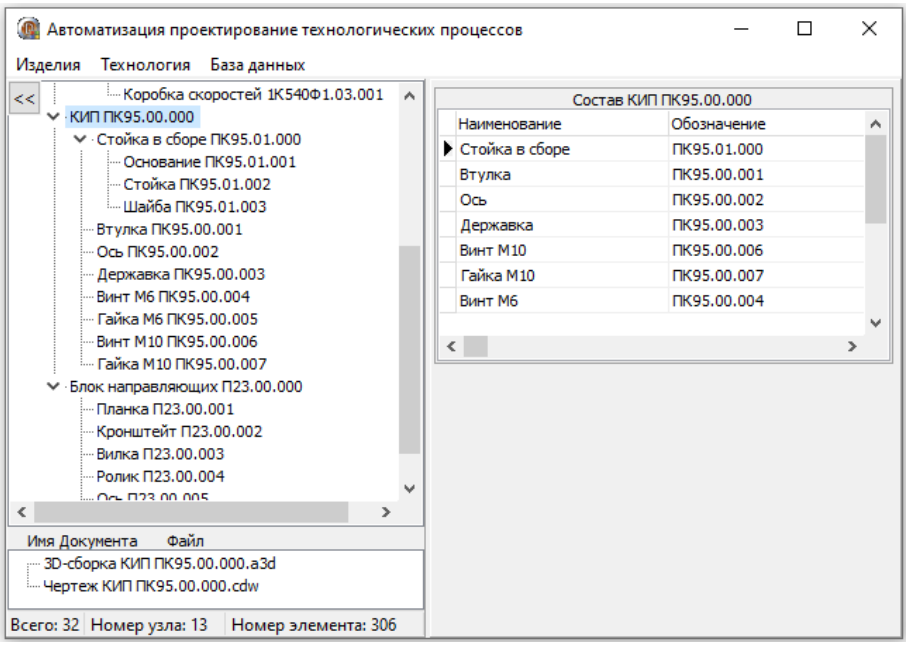

Рис. 1.5

Состав узла

Выбор элемента по сочетанию полей:

str:='SELECT Элементы.Наименование, Элементы.Обозначение '
+' FROM Элементы AS Элементы_1 INNER JOIN (Элементы INNER
JOIN Связи ON Элементы.Id = Связи.КодЭлемента) ON Элементы_1.Id
= Связи.КодСЕ Where(((Элементы_1.Наименование+'+'""
""'+'+Элементы_1.Обозначение)=:par));';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.Ftxt;//критерий выбора.
ADOQuery1.Open;

Показать в дереве документы-файлы, разработанные для данного элемента (рис. 1.6):

```
str:='SELECT Документы.* FROM Документы WHERE (((Документы.Fid)=:par));';
ADOQuery4.Close;
ADOQuery4.SQL.Clear;
ADOQuery4.SQL.Add(str);
ADOQuery4.Parameters[0].Value:=Tr.FidELem;
ADOQuery4.Open;
Tr.FidDoc:= ADOQuery4.Fields[0].AsInteger;
//Отразить в дереве файлы
TreeView2.Items.Clear;
while not Form2.ADOQuery4.Eof do begin
  Str:= ADOQuery4.Fields[1].AsString +' '+ADOQuery4.Fields[3].AsString;
  TreeView2.Items.Add(nil,str);
  ADOQuery4.Next
end;
```

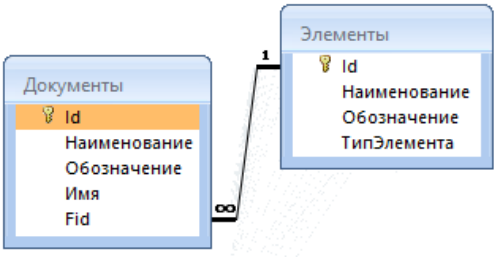

Рис. 1.6

Фрагмент базы данных

Выбрать из списка документов заданный документ по выделенному узлу дерева документов:

```
str:= 'SELECT Документы.Id FROM Документы WHERE (((Наименование+" "+Имя)) =:par);';
ADOQuery4.Close;
ADOQuery4.SQL.Clear;
ADOQuery4.SQL.Add(str);
ADOQuery4.Parameters[0].Value:= TreeView2.Selected.Text;
ADOQuery4.Open;
Tr.FidDoc:= ADOQuery4.Fields[0].AsInteger;
```

Показать переменные 3D-модели выбранного элемента и показать в сетке *DBGrid* (рис. 1.7):

```
str:='SELECT Номер, Имя, Значение, Примечание FROM Параметры
WHERE (((Fid)=:par));';
ADOQuery1.Close;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(str);
ADOQuery1.Parameters[0].Value:=Tr.FidDoc;//код документа.
```