В.Г. МОКРОЗУБ

ГРАФОВЫЕ СТРУКТУРЫ И РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ В АВТОМАТИЗИРОВАННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Москва
Издательский дом "Спектр"
2011

Научное издание

МОКРОЗУБ Владимир Григорьевич

ГРАФОВЫЕ СТРУКТУРЫ И РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ В АВТОМАТИЗИРОВАННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Редактор Л.В. К о м б а р о в а
Инженер по компьютерному макетированию И.В. Е в с е е в а

Сдано в набор 22.11.2011 г. Подписано в печать 22.11.2011 г.
Формат 60 × 84/16. Бумага офсетная. Гарнитура Times New Roman
Усл. печ. л. 6,27. Уч.-изд. л. 6,75
Тираж 400 экз. Заказ № 515

ООО "Издательский дом "Спектр", 119048, Москва, ул. Усачева, д. 35, стр. 1
Http://www.idspektr.ru. E-mail: idspektr@rambler.ru

Подготовлено к печати и отпечатано в Издательско-полиграфическом центре
ФГБОУ ВПО "ТГТУ"
392000, г. Тамбов, ул. Советская, д. 106, к. 14

По вопросам приобретения книги обращаться по телефону 8(4752)638108
E-mail: izdatelstvo@admin.tstu.ru

В.Г. МОКРОЗУБ

ГРАФОВЫЕ СТРУКТУРЫ И РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ В АВТОМАТИЗИРОВАННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Москва, 2011

УДК 681.142(075.8)
ББК 32.973я73
М168

Р е ц е н з е н т ы :

Доктор технических наук, профессор, заведующий кафедрой компьютерно-интегрированных систем в химической технологии Российского химико-технологического университета им. Д.И. Менделеева
В.Ф. Егоров

Доктор технических наук, профессор, проректор по информатизации ФГБОУ ВПО "ТГТУ"
В.Е. Подольский

Мокрозуб В.Г.

М168 Графовые структуры и реляционные базы данных в автоматизированных интеллектуальных информационных системах. – М.: Издательский дом "Спектр", 2011. – 108 с. – 400 экз. – ISBN 978-5-904270-89-6.

Отражены вопросы создания математического и программного обеспечения автоматизированных интеллектуальных информационных систем. В качестве базового программного обеспечения используются реляционные базы данных. Основа математического обеспечения – различные виды графов. Рассмотрены конкретные примеры элементов автоматизированных интеллектуальных информационных систем, предназначенных для использования в различных прикладных областях.

Предназначена для студентов, аспирантов и специалистов, занимающихся созданием автоматизированных информационных систем.

УДК 681.142(075.8)
ББК 32.973я73

ISBN 978-5-904270-89-6

ВВЕДЕНИЕ

Современное общество невозможно представить без развитых информационных технологий, которые используются в настоящее время практически во всех областях человеческой деятельности. Эти технологии объединяют в себе знания и труд специалистов как в области самих информационных технологий, так и в тех областях, в которых они используются (медицина, обучение, техника и др.).

В последнее десятилетие информационные технологии активно развиваются в направлении, связанном с их применением в интеллектуальной деятельности человека. Результат этого развития – экспертные системы и онтологии, которые аккумулируют в себя знания специалистов различных предметных областей (медицина, проектирование технических объектов, управление, диагностика оборудования и др.).

Интеллектуализация информационных систем требует развития:

– подходов к описанию знаний специалистов;

– способов представления и обработки этих знаний в информационных системах;

– способов взаимодействия информационной системы как с человеком, так и с другими информационными системами.

Любая информационная система представляет собой совокупность различных взаимосвязанных видов обеспечения: программное, математическое, техническое, организационное и др., которые оказывают влияние друг на друга и должны разрабатываться совместно.

Настоящая работа посвящена описанию математического и программного обеспечения информационных систем, предназначенных для проектирования технических объектов и управления техническими объектами.

Математическое обеспечение базируется на представлении объектов различной природы гиперграфами. В качестве базового программного обеспечения приняты реляционные базы данных.

Выбор автором предметной области обусловлен спецификой его работы (кафедра "Автоматизорованное проектирование технологичекого оборудования"). Представленные результаты могут быть использованы и в других предметных областях, где требуется описание и представление в информационной системе структуры объектов, свойств объектов, взаимного влияния объектов друг на друга и др.

В первом разделе описываются графы с ограничениями и их представление в реляционной базе данных. Приведены примеры ограничений простых ориентированных графов, неориентированных и ориентированных гиперграфов, неориентированных и ориентированных ультраграфов. Введено понятие N-ориентированного гиперграфа. Для всех перечисленных видов графов представлены структуры реляционных баз данных, в которых ограничения поддерживаются доменной и ссылочной целостностью базы.

Во втором разделе описываются операции над N-ориентированными гиперграфами в реляционной базе данных: добавление новой вершины, добавление нового ребра, удаление вершины и удаление ребра. Для указанных операций представлены запросы на языке SQL, позволяющие выполнять их, не нарушая целостность базы данных.

В третьем разделе описывается представление в реляционной базе данных ограничений при решении задачи размещения объектов в пространстве. Приведенная классификация правил размещений позволяет разрабатывать программное обеспечение информационной системы, предназначенное для проверки правильности размещения объекта, независимо от его природы. Естественно, что информационное обеспечение в этом случае определяется природой объекта размещения: здания в городе, деревья на дачном участке, металлообрабатывающие станки или химические аппараты в производственном помещении.

Четвертый раздел посвящен описанию способов разработки интерфейса пользователя, который не знает язык структурированных запросов (SQL), с реляционной базой данных. Подобный интерфейс необходим, как для ввода правил, например, размещения объекта, так и для составления обращения (запроса) к информационной системе.

В пятом разделе описывается применение N-ориентированных гиперграфов для структурного и параметрического синтеза технических систем. Исходными данными для структурного синтеза являются функции системы, заданные в техническом задании на создании объекта. Приведена классификация правил в зависимости от операции, например, ребро вводится в граф, вершина добавляется в ребро и др.

В шестом разделе описывается применение N-ориентированных гиперграфов для разработки информационно-логических и процедурных моделей планирования выпуска готовой продукции и ремонта оборудования многоассортиментных химических производств.

Седьмой раздел посвящен описанию структуры технических объектов. Описаны реляционные базы данных для представления простой спецификации изделия, групповой спецификации и спецификации изделия с взаимно заменяемыми элементами. Для представления групповой спецификации предложена модель, отличающаяся тем, что для описания принадлежности одного элемента группе других элементов введена специальная логическая функция принадлежности. В изделиях с взаимно заменяемыми элементами имеются позиции спецификации, на которых стоят несколько элементов. Для представления таких спецификаций предложена модель описания влияния элементов друг на друга и алгоритм обработки этой модели, позволяющий по исходным данным получить спецификацию, в которой на каждой позиции стоит только один элемент.

Работа выполнена в рамках федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" на 2009 – 2013 гг.

1. ГРАФЫ С ОГРАНИЧЕНИЯМИ И ИХ ПРЕДСТАВЛНИЕ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Графы, теория и методы, основанные на графах, часто применяются в автоматизированных системах. В работах Овчинникова В.А., Ивановой Г.А., Батищева Д.И., Старостина Н.В. для моделирования и синтеза сложных структур предложены методы, основанные на графах, гиперграфах и ультраграфах.

В [7] представлены графовые модели задач структурного синтеза. В работе [8] описаны представления графов, мультиграфов, неориентированных и ориентированных гипер- и ультраграфов матрицами смежности и инцидентности. В [27] предлагается единый подход к определению понятий ультра-, гипер- и обыкновенных графов. Приведены формальные правила, связывающие матричное и аналитическое представления для каждого вида графов, а также выражения для оценки характеристик компонент графа. Для представления дополнительной информации в [1] на вершинах и гиперребрах гиперграфа задается множество атрибутов. Для формализованного представления свойств множества и его элементов Павловым В.В. разработан аппарат теории полихроматических множеств и графов [29 – 31]. Применение многодольных графов для решения задач структурного синтеза на элементах с ограниченной сочетаемостью представлено в [3]. В [1] подробно представлены методы декомпозиции гиперграфовых структур, в [28] описаны операции над ультра- и гиперграфами, представленными в аналитическом виде.

Указанные публикации описывают операции над гиперграфами, не привязываясь к программным продуктам, в среде которых предполагается реализация автоматизированной системы. Между тем, выбранная программная среда оказывает значительное влияние на последовательность действий при выполнении операций над гиперграфами.

В настоящем разделе в качестве такой программной среды используюся реляционные базы данных. Операции над графовыми структурами описываются как с помощью теории множеств, так и непосредственно операторами языка структурированных запросов, в качестве которого выбран Transact-SQL (релиз SQL фирмы Microsoft).

Представленные структуры баз данных позволяют описывать различные виды графов (простые графы, гиперграфы и ультраграфы) с ограничениями. Причем ограничения поддерживаются доменной и ссылочной целостностью базы данных.

1.1. ПРОСТЫЕ ОРИЕНТИРОВАННЫЕ ГРАФЫ

Обозначим простой граф $G_o(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин графа, x_i – i-я вершина графа; $U = \{u_m(x_i, x_j)\}$, $m = \overline{1, M}$, $i = \overline{1, I}$, $j = \overline{1, I}$ – множество ребер графа, $u_m(x_i, x_j)$ – m-е ребро графа, соединяющее вершины x_i и x_j.

Ребро, соединяющее вершину саму с собой, $u_m(x_k, x_k)$, $m \in \overline{1, M}$, $k \in \overline{1, I}$, называется петлей. Простой граф не будет иметь петель, если выполняется ограничение

$$O_1 : \neg\exists u_m(x_k, x_k) \in U;\ m = \overline{1, M};\ k = \overline{1, I}.$$

Если две вершины могут иметь несколько связывающих их ребер, то такой граф называется мультиграфом. Таким образом, простой граф не будет мультиграфом, если выполняется ограничение

$$O_2 : \forall u_m(x_i, x_j) \in U;\ m = \overline{1, M};\ i \in \overline{1, I};\ j \in \overline{1, I} \Rightarrow$$

$$\neg\exists u_l(x_i, x_j) \in U;\ l \neq m;\ l \in \overline{1, M}.$$

Граф называется ориентированным, если ребро имеет направление, например от x_i (вершина-источник) к x_j (вершина-приемник). Если вершина-приемник связана только одним ребром с только одной вершиной-источником, то такой граф называется направленным деревом.

Введем ограничение O_3 – вершина-приемник связана только с одной вершиной-источником (ограничения на число связей нет)

$$O_3 : \forall u_m(x_i, x_j) \in U;\ m \in \overline{1, M};\ i \in \overline{1, I};\ j \in \overline{1, I} \Rightarrow$$

$$\neg\exists u_n(x_k, x_j) \in U;\ m \neq n;\ i \neq k;\ n \in \overline{1, M};\ k \in \overline{1, I}.$$

С учетом введенных ограничений граф будет направленным деревом, если одновременно выполняются условия O_1, O_2, O_3.

На рисунке 1.1 представлены примеры простых ориентированных графов.

Добавим к классическому обозначению графа $G(X, U)$ перечисленные выше ограничения. С учетом ограничений графы на рис. 1.1 формально можно представить следующим образом:

- $G(X, U, O_1 \wedge O_2 \wedge O_3)$ – рис. 1.1, *а*;
- $G(X, U, O_1 \wedge O_3)$ – рис. 1.1, *б*;
- $G(X, U, O_1 \wedge O_2)$ – рис. 1.1, *в*;
- $G(X, U, O_1)$ – рис. 1.1, *г*;
- $G(X, U, O_1 \wedge O_2)$ – рис. 1.1, *д*.

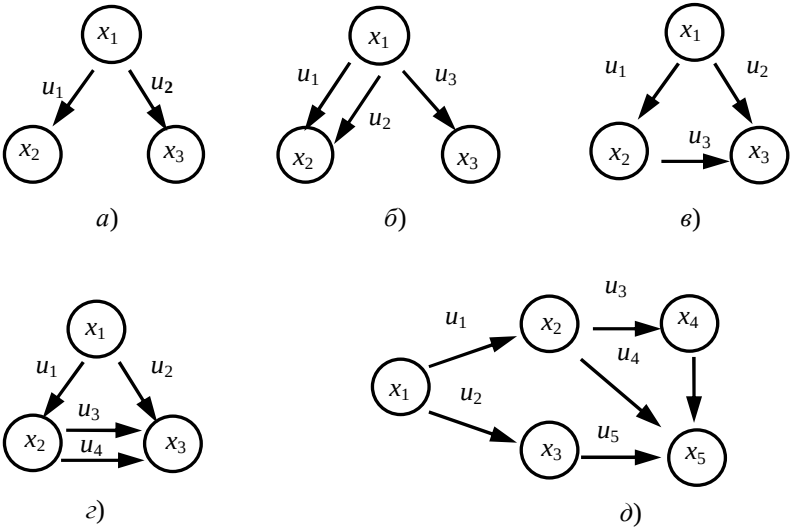


Рис. 1.1. Примеры ориентированных графов

Реляционная база данных (РБД) BDo, описывающая графы на рис. 1.1, представляет собой кортеж:

BDo=<X,U,G>,

где X,U,G – следующие отношения, X – вершины графа, U – ребра графа, G – граф (связи вершин и ребер).

В свою очередь отношения X,U,G можно представить как:

X={PK_ID_X, ID_X, Наименование_вершины}, где PK_ID_X – первичный ключ, ID_X – поле первичного ключа;

U={PK_ID_U, ID_U, Наименование_связи}, где PK_ID_U – первичный ключ, ID_U – поле первичного ключа;

G={PK_ID_G, FK_ID_U, FK_ID_XI, FK_ID_XJ, ID_G, ID_U, ID_XI, ID_XJ}, где PK_ID_G – первичный ключ, FK_IDU, FK_ID_XI, FK_ID_XJ – внешние ключи (по полям ID_U, ID_XI, ID_XJ), ID_G – поле первичного ключа, ID_XI, ID_XJ – поля первичного ключа вершины-источника и вершины-приемника.

Так как ребро в графе всегда уникально, то нет необходимости вводить отношение U, достаточно в отношении G добавить поле Наименование_связи. Тогда база данных, описывающая графы, представленные на рис. 1.1, описывается двумя отношениями

BDo=<X,G>,

где X={PK_ID_X, ID_X, Наименование_вершины}, G={PK_ID_G, FK_ID_XI, FK_ID_XJ, ID_G,ID_XI, ID_XJ, Наименование_связи}. Схема данных этой базы представлена на рис. 1.2.

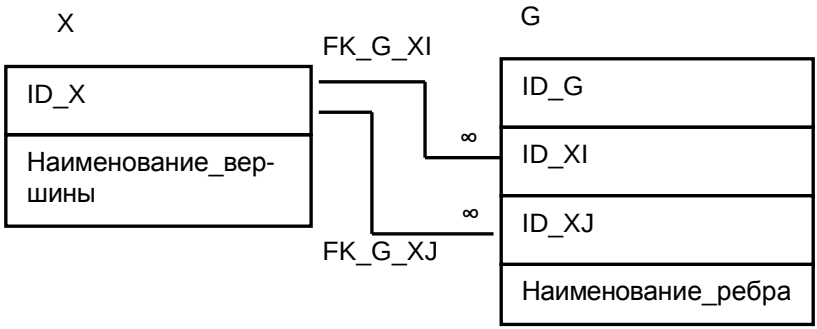


Рис. 1.2. Схема данных базы, описывающей ориентированные графы

Описанная база данных не включает ограничения O_1, O_2, O_3. Внешние ключи FK_ID_XI, FK_ID_XJ гарантируют ссылочную целостность базы, т.е в таблице G не будет значений ID_XI и ID_XJ, не входящих в домен ID_X таблицы X. Ограничения O_1, O_2, O_3 относятся к доменной целостности базы данных. Рассмотрим представление этих ограничений в базе данных BDo.

Проверка доменной целостности в РБД может осуществляться при выполнении операций удаления, редактирования и добавления записей.

Ограничение O_1 – граф не должен иметь петель. Это означает, что в отношении G не может быть записи, у которой G.ID_XJ=G.ID_XI. Это ограничение может быть добавлено к BDo следующим выражением:

```
alter table G add constraint CK_O1 check (ID_XI<>ID_IG)        (1.1)
```

Ограничение O_2 – граф не является мультиграфом. Это ограничение может быть добавлено двумя способами:

а) добавлением к отношению G уникального индекса по полям ID_XI и ID_XJ:

```
create unique index IX_O2 on G (ID_XI, ID_XJ);                 (1.2)
```

б) проверкой истинности условия:

```
(not exists (select *  from G
    where G.ID_XI=@ID_XI and G.ID_XJ = @ID_XJ )),              (1.3)
```

где @ID_XI, @ID_XJ – значения G.ID_XI и G.ID_XJ добавляемой записи (или новые значения редактируемой записи).

Ограничение O_3 – вершина-приемник связана только с одной вершиной-источником (ограничения на число связей нет). Это ограничение

может быть нарушено при добавлении или редактировании записи. Ограничение O_3 не будет нарушено если:

– добавляется ребро с вершиной приемником, у которой нет связей;

– связываются две вершины, у которых уже есть связь.

Обозначим @ID_XI, @ID_XJ – значения G.ID_XI и G.ID_XJ добавляемой записи (или новые значения редактируемой записи). Тогда ограничение O_3 не будет нарушено, если истинно выражение:

```
(not exists (select * from G   where G.ID_XJ=@ID_XJ)) or
(exists (select *  from G where G.ID_XI=@ID_XI                (1.4)
    and G.ID_XJ = @ID_XJ )
```

Выражения (1.1) и (1.2) создают отдельные объекты РБД, а именно CK_O1, IX_O2. Выражения (1.3) и (1.4) могут проверять тригеры. Обозначим тригеры, проверяющие выражения (1.3) и (1,4), как TR_O2 и TR_O3.

Обозначим G1={PK_ID_G, FK_ID_U, FK_ID_XI, FK_ID_XJ, ID_G, ID_U, ID_XI, ID_XJ}. Тогда с учетом ограничений таблица G для графов, представленных на рис. 1.1 будет выглядеть следующим образом:

– G={G1, CK_O1, IX_O2 или TR_O2, TR_O3} – рис. 1.1, *а*;

– G={G1, CK_O1, TR_O3} – рис. 1.1, *б*;

– G={G1, CK_O1, IX_O2 или TR_O2} – рис. 1.1, *в*;

– G={G1, CK_O1} – рис. 1, *г*;

– G={G1, CK_O1, IX_O2 или TR_O2} – рис. 1.1, *д*.

1.2. НЕОРИЕНТИРОВАННЫЕ ГИПЕРГРАФЫ С ОГРАНИЧЕНИЯМИ

Обозначим гиперграф $G_g(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин гиперграфа, x_i – i-я вершина; $U = \{u_m(Y_m)\}$, $m = \overline{1, M}$ – множество ребер гиперграфа, $u_m(Y_m)$ – m-е ребро гиперграфа, Y_m – множество вершин, инцидентных m-му ребру $Y_m \subseteq X$, $Y_m = \{x_k\}$, $k \in K_m$, $K_m \subseteq \overline{1, I}$. При этом два множества вершин равны $Y_m = Y_n$, если у них одинаковое число элементов, т.е. равны их мощности $|Y_m| = |Y_n|$, и для любого элемента $x_k \in Y_m$, $\overline{k \in 1, I}$ во множестве Y_n существует точно такой же элемент. Например, множества $\{x_1, x_5, x_3\}$ и $\{x_3, x_1, x_5\}$ равны.

Примеры неориентированных гиперграфов представлены на рис. 1.3.

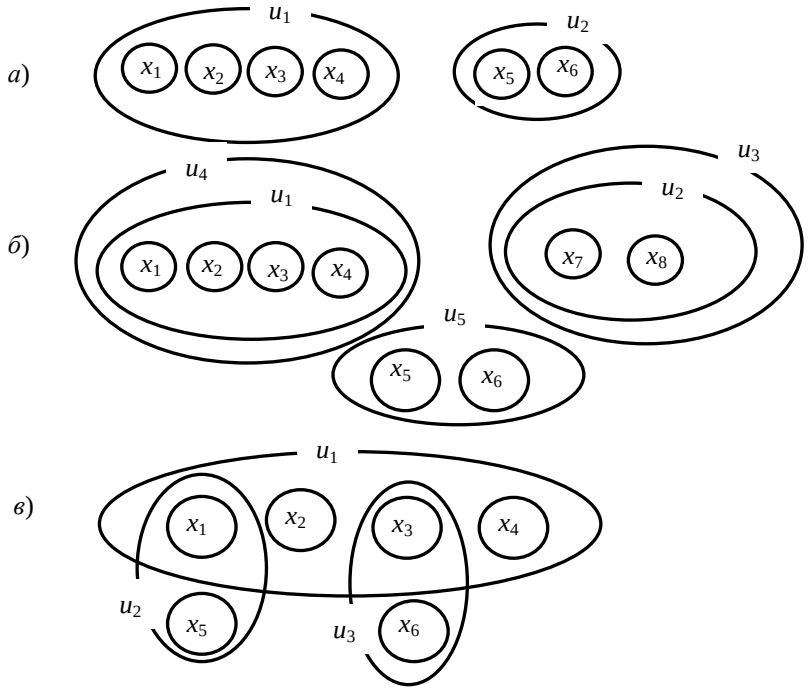


Рис. 1.3. Примеры неориентированных гиперграфов

Два ребра, имеющие одинаковые вершины, называются смежными. Гиперграф не будет иметь смежных ребер (рис. 1.3, *а*), если выполняется ограничение

$$O_4 : \forall u_m(Y_m) \in U;\ m \in \overline{1,M} \Rightarrow$$
$$\neg\exists u_n(Y_n) \in U;\ n \in \overline{1,M};\ m \neq n;\ Y_m \cap Y_n \neq \varnothing.$$

На рисунке 1.3, *б* представлен гиперграф, который используется в задачах размещения оборудования в производственных помещениях, когда определенная группа аппаратов (вершины) обладает одним и тем же свойством (первое ребро), например, пожароопасность и, следовательно, должна размещаться в соответствующем помещении (второе ребро). Описанная ситуация может быть учтена следующим ограничением. Определенный набор вершин $Y \subseteq X$ инцидентен определенному набору ребер $U' \subseteq U$, причем нет вершин, принадлежащих Y, инцидентных ребрам другого набора вершин.

$$O_5:\ \exists U' = \{u_p(Y_p)\} \subseteq U;\ Y_p \subseteq X;\ p \in P \subseteq \overline{1,M} \Rightarrow$$

$$\neg\exists u_n(Y_n) \notin U';\ n \in \overline{1,M};\ n \neq p;\ Y_n \subseteq X;\ Y_n \cap Y_p \neq \varnothing.$$

Ребра, имеющие равные множества вершин, называются кратными. Гиперграф не будет иметь кратных ребер (рис. 1.3, *в*), если выполняется ограничение

$$O_6:\ \forall u_m(Y_m) \in U;\ m \in \overline{1,M} \Rightarrow$$

$$\neg\exists u_n(Y_n) \in U;\ m \neq n;\ n \in \overline{1,M};\ Y_m = Y_n.$$

С учетом введенных ограничений гиперграфы, приведенные на рис. 1.3, можно записать в виде:

- $G_g(X, U, O_4)$ – рис. 1.3, *а*;
- $G_g(X, U, O_5)$ – рис. 1.3, *б*;
- $G_g(X, U, O_6)$ – рис. 1.3, *в*.

База данных BDg, описывающая графы на рис. 1.3, представляет собой следующий кортеж (схема данных представлена на рис. 1.4):

BDg=<X,U,G>,

где X,U,G – следующие отношения, X – вершины графа, U – ребра графа, G – граф (связи вершин и ребер).

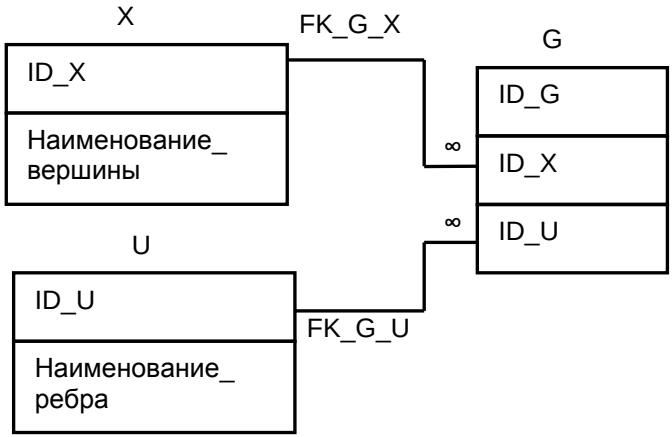


Рис. 1.4. Схема данных базы, описывающей неориентированные гиперграфы

В свою очередь отношения X,U,G можно представить как:

X={PK_ID_X, ID_X, Наименование_вершины}, где PK_ID_X – первичный ключ, ID_X – поле первичного ключа;

U={PK_ID_U, ID_U, Наименование_ребра}, где PK_ID_U – первичный ключ, ID_U – поле первичного ключа;

G={PK_ID_G, FK_G_U, FK_G_X, ID_G, ID_U, ID_XI}, где PK_ID_G – первичный ключ, FK_G_U, FK_G_X – внешние ключи (по полям ID_U, ID_X), ID_G – поле первичного ключа, ID_U, ID_X – поля первичного ключа ребра и вершины.

Внешние ключи FK_G_U, FK_G_X обеспечивают только ссылочную целостность данных и не обеспечивают выполнение ограничений O_4, O_5, O_6. Рассмотрим способы представления ограничений O_4, O_5, O_6 в РБД.

Ограничение O_4 – гиперграф не должен иметь смежных ребер (рис. 1.3, *а*). Это значит, что в отношении G может существовать только одна запись (кортеж) с заданной вершиной. Обозначим @ID_X – значения G.ID_X добавляемой в G записи (или новое значение редактируемой записи). Ограничение O_4 не будет нарушено, если истинно выражение:

```
(not exists (select * from G where  G.ID_X=@ID_X ))
```
(1.5)

Вторым способом обеспечения выполнения ограничения O_4 является создание уникального индекса в таблице G по полю ID_X:

```
create unique index IX_O4 on G (ID_X)
```

Ограничение O_5 – определенный набор вершин $Y \subseteq X$ инцидентен определенному набору ребер $U' \subseteq U$, причем нет вершин, принадлежащих Y, инцидентных ребрам другого набора (рис. 1.3, *б*). Рассмотрим ситуации, при которых возможно нарушение ограничения O_5.

Добавить (*удалить*) *вершину в ребро*. Обозначим @ID_U – ребро, в которое добавляется вершина. Здесь возможны два случая:

а) если набор вершин корректируемого ребра инцидентен только одному ребру (ребро U_5, рис. 1.3, *б*), то вершина без проблем добавляется (удаляется) в ребро. Набор вершин инцидентен только одному ребру @ID_U, если истинно условие:

```
(not exists (select ID_U from G as b,
    (select ID_X from G where ID_U=@ID_U) as a
    where b.ID_X=a.ID_X and b.ID_U<>@ID_U));
```
(1.6)

б) если набор вершин ребра инцидентен нескольким ребрам, то ограничение O_5 не будет нарушено, если добавить (удалить) вершину ко

всем этим ребрам. Для гиперграфа на рис. 1.3, *б* при добавлении новой вершины в ребро U_1 необходимо добавить эту вершину также и в ребро U_5. Предположим надо добавить вершину @ID_X в ребро @ID_U, тогда команда добавления будет следущая:

```
insert into G (ID_X, ID_U) (select distinct @ID_X, G.ID_U from G
 inner join G as G1 on G1.ID_U= @ID_U where G.ID_X=G1.ID_X).  (1.7)
```

Добавить новое ребро. Обозначим X1={ID_X1, ID_X) – таблица вершин добавляемого ребра, @M1 – число вершин (число записей в X1). Добавить новое ребро можно:

а) если все вершины X1 инцидентны хотя бы одному ребру (например, добавление ребра U_3, когда существует ребро U_2). Все вершины X1 инцидентны хотя бы одному ребру, если истинно выражение:

@M1=@M2,

где @M2 – число вершин существующего ребра, которые равны вершинам X1. Определение @M2 осуществляется выражением:

```
select @M2=count(*) from G, X1,
    (select top 1 ID_U from G, X1 where G.ID_X=X1.ID_X) as a
where G.ID_U=a.ID_U and G.ID_X=X1.ID_X;                    (1.8)
```

б) если в G нет ни одной вершины из X1, т.е. истинно выражение:

```
(not exists (select G.ID_X from G,X1 where X1.ID_X=G.ID_X)).   (1.9)
```

Ограничение O_6 – гиперграф не должен иметь кратных ребер (рис. 1.3, *в*). Рассмотрим следующие ситуации, которые могут привести к нарушению ограничения O_6.

Добавить новое ребро. Обозначим вершины добавляемого ребра, как отношение X1={ID_X}, @M1 – число вершин (число записей в X1). Добавить новое ребро можно:

а) если хотя бы одна вершина из X1 не входит ни в одно ребро (не присутствует в G), т.е. истинно выражение:

```
(exists (select X1.ID_X from X1 where X1.ID_X not in
     select X1.ID_X from G,X1 where G.ID_X=X1.ID_X)));        (1.10)
```

б) если не существует ребра с вершинами X1, т.е. истинно выражение:

```
(not exists (select count(*)) from X1,G where G.ID_X=X1.ID_X
          group by G.ID_U having (count(*))= @M1)            (1.11)
```

Добавить вершину в ребро или удалить вершину из ребра. Обозначим @ID_X – ID_X добавляемой (удаляемой) вершины, @ID_U – ID_U корректируемого ребра.

Добавить вершину в ребро можно, если вершина не существует в G, т.е. истинно выражение:

(not exists (select * from G where G.ID_X=@ID_X)) (1.12)

Добавить (удалить) вершину из ребра можно также, если нет ни одного ребра, все вершины которого равны вершинам получаемого ребра. Обозначим X1 – вершины получаемого ребра, тогда добавить (удалить) вершину из ребра можно, если выражение (1.11) истинно.

Обозначим TR_O4 – триггер, осуществляющий проверку выражения (5) (ограничение O_4), TR_O5 – триггер, осуществляющий проверку выполнимости ограничения O_5 (выражения (1.6) – (1.9)), TR_O6 – триггер, осуществляющий проверку выполнимости ограничения O_6 (выражения (1.10) – (1.12)).

Обозначим G1={PK_ID_G, FK_G_U, FK_G_X, ID_G, ID_U, ID_X}, тогда с учетом ограничений отношение G для графов, представленных на рис. 1.3, будет выглядеть следующим образом:

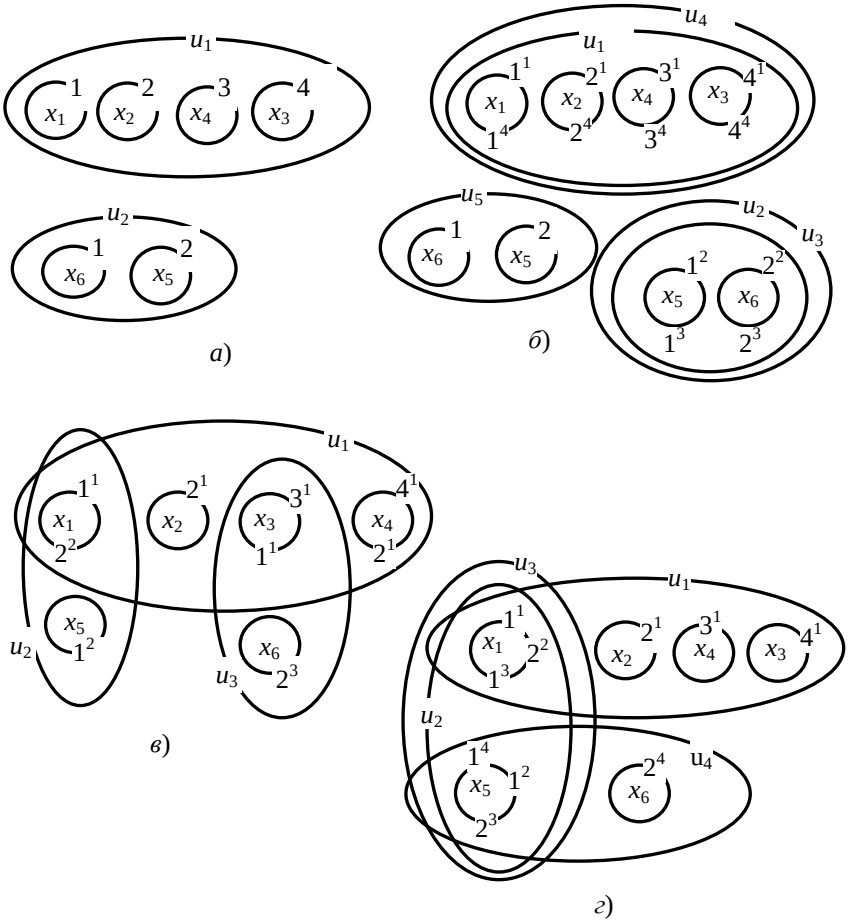
- G={G1, TR_O4 или IX_O4} – рис. 1.3, *а*;
- G={G1, TR_O5} – рис. 1.3, *б*;
- G={G1, TR_O6} – рис. 1.3, *в*.

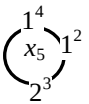
1.3. ОРИЕНТИРОВАННЫЕ ГИПЕРГРАФЫ С ОГРАНИЧЕНИЯМИ

Обозначим ориентированный гиперграф $G_{go}(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин гиперграфа, x_i – i-я вершина; $U = \{u_m(Y_m)\}$, $m = \overline{1, M}$ – множество ребер гиперграфа, $u_m(Y_m)$ – m-е ребро гиперграфа, Y_m – множество вершин инцидентных m-му ребру $Y_m \subseteq X$, $Y_m = \{x_k^l\}$, $k \in K_m$, $K_m \in \overline{1, I}$, l, – номер вершины в ребре ориентированного гиперграфа $l = \overline{1, L_m}$. При этом будем считать, что наборы вершин $\{x_3, x_2, x_4\}$ и $\{x_3, x_4, x_2\}$ разные, т.е. наборы вершин Y_k и Y_n равны, если

$$Y_k = \{x_p^l\};\ Y_n = (x_t^s\};\ p, t \in \overline{1, I};\ l = \overline{1, L_k};\ s = \overline{1, L_n};\ L_k = L_n;\ l = s \Rightarrow p = t. \quad (1.13)$$

Примеры ориентированных гиперграфов представлены на рис. 1.5.





– вершина x_5 принадлежит ребрам u_2, u_3, u_4 и ее номер в этих ребрах 1, 2 и 1 соответственно

Рис. 1.5. Примеры ориентированных гиперграфов

Для пояснения рисунков в таблице 1.1 приведены наборы вершин, инцидентных ребрам ориентированного гиперграфа, представленного на рис. 1.5, *г*.

Рассмотрим некоторые ограничения, которые могут накладываться на вершины и ребра ориентированного гиперграфа.

1.1. Наборы вершин, инцидентных ребрам ориентированного гиперграфа, представленного на рис. 1.5, *г*

Ребро гиперграфа	Вершины инцидентные ребру
u_1	$Y_1 = \{x_1, x_2, x_4, x_3\}$
u_2	$Y_2 = \{x_5, x_1\}$
u_3	$Y_3 = \{x_1, x_5\}$
u_4	$Y_4 = \{x_5, x_6\}$

Два ребра, имеющие одинаковые вершины, называются смежными. Гиперграф не будет иметь смежных ребер (рис. 1.1, *а*), если выполняется ограничение

$$O_7 : \forall u_m(Y_m) \in U;\ m \in \overline{1, M} \Rightarrow$$
$$\neg\exists u_n(Y_n) \in U;\ n \in \overline{1, M};\ m \neq n;\ Y_m \cap Y_n \neq \varnothing.$$

Операция $Y_m \cap Y_n$ проводится с учетом выражения (1.13). Ребра u_1 и u_3 на рис. 1.5, *г* имеют одинаковую вершину x_1. В то же время вершина x_1 не одинакова для ребер u_1 и u_2, так ее номер в этих ребрах один и два соответственно.

Ограничение O_7 можно представить и так. Вершина может быть инцидентна только одному ребру. Для ориентированного гиперграфа с учетом выражения (1.13) это означает, что любая вершина на конкретной позиции во всех ребрах может быть только один раз.

$$O_7 :\ \exists x_k;\ k \in \overline{1, I};\ u_m(Y_m) \in U;\ m \in \overline{1, M};\ Y_m = \{x_n^t\};\ n \in K_m;$$
$$t = \overline{1, L_m};\ x_n^{l1} = x_k;\ l1 \in \overline{1, L_m} \Rightarrow$$
$$\neg\exists u_p(Y_p) \in U;\ p \in \overline{1, M};\ Y_p = \{x_r^s\};\ r \in K_p;\ s = \overline{1, L_p};\ x_k \in Y_p;\ x_r^{l1} = x_k.$$

На рисунке 1.5, *б* представлен гиперграф, который используется в задачах размещения оборудования в производственных помещениях. Например, ребро u_1 задает этаж, на котором располагается цепочка емкостных аппаратов x_1, x_2, x_3, x_4. Номер вершины в ребре u_1 задает ее порядковый номер (координату) в цепочке. Номера вершин в ребре u_2 задают последовательность выполнения стадий периодического технологического процесса, который осуществляется в указанных аппаратах, причем номер аппарата в цепочке должен совпадать с номером стадии.

Описанная ситуация может быть учтена следующим ограничением. Определенный набор вершин $Y \subseteq X$ инцидентен определенному набору ребер $U' \subseteq U$, причем нет вершин, принадлежащих Y, инцидентных ребрам другого набора вершин.

$$O_8 : \exists U' = \{u_p(Y_p)\} \subseteq U;\ Y_p \subseteq X;\ p \in P \subseteq \overline{1,M} \Rightarrow$$

$$\neg\exists u_n(Y_n) \notin U';\ n \in \overline{1,M};\ n \neq p;\ Y_n \subseteq X;\ Y_n \cap Y_p \neq \varnothing.$$

Операция $Y_p \cap Y_n$ проводится с учетом выражения (1.13).

Ребра, имеющие равные множества вершин, называются кратными. Гиперграф не будет иметь кратных ребер (рис. 1.3, *в*), если выполняется ограничение

$$O_9 : \forall u_m(Y_m) \in U;\ m \in \overline{1,M} \Rightarrow$$

$$\neg\exists u_n(Y_n) \in U;\ m \neq n;\ n \in \overline{1,M};\ Y_m = Y_n.$$

С учетом введенных ограничений гиперграфы, приведенные на рис. 1.5, можно записать в виде:

- $G_{go}(X, U, O_1)$ – рис. 1.5, *а*;
- $G_{go}(X, U, O_2)$ – рис. 1.5, *б*;
- $G_{go}(X, U, O_3)$ – рис. 1.5, *в*;
- $G_{go}(X, U)$ – рис. 1.5, *г*.

База данных BDgo, описывающая графы на рис. 1.5, представляет собой следующий кортеж отношений (схема данных представлена на рис. 1.6).

BDgo=<X,U,G>,

где X – вершины графа, U – ребра графа, G – граф (связи вершин и ребер).

В свою очередь отношения X,U,G можно представить как:

X={PK_ID_X, ID_X, Наименование_вершины}, где PK_ID_X – первичный ключ, ID_X – поле первичного ключа;

U={PK_ID_U, ID_U, Наименования_ребра}, где PK_ID_U – первичный ключ, ID_U – поле первичного ключа;

G={PK_ID_G, FK_G_U, FK_G_X, ID_G, ID_U, ID_X, NPP, IX_1}, где PK_ID_G – первичный ключ, FK_G_U, FK_G_X – внешние ключи (по полям ID_U, ID_X), ID_G – поле первичного ключа, ID_U, ID_X – поля первичного ключа ребра и вершины, NPP – порядковый номер вершины в ребре, IX_1 – уникальный индекс по полям ID_U и NPP, гарантирующий, что в одном ребре не будет вершин с одинаковым порядковым номером.

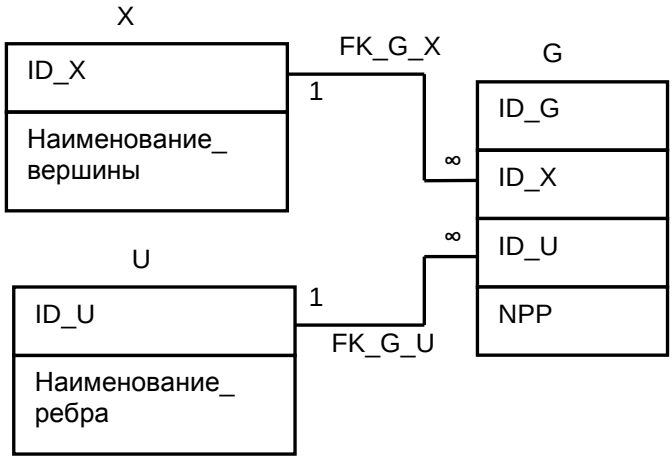


Рис. 1.6. Схема данных базы, описывающей ориентированные гиперграфы

Для обеспечения целостности базы рассмотрим представление ограничений O_7, O_8, O_9 в BDgo.

Ограничение O_7 – вершина может быть инцидентна только одному ребру (рис. 1.5, *а*). Это значит, что в отношении G может существовать только одна запись (кортеж) с заданной вершиной на заданном месте. В данном случае целостность базы (несоблюдение ограничения O_7) может быть нарушена при операции добавления и редактирования записей в таблицу G. Обозначим @ID_X – значения G.ID_X добавляемой записи (или новое значение редактируемой записи), @NPP – порядковый номер вершины в ребре. Ограничение O_7 не будет нарушено, если истинно выражение:

```
(not exists (select * from G
    where G.ID_X=@ID_X and G.NPP=@NPP))
```
(1.14)

Вторым способом обеспечения выполнения ограничения O_7 является создание уникального индекса в таблице G по полям ID_X, NPP

```
create unique index IX_O4 on G (ID_X, NPP)
```

Ограничение O_8 – определенный набор вершин $Y \subseteq X$ инцидентен определенному набору ребер $U' \subseteq U$, причем нет вершин, принадлежа-

щих Y, инцидентных ребрам другого набора (рис. 1.5, *б*). Рассмотрим ситуации, при которых возможно нарушение ограничения O_2.

Добавить новое ребро. Обозначим X1={ID_X, NPP} – таблица вершин добавляемого ребра, @M1 – число вершин (число записей в X1). Добавить новое ребро можно:

а) если весь набор вершин X1 равен (выражение (1.13)) набору вершин хотя бы одного имеющегося ребра (например, добавление ребра U_3, когда существует ребро U_2, рис. 1.5, *б*). Весь набор вершин X1 равен набору вершин имеющегося ребра, если истинно выражение:

@M1=@M2, (1.15)

где @M2 – число вершин существующего ребра, которые равны вершинам X1, причем @M2 определяется по выражению

```
select @M2=count(*) from G, X1, (select top 1 ID_U from G, X1
    where G.ID_X=X1.ID_X and G.NPP=X1.NPP) as a
where G.ID_U=a.ID_U and G.ID_X=X1.ID_X and G.NPP=X1.NPP;
```

б) если в G нет ни одной вершины из X1, т.е. истинно выражение:

```
(not exists (select G.ID_X from G,X1
    where X1.ID_X=G.ID_X and X1.NPP=G.NPP))
```
(1.16)

Добавить (*удалить*) *вершину в ребро*. Обозначим @ID_U – ребро, в которое добавляется вершина. Здесь возможны следующие варианты:

– первый вариант, вершина добавляется в начало или внутрь списка вершин ребра (не в конец), т.е. необходима перенумерация вершин в ребре;

– второй вариант, вершина добавляется в конец списка вершин ребра, т.е. порядковые номера существующих вершин ребра не изменяются;

Первый вариант целесообразно свести к добавлению нового ребра. Порядок действий при этом может быть следующим:

– копировать редактируемое ребро в таблицу X1;

– вставить новую вершину в X1 и перерумеровать вершины;

– добавить новое ребро (проверка ограничения при добавлении нового ребра описана выше).

Рассмотрим второй вариант, т.е. добавление вершины в конец списка вершин ребра. Если набор вершин корректируемого ребра инцидентен только этому ребру, то вершина добавляется (удаляется) в ребро (ребро U_5, рис. 1.5, *б*). Набор вершин инцидентен только одному ребру со значением первичного ключа, равным @ID_U, если истинно условие:

```
(not exists (select  ID_U from G as b, (select ID_X, NPP  from G
    where ID_U=@ID_U) as a where  b.ID_X=a.ID_X and
    b.NPP=a.NPP and b.ID_U<>@ID_U)
```
(1.17)

Если набор вершин ребра инцидентен нескольким ребрам, то надо добавить (удалить) вершину ко всем ребрам или не добавлять (удалять) ее вообще. Команда на добавление вершины будет следующей

```
insert into G (ID_X, ID_U) (select distinct @ID_X, G.ID_U from G
    inner join G as G1 on G1.ID_U= @ID_U where G.ID_X=G1.ID_X)
```

Ограничение O_3 – существует только одно ребро, инцидентное определенному набору вершин (рис. 1.5, *в*). Рассмотрим ситуации, при которых возможно нарушение ограничения O_3.

Добавить новое ребро. Обозначим X1={ID_X, NPP} – вершины добавляемого ребра, @M1 – число вершин (число записей в X1). Добавить новое ребро можно:

а) если хотя бы одна вершина из X1 не содержится в G, т.е. истинно выражение:

```
(exists (select X1.ID_X from X1 where X1.ID_X not in
    (select X1.ID_X from G,X1 where G.ID_X=X1.ID_X and
    G.NPP=X1.NPP);
```
(1.18)

б) если не существует ребра, все вершины которого равны X1, т.е. истинно выражение:

```
(not exists (select count(*) from X1,G where G.ID_X=X1.ID_X and
    G.NPP=X1.NPP group by G.ID_U having (count(*))=@M1))
```
(1.19)

Добавить (*удалить*) *вершину в ребро* (*из ребра*). Обозначим @ID_X – значение первичного ключа (ID_X) добавляемой (удаляемой) вершины, @ID_U – значение первичного ключа (ID_U) ребра, @NPP – порядковый номер вершины в ребре.

Добавить вершину в ребро можно, если вершина не существует в G, т.е. истинно выражение:

```
(not exists (select * from G where G.ID_X=@ID_X and
    G.NPP=@NPP))
```
(1.20)

Добавить (удалить) вершину из ребра можно, если нет ни одного ребра, все вершины которого равны вершинам получаемого ребра. Обозначим X1 – вершины получаемого ребра, тогда добавить (удалить) вершину из ребра можно, если выражение (1.19) истинно.

Обозначим TR_O1 – триггер, осуществляющий проверку выражения (1.14) (ограничение O_7), TR_O2 – триггер, осуществляющий проверку

выполнимости ограничения O_8 (выражения (1.15) – (1.17)), TR_O3 – триггер, осуществляющий проверку выполнимости ограничения O_9 (выражения (1.18) – (1.20)).

Обозначим G1={PK_ID_G, FK_G_U, FK_G_X, ID_G, ID_U, ID_X. NPP}, тогда с учетом ограничений таблица G для графов, представленных на рис. 1.5 будет выглядеть следующим образом:

- G={G1, TR_O1 или IX_O1} – рис. 1.5, *а*;
- G={G1, TR_O2} – рис. 1.5, *б*;
- G={G1, TR_O3} – рис. 1.5, *в*;
- G={G1} – рис. 1.5, *г*.

1.4. НЕОРИЕНТИРОВАННЫЕ УЛЬТРАГРАФЫ

Обозначим ультраграф $G_u(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин ультраграфа x_i – i-я вершина; $U = \{u_m(Y_m)\}$, $m = \overline{1, M}$ – множество ребер ультраграфа, $u_m(Y_m)$ – m-е ребро ультраграфа, Y_m – множество вершин, инцидентных m-му ребру $Y_m \subseteq X$, $Y_m = \{x_{kp}\}$, $k \in K_m$, $K_m \subseteq \overline{1, I}$, $p = \{1, 2\}$ – тип вершины, $p = 1$ – вершина-приемник, $p = 2$ – вершина-источник. При этом будем считать, что наборы вершин $\{x_{1,1}, x_{2,2}, x_{3,1}\}$ и $\{x_{1,2}, x_{2,2}, x_{3,1}\}$ разные (у них разные элементы), наборы $\{x_{1,1}, x_{2,2}, x_{3,1}\}$ и $\{x_{2,2}, x_{1,1}, x_{3,1}\}$ – одинаковые (у них одинаковые элементы, место положения элемента не учитываем), т.е. наборы вершин Y_k и Y_n равны, если

$$Y_k = \{x_{pl}\};\ Y_n = \{x_{ts}\};\ p, t \in \overline{1, I};\ l \in \overline{1,2};\ s \in \overline{1,2} \Rightarrow$$

$$\forall x_{pl} \in Y_k\ \exists x_{ts} \in Y_n;\ t = p;\ l = s;\ x_{ts} \in Y_n,\ |Y_k| = |Y_n|.$$

Примеры неориентированных ультраграфов представлены на рис. 1.7. Например, в ультраграфе рис. 1.7, *г* вершина x_5 является приемником для ребер u_3, u_4 и источником для ребра u_2.

Неориентированный ультраграф можно свести к ориентированному гиперграфу, номера вершин которого принимают только значение 1 (вершина-источник) или 2 (вершина-приемник). Тогда неориентированные ультраграфы, представленные на рис. 1.7, можно представить ориентированными гиперграфами, рис. 1.8.

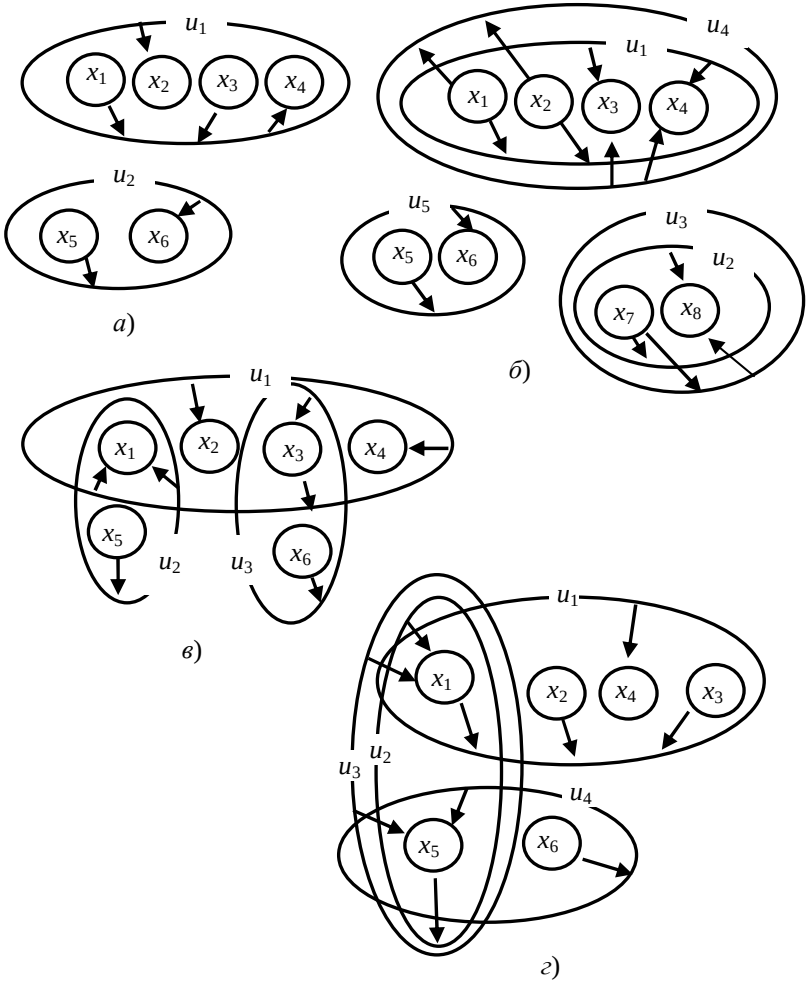


Рис. 1.7. Примеры неориентированных ультраграфов

Таким образом, ограничения O_7, O_8, O_9 будут справедливы и для неориентированных ультраграфов. Структура базы данных, описывающей неориентированные ультраграфы BDu, будет аналогичной структуре базы данных BDgo, описывающей ориентированные гиперграфы, если принять что поле G.NPP может принимать только значение 1 или 2. Отличие будет в том, что в таблице BDu.G не будет уникальтного индекса IX_1 по

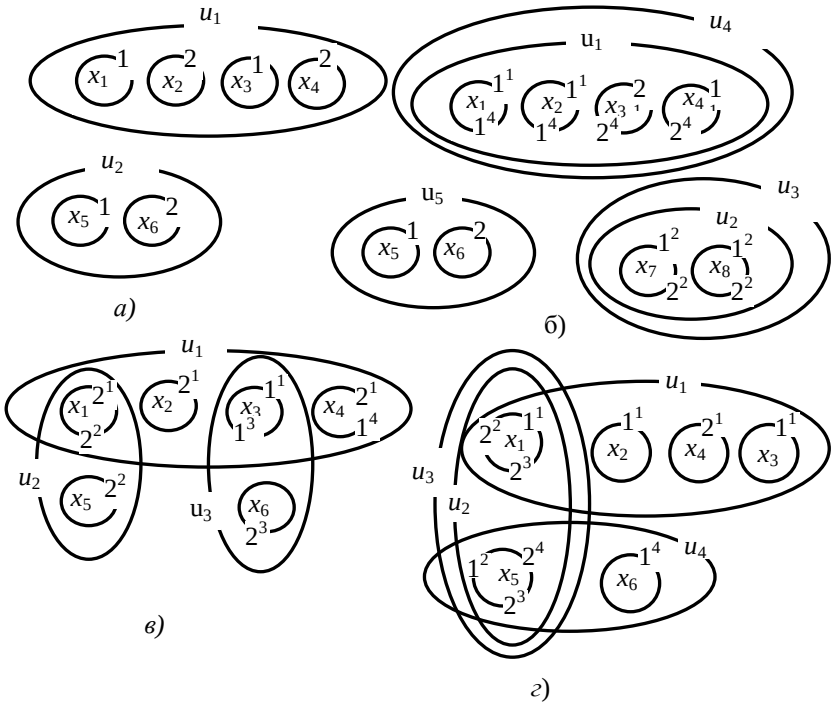


Рис. 1.8. Ориентированные гиперграфы, полученные из ультраграфов, рис. 1.7

полям ID_U и NPP, гарантирующего, что в одном ребре не будет вершин с одинаковым порядковым номером, но будет дополнительно ограничение на допустимые значения поля G.NPP (допустимые значения 1 или 2).

1.5. ОРИЕНТИРОВАННЫЕ УЛЬТРАГРАФЫ

Обозначим ориентированный ультраграф $G_{uo}(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин ультраграфа, x_i – i-я вершина; $U = \{u_m(Y_m)\}$, $m = \overline{1, M}$ – множество ребер ультраграфа, $u_m(Y_m)$ – m-е ребро ультраграфа, Y_m – множество вершин инцидентных m-му ребру $Y_m \subseteq X$, $Y_m = \{x_{kp}^l\}$, $k \in K_m$, $K_m \subseteq \overline{1, I}$, $p = \{1, 2\}$ – тип вершины, $p = 1$ – вершина-приемник, $p = 2$ – вершина-источник; l – номер вершины в ребре $l = \overline{1, L_m}$. При этом будем считать, что наборы вершин

$\{x^1_{1,1},\ x^2_{2,2},\ x^3_{3,1}\}$ и $\{x^1_{1,2},\ x^2_{2,2},\ x^3_{3,1}\}$ разные, т.е. наборы вершин Y_k и Y_n равны, если

$$Y_k = \{x^r_{hl}\};\ Y_n = \{x^m_{ts}\};\ h, t \in \overline{1,I};\ l \in \overline{1,2};\ s \in \overline{1,2};\ r = \overline{1,L_k};\ m = \overline{1,L_n} \Rightarrow$$
$$\forall x^r_{hl} \in Y_k\ \exists x^m_{ts} \in Y_n;\ h = t;\ l = s;\ r = m;\ |Y_k| = |Y_n|. \qquad (1.21)$$

На рисунке 1.9 представлены примеры ориентированных ультраграфов.

Вершина x_5 ультраграфа рис. 1.9, *г* является приемником для ребер u_2, u_3 и источником для ребра u_4. Кроме того, порядковый номер вершины в ребре u_2 первый, ребре u_3 второй, в ребре u_4 второй.

Ориентированный ультраграф можно рассматривать как дважды ориентированный гиперграф. Если в ориентированном гиперграфе порядок следования вершин в ребре можно рассматривать как их положение на оси, то в дважды ориентированном гиперграфе вершины в ребре располагаются на плоскости, причем порядок обхода вершин ребра задается одной из координат. На рисунке 1.10 представлены ориентированные ультраграфы (рис. 1.9, *а* и 1.9, *б*) как дважды ориентированные гиперграфы.

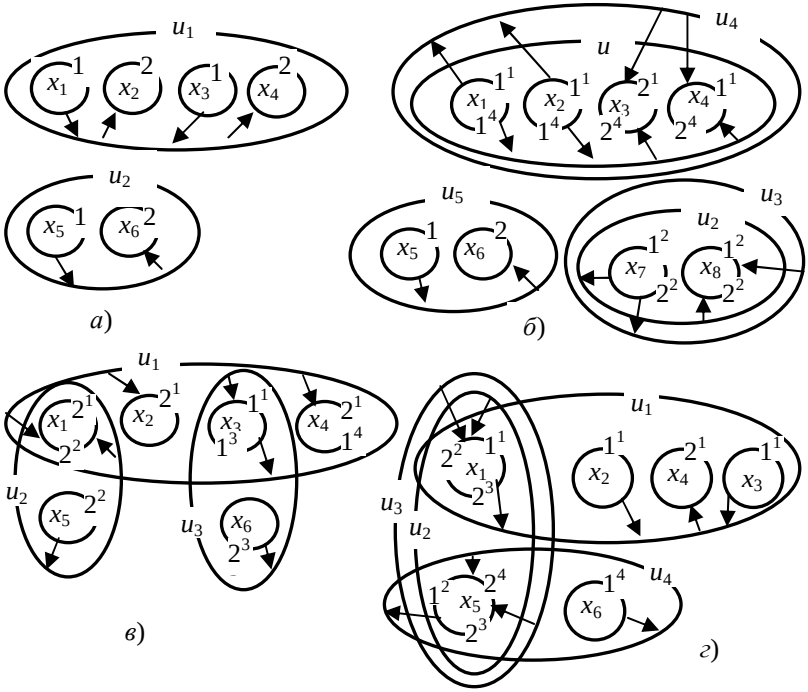
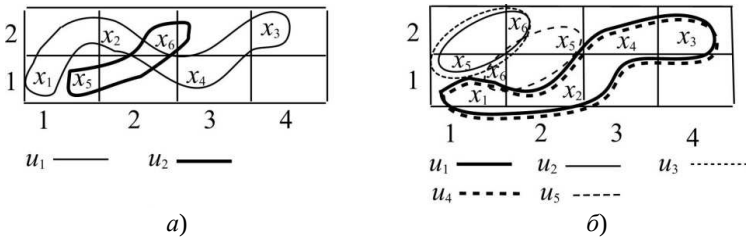


Рис. 1.9. Примеры ориентированных ультраграфов



а) *б*)

Рис. 1.10. Примеры дважды ориентированных гиперграфов

Таким образом, ограничения O_7, O_8, O_9 для ориентированных ультраграфов будут отличаться от ограничений ориентированного гиперграфа добавлением еще одного индекса в выражения, описывающие набор вершин, инцидентных гиперребру.

Для примера рассмотрим ограничение O_{10} – вершина, как приемник-источник, может быть инцидентна только одному ребру (рис. 1.9, *а*). Для ориентированного ультраграфа это означает, что любая вершина на конкретной позиции во всех ребрах с признаком приемник-источник может быть только один раз. Отличие этого ограничения от ограничения O_7 заключается в добавлении еще одного индекса (1 – источник, 2 – приемник) к вершинам, инцидентным определенным ребрам

$$O_{10}: \exists x_k;\ k \in \overline{1, I};\ u_m(Y_m) \in U;\ m \in \overline{1, M};\ Y_m = \{x_n^{t,a}\};\ n \in K_m;$$
$$t = \overline{1, L_m};\ a \in 1, 2;\ x_n^{l1,a1} \in Y_m;\ x_n^{l1,a1} = x_k;\ l1 \in \overline{1, L_m};\ a1 \in 1, 2 \Rightarrow \quad (1.22)$$
$$\neg\exists u_p(Y_p) \in U;\ p \in \overline{1, M};\ Y_p = \{x_r^{s,b}\};\ r \in K_p;\ s = \overline{1, L_p};$$
$$x_k \in Y_p;\ x_r^{l1,a1} = x_k.$$

Аналогично можно трансформировать ограничения O_8 и O_9.

Схема данных базы, описывающей ориентированные ультраграфы BDuo, представлена на рис. 1.11. Отличие ее от базы, описывающей ориентированный гиперграф (BDgo), заключается в добавлении поля NPP1, которое принимает значение 1 для вершины источника и 2 для вершины приемника.

Преобразуем выражение (1.14) в ограничение O_{10}. Обозначим @ID_X – значения G.ID_X добавляемой записи (или новое значение редактируемой записи), @NPP – порядковый номер вершины в ребре, @NPP1 – значение, задающее характер вершины (источник или приемник).

Ограничение O_{10} не будет нарушено, если истинно выражение:

```
(not exists (select * from G where G.ID_X=@ID_X
    and G.NPP=@NPP and G.NPP1=@NPP1))
```

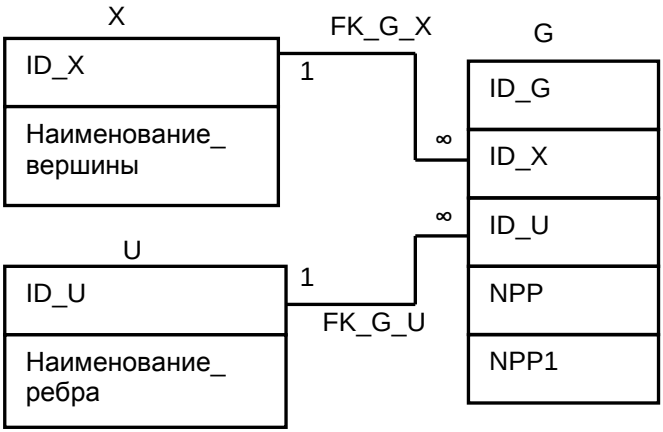


Рис. 1.11. Схема данных базы, описывающей ориентированные ультраграфы

Вторым способом обеспечения выполнения ограничения O_{10} является создание уникального индекса в таблице G по полям ID_X, NPP, NPP1:

```
create unique index IX_O4 on G (ID_X, NPP, NPP1)
```

Аналогично нетрудно преобразовать выражения (1.15) – (1.20).

1.6. N-ОРИЕНТИРОВАННЫЕ ГИПЕРГРАФЫ С ОГРАНИЧЕНИЯМИ

Обозначим N-ориентированный гиперграф $G_{gon}(X,U)$, где $X=\{x_i\}$, $i=\overline{1,I}$ – множество вершин гиперграфа, x_i – i-я вершина; $U=\{u_m(Y_m)\}$, $m=\overline{1,M}$ – множество ребер гиперграфа, $u_m(Y_m)$ – m-е ребро гиперграфа, Y_m – множество вершин, инцидентных m-му ребру $Y_m \subseteq X$, $Y_m=\{x_k^{\overline{L}}\}$, $k \in K_m$, $K_m \in \overline{1,I}$, $\overline{L}$ – номер вершины в ребре ориентированного гиперграфа представляет собой вектор, $\overline{L}=\{l_n\}$, $n=\overline{1,N}$, мощность которого N.

По аналогии со схемой данных дважды ориентированного гиперграфа составим схему данных N-ориентированного гиперграфа, рис. 1.12.

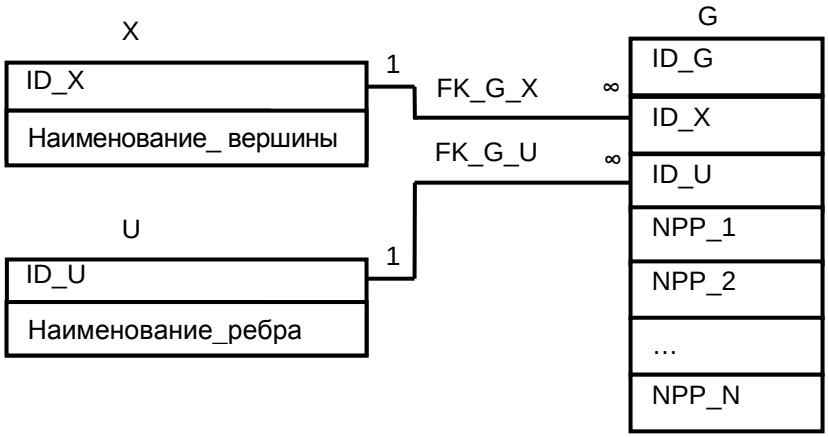


Рис. 1.12. Схема данных N-ориентированного гиперграфа

Представление ограничений в РБД для N-ориентированного гиперграфа рассмотрим на примере ограничения O_7.

Ограничение O_7 – вершина может быть инцидентна только одному ребру. Для N-ориентированного гиперграфа это означает, что не существует двух ребер, у которых набор порядковых номеров одной и той же вершины одинаков. Для изготовления унифицированных шаблонов проверки ограничения O_7 порядок ориентации гиперграфа N является исходными данными.

Обозначим @N – порядок ориентации гиперграфа. При добавлении или изменении записи в таблице G новые значения полей N_1, …, N_i, …, N_N будем держать в таблице T={ID_T,NPP} таким образом, что в первой записи этой таблицы ID_T=1 и NPP=N_1, во второй ID_T=2 и NPP=N_2 и т.д. Тип поля NPP varchar. Число записей в таблице T равно @N.

Ограничение O_7 не будет нарушено, если истинно выражение:

```
(not exists (select * from G where G.ID_X=@ID_X,
    NPP_1=select NPP from T where ID_T=1,
    NPP_2=select NPP from T where ID_T=2,                    (1.23)
    ...
    NPP_N=select NPP from T where ID_T=@N)
```

Выражение (1.23) в виде текстовой строки можно сформировать следующим образом:

'(not exists (select * from G where G.ID_X=@ID_X) ' +@S , (1.24)

где @S находится следующим образом:

```
Declare @I int, @S varchar(max)
    select @I=1, @S=''
    while @I<>@N
        select @S=@S+' and NPP_'+convert(varchar(4), @I)+'='+
        (select NPP from T where ID_T=@I ), @I=@I+1
```

В выражении (1.25) представлен пример добавления записи в таблицу G с проверкой выполнения ограничения O_7:

```
Declare @I int, @S varchar(max)
    select @I=1, @S=''
    while @I<>@N                                              (1.25)
      select @S=@S+' and NPP_'+convert(varchar(4), @I)+'='+
      (select NPP from T where ID_T=@I), @I=@I+1
    execute('Insert G vaues( )  where (not exists (select * from G
       where G.ID_X=@ID_X ) ' +@S
```

Вторым способом обеспечения выполнения ограничения O_7 является создание уникального индекса в таблице G по полям ID_X, NPP_1, …, NPP_N:

```
Declare @I int,  @S varchar(max)
    select @I=1, @S=''                                        (1.26)
    while @I<=@N
      select @S=@S + ',NPP_' + convert(varchar(4), @I),@I=@I+1
    execute('create unique index IX_O1 on G (ID_X' + @S + ')' )
```

Ограничение O_8 – определенный набор вершин $Y \subseteq X$ инцидентен определенному набору ребер $U' \subseteq U$, причем нет вершин, принадлежащих Y, инцидентных ребрам другого набора.

Рассмотрим ситуации, при которых возможно нарушение ограничения O_8.

Добавить новое ребро. Обозначим X1={ID_X, NPP_1, …NPP_N) – таблица вершин добавляемого ребра, @M1 – число вершин (число записей в X1). Добавить новое ребро можно:

а) если все вершины X1 инцидентны хотя бы одному ребру (например, добавление ребра U_3, когда существует ребро U_2, рис. 1.11, *б*). Все вершины X1 инцидентны хотя бы одному ребру, если истинно выражение:

@M1=@M2, (1.27)

где @M2 – число вершин существующего ребра, которые равны вершинам X1. Определить @M2 позволяет выражение:

```
Declare @S varchar(max), @I int, @S1 varchar(max)
    select @I=1,  @S=''
    while @I<=@N                                                    (1.28)
      select @S=@S + ' and G.NPP_' + convert(varchar(4), @I) +
           '= X1.NPP_' +  convert(varchar(4), @I),
            @I=@I+1
    select @S1='declare @M2 int; select @M2=count(*) from G, X1,
    (select top 1 ID_U from G, X1 where G.ID_X=X1.ID_X)
       as a  where G.ID_U=a.ID_U and G.ID_X=X1.ID_X' + @S
    execute (@S1);
```

б) если в G нет ни одной вершины из X1, т.е. истинно выражение:

```
execute ( '(not exists (select G.ID_X from G.X1
     where X1.ID_X=G.ID_X ' + @S + ')') ,
```

где @S определяется аналогично (1.28).

Используя динамически формируемую программу, нетрудно проверить другие ограничения, накладываемые на ребра N-ориентированного гиперграфа.

Вывод – все действия можно отлаживать на ориентированном гиперграфе, а потом, используя динамическое формирование программного кода, перенести эти действия на N-ориентированный гиперграф.

2. ОСНОВНЫЕ ОПЕРАЦИИ НАД N-ОРИЕНТИРОВАННЫМИ ГИПЕРГРАФАМИ В РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

Операции над графовыми структурами являются неотъемлемыми при использовании геперграфов в атоматизированных системах. В [1] подробно представлены методы декомпозиции гиперграфовых структур, в [28] описаны операции над ультра- и гиперграфами, представленными в аналитическом виде. Указанные публикации описывают операции над гиперграфами, не привязываясь к программным продуктам, в среде которых предполагается реализация автоматизированной системы. В данном разделе представлены основные операции над N-ориентированными гиперграфами в реляционной базе данных, выполнеямые таким образом, чтобы сохранить целостность базы данных. Операции описаны в терминах теории множеств и на языке Transact SQL.

По аналогии с ориентированным гиперграфом обозначим N-ориентированный гиперграф $G_{gon}(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин гиперграфа, x_i – i-я вершина; $U = \{u_m(X1_m)\}$, $m = \overline{1, M}$ – множество ребер гиперграфа, $u_m(X1_m)$ – m-е ребро гиперграфа, $X1_m$ – множество вершин, инцидентных m-му ребру $X1_m \subseteq X$, $X1_m = \{x_k^{\overline{L}}\}$, $\forall k \in K_m$, $K_m \subseteq \overline{1, I}$, $\overline{L}$ – номер вершины в ребре ориентированного гиперграфа представляет собой вектор, $\overline{L} = \{l_n\}$, $n = \overline{1, N}$, мощность которого N. В общем случае номер вершины в ребре не обязательно должен иметь значение "номер по порядку" и может отражать определенное свойство вершины, которое принимает конкретное значение при включении вершины в ребро. При этом предполагается, что все вершины в ребрах имеют одинаковый набор свойств.

Для простоты координаты (свойства) вершин будем обозначать номерами от 1 до N. Значение свойства n, $n \in \overline{1, N}$ вершины x_i, $x_i \in X$ в ребре u_m, $u_m \in U$ будем обозначать $s[n, i, m]$. С учетом сказанного, множество вершин, инцидентных m-му ребру $X1_m \subseteq X$, с множеством значений свойств каждой вершины в ребре будем обозначать $X1_m = \{x_k, S_k\}$, где $\forall k \in K_m$, $K_m \subseteq \overline{1, I}$, $S_k = \{s[n, k, m]\}$ – множество значений свойств k-й вершины в m-м ребре, $n = \overline{1, N}$.

По аналогии со схемой данных ориентированного гиперграфа [14] представим схему данных N-ориентированного гиперграфа, рис. 2.1.

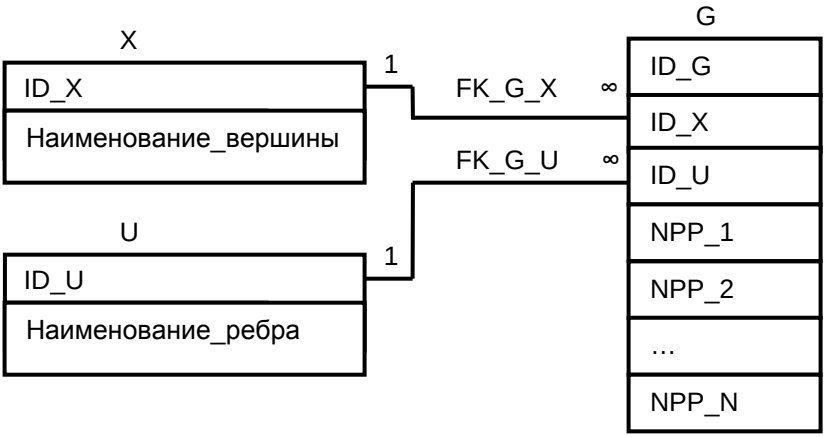


Рис. 2.1. Схема данных N-ориентированного гиперграфа

Примем, что значения полей первичного ключа ID_U, ID_X, ID_G определяются автоматически (например, свойство IDENTITY в Transact-SQL или счетчик в MS-Access). Будем также считать, что наименование вершины в таблице X и наименование ребра в таблице U уникальны. FK_G_X, FK_G_U – внешние индексы, связывающие таблицы G и X, G и U.

Поля NPP_1, NPP_2, … NPP_N таблицы G представляют собой координаты (или свойства) вершины ID_X в ребре ID_U. Число координат определяется прикладной задачей. Максимальное число координат N определяется ограничениями выбранной СУБД. Так, например, для СУБД MS-SQL Server 2005 максимальное число столбцов в таблице – 1024, что достаточно для решения практических задач по проектированию технологического оборудования и управления предприятиями химического и машиностроительного профилей.

2.1. ДОБАВЛЕНИЕ НОВОЙ ВЕРШИНЫ

Добавление новой вершины складывается из двух этапов:

– добавление вершины в множество вершин $X = \{x_i\},\ i = \overline{1, I}$;

– добавление вершины в множество инцидентных ей ребер $U1 \subseteq U$, где $U1$ – множество ребер, которым инцидентна добавляемая вершина.

При добавлении вершины в существующие ребра возможны следующие ситуации:

– координаты вершин, входящих в ребро, изменяются, например, координата "порядковый номер вершины" изменяется у всех существующих

вершин (увеличивается на единицу) при добавлении новой вершины с порядковым номером один;

– координаты существующих вершин не изменяются.

Обозначим x_t – добавляемая вершина, $U1=\{u_{m1}(X1_{m1})\}$, $\forall m1\in M1,\ M1\subseteq\overline{1,M}$ – множество ребер гиперграфа, которым инцидентна вершина x_t, $X1_{m1}=\{x_k, S_k\}$ – множество вершин и свойств вершин ребер, инцидентных добавляемой вершине, $S_k=\{s[n, k, m1]\}$, $n=\overline{1,N}$, $\forall k\in K_{m1},\ K_{m1}\in\overline{1,I}$. Вершина x_t добавляется в каждое ребро со своим набором координат (свойств), которые обозначим как $S1_t=\{s[n, t, m1]\}$, $n=\overline{1,N}$.

Для представления операции добавления новой вершины в среде Transact-SQL введем таблицу свойств добавляемой вершины S1={ID_S1, ID_U, NPP_1, NPP_2, …, NPP_N}, где ID_S1 – поле первичного ключа, ID_U – поле первичного ключа ребер, инцидентных добавляемой вершине, NPP_1, NPP_2, …, NPP_N – значения свойств добавляемой вершины в ребре ID_U. Наименование добавляемой вершины будем обозначать переменной @Наименование_вершины, значение поля первичного ключа – @ID_X.

С учетом введенных обозначений последовательность действий и реализация этих действий в среде Transact-SQL при добавлении новой вершины в ребро следующая:

1) Если $x_t\notin X$, то добавление x_t в X:

```
if not exists (select * from  X where
Наименование_вершины=@Наименование_вершины)
   begin
     insert X values(@Наименование_вершины)
     select @ID_X=@@Identity
   end
```

2) Добавление вершины x_t в каждое ребро множества ребер $U1=\{u_{m1}(X1_{m1})\}$ со своим набором свойств $S1_t=\{s[n, t, m1]\}$, $m1\in\overline{M1}$, $M1\subseteq\overline{1,M}$, $n=\overline{1,N}$, т.е. в результате получаем

$$U=\{u_m(X2_m)\},\quad X2_m=X_{m2}\cup(X_{m1}\cap x_t),\quad \forall m2\in M\setminus M1,\ \forall m1\in M1,$$

где $m1, m2$ – индексы ребер, инцидентных и неинцидентных вершине x_t.

```
Insert G (ID_X, ID_U, NPP_1, NPP_2,…, NPP_N)
        (select @ID_X, ID_U, NPP_1, NPP_2,.., NPP_N  from S1)
```

3) Корректировка свойств S_k вершин множества ребер $U1=\{u_{m1}(Y1_{m1})\}$ в зависимости от свойств вершины x_t. Этот пункт полностью определяется характером решаемой задачи. Для примера обозначим функцию, позволяющую получить новое значение свойства NPP вершины ID_X в ребре ID_U, как dbo.F_NPP(NPP, ID_X, ID_U). Тогда корректировку свойств вершин можно записать следующим образом.

$$S_k=\{s[n,k,m]=f\_npp(n,k,m)\}\quad n=\overline{1,N},\quad k\in K_{m1},\quad K_{m1}\in\overline{1,I},$$
$$\forall m\in M1,\quad M1\subseteq M,$$

где $f\_npp(n,k,m)$ – функция, позволяющая найти новое значение свойства n вершины k в ребре m.

```
Update G set NPP_1=dbo.F_NPP('NPP_1', G. ID_X, S1.ID_U),
    NPP_2=dbo.F_NPP('NPP_2', G. ID_X, S1.ID_U),
    … ,
    NPP_N=dbo.F_NPP('NPP_N', G.ID_X, S1.ID_U )
from S1 where  G.ID_U=S1.ID_U and G.ID_X<>@ID_X
```

2.2. ДОБАВЛЕНИЕ НОВОГО РЕБРА

При добавлении нового ребра возможны два основных случая:

- новое ребро содержит только существующие вершины;
- новое ребро содержит новые вершины.

Обозначим u_t – добавляемое ребро, $X1_t=\{x_k, S_k\}$, $\forall k\in\overline{1,I}\cup K_n$ – множество вершин ребра с их свойствами, $S_k=\{s[n,k,t]\}$, $n=\overline{1,N}$ – множество свойств k-й вершины в ребре u_t, K_n – множество индексов новых вершин в добавляемом ребре $K_n\cap\overline{1,I}=\varnothing$.

Для представления операции добавления в Transact-SQL введем таблицу вершин добавляемого ребра:

X1={ID_X1, ID_X, Наименование_вершины}. В таблице X1 поле ID_X имеет значение NULL, если вершина новая (не содержится в таблице X). Введем таблицу свойств добавляемого ребра U1 = (ID_U1, ID_X, NPP_1, NPP_2, …, NPP_N), где ID_U1 поле первичного ключа, ID_X – поле первичного ключа вершины инцидентной добавляемому ребру, NPP_1, NPP_2, …, NPP_N – значения свойств вершин в добавляемом ребре. Обозначим наименование нового ребра как @Наименование_ребра.

С учетом введенных обозначений последовательность действий и реализация этих действий в среде Transact-SQL при добавлении нового ребра следующая.

1) Добавление в множество X новых вершин, если они есть в добавляемом ребре – результат множество $X2$.

Если $K_n \neq \varnothing$, то $X2 = X \cup \{x_k\}$, $\forall k \in K_n$.

```
Insert X (Наименование_вершины) (select Наименование_вершины
    from X1 where not exists(select * from X where
        X.Наименование_вершины=X1.Наименование_вершины))
```

2) Формирование множества вершин и свойств вершин добавляемого ребра $X1_t = \{x_k, S_k\}$, $x_k \in X2$.

```
Insert U1 (ID_X) (select X.ID_X from X,X1 where
    X.Наименование_вершины=X1.Наименование_вершины).
    Exec INPUT_U1_NPP,
```

где INPUT_U1_NPP – процедура, позволяющая ввести значения свойств вершин добавляемого ребра.

3) Добавление ребра u_t, результат – множество $U2 = U \cup u_t$.

```
Insert U values(@Наименование_ребра) set @ID_U=@@Identity
Insert G (ID_X,ID_U, NPP_1, NPP_2,…, NPP_N)
    (select ID_X, @ID_U, NPP_1, NPP_2,.., NPP_N  from U1)
```

2.3. УДАЛЕНИЕ ВЕРШИНЫ

В [1] предлагается называть слабым удалением – удаление вершины из ребра с сохранением ребер, сильным удалением – удаление вершин вместе с инцидентными ребрами.

Будем рассматривать следующие типовые операции удаления вершины:

– удаление вершины из определенного ребра с сохранением ребра – простое удаление вершины;

– удаление вершины из ребра вместе с ребром (равнозначно удалению ребра, инцидентного вершине) – составное удаление;

– удаление вершины из всех ребер с сохранением ребер – слабое удаление;

– удаление выршины из всех ребер вместе с ребрами (равнозначно удалению ребер, инцидентных вершине) – сильное удаление.

При этом возможно как удаление вершины из множества X, так и сохранение ее в множестве X. Удаление вершины из множества X возможно только при слабом и сильном удалении вершины, в противном случае будет нарушена целостность базы. Кроме того, удаление вершины из ребра, как и в случае добавления вершины в ребро, может привести к изменению координат (свойств) других вершин в этом ребре.

Обозначим x_t, $t \in \overline{1,I}$ – удаляемая вершина, u_z, $z \in \overline{1,M}$ – ребро, из которого удаляется вершина, $X1_j$, $X1_z$ – множество вершин инцидентных j-му и z-му ребру, $U2$ – множество ребер после удаления, $G_{gon}(X \setminus x_t, U2)$ – получаемый гиперграф при удалении вершины из множества X, $G_{gon}(X, U2)$ – получаемый гиперграф без удаления вершины из множества X, $U2$ – множество ребер гиперграфа, получаемое после удаления вершины x_t.

Как и в случае добавления вершины, корректировка свойств S_z вершин ребра $u_z(X1_z)$ в зависимости от свойств вершины x_t полностью определяется характером решаемой задачи.

Обозначим функцию, позволяющую получить новое значение свойства NPP вершины ID_X в ребре ID_U, как dbo.F_NPP(NPP, ID_X, ID_U). Обозначим @Наименование_вершины – наименование удаляемой вершины, @Наименование_ребра – наименованте ребра, из которого удаляется вершина.

Простое удаление вершины – удаление вершины из определенного ребра с сохранением ребра.

$$U2 = \{u_j(X1_j) \cup u_z(X1_z \setminus x_t)\}, \ \forall j \in K_j, \quad K_j = \overline{1,M} \setminus z.$$

```
delete from G where ID_X=(select ID_X from X
   where Наименование_вершины=@Наименование_вершины) and
   ID_U=(select ID_U  from U
      where Наименование_ребра=@Наименование_ребра)
```

Корректировку свойств вершин можно записать следующим образом.

$S_k = \{s[n, k, z] = f\_npp(n, k, z)\} \quad n = \overline{1,N}, \ \forall k \in K_{m1}, \ K_{m1} \in \overline{1,I}$, где $f\_npp(n,k,z)$ – функция, позволяющая найти новое значение свойства n вершины k в ребре z. В нотации Transact-SQL корректировка свойств:

```
Update G set NPP_1=dbo.F_NPP( 'NPP_1', ID_X , S1.ID_U) ,
                NPP_2= dbo.F_NPP( 'NPP_2', ID_X , S1.ID_U) ,
                  … ,
                NPP_N=dbo.F_NPP( 'NPP_N', ID_X,  S1.ID_U)
 from G where G.ID_U= (select top 1 ID_U from U
       where Наименование_ребра=@Наименование_ребра)
```

Составное удаление вершины – удаление вершины из ребра вместе с ребром можно записать как:

$$U2 = \{u_j(X1_j)\},\ \ \forall j \in K_j,\ \ K_j = \overline{1,M} \setminus z$$

и в нотации Transact-SQL как:

```
delete from G where ID_U=(select ID_U from U where
    Наименование_ребра=@Наименование_ребра)
```

Так как ребро удаляется, то корректировку свойств его вершин проводить не надо.

Слабое удаление вершины – удаление вершины из всех ребер с сохранением ребер можно записать как:

$$U2 = \{u_j(X1_j) \cup u_r(X1_r \setminus x_t)\},\ \ \forall j \in K_j,\ \ \forall r \in K_r,$$
$$K_j \subseteq \overline{1,M},\ \ K_r \subseteq \overline{1,M},$$

где $K_r = M \setminus K_j$, где K_j – множество индексов ребер, не инцидентных вершине x_t, K_r – множество индексов ребер, инцидентных вершине x_t.

В нотации Transact-SQL слабое удаление вершины:

```
delete from G where ID_X=(select ID_X from X
    where Наименование_вершины=@Наименование_вершины)
```

Удаление вершины из множества X, т.е $G_{gon}(X \setminus x_t, U2)$, можно записать в виде

```
delete from X where Наименование_вершины =
    @Наименование_вершины.
```

Если при этом внешний ключ FR_G_X описать как

```
alter table G add constraint FK_G_X foreign key (ID_X)
references X(ID_X) on delete cascade,
```

то удаление вершины из всех ребер будет сделано при удалении вершины из множества *X*.

Корректировку свойств вершин можно записать следующим образом.

$$S_k = \{s[n, k, m] = f\_npp(n, k, m)\},\ \ n = \overline{1,N},\ \ \forall k \in K_{m1},\ \ K_{m1} \in \overline{1,I},$$
$$\forall m \in K_r,\ \ K_r \subseteq M.$$

```
select @ID_X=ID_X from X where
    Наименование_вершины=@Наименование_вершины
    Update G set NPP_1=dbo.F_NPP( 'NPP_1', ID_X , ID_U),
                 NPP_2=dbo.F_NPP( 'NPP_2', ID_X , ID_U),
                   … ,
                 NPP_N=dbo.F_NPP( 'NPP_N', ID_X, ID_U)
     from G where G.ID_U in  (select ID_U from G
         where ID_X=@ID_X)
```

Сильное удаление вершины – удаление выршины из всех ребер вместе с ребрами.

$$U2 = \{u_j(X1_j)\}, \quad \forall j \in K_j, \ \forall r \in K_r, \ K_j \subseteq \overline{1,M} \setminus K_r, \ K_r \subseteq \overline{1,M},$$

где K_j – множество индексов ребер, не инцидентных вершине x_t, K_r – множество индексов ребер, инцидентных вершине x_t.

```
select @ID_X=ID_X from X where
    Наименование_вершины=@Наименование_вершины
    delete from G where ID_U in (select ID_U from G
            where ID_X=@ID_X).
```

Так как ребра удаляются, то корректировку свойств их вершин производить не надо.

2.4. УДАЛЕНИЕ РЕБРА

По аналогии с удалением вершины будем различать следующие операции удаления ребер:

– простое удаление ребра – удаление ребра из множества G_{gon};

– составное удаление – удаленние ребра из множества G_{gon} и удаление всех вершин удаляемого ребра из других ребер;

– жесткое удаление – удаление ребра из множества G_{gon} и удаление всех ребер, в которые входят вершины удаляемого ребра.

При этом возможно как сохранение, так и удаленеие ребра и соответствующих вершин из множеств U и X.

Обозначим u_t – удаляемое ребро $t \in \overline{1,M}$, $X1_t$ – множество вершин ребра u_t, $U2$, $X2$, $G2_{gon}(X2, U2)$ – множество ребер, множество вершин и граф, получаемые после удаления ребра u_t.

Простое удаление ребра – удаление ребра из множества G_{gon} $G2_{gon}(X2, U \setminus u_t)$, $U2 = U \setminus u_t$ – при удалении u_t из множества U или $U2 = U$ – при сохранении, $X2 = X$ – при сохранении вершин удаляемого ребра в множестве X или $X2 = X \setminus X1_t$ – при удалении вершин ребра u_t из множества X.

При удалении вершин удаляемого ребра из множества X, для обеспечения целостности базы данных необходимо удалить эти вершины и из всех ребер, в которые они входят, т.е. $U2 = \{u_j(X1_j \setminus X3_j)\}$, где $X3_j = X1_j \cap X1_t$, $X3_j \neq \varnothing$, $\forall j \in \overline{1,M}$. В этом случае необходимо также провести корректировку свойств оставшихся вершин. Обозначим K_t – множество индексов вершин удаляемого ребра. Тогда корректировку свойств оставшихся вершин можно записать следующим образом:

$$S_k = \{s[n, k, j] = f\_npp(n, k, j)\}, \quad n = \overline{1,N}, \ \forall k \in \overline{1,I} \setminus K_t, \ \forall j \in \overline{1,M},$$
$$X3_j \neq \varnothing, \quad X3_j = X1_j \cap X1_t .$$

Обозначим X3={ID_X3, ID_X, NPP_1, …, NPP_N} – таблица, содержащая ID_X вершин и свойства вершин удаляемого ребра, @ID_U – ID_U удаляемого ребра.

```
select @ID_U=ID_U from U where
    Наименование_ребра=@Наименование_ребра
    /* Запоминание множества вершин удаляемого ребра */
select  ID_X, NPP_1, …, NPP_N into X3 from G where ID_U=@ID_U
    /* Удаление ребра */
delete from G where ID_U=@ID_U
```

При удалении вершин удаляемого ребра из множества X, для обеспечения целостности базы данных необходимо удалить эти вершны из всех ребер гиперграфа и, соответственно, произвести корректировку свойств оставшихся в этих ребрах вершин. Причем сначала надо произвести корректировку вершин, а потом их удаление. В этом случае простое удаление трансформируется в составное удаление.

```
    /* Корректировка свойств вершин */
update G set NPP_1=dbo.F_NPP( 'NPP_1', ID_X , ID_U) ,
             NPP_2=dbo.F_NPP( 'NPP_1', ID_X , ID_U) ,
               … ,
             NPP_N=dbo.F_NPP( 'NPP_N', ID_X,  ID_U)
from G, X3  where ID_U= (select ID_U from G where
    ID_X in (Select ID_X from X3)) and ID_U<>@ID_U
delete from X where ID_X in (select ID_X from X3)
```

Если внешний ключ FR_G_X описать как

```
alter table G  add  constraint FK_G_X foreign key (ID_X)
references X(ID_X) on delete cascade,
```

то удаление вершин из всех ребер будет сделано при удалении вершины из множества *X*, в противном случае можно воспользоваться оператором

```
delete from G where ID_U<>@ID_U and ID_X in (select ID_X from X3)
```

Составное удаление ребра – удаленние ребра и удаление всех вершин удаляемого ребра из других ребер. Составное удаление ребра – это простое удаление с удалением вершин ребра из множества *X*, которое описано выше.

Жесткое удаление ребра – удаление ребра и удаление всех ребер, в которые входят вершины удаляемого ребра, сводится к получению гиперграфа

$$G2_{gon}(X2, U \setminus U3),$$

где $X2 = X$ – при сохранении вершин удаляемого ребра в множестве X или $X2 = X \setminus X1_t$ – при удалении вершин ребра u_t из множества *X*, $U3$ – множество ребер, в которые входят вершины удаляемого ребра,

$$U3 = \{u_j(X1_j)\}, \quad X1_j \cap X1_t \neq \varnothing, \quad \forall j \in \overline{1,M}.$$

Так как ребра, содержащие вершины удаляемого ребра, удаляются, то производить корректировку свойств их вершин не надо.

Обозначим X3={ID_X3, ID_X} – таблица, содержащая ID_X вершин удаляемого ребра, @ID_U – ID_U удаляемого ребра, тогда жесткое удаление ребра можно записать как

```
select @ID_U=ID_U from U where
    Наименование_ребра=@Наименование_ребра
select distinct  ID_X into X3 from G where ID_U=@ID_U
delete from G where ID_U in (select ID_U from G
    where ID_X in (select ID_X from X3))
```

3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ РАЗМЕЩЕНИЯ ОБЪЕКТОВ В ПРОСТРАНСТВЕ, ИНВАРИАНТНОЕ К ПРЕДМЕТНОЙ ОБЛАСТИ

Решение разнообразных задач размещения (компоновки) объектов в пространстве часто требуется в различных областях человеческой деятельности. К таким задачам относятся, например, планировка дачного участка (где и какие деревья и кустарники посадить), создание проекта городского строительства (где и какие здания строить), создание проекта цеха химического или машиностроительного предприятия (где и какие аппараты или станки надо разместить).

Общим в перечисленных задачах является то, что имеется набор сущностей (объектов), которые надо разместить (деревья, здания, аппараты, станки) и имеется набор сущностей (объектов), в которых размещаются первые (территория дачного участка или города, помещения химического или машиностроительного предприятия).

Размещаемые объекты и объекты, в которых происходит размещение, обладают определенными характеристиками или свойствами (размеры, назначение, категория).

Объект считается размещенным, если однозначно определено его положение в пространстве. Это могут быть координаты некоторой характерной точки объекта, например, координаты центра ствола дерева на земле или координаты диагональных углов длинного объекта.

На возможные координаты размещаемых объектов накладываются ограничения, например, минимальное расстояние от ствола высокорослого дерева до границы соседнего участка на даче 4 м, расстояние между домами в строительстве при наличии окон в противоположно расположенных зданиях не может быть меньше 15 м, тяжелое оборудование надо размещать на нижних этажах и др. Эти ограничения содержатся в нормативных документах: СНиП (сборники нормативов и правил), ПБ (правила проектирования безопасных технических объектов) и др., которые определены в каждой предметной области.

При наличии опыта, человек (проектировщик), занимающийся размещением, может дополнительно накладывать свои ограничения, которых нет в нормативных документах. Кроме того, существует по крайней мере одно ограничение общего характера – это непересечение размещаемых объектов друг с другом.

В случае если существует много вариантов размещения, то выбор наилучшего осуществляется с использованием некоторого критерия предпочтения (критерия оптимизации), например, максимум освещенной

солнцем поверхности под огород, минимум стоимости монтажа оборудования, минимум стоимости соединительных трубопроводов и др.

В работах [5, 6] представлены аналитические и процедурные модели, позволяющие находить оптимальные компоновочные решения оборудования химических производств. В качестве критерия оптимальности предлагается использовать критерий приведенных затрат, включающий в себя капитальные и эксплуатационные затраты, зависящие от принимаемых компоновочных решений (затраты на строительные конструкции, монтаж оборудования, трубопроводную арматуру, электроэнергию, затрачиваемую на транспортировку веществ и др.).

В настоящее время существует ряд программных средств, предназначенных для:

– ландшафтного дизайна: Наш Сад 9.0 Рубин (Dicomp), Sierra Land Designer 3D 7.0 (ActiVision), Home and Landscape Design (PUNCH!);

– автоматизации инженерного проектирования объектов промышленности PDS (Intergraph), PDMS (AVEVA), CADPIPE (Orange Technologies), CADWORX (COADE), AutoPLANT (Bentley), PLANT-4D (CEA Technology), ABS (Autodesk);

– универсальные программы, например SketchUp (Google).

Эти программы обладают мощным графическим интерфейсом, множеством инструментов для ручного проектирования, базами данных элементов (деревья, строительные конструкции, арматура). Однако в них отсутствуют ограничения, которые определены нормативными документами или отсутствует возможность проектировщику самостоятельно добавлять новые ограничения, что не позволяет решать оптимизационные задачи размещения в автоматическом (или с минимальным участием человека) режиме.

Между тем, принципиальная общность перечисленных задач размещения позволяет создать такую автоматизированную систему, которая может настраиваться пользователем на конкретную предметную область: ландшафтный дизайн, градостроительство или проектирование промышленных предприятий. Основным компонентом этой системы является база ограничений.

Цель настоящего раздела – разработка инвариантных к предметной области:

– представлений ограничений в задачах размещения на языке SQL;

– структуры базы для хранения ограничений;

– способов обработки базы данных с целью ответа на запросы о соблюдении ограничений предметной области при размещении объектов.

Область применения – разработка программного обеспечения автоматизированных систем, предназначенных для решения компоновочных задач промышленности, градостроительства, ландшафтного дизайна и др.

3.1. СТРУКТУРА БАЗЫ ДАННЫХ ДЛЯ ПРЕДСТАВЛЕНИЯ РАЗМЕЩЕНИЯ ОБЪЕКТОВ

В качестве программно-независимого способа представления информационных объектов выбран N-ориентированный гиперграф. Ограничения представляются продукционными правилами вида "Если А, то В". В качестве базового программного обеспечения использована реляционная база данных. Продукции (правила) представлены теоретико-множественным описанием и описанием на языке структурированных запросов Transact-SQL (релиз SQL фирмы Microsoft).

Обозначим N-ориентированный гиперграф [21] $G(X,U)$, где $X=\{x_i\}, i=\overline{1,I}$ – множество вершин гиперграфа, x_i – *i*-я вершина; $U=\{u_m(X1_m)\}$, $m=\overline{1,M}$ – множество ребер гиперграфа, $u_m(X1_m)$ – *m*-е ребро гиперграфа, $X1_m$ – множество вершин инцидентных *m*-му ребру $X1_m \subseteq X$, $X1_m=\{x_k^{\overline{L}}\}$, $\forall k \in K_m$, $K_m \subseteq \overline{1,I}$, $\overline{L}$ – номер вершины в ребре ориентированного гиперграфа представляет собой вектор, $\overline{L}=\{l_n\}$, $n=\overline{1,N}$, мощность которого N. В общем случае номер вершины в ребре не обязательно должен иметь значение номер по порядку и может отражать определенное свойство вершины, которое принимает конкретное значение при включении вершины в ребро. При этом предполагается, что все вершины в ребрах имеют одинаковый набор свойств. Например, при решении задачи размещения элементов в пространстве, элементы имеют одинаковые наборы свойств, а именно, координаты элементов.

Требование одинакового набора свойств элементов существенно сужает область прикладных задач. Каждый элемент реальной технической системы обладает собственным набором свойств, который отличается от наборов свойств других элементов, например, каждый размещаемый аппарат имеет собственный набор размеров (для горизонтального емкостного аппарата – это диаметр и длина, для вертикального – диаметр и высота):

Обозначим $S=\{s_j\}$, $j=\overline{1,J}$ – множество всех возможных свойств вершин и ребер, $SX_i=\{s_{r1}\} \subset S$, $i=\overline{1,I}$, $r1 \in J1_i \subset \overline{1,J}$ – множество свойств *i*-й вершины, $SU_m=\{s_{r2}\} \subset S$, $m=\overline{1,M}$, $r2 \in J2_m \subset \overline{1,J}$ – множество свойств *m*-го ребра. Под свойством здесь понимается контейнер для хранения значения свойства. Например, свойство вес аппарата может иметь значение 25 000 кг. Таким образом, для каждой вершины множество ее номеров $\overline{L}=\{l_n\}$, $n=\overline{1,N}$ заменяется множеством свойств SX_i, $i=\overline{1,I}$. Кроме того, каждому ребру $u_m(X1_m)$, также ставится в соответствие свой набор свойств SU_m, $m=\overline{1,M}$.

Каждое свойство может иметь значения из определенного множества, например масса аппарата – это числовые значения больше нуля, группа аппарата в зависимости от расчетного давления, температуры стенки и характера среды принимает значения 1, 2, 3, 4, 5а, 5б (ПБ 03-584–03 Правила проектирования, изготовления и приемки сосудов и аппаратов стальных сварных), а категория помещения по взрывопожарной и пожарной опасности – А, Б, В1, В2, В3, В4, Г, Д (Нормы пожарной безопасности НПБ 105–03). Обозначим $Z_j = \{z_{j,t}\}$, $t = \overline{1,T_j}$ – множество возможных значений свойства s_j.

Обозначим $z[s_j, x_i] \in Z_j$ – значение свойства s_j вершины x_i. Значение свойства s_j ребра u_m обозначим $z[s_j, u_m] \in Z_j$. Введем Θ – логический оператор, один из элементов множества $\Omega = \{=, \neq, <, >, \leq, \geq\}$, $\Theta \in \Omega$. Запись $z[s_{j1}, x_{i1}] = z_{j1,t1}$ обозначает, что значение свойства s_{j1} для вершины x_{i1} равно $z_{j1,t1}$ (или принимает значение $z_{j1,t1}$). Запись $z[s_{j1}, x_{i1}] \Theta z_{j1,t1}$ означает, что значение свойства s_{j1} для вершины x_{i1} находится в определенном отношении со значением $z_{j1,t1}$, причем это отношение ограничено элементами множества $\Omega = \{=, \neq, <, >, \leq, \geq\}$.

Здесь и далее в качестве примера используется компоновка химического оборудования, только потому, что эта предметная область более известна авторам, чем, например, градостроительство. Кроме того, предполагается, что размещение аппаратов осуществляются в два этапа [5, 6]. На первом этапе определяется этаж или помещение, в котором будет размещен аппарат, на втором – координаты аппарата на этаже.

Подобная двухуровневая декомпозиция исходной задачи не сужает область применения, так как и в других предметных областях она присутствует, например, разделение участка на сад и огород или группировка станков по отделениям (токарное, фрезерное). Теоретически можно предложить N-уровневую декомпозицию, однако для реальных практических задач двухуровневой декомпозиции вполне достаточно.

Структура базы данных для хранения описанного выше N-ориентированного гиперграфа представлена на рис. 3.1.

Имена таблиц соответствуют именам введенных ранее множеств X, U, G, S, SX, SU, Z. Текст в именах таблиц после подчеркивания следует рассматривать как комментарий, например, действительное имя таблицы G_(Аппараты_в_помещениях) будет G. Поле Z_Значение_свойства в приведенных ниже примерах обозначено как Z.

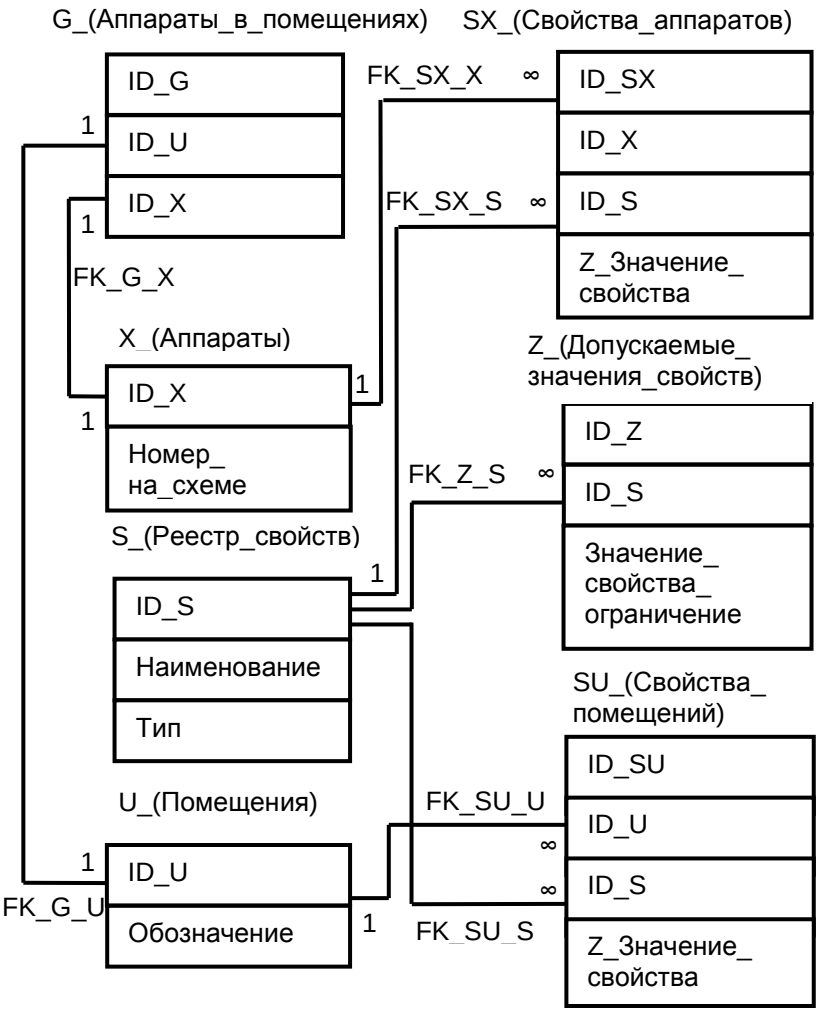


Рис. 3.1. Структурная схема базы данных, предназначенной для размещения объектов в пространстве

Поле S.Тип позволяет различать два типа свойств:

– тип 1. Свойства, которые могут иметь бесконечное множество значений. Они вводятся вручную, например, масса аппарата, объем аппарата, температура среды в аппарате. Таблица Z для них будет содержать ограничение на допустимое значение свойства, например, значение массы аппарата всегда больше нуля;

– тип 2. Свойства, значения которых ограничено конечным списком, например, для свойства "категория помещения" список возможных значений – А, Б, В1, В2, В3, В4, Г, Д. Этот список будет содержаться в таблице Z.

Таблица Z позволяет поддерживать доменную целостность базы. Ссылочная целостность поддерживается первичными ключами по полям X.ID_X, U.ID_U, S.ID_S, Z.ID_Z, SU.ID_SU, SX.ID_SX и внешними ключами, наименование которых на рис. 3.1 имеет префикс FK. Кроме того, для сохранения доменной целостности базы необходимо создать уникальные индексы по полям:

– SX.ID_X, SX.ID_S и SU.ID_U, SU.ID_S, конкретное свойство вершины или ребра присутствует в таблице SX или SU один раз;

– G.ID_U, конкретное ребро в графе может быть один раз;

– G.ID_X, аппарат не может находиться одновременно в разных местах.

Представленная на рис. 3.1 структура базы данных упрощена. Строго говоря, таблицы SX и SU следует дополнить полем ID_Z для хранения первичного ключа свойств типа 2 (поле Z.ID_Z). Это усложнит приведенное ниже изложение, но никак не повлияет на суть предлагаемого подхода.

Свойства вершин и свойства ребер дополнительно можно разделить на следующие группы:

1) свойства вершины, которые не зависят от ее принадлежности ребру (масса аппарата, размеры);

2) свойства ребра, которые не зависят от вершин этого ребра (номер этажа, размеры помещения);

3) свойства ребра, зависящие от вершин этого ребра (категория помещения в зависимости от свойств аппаратов, находящихся в помещении);

4) свойства вершины, зависящие от ее принадлежности ребру (координаты аппарата в конкретном помещении);

5) свойства вершины, зависящие от собственных свойств или свойств других вершин (при размещении однотипных аппаратов в ряд, после установки первого аппарата определена одна из координат (x или y) всех остальных аппаратов ряда;

6) свойства ребра, зависящие от собственных свойств, или свойств других ребер.

Свойства первой и второй группы являются исходными данными и вводятся до начала процесса размещения. Свойства третьей, четвертой, пятой и шестой групп определяются в процессе размещения.

Представленная классификация свойств необходима для дальнейшей программной реализации способов их определения. Предполагается, что

свойства третьей, четвертой, пятой и шестой групп в начальный момент содержатся в таблицах SX и SU и имеют значения NULL. То есть в процессе решения задачи эти свойства не добавляются, а редактируются, точнее редактируется их значение (поле Z_Значение_свойства).

3.2. ФУНКЦИИ БАЗЫ ОГРАНИЧЕНИЙ

Под базой ограничений здесь понимается программно-информационный модуль, который в процессе решения задачи размещения должен отвечать на определенные запросы, поступающие из внешней программы, которая решает задачу размещения. Ответ на запросы база ограничений осуществляет, используя хранящуюся в ней информацию. Структура этой информации инвариантна предметной области. Смысловое содержание определяется предметной областью.

В качестве "внешней программы" может выступать и проектировщик, который расставляет объекты "вручную". База ограничений должна предостеречь его о возможных ошибках. Например, если проектировщик разместил тяжелый аппарат не на первом этаже, база ограничений должна сообщить ему, что здесь аппарат размещать нельзя, так как существует правило, согласно которому тяжелые аппараты следует размещать на первом этаже.

Поскольку осуществлена двухуровневая декомпозиция исходной задачи, возможные запросы управляющей программы и ответы базы ограничений также классифицируем по двум уровням. Подобная классификация необходима для представления возможных запросов в SQL-нотации. Возможны следующие запросы управляющей программы на первом уровне:

– тип 1. Исходные данные – помещение и аппарат. Ответ – можно или нельзя размещать аппарат в помещении;

– тип 2. Исходные данные – аппарат. Ответ – список помещений, куда ставить аппарат можно;

– тип 3. Исходные данные – помещение. Ответ – список аппаратов, которые ставить в этом помещении можно;

– тип 4. Исходные данные – аппарат. Ответ – список помещений, куда ставить аппарат нельзя;

– тип 5. Исходные данные – помещение. Ответ – список аппаратов, которые ставить в этом помещении нельзя.

Ответы на перечисленные запросы выдаются на основании правил (ограничений), которые связывают свойства вершин и свойства ребер. Часть этих свойств не зависит от решаемой задачи (свойства группы 1 и 2), другие (группы 3 – 6) изменяются или определяются в процессе решения задачи. Таким образом, кроме ответа на перечисленные выше запросы, база ограничений должна "уметь" изменять (или определять) свойства вершин и ребер в зависимости от своего текущего состояния, полу-

чаемого в процессе размещения. Текущее состояние определяется значениями полей G.ID_U, G.ID_X (принадлежность аппаратов помещениям), SX.ID_S, SX.Z (свойства аппаратов), SU.ID_S, SU.Z (свойства помещений) на определенном итерационном этапе решения задачи размещения.

Ниже рассматриваются только правила первого уровня. Решение задачи второго уровня (размещение аппаратов на этаже) можно свести к задаче первого уровня, если все пространство этажа разделить на зоны (ребра) со своими свойствами. Например, для задачи размещения технологического оборудования такими зонами являются типовые строительные клетки размером 6х6 метров (участок между колоннами).

3.3. ЭЛЕМЕНТАРНЫЕ ОГРАНИЧЕНИЯ И ИХ ПРЕДСТАВЛЕНИЕ В SQL-НОТАЦИИ

Введем понятия ограничения на значение свойства s_{j1} вершины x_{i1} (например, ограничение на массу аппарата) и ограничения на значение свойства s_{j2} ребра u_{m1} (например, ограничение на номер этажа помещения).

Ограничение на значение свойства вершины.

$$z[s_{j1}, x_{i1}]\Theta z_{j1,t1},\ \ z_{j1,t1} \in Z_j,\ \ j1 \in J1 \subset J,\ \ i1 \in I,\ \ t1 \in \overline{1,T_{j1}}. \qquad (3.1)$$

Запишем проверку ограничения (3.1) для заданной вершины в SQL-нотации. Будем считать, что заданы @ID_X (вершина), @ID_S (свойство вершины) и @z (значение свойства), которые соответствуют $x_{i1}, j1, z_{j1,t1}$ в выражении (3.1).

```
exists (select * from SX
    where SX.ID_X=@ID_X  and SX.ID_S=@ID_S
    and SX.Z  Θ @z1)                                    (3.2)
```

Запишем выбор всех вершин, удовлетворяющих условию (3.1).

```
select * from X where exists (select * from SX
        where X.ID_X=SX.ID_X  and SX.ID_S=@ID_S
        and SX.Z  Θ @z)                                 (3.3)
```

Введем понятие элементарного ограничения на свойства вершины Ox1(x), такое что

```
Ox1(@ID_X) ≡  exists (select * from SX where SX.ID_X=@ID_X
                    and SX.ID_S=@ID_S and SX.Z  Θ  @z),
```

Ox1(X.ID_X) ≡ exists (select * from SX where SX.ID_X=X.ID_X
and SX.ID_S=@ID_S and SX.Z Θ @z).

С учетом введенного Ox1(x), выражение (3.2) запишется как Ox1(@ID_X), выражение (3.3) как select * from X where Ox1(X.ID_X).

Ограничение на значение свойства ребра

$$z[s_{j2}, u_{m1}]\Theta z_{j2,t2},\ \ z_{j2,t2} \in Z_{j2},\ \ j2 \in J2 \subset J,\ \ m1 \in M,\ \ t2 \in \overline{1,T_{j2}}. \quad (3.4)$$

Запишем проверку ограничения (3.4) для заданной вершины в нотации SQL. Заданы @ID_U, @ID_S, @z2, которые соответствуют $m1, j2, z_{j1,t1}$ в выражении (3.4).

```
exists (select * from  SU
    where SU. ID_U=@ID_U and SU.ID_S=@ID_S                    (3.5)
        and SU.Z Θ @z)
```

Запишем выбор всех ребер, удовлетворяющих условию (3.4):

```
select * from U where exists (select * from SU
    where U.ID_U=SU.ID_U and SU.ID_S=@ID_S                    (3.6)
        and SU.Z Θ @z)
```

По аналогии с Ox1(x) введем обозначение Ou1(u), такое что:

```
Ou1(@ID_U) ≡ exists (select * from SU where U.ID_U=@ID_U
                and SU.ID_S=@ID_S and SU.Z Θ @z),
Ou1(U.ID_U) ≡ exists (select * from SU where SU.ID_U=U.ID_U
                and SU.ID_S=@ID_S and SU.Z Θ @z)
```

С учетом введенного Ou1(u), выражения (3.5) и (3.6) запишутся как Ou1(@ID_U) и select * from U where Ou1(U.ID_U).

Комбинации ограничений. Введем функцию $F1(z[s_{j1}, x_{i1}]\,\Theta_{r1}\, z_{j1,t1})$, соединяющую логическим "И" несколько элементарных ограничений одной вершины:

$$F1(z[s_{j1}, x_{i1}]\,\Theta_{r1}\, z_{j1,t1}) = \bigcap_{r1=1}^{R1}(z[s_{j1}, x_{i1}]\Theta_{r1}\, z_{j1,t1}),\quad j1 \in J1 \subset J,\quad i1 \in I,$$

$$r1 = \overline{1,R1},\quad z_{j1,t1} \in Z_{j1},\quad t1 \in \overline{1,T_{j1}}, \quad (3.7)$$

где $J1$ – множество свойств соединяемых ограничений, $R1$ – количество соединяемых ограничений. Для упрощения в дальнейшем $F1(z[s_{j1}, x_{i1}]\,\Theta_{r1}\,z_{j1,t1})$ будем обозначать $F1()$.

Рассмотрим пример комбинации двух ограничений вершины – "среда в аппарате взрывоопасна и токсична". Для удобного представления в базе данных запишем эти ограничения в виде "свойство вершины взрывоопасность среды=Да И свойство вершины токсичность среды=Да".

Пусть для свойств "взрывоопасность среды в аппарате" ID_S=10, "токсичность среды в аппарате" ID_S=11, тогда в выражении (3.7) $J1=\{10, 11\}$, $R1=2$, Θ_1 и Θ_2 соответствуют знаку =. Таким образом, для рассматриваемого примера

$$F1() = (z[s_{10}, x_{i1}] = "Да")\ \ И\ (z[s_{11}, x_{i1}] = "Да"),\ i1 \in I. \qquad (3.8)$$

Запишем проверку условия (3.8) для аппарата @ID_X в SQL-нотации

OOx(@ID_X) ≡ Ox1(@ID_X) and Ox2(@ID_X),

где OOx(@ID_X) – комбинация ограничений вершины,

Ox1(@ID_X) ≡ exists (select * from SX where SX.ID_X=@ID_X and SX.ID_S=10 and SX.Z="Да"),

Ox2(@ID_X) ≡ exists (select * from SX where SX.ID_X=@ID_X and SX.ID_S=11 and SX.Z="Да").

Поиск всех аппаратов, удовлетворяющих условию (3.8) запишется как

select * from X where OOx(X.ID_X),

где OOx(X.ID_X) ≡ Ox1(X.ID_X) and Ox2(X.ID_X),

Ox1(X.ID_X) ≡ exists (select * from SX where SX.ID_X=X.ID_X and SX.ID_S=10 and SX.Z="Да"),

Ox2(ID_X) ≡ exists (select * from SX where SX.ID_X=X.ID_X and SX.ID_S=11 and SX.Z="Да").

В общем случае $F1()$, $j1 \in J1 \subset J$, $i1 \in I$, $r1 = \overline{1, R1}$, в SQL-нотации запишется в виде

OOx1(x) ≡ Ox1(x) and Ox2(x) … and Oxr(x) … and OxR1(x) (3.9)

По аналогии с $F1()$ введем функцию $F2() = F2(z[s_{j2}, u_{m1}]\,\Theta_{r2}\,z_{j2})$, соединяющую логическим "И" несколько элементарных ограничений одного ребра:

$$F2() = \bigcap_{r2=1}^{R2} (z[s_{j2}, u_{m1}] \Theta_{r2} z_{j2,t2}), \quad j2 \in J2 \subset J, \quad m1 \in M, \quad r2 = \overline{1, R2},$$

$$z_{j2,t2} \in Z_{j2}, \quad t2 \in \overline{1, T_{j2}}, \tag{3.10}$$

где $J2$ – множество свойств соединяемых ограничений, $R2$ – количество соединяемых ограничений.

В общем случае $F2()$, $j2 \in J2 \subset J$, $m1 \in M$, $r2 = \overline{1, R2}$, $z_{j2,t2} \in Z_{j2}$, $t2 \in \overline{1, T_{j2}}$ в SQL-нотации запишется в виде

$$\text{OOu1(u)} \equiv \text{Ou1(u) and Ou2(u) … and Our(u)… and OuR2(u)} \tag{3.11}$$

Введем функцию $F3(F1(z[s_{j1}, x_{i1}] \Theta_{r1} z_{j1,t1}))$, соединяющую логическим И группы ограничений разных вершин:

$$F3(F1()) = \bigcap_{\forall i1 \in I1} F1_{i1}(z[s_{j1}, x_{i1}] \Theta_{r1} z_{j1,t1}) = \bigcap_{\forall i1 \in I1} \bigcap_{r1=1}^{R1_{i1}} (z[s_{j1}, x_{i1}] \Theta_{r1} z_{j1,t1})$$

$$j1 \in J1_{i1} \subset J, \ i1 \in I1 \subset I, \ r1 = \overline{1, R1}_{i1}, \ z_{j1,t1} \in Z_{j1}, \ t1 \in \overline{1, T_{j1}}, \tag{3.12}$$

где $I1$ – множество соединяемых вершин, J_{i1} – множество соединяемых свойств вершины $i1$, $R1_{i1}$ – количество соединяемых ограничений вершины $i1$.

Запишем проверку истинности выражения (3.12) в SQL-нотации:

```
OOOx≡ exists (select * from X where OOx1(ID_X)) and
      exists (select * from X where OOx2(ID_X)) and
      …                                                    (3.13)
      exists (select * from X where OOxI1(ID_X))
```

Например, если среда в одном аппарате взрывоопасна, а в другом токсична и для свойства взрывоопасность ID_S=10, для свойства токсичность ID_S=11, $R_{i1} = 1$, $R_{i2} = 1$ выражение (3.12) примет вид

$$F3(F1()) = (z[s_{10}, x_{i1}] = "Да") \ И \ (z[s_{11}, x_{i2}] = "Да"), \ i1 \neq i2, \ i1, \ i2 \in I1 \subset I.$$

В SQL-нотации последнее выражение запишется следующим образом:

```
Ox1(X.ID_X)≡exists (select * from SX where SX.ID_X=X.ID_X
            and SX.ID_S=”10” and SX.Z=”Да”)
Ox2(X.ID_X)≡exists (select * from SX where SX.ID_X=X.ID_X
            and SX.ID_S=”11” and SX.Z=”Да”)
OOx1(X.ID_X)≡Ox1(X.ID_X)
OOx2(X.ID_X)≡Ox2(X.ID_X)
OOOx≡exists (select * from X where OOx1(ID_X)) and
      exists (select * from X where OOx2(ID_X))
```

В реальных практических задачах (по крайней мере в предметной области компоновки химического оборудования) авторам не известны ограничения, которые объединяли бы свойства нескольких помещений), не смотря на это, для общности введем по аналогии с $F3(F1())$ функцию $F4(F2())$, которая логически связывает группы свойств разных ребер:

$$F4(F2()) = \bigcap_{\forall m1 \in M1} F2_{m1}(z[s_{j2}, u_{m1}]\Theta_{r2}\, z_{j2,t2}) = \bigcap_{\forall m1 \in M1} \bigcap_{r2=1}^{R2_{m1}} (z[s_{j1}, u_{m1}]\Theta_{r2}\, z_{j2,t2}) \qquad (3.14)$$

$$j2 \in J2_{m1} \subset J, \quad m1 \in M1 \subset M, \quad r2 = \overline{1, R2}_{m1},$$

$$z_{j2,t2} \in Z_{j2}, \quad t1 \in \overline{1, T_{m1}},$$

где $M1$ – множество соединяемых ребер, J_{m1} – множество соединяемых свойств ребра $m1$, $R2_{m1}$ – количество соединяемых ограничений ребра $m1$.

По аналогии с (3.12) и (3.13) запишем проверку выражения (3.14):

```
OOOu ≡ exists (select * from U where OOu1(ID_U)) and
       exists (select * from U where OOu2(ID_U)) and
       …                                                    (3.15)
       exists (select * from U where OOuM1(ID_U))
```

Выражения Ox1, Ou1, OOx1, OOu1, OOOx, OOOu представляют собой языковые конструкции, которые легко получить программным путем.

3.4. ПРАВИЛА, ОГРАНИЧИВАЮЩИЕ ПРИНАДЛЕЖНОСТЬ ВЕРШИНЫ РЕБРУ

Правила (ограничения) первого уровня декомпозиции, которые определяют принадлежность вершины ребру можно разделить на три группы:

– принадлежность вершины ребру определяется одним свойством вершины и одним свойством ребра;

– принадлежность вершины ребру определяется группой свойств вершины и группой свойств ребра;

– принадлежность одной вершины ребру определяется принадлежностью ребру другой вершины.

Принадлежность вершины x_{i1} ребру u_{m1}, $x_{i1} \in X1_m$ определяется одним свойством s_{j1} вершины x_{i1} и свойством s_{j2} ребра.

Например, если масса аппарата > 100 000 кг, то аппарат следует располагать на первом этаже.

$$z[s_{j1}, x_{i1}]\Theta_1 z_{j1,t1} \wedge x_{i1} \in X1_{m1},\ m1 \in M,\ z_{j1,t1} \in Z_{j1},$$

$$j1 \in J1 \subset J,\ t1 = \overline{1, T_{j1}},\ i1 \in I,\ m1 \in M \Rightarrow$$

$$z[s_{j2}, u_{m1}]\Theta_2 z_{j2,t2},\ j2 \in J2 \subset J,\ z_{j2,t2} \in Z_{j2},\ t2 = \overline{1, T_{j2}}. \qquad (3.16)$$

Представим синтаксис запросов типа 1 – 3, основанных на правилах (3.16) в нотации SQL. Синтаксис запросов типов 4 и 5 достаточно легко получается из запросов типа 2, 3 отрицанием ограничений.

Заданы @ID_S1 – $j1$ свойство вершины, @z1 – $z_{j1,t1}$ значение свойства вершины, @ID_S2 – $j2$ свойство ребра, @z2 – $z_{j2,t2}$ – значение свойства ребра.

Тип 1. Ответ на вопрос можно ли размещать аппарат @ID_X в помещении @ID_U.

Введем переменную @ret = 1, если аппарат в заданном помещении располагать можно, @ret = 0 в противном случае.

```
if Ox1(@ID_X)
   if  Ou1 (@ID_U)
        set @ret=1                                    (3.17)
else set @ret=0
```

Тип 2. Получить все помещения, в которых можно размещать аппарат @ID_X.

```
If Ox1(@ID_X)  select ID_U from U where Ou1(U.ID_U)   (3.18)
```

Тип 3. Получить список аппаратов, которые необходимо ставить в помещении @ID_U.

```
if Ou1(@ID_U) select ID_X from X where Ox1(ID_X)      (3.19)
```

Представленное разделение запросов на типы позволяет ввести правило один раз и автоматически сформировать рассмотренные конструкции запросов. Рассмотрим это на следующем примере.

Правило "Если масса аппарата > 100 000, то аппарат надо располагать на первом этаже". В нотации удобной для ввода в базу это правило выглядит следующим образом: "Если свойство аппарата = "масса" и значение свойства > 100 000, то свойство помещения = "этаж" и значение свойства = "первый".

Предположим, что ID_S=6 для свойства "масса", ID_S=7 для свойства "этаж". Программно не трудно сформировать проверку элементарных ограничений.

```
Ox1(@ID_X) ≡ exists (select * from SX where SX.ID_X=@ID_X
                  and SX.ID_S=6  and SX.Z>10000),
```

Ox1(SX.ID_X) ≡ exists (select * from SX where SX.ID_X=X.ID_X
and SX.ID_S=6 and SX.Z>10000),
Ou1(@ID_U) ≡ exists (select * from SU where SU.ID_U=@ID_U
and SU.ID_S=7 and SU.Z="первый"),
Ou1(U.ID_U) ≡ exists (select * from SU where SU.ID_U=U.ID_U
and SU.ID_S=7 and SU.Z="первый")

Подставив полученные выражения в языковые конструкции (3.17) – (3.19), получаем синтаксис запросов, основанных на одном правиле – "Если масса аппарата > 100 000, то аппарат надо располагать на первом этаже".

Принадлежность вершины x_{i1} *ребру* $u_{m1}, x_{i1} \in X1_m$ *определяется группой свойств вершины* $\{s_{j1}\} \subset S,\ j1 \in J1 \subset J,$ *соединенных логическими условиями "И" и группой свойств ребра* $\{s_{j2}\} \subset S,\ j2 \in J2 \subset J,$ *соединенных логическими условиями "И"*.

Пример правила: "Если среда в аппарате взрывоопасна и токсична, то помещение должно быть оборудовано принудительной вентиляцией и иметь эвакуационный выход".

$$F1(z[s_{j1},\ x_{i1}]\Theta_{r1}\, z_{j1,t1}) \wedge x_{i1} \in X1_{m1},\ m1 \in M\ ,\ z_{j1,t1} \in Z_{j1},$$
$$j1 \in J1 \subset J,\ t1 = \overline{1, T_{j1}},\ i1 \in I,\ r1 = \overline{1, R1},\ m1 \in M \Rightarrow \qquad (3.20)$$
$$F2(z[s_{j2},\ u_{m1}]\Theta_{r2}\, z_{j2,t2}),\ z_{j2,t2} \in Z_{j2},\ j2 \in J2 \subset J,$$
$$t2 = \overline{1, T_{j2}},\ r2 = \overline{1, R2}\ .$$

Проверка истинности логических функций $F1()$ и $F2()$ в нотации SQL запишется конструкциями OOx1(x) и OOu1(u), выражения (3.9) и (3.11).

SQL-выражения для всех типов запросов первого уровня декомпозиции (тип 1 – 3) легко получить из (3.17) – (3.19) заменой конструкций Ox1(X.ID_X), Ox1(@ID_X), Ou1(U.ID_U), Ou1(@ID_U) на OOx1(X.ID_X), OOx1(@ID_X), OOu1(U.ID_U), OOu1(@ID_U).

Принадлежность вершины x_{i1} *ребру* u_{m1}, $x_{i1} \in X1_{m1}$ *определяется принадлежностью этому ребру другой вершины* x_2, $x_2 \in X1_{m1}$. Вершина x_{i1} обладает свойствами $\{s_{j1}\} \subset S,\ j1 \in J1 \subset J,$ вершина x_2 – свойствами $\{s_{j2}\} \subset S,\ j2 \in J2 \subset J,$ для которых выполняются ограничения типа $F1()$, выражение (3.7). Пример правила: "Токарные станки следует располагать в одном помещении". По другому это правило можно сформули-

ровать так: "Если имеется станок, тип которого токарный, и он расположен в помещении u_{m1}, то другие станки с типом токарный следует располагать в помещении u_{m1}". Функция $F1()$ в этом случает ограничивает свойство "тип станка" значением "токарный".

$$F1(z[s_{j1}, x_{i1}]\Theta_{r1}\, z_{j1,t1}) \wedge F1(z[s_{j2}, x_{i2}]\Theta_{r2}\, z_{j2,t2}) \wedge (x_{i2} \in X_{m1}),$$

$$z_{j1,t1} \in Z_j, \quad j1 \in J1 \subset J, \quad t1 = \overline{1,T_{j1}}, \quad i1 \in I, \quad z_{j2,t2} \in Z_{j2}, \qquad (3.21)$$

$$j2 \in J2 \subset J, \quad t2 = \overline{1,T_{j2}}, \quad i2 \in I, \quad m1 \in M \Rightarrow x_{i1} \in X_{m1}.$$

Представим синтаксис всех запросов, основанных на правилах (3.21) в нотации SQL.

Тип 1. Ответ на вопрос можно ли размещать аппарат @ID_X в помещении @ID_U. Введем переменную @ret = 1, если аппарат в заданном помещении располагать можно, @ret = 0 в противном случае.

```
if Ox1(@ID_X)
    if exists (select * from G,X
    where G.ID_X=X.ID_X and G.ID_U=@ID_U and  Ox1(X.ID_X))
        set @ret=1
     else set @ret=0
```

Тип 2. Получить все помещения, в которых можно размещать аппарат @ID_X.

```
if Ox1(@ID_X)
    select U.ID_U from U where exists (select G.ID_U from G, X
        where G.ID_X=X.ID_X and X.ID_X<>@ID_X
        and U.ID_U=G.ID_U and OX1(X.ID_X))
```

Тип 3. Получить список аппаратов, которые необходимо ставить в помещении @ID_U.

```
select X1.ID_X from X, X1 where Ox1(X1.ID_X) and
    exists (select G.ID_U from G, X where G.ID_U=@ID_U
            and X.ID_X=G.ID_X and Ox1(X.ID_X))
```

3.5. ПРАВИЛА, ОПРЕДЕЛЯЮЩИЕ СВОЙСТВА РЕБРА В ЗАВИСИМОСТИ ОТ СВОЙСТВ ВЕРШИНЫ

Это правила, позволяющие по известным определенным свойствам вершины x_{i1}, принадлежащей ребру u_{m1}, найти определенные свойства этого ребра. Например, "Если взрывоопасный аппарат находится в помещении, то помещение категории А".

В практических задачах существуют следующие группы правил, определяющих свойства ребра в зависимости от свойств вершин:

Свойство s_{j2} *ребра* u_{m1} *определяется одним свойством* s_{j1} *вершины* x_{i1} *этого ребра* $x_{i1} \in X1_{m1}$:

$$z[s_{j1}, x_{i1}]\Theta z_{j1,t1} \wedge x_{i1} \in X1_{m1},\ m1 \in M,\ z_{j1,t1} \in Z_{j1},\quad j1 \in J1 \subset J,$$

$$i1 \in I,\ t1 \in \overline{1,T_{j1}} \Rightarrow z[s_{j2}, u_{m1}] = z_{j2,t2},\quad z_{j2,t2} \in Z_{j2},$$

$$j2 \in J2 \subset J,\quad t2 \in \overline{1,T_{j2}}.$$

SQL-оператор изменения свойств ребер запишется следующим образом. Заданы @ID_S1 – $j1$ свойство вершины, @z1 – $z_{j1,t1}$ значение свойства вершины, @ID_S2 – $j2$ свойство ребра, @z2 – $z_{j2,t2}$ – значение свойства ребра. Необходимо во всех ребрах, которые имеют свойство @ID_S2, изменить значение этого свойства на @z2, если ребра содержат вершины, для которых справедливо элементарное ограничение Ox1(X.ID_X):

```
update SU set Z=@z2 where ID_S=@IZ_S2
        and ID_U in (select G.ID_U from G, X                    (3.22)
where X.ID_X=G.ID_X and Ox1(X.ID_X))
```

Свойство s_{j2} *ребра* u_{m1} *определяется группой свойств* $\{s_{j1}\} \subset S$, $j1 \in J1 \subset J$ *вершины* x_{i1}.

$$F1(z[s_{j1}, x_{i1}]\Theta_{r1} z_{j1,t1}) \wedge x_{i1} \in X1_{m1},\ m1 \in M,\ z_{j1,t1} \in Z_j,$$

$$\forall j1 \in J1 \subset J,\quad i1 \in I,\quad t1 \in \overline{1,T_j},\quad r1 = \overline{1,R1} \Rightarrow$$

$$z[s_{j2}, u_{m1}] = z_{j2,t2},\quad z_{j2,t2} \in Z_j,\quad j2 \in J,\quad t2 \in \overline{1,T_j},$$

где $F1(z[s_{j1}, x_{i1}]\Theta_{r1} z_{j1,t1}) = \bigcap_{r1=1}^{R1}(z[s_{j1}, x_{i1}]\Theta_{r1} z_{j1,t1})$ – логическая функция, связывающая значения свойств вершины, $R1$ – количество связанных свойств.

В нотации SQL корректировка свойства ребра в этом случае получается из выражения (3.2.2), в котором Ox1(ID_X) заменяется на OOx1(ID_X), выражение (3.9).

Свойство s_{j2} *ребра* u_{m1} *определяется группой свойств* $\{s_{j1}\} \subset S$, $j1 \in J1 \subset J$ *группы вершин* $X1 = \{x_{i1}\}$, $i1 \in I1 \subset I$ *этого ребра* (т.е. свойства могут принадлежать разным вершинам ребра):

$$\bigcap_{\forall i1 \in I1} F_{i1}(z[s_{j1}, x_{i1}]\Theta_{i1,r1} z_{j1,t1}) \wedge \bigcap_{\forall i1 \in I1} x_{i1} \in X1_{m1},\ m1 \in M,\ z_{j1,t} \in Z_{j1},$$

$$\forall j1 \in J1_{i1} \subset J,\quad \forall i1 \in I1 \subset I,\quad r1 = \overline{1, R1_{i1}},\quad t1 = \overline{1, T1_{i1}} \Rightarrow$$

$$z[s_{j2}, u_{m1}] = z_{j2,t2},\quad z_{j2,t2} \in Z_{j2},\quad j2 \in J1_{m1} \subset J,\quad t2 \in \overline{1, T2}.$$

Корректировка значения свойства ребер в нотации SQL в этом случае получается из выражения (3.22), в котором Ox1(ID_X) заменяется на OOOx, выражение (3.13).

3.6. ПРАВИЛА, ОПРЕДЕЛЯЮЩИЕ СВОЙСТВА ВЕРШИН В ЗАВИСИМОСТИ ОТ СВОЙСТВ РЕБРА

Эти правила позволяют по известным определенным свойствам ребра u_{m1} найти определенные свойства принадлежащих ему вершин $x_{i1} \in X_m$. В практических задачах существуют следующие группы правил, определяющих свойства вершин в зависимости от свойств ребра.

Свойство s_{j1} *вершин* $x_{i1} \in X_m$ *ребра* u_{m1} *определяется одним свойством ребра* s_{j2}:

$$z[s_{j2}, u_{m1}]\Theta z_{j2,t2},\ m1 \in M,\quad z_{j2,t2} \in Z_j,\quad j2 \in J2 \subset J,\quad t2 \in \overline{1, T2} \Rightarrow$$

$$z[s_{j1}, x_{i1}] = z_{j1,t1},\quad z_{j1,t1} \in Z_j,\quad j1 \in J1 \subset J,\quad t1 \in \overline{1, T1},\quad x_{i1} \in X1_{m1}.$$

SQL-оператор изменения свойств ребер запишется следующим образом. Заданы @ID_S2 – $j2$ свойство ребра, @z2 – $z_{j2,t2}$ значение свойства ребра, @ID_S1 – $j1$ свойство вершины, @z1 – $z_{j1,t1}$ – значение свойства вершины. В случае истинности элементарного ограничения Ou1(U.ID_U) необходимо у всех вершин ребра, у которых есть свойство SX.ID_S=@ID_S1, изменить его значение SX.Z=@z1.

```
update SX set Z=@z1 where ID_S=@ID_S1 and ID_X
    in (select G.ID_X (from U, G where U.ID_U=G.ID_U          (3.23)
       and  Ou1(U.ID_U))
```

Свойство s_{j1} *вершин* $x_{i1} \in X_{m1}$ *ребра* u_{m1} *определяется группой свойств* $\{s_{j2}\} \subset S,\ j2 \in J2 \subset J$ *этого ребра*:

$$F2(z[s_{j2}, u_{m1}]\,\Theta_{r2}\, z_{j2,t2}),\ \ m1 \in M,\ \ z_{j2,t2} \in Z_j,\ \ j2 \in J2 \subset J,$$

$$t2 \in \overline{1,T2}\ ,\ r2 = \overline{1,R2} \Rightarrow$$

$$z[s_{j1}, x_{i1}] = z_{j1,t1},\ \ z_{j1,t1} \in Z_j,\ \ j1 \in J1 \subset J,\ \ t1 \in \overline{1,T1},\ \ x_{i1} \in X1_{m1},$$

где $F2(z[s_{j2}, u_{m1}]\Theta_{r2}\, z_{j2,t2}) = \bigcap_{r2=1}^{R2}(z[s_{j2}, u_{m1}]\Theta_{r2}\, z_{j2,t2})$ – логическая функция, связывающая значения свойств ребра, $R2$ – число элементарных ограничений (число свойств ребра, определяющих свойства вершины).

SQL-оператор изменения свойств вершин получается заменой Ou1(U.ID_U) в (3.23) на OOu1(U.ID_U), выражение (3.11).

3.7. ПРАВИЛА, ОПРЕДЕЛЯЮЩИЕ СВОЙСТВА ВЕРШИНЫ В ЗАВИСИМОСТИ ОТ СВОЙСТВ ДРУГИХ ВЕРШИН

Правила, позволяющие найти значение свойства $j3$ одной вершины x_{i3} по известным свойствам $\{s_{j1}\} \subset S,\ j1 \in J1 \subset J$ другой вершины x_{i1}, причем вершины принадлежат одному ребру $x_{i1}, x_{i3} \in X1_{m1}$. В общем случае может быть, что $i1 = i3$, т.е. некоторое свойство вершины определяется через другие ее свойства, тогда определяемое свойство $j3$ не должно принадлежать множеству определяющих свойств $J1$, $j3 \notin J1$. Пример правила: Однотипные аппараты следует располагать в ряд. Это означает, что если аппараты располагаются в ряд по оси y и найдена (предложена) координата x одного аппарата, то и другие аппараты этого типа будут иметь такую же координату x.

Если $i3 = i1,\ j3 \notin J1$

$$F1(z[s_{j1}, x_{i1}]\Theta_{r1}\, z_{j1,t1}) \wedge x_{i1} \in X1_{m1},\ \ m1 \in M,\ \ z_{j1,t1} \in Z_j,$$

$$j1 \in J1 \subset J,\ \ i1 \in I,\ \ t1 \in \overline{1,T_j},\ \ r1 = \overline{1,R1} \Rightarrow$$

$$z[s_{j3}, x_{i3}] = z_{j3,t3},\ \ z_{j3,t3} \in Z_j,\ \ j3 \in J,\ \ t3 \in \overline{1,T_{j3}},\ \ i3 \in I,\ \ x_{i3} \in X1_{m1},$$

где $F1(z[s_{j1}, x_{i1}]\Theta_{r1}\, z_{j1,t1}) = \bigcap_{r1=1}^{R1}(z[s_{j1}, x_{i1}]\Theta_{r1}\, z_{j1,t1}).$

Для записи этого правила в нотации SQL введем обозначения. Известны @ID_X – определяющая вершина, набор свойств $J1$ и их значения для определяющей вершины, @ID_S3 – определяемое (изменяемое) свойство, @Z3 – новое значение определяемого свойства. Запишем правило для условия $i3 \neq i1$.

```
if OOx1(@ID_X)
    update SX set Z=@Z3 where ID_S=@ID_S3 and
            ID_X in (select G.ID_X from G where G.ID_X<>@ID_X
                    and G.ID_U in (select G1.ID_U from G1
                                where G1.ID_X=@ID_X))
```

Условие G.ID_X<>@ID_X соответствует условию $i3 \neq i1$.

3.8. ХРАНЕНИЕ И ОБРАБОТКА ПРАВИЛ

На рисунке 3.2 представлена структура базы данных для хранения правил. Правила записаны в нотации SQL и хранятся в таблицах PP, PR и PW в поле Текст_на_SQL в текстовом формате (nvarchar(max)). В поле Текст_на_ЕЯ хранится текст правила на естественном языке, который необходим для выдачи пояснения, почему принято то или иное решение.

Таблица PP содержит правила, определяющие принадлежность вершин ребру. Поле PP.Тип определяет характер запроса, на который отвечает правило (тип 1 – 5). В таблицах PR и PW хранятся правила, которые позволяют найти свойства ребер и вершин в зависимости от текущего состояния базы данных. Поле ID_S в таблицах PR и PW содержит первичный ключ искомого свойства.

Таблицы SPP, SPR и SPW содержат первичные ключи свойств ID_S, которые присутствуют в тексте правил. Эти таблицы нужны для обеспечения целостности базы, они не позволяют удалять свойство из таблицы S, если оно присутствует в тексте правила.

После поступления запроса любого типа сначала изменяются свойства вершин и ребер в зависимости от текущей ситуации решения задачи размещения. Для этого в цикле обрабатываются последовательно все правила в таблицах PR и PW до тех пор, пока не будет выполнено ни одно правило. Затем осуществляется однократный проход по правилам таблицы PP. Если запрос первого типа, то результ 1 или 0 (можно или нельзя размещать вершину в ребре). Для остальных типов правил результаты работы каждого правила запоминаются во временной таблице, из которой затем удаляются дубликаты.

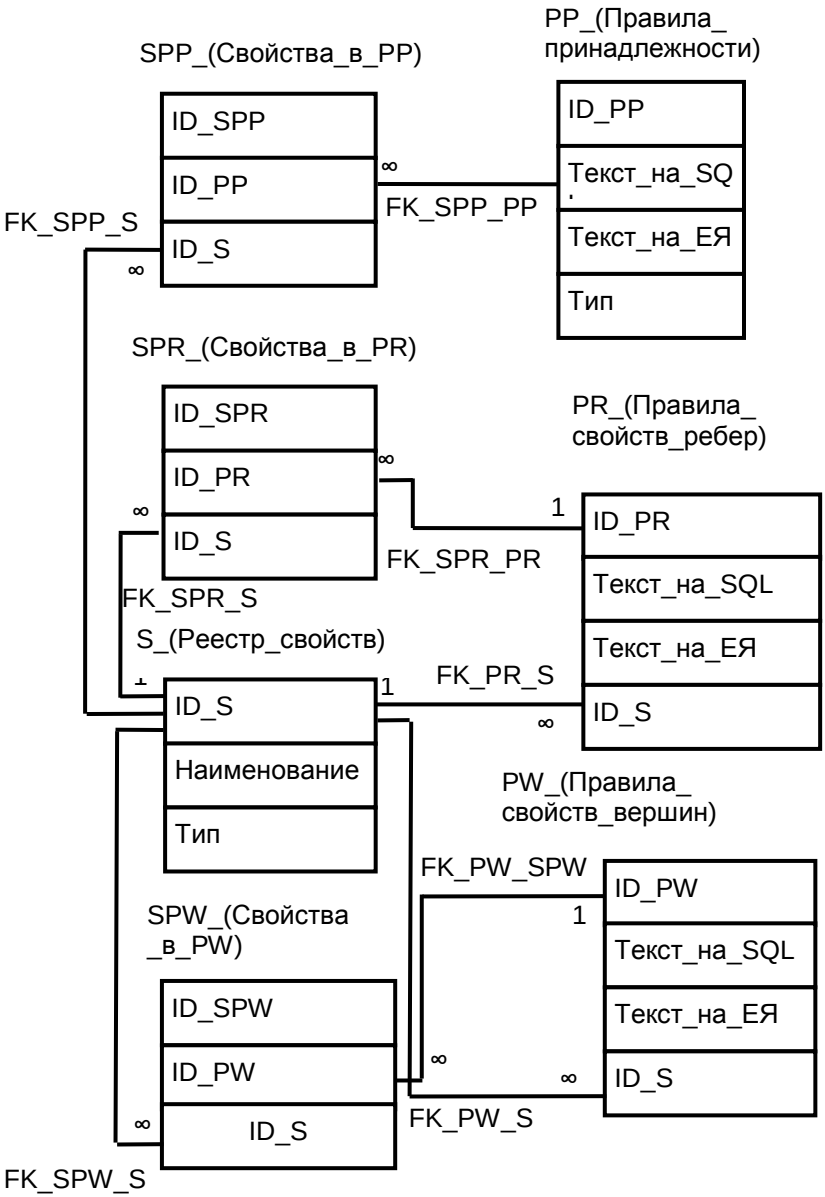


Рис. 3.2. Структура базы данных для хранения правил

Процедура формирование ответа для запроса первого типа (можно ли размещать ребро в вершине) представлена ниже.

```
create procedure PP_1 @ID_X1 int, @ID_U1 int, @ret1 int output
 /* @ret1 = 1, если вершину @ID_X1 МОЖНО размещать
    в ребре @ID_U1
    @ret1 = 0, если вершину @ID_X1 НЕЛЬЗЯ  размещать
    в ребре @ID_U1
 */
     as
     declare @PP nvarchar(max)  -- текст правила
     declare @param nvarchar(max)  --  параметры правила
     -- cursor_PP набор правил первого типа
     declare cursor_PP cursor for select Текст_на_SQL from PP
               where Тип=1
     set @param=N'@ID_X int, @ID_U int, @ret int output'
     set @ret1=1 --  предполагаем, что размещать
                    -- вершину в ребре можно
     open cursor_PP
     Fetch Next from cursor_PP into @PP -- чтение первого правила

     /* фукция  @@fetch_status = 0, если чтение
     правила прошло успешно */

     while @@fetch_status=0 and @ret1=1
       begin
         --  выполнение правила
         execute sp_executesql @PP, @param, @ID_X=@ID_X1,
         @ID_U=@ID_U1, @ret=@ret1 output
          Fetch Next from cursor_PP into @PP -- чтение
                                                -- следующего правила
       end
     close cursor_PP
     deallocate cursor_PP
```

Представленная процедура существенно упрощена по сравнению с реальной (промышленной). Так, например, в ней не приведен механизм формирования пояснения, почему вершину нельзя размещать в указанном ребре. Кроме того, процедура прекращает работать как только найдено первое правило, запрещающее размещать вершину @ID_X в ребре @ID_U. Просмотр всех правил позволяет пользователю получить не одно, а все условия, которые запрещают размещение вершины @ID_X в ребре @ID_U, что необходимо для последующего анализа. В реальной процедуре пользователь сам вибирает режим работы, – выход по первому запрещающему условию или получение всех запрещающих условий.

Рассмотренный подход может быть полезен при разработке новых автоматизированных систем размещения объектов в пространстве, или при разработке дополнений к уже существующим информационным системам размещения. Он использован автором при создании автоматизированной системы размещения химического оборудования в производственных помещениях.

4. СИНТАКСИС ЗАПРОСОВ КОНЕЧНЫХ ПОЛЬЗОВАТЕЛЕЙ К РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

В предыдущих разделах описаны элементы информационных систем, задача которых хранение и обработка знаний. В качестве базового программного обеспечения используются реляционные базы данных. Для использования этих элементов в информационных системах поддержки принятия решений необходимо разработать модули диалога, которые должны позволить конечным пользователям формулировать свой вопрос.

При этом программист должен по возможности учесть все пожелания конечных пользователей по обработке данных и получению отчетов этой обработки.

В настоящем разделе описывается синтаксис запросов конечных пользователей и интерпретация этих запросов в синтаксис SQL. В найденных автором публикациях, посвященных этой теме, в большей степени описывается [9], что надо сделать для того, чтобы конечный пользователь мог формировать запросы к базе данных на понятном ему языке, и в меньшей степени [15], как это сделать с минимальными затратами на программирование.

В зависимости от структуры разделим РБД на два класса [15, 25]:

– традиционные базы данных или базы данных с плоскими таблицами;

– базы данных типа ОБЪЕКТ–АТРИБУТ–ЗНАЧЕНИЕ.

Под традиционными РБД будем понимать базы данных, в которых сущности представлены широкими таблицами. Например, информационная сущность "Сдача цехом продукции на склад" характеризуется атрибутами: "Продукт", "Количество", "Цех_изготовитель", "Дата_изготовления", "Дата_сдачи" и др. Тогда в РБД будет существовать отношение (таблица), в которой с учетом нормализации представлены все перечисленные атрибуты (поля).

Базы данных типа ОБЪЕКТ–АТРИБУТ–ЗНАЧЕНИЕ представлены тремя основными таблицами: "Реестр объектов", "Реестр атрибутов", "Значения атрибутов объектов". Последняя таблица имеет связи с первыми двумя – типа многие к одному.

4.1. ОРГАНИЗАЦИЯ ДИАЛОГА В ТРАДИЦИОННЫХ БАЗАХ ДАННЫХ

Представим нормализованную РБД BDT в следующем виде:

$$BDT = \{R_1, ..., R_i, ..., R_I\}, \quad i = \overline{1, I},$$

$$R_i = \{PK_i, FK_i, D_i\},$$

$$FK_i = \{FK_{i1}, ..., FK_{ij}, ..., FK_{iJ}\}, \quad j = \overline{1, J},$$

$$D_i = \{D_{i1}, ..., D_{ik}, ..., D_{iK}\}, \quad k = \overline{1, K},$$

где R_i – i-е отношение (таблица), I – число отношений в базе данных, PK_i – ключевой атрибут i-го отношения, FK_i – множество атрибутов для внешних ключей i-го отношения, J – число атрибутов внешних ключей, D_i – множество атрибутов данных i-го отношения, K – число атрибутов данных (индекс i у J и K для упрощения записи не указан).

Обозначим $W = \{W_1, ..., W_t, ..., W_T\}$, $t = \overline{1, T}$, $W_t \in \bigcup_i^I D_i$ – множество атрибутов, по которым могут осуществляться запросы или выводиться отчеты. Под элементарным условием p запроса будем понимать запись $p = < a, f, z >$, где a – атрибут $a \in W$, z – множество значений атрибута, f – условие (равно, не равно, больше, меньше и т.д.), например <Город = 'Москва'> или <Дата= '01.01.2008'> .

Запрос пользователя S представим как совокупность элементарных запросов, соединенных логическими операторами $И, ИЛИ, НЕ$:

$$S = F(p_1, ..., p_m, ..., p_M), \quad m = \overline{1, M},$$

где F – функция, связывающая элементарные условия запроса m – номер элементарного условия запроса, M – число элементарных условий запроса, например, $S = p_1 \; И \; (p_2 \; ИЛИ \; p_3) \; И \; (НЕ \; p_4)$.

Отчет запроса составляется по элементам множества B, составленного из элементов множества W:

$$B = \{B^1, B^2\},$$

$$B^1 = \{b_1^1, ..., b_c^1, ..., b_C^1\}, \quad b_c^1 \in W, \quad c = \overline{1, C},$$

$$B^2 = \{\varphi_1(b_1^2), ..., \varphi_l(b_l^2), ..., \varphi_L(b_L^2)\}, \quad b_l^2 \notin W, \quad b_l^2 \notin B^1, \quad l = \overline{1, L},$$

где B^1 – множество атрибутов данных, по которым составляется отчет запроса, и к которым не применяются агрегатные функции (Sum, Max и др.); b_l^2, $l = \overline{1, L}$ – множество атрибутов данных, по которым составляется отчет, и к которым применяются агрегатные функции; φ_l, $l = \overline{1, L}$ –

агрегатные функции, применяемые к элементам b_l^2. Например B^1={Продукт, Город}, B^2={Sum(Количество)}.

С учетом сказанного, формирование запроса пользователя и получение результата разбиваются на следующие этапы:

– ввод элементарных условий, $\{p_1, ..., p_m, ..., p_M\}$, $m = \overline{1, M}$;

– ввод строки запроса, $S = F(p_1, ..., p_m, ..., p_M)$, $m = \overline{1, M}$;

– ввод полей отчета $B^1 = \{b_1^1, ..., b_c^1, ..., b_C^1\}$, $b_c^1 \in W$, $c = \overline{1, C}$;

– ввод полей отчета $B^2 = \{\varphi_1(b_1^2), ..., \varphi_l(b_l^2), ..., \varphi_L(b_L^2)\}$, $b_l^2 \in W$, $b_l^2 \notin B^1$, $l = \overline{1, L}$;

– выполнение запроса и получение отчета.

Для преобразования строки запроса S в формат SQL элементарные запросы, сформированные пользователем в виде $< a, f, z >$, представляются в виде $< aID, f, zID >$, где aID – наименование поля первичного ключа для a, zID – множество значений поля первичного ключа для множества значений атрибута z. Например, элементарное условие <Город = 'Москва'> в программной реализации необходимо заменить на <IDГород = IDМосква>, где IDМосква – значение ключевого поля для значения 'Москва'. В то же время элементарное условие <Дата ≥ 01.01.2008> остается без изменения. В первом случае назовем элементарное условие *условием для ID*, во втором случае – *условием для значения*. Преобразованную таким образом строку запроса S назовем SI.

Для организации диалога с пользователем при вводе элементарных условий необходимо обеспечить выбор атрибута a и ввод (выбор) множества значений атрибута z. Для выбора a создадим таблицу атрибутов запросов $G = \{GID, A, AID\}$. Значениями (доменом) поля A является множество $W = \{W_1, ..., W_t, ..., W_T\}$. Домен поля AID – наименования ключевых полей для $W = \{W_1, ..., W_t, ..., W_T\}$, если элементарное условие для ID или наименование поля, если элементарное условие для значения. Элементарные условия сохраняются в таблице $Y\{Имя\_условия, A, AID, f, Z, ZID\}$.

Создадим $V\{PK_1, ..., PK_i, ..., PK_n, W_1, ..., W_t, ..., W_T\}$ – представление (View), по которому составляется отчет. Тогда запрос на SQL можно записать как:

select B from V where SI group by B^1 order by B^1.

Рассмотрим разработку программы диалога на примере отгрузки готовой продукции потребителям. Каждая единица готовой продукции изготавливается определенным цехом. База данных представляется как:

$BDT = \{R_1, R_2, R_3, R_4, R_5\}$,

R_1 (IDR1, IDПокупатель, IDПродукт, Количество, Дата_отгрузки) – отгрузка продукции,

R_2 (IDПродукт, IDЦех, Наименование_продукта) – продукты,

R_3 (IDПокупатель, IDГород, Наименование_покупателя) – покупатели,

R_4 (IDГород, Наименование_города) – города,

R_5 (IDЦех, Наименование_цеха) – цеха.

Таким образом, отгрузка характеризуется шестью атрибутами "Покупатель", "Продукт", "Количество", "Дата отгрузки", "Цех_производитель", "Город" .

Конечный пользователь может сформулировать запрос на получение отчета по любой комбинации атрибутов, например отчет по отгрузке продуктов, выпущенных цехами 1, 3, 10, в города Москва, Липецк, Тамбов в январе 2008 г. и январе 2009 г. Форма отчетов по этому запросу может быть разнообразной, например, Продукт–количество, Город–Продукт–Количество, Продукт–Город–Количество и т.д.

Множество

W ={Наименование_покупателя, Наименование_продукта,
Наименовнаие_города, Наименование_цеха,
Дата_отгрузки, Количество}.

Для рассматриваемого примера таблица атрибутов запросов G выглядит следующим образом, табл. 4.1.

4.1. Таблица атрибутов запросов *G*

IDG	*A*	*AID*
1	Наименование_покупателя	IDПокупатель
2	Наименование_ продукта	IDПродукт
3	Наименование_цеха	IDЦех
4	Наименование_города	IDГород
5	Дата_отгрузки	Дата_отгрузки
6	Количество	Количество

Представление, по которому составляется отчет:

$V\{PK_1, ..., PK_i, ... PK_n, W_1, ..., W_t, ... W_T\} =$

$V\{$ IDR1, IDПокупатель, Наименование_покупателя, IDПродукт, Наименовнаие_продукта, IDГород, Наименовнаие_города, ID_цех, Наименование_цеха, Дата_отгрузки, Количество$\}$

получается в результате запроса:

```
select IDR1, IDПокупатель, Наименование_покупателя, IDПродукт,
       Наименовнаие_продукта, IDГород, Наименовнаие_города,
       ID_цех, Наименование_цеха, Дата_отгрузки, Количество
  from R1, R2, R3, R4, R5
  where R1.IDПокупатель= R3.IDПокупатель
        and R1.IDПродукт= R2.IDПродукт
        and R2.IDЦех= R5.IDЦех
        and R3.IDГород= R4.IDГород
```

Строка запроса пользователя представляет собой логическое выражение из элементарных условий. Например, запрос "Отгрузка продуктов, выпущенных цехами 1, 3, 10 (IDЦех – 2,5,8 соответственно) в города Москва, Липецк, Тамбов (IDГород – 10, 11, 12 соответственно) в январе 2008 г. и январе 2009 г." будет выглядеть так:

$$S = F(p_1, p_2, p_3, p_4, p_5, p_6) = p_1 \; И \; p_2 \; И \; (p_3 \; И \; p_4 \; ИЛИ \; p_5 \; И \; p_6),$$

где p_1 – цех=(цех1, цех 3, цех 10), p_2 – Город=(Москва, Липецк, Тамбов),

p_3 – Дата_отгрузки≥01.01.2008, p_4 – Дата_отгрузки≤31.01.2008,

p_5 – Дата_отгрузки≥01.01.2009, p_6 – Дата_отгрузки≤31.01.2009.

Ввод каждого элементарного условия $< a, f, z >$ состоит из следующих действий:

- выбор атрибута a из таблицы атрибутов запросов G;
- назначение условия, f;
- ввод значения или множества значений, z.

При этом создается табл. 4.2, столбцы 1, 2, 4, 5 которой представляются пользователю.

Далее пользователь вводит строку условий запроса $S = p_1 \; И \; p_2 \; И \; (p_3 \; И \; p_4 \; ИЛИ \; p_5 \; И \; p_6)$. Безусловно, что элементарные логические выражения он должен уметь составлять. Практика показала, что после получасового обучения у пользователей любого уровня знаний здесь проблем не возникает.

4.2. Таблица (отношение) *Y*

Имя условия	*A*	*AID*	*f*	*Z*	*ZID*
1	2	3	4	5	6
p_1	Наименование_цеха	IDЦех	=	цех1, цех 3, цех 10	2, 5, 8
p_2	Наименование_города	IDГород	=	Москва, Липецк, Тамбов	10, 11, 12
p_3	Дата_ отгрузки	Дата_ отгрузки	≥	01.01.2008	01.01.2008
p_4	Дата_ отгрузки	Дата_ отгрузки	≤	31.01.2008	31.01.2008
p_5	Дата_ отгрузки	Дата_ отгрузки	≥	01.01.2009	01.01.2009
p_6	Дата_ отгрузки	Дата_ отгрузки	≤	31.01.2009	31.01.2009

Поля отчета (поля представления *V*) также выбираются пользователем из таблицы атрибутов запросов *G*. В результате создаются множества:

B = {Наименование_продукта, Наименование_цеха, sum(Количество)},

B^1 = {Наименование_продукта, Наименование_цеха},

B^2 = {sum(Количество)}.

Строка условий запроса S на основании таблицы Y (табл. 4.2) преобразуется в:

$SI = pi_1$ and pi_2 and (pi_3 and pi_4 or pi_5 and pi_6),

где pi_1 – IDцех in zi_1, pi_2 – IDГород in zi_2, pi_3 – Дата_отгрузки≥ zi_3,

pi_4 – Дата_отгрузки≤ zi_4, pi_5 – Дата_отгрузки≥ zi_5,

pi_6 – Дата_отгрузки≤ zi_6,

zi_1 = (2, 5, 8), zi_2 = (10, 11, 12),

zi_3 = 01.01.2008, zi_4 = 31.01.2008, zi_5 = 01.01.2009, zi_6 = 31.01.2009

В результате формируем запрос:

select B from V where SI group by B^1 order by B^1

4.2. ОРГАНИЗАЦИЯ ДИАЛОГА В БАЗАХ ДАННЫХ ТИПА ОБЪЕКТ–АТРИБУТ–ЗНАЧЕНИЕ АТРИБУТА

Примером базы данных с подобной структурой являются база оборудования предприятия, справочник свойств веществ и др. Запрос к базе данных заключается в поиске таких объектов, атрибуты которых удовлетворяют заданным условиям. Например, в базе оборудования предприятия найти все емкостные аппараты, у которых имеется якорная мешалка и объем больше 10 м^3. База данных BDO в этом случае может быть представлена как:

$$BDO = \{O, S, SO, SZ\},$$

где O (IDO, Наименование_объекта) – объекты,

SW (IDSW, Наименование_атрибута) – аттрибуты,

SO (IDSO, IDO, IDSW, Значение_атрибута, IDSZ) – значения атрибутов конкретных объектов,

SZ (IDSZ,IDSW, Возможное_значение) – список значений атриибутов.

Отношение SZ вводится для тех атрибутов, значения которых выбираются из определенного списка, например, тип мешалки аппарата выбирается из списка: лопастная, турбинная, рамная и др.

Таблица атрибутов запросов здесь присутствует в явном виде (SW) и ее не надо создавать, как это было выше. Также не надо создавать представление V, так как его роль играют таблицы O и SO.

Последовательность составления запроса пользователем и его обработку рассмотрим на следующем примере. В базе оборудования завода найти аппараты объемом менее 10 м^3 с турбинной мешалкой.

Запрос пользователя S будет состоять из двух элементарных условий соединенных логическим $И$: "объем аппарата меньше 10 м^3 " $И$ "тип мешалки – турбинная".

Ввод каждого элементарного условия $< a, f, z >$ состоит из следующих действий:

– выбор атрибута a из таблицы атрибутов SW;

– назначение условия, f;

– ввод значения z или выбор его из таблицы SZ.

При этом создается отношение Y (табл. 4.3), столбцы 1, 2, 4, 5 которой представляются пользователю.

Далее пользователь составляет запрос:

$$S = p_1 \; И \; p_2,$$

где p_1 – Тип мешалки – турбинная, p_2 – Объем аппарата<10.

4.3. Таблица (отношение) *Y*

Имя условия	*A* (S.Наименование_атрибута)	*AID* (S.IDS)	*f*	*Z* (SZ.Возможное_значение или константа)	*ZID* (SZ.IDS или Null)
1	2	3	4	5	6
p_1	Тип мешалки	5	=	Турбинная	15
p_2	Объем аппарата м^3	7	<	10	Null

Строка условий запроса на основании таблицы *Y* (табл. 4.3) преобразуется в строку:

$$SI = pi_1 \text{ and } pi_2,$$

где pi_1 – exists (select * from SO where O.IDO=SO.IDO
and SO.IDS=5 and SO.IDSZ=15),
pi_2 – exists (select * from SO where O.IDO=SO.IDO
and SO.IDS=7 and SO.Значение_атрибута<10).

Составить программно-элементарные условия *pi* не представляет особого труда, так как они содержат постоянную часть "exists (select * from SO where O.IDO=SO.IDO and " и переменную часть, которая легко составляется из табл. 4.3.

Таким образом, запрос на поиск объектов, удовлетворяющих *SI*, можно записать в виде

select * from *O* where *SI*

5. ПРИМЕНЕНИЕ N-ОРИЕНТИРОВАННЫХ ГИПЕРГРАФОВ И РЕЛЯЦИОННЫХ БАЗ ДАННЫХ ДЛЯ СТРУКТУРНОГО И ПАРАМЕТРИЧЕСКОГО СИНТЕЗА ТЕХНИЧЕСКИХ СИСТЕМ

Структурный и параметрический синтез – основные этапы создания технических систем (ТС). Под структурным синтезом ТС будем понимать задачу нахождения такого состава ее элементов, который обеспечивает выполнение функций системы. Под задачей параметрического синтеза будем понимать нахождение геометрических, механических и других параметров элементов ТС, определенных в задаче структурного синтеза.

Для представления структуры и решения задач структурного синтеза ТС используются различные виды графов. Так, в [26] описано моделирование структур ТС с помощью аналитического представления графов, в том числе гипер- и мультиграфов. Использованию реляционных баз данных для описания структур ТС посвящена работа [12], в [14] описано представление гиперграфов в реляционных базах. Применение многодольных графов для решения задач структурного синтеза на элементах с ограниченной сочетаемостью представлено в [3].

Разработка информационной системы для решения задач структурного и параметрического синтеза ТС включает в себя много разных этапов, в том числе и следующие:

– выбор способа представления структуры ТС;

– выбор способов или алгоритмов получения структуры ТС;

– выбор базового программного обеспечения для представления структуры и алгоритмов в информационной системе.

В перечисленных выше источниках эти этапы рассматриваются каждый в отдельности с разной степенью детализации, что не совсем корректно, так как имеется их взаимное влияние друг на друга.

В настоящем разделе представлено совместное описание перечисленных выше этапов создания информационной системы для решения задач структурного и параметрического синтеза ТС.

Область применения – разработка программного обеспечения систем автоматизированного проектирования (САПР), предназначенных для:

– конструирования оборудования из типовых элементов (емкостные аппараты, теплообменники, колонны);

– проектирования технологических схем (определение типов аппаратов, необходимых для выпуска продукта по заданной технологии, например, красителей в химической или молочных продуктов в пищевой прмышленности);

– размещения единиц оборудования в производственном помещении, например, аппаратов в цехе химического или станков в цехе машиностроительного предприятия.

В качестве способа представления структуры ТС выбран N-ориентированный гиперграф [14], синтез структуры ТС осуществляется с помощью продукционных правил типа "Если А, то В", где А – логическое выражение-условие, В – логическое выражение-следствие. В качестве базового программного обеспечения используется реляционная база данных.

Таким образом, в контексте настоящей работы задача структурного синтеза заключается в определении ребер и вершин гиперграфа (построение гиперграфа), а задача параметрического синтеза – в определении свойств ребер и вершин.

Ассоциация ребра и вершины с реальными объектами зависит от решаемой задачи. Например, в задаче конструирования технологического оборудования в качестве ребер гиперграфа выступают узлы или сборочные единицы, вершины – детали. В задаче размещения единиц оборудования в производственном помещении ребра – отделения или этажи цеха, вершины – отдельные единицы оборудования.

Обозначим N-ориентированный гиперграф $G(X, U)$, где $X = \{x_i\}$, $i = \overline{1, I}$ – множество вершин гиперграфа, x_i – i-я вершина; $U = \{u_m(X1_m)\}$, $m = \overline{1, M}$ – множество ребер гиперграфа, $u_m(X1_m)$ – m-е ребро гиперграфа, $X1_m$ – множество вершин, инцидентных m-му ребру $X1_m \subseteq X$, $X1_m = \{x_k^{\overline{L}}\}$, $\forall k \in K_m$, $K_m \subseteq \overline{1, I}$, $\overline{L}$ – номер вершины в ребре ориентированного гиперграфа представляет собой вектор, $\overline{L} = \{l_n\}$, $n = \overline{1, N}$, мощность которого N. В общем случае номер вершины в ребре не обязательно должен иметь значение номер по порядку и может отражать определенное свойство вершины, которое принимает конкретное значение при включении вершины в ребро. При этом предполагается, что все вершины в ребрах имеют одинаковый набор свойств. Например, при решении задачи размещения элементов в пространстве, элементы имеют одинаковые наборы свойств, а именно, координаты элементов.

Требование одинакового набора свойств элементов существенно сужает область прикладных задач. Каждый элемент реальной технической системы обладают собственным набором свойств, который отличается от наборов свойств других элементов, например, каждая деталь изделия имеет собственный набор размеров.

Обозначим $S = \{s_j\}$, $j = \overline{1, J}$ – множество всех возможных свойств вершин и ребер, $SX_i = \{s_r\} \subset S$, $i = \overline{1, I}$, $r \in \overline{1, J}$ – множество свойств i-й

вершины, $SU_m = \{s_r\} \subset S$, $m = \overline{1, M}$, $r \in \overline{1, J}$ – множество свойств m-го ребра. Под свойством здесь понимается контейнер для хранения значения свойства. Например, свойство тип фланца может иметь значение "плоский приварной". Таким образом, для каждой вершины множество ее номеров $\overline{L} = \{l_n\}$, $n = \overline{1, N}$ заменяется множеством свойств SX_i, $i = \overline{1, I}$. Кроме того, каждой дуге также ставится в соответствие свой набор свойств.

5.1. ГЕНЕРАЦИЯ ВАРИАНТОВ ТЕХНИЧЕСКОЙ СИСТЕМЫ. СТРУКТУРНЫЙ СИНТЕЗ

Любая вновь создаваемая ТС должна выполнять заданные функции в заданных условиях, которые определены, например, в техническом задании (ТЗ) на создание ТС. Если таких ТС получается несколько, то предпочтение отдается решению с экстремальным значением некоторого показателя, например, минимума затрат на создание и эксплуатацию ТС, т.е. решается оптимизационная задача. Независимо от того, решается ли задача оптимизации или нет, необходимо уметь генерировать варианты ТС. Генерация варианта ТС состоит из этапа структурного и этапа параметрического синтеза.

Как указано выше, задача структурного синтеза заключается в определении ребер и вершин гиперграфа (построение гиперграфа). При этом вершины выбираются из множества допустимых вершин $X \subset XD = \{xd_{di}\}$, $di = \overline{1, DI}$, а ребра – из множества допустимых ребер $U \subset UD = \{ud_{dm}\}$, $dm = \overline{1, DM}$. Этот выбор осуществляется на основании данных ТЗ, которые можно представить как множество двоек $TZ = \{t, z\}$, где t – некоторое свойство ТС, z – значение этого свойства, например рабочее давление (свойство) равно 0,6 МПа (значение). Функции аппарата также можно отнести к свойствам, при этом z становится вектором, например, $z = \{$*загрузить*, *нагреть*, *пермешать*, *выгрузить*$\}$.

Выбор вершин и ребер по данным ТЗ осуществляется с использованием продукционных правил, например, "Если необходимо перемешивать, то должна быть мешалка". С учетом принятых обозначений это правило можно записать как "Если $перемешать \in TZ$, то $мешалка \in X$ ".

Задача параметрического синтеза заключается в определении свойств ребер и вершин. Обозначим $x_{i,m}(s_r)$ – значение свойства s_r вершины x_i в ребре u_m, $u_m(s_r)$ – значение свойства s_r ребра u_m, тогда задача параметрического синтеза заключается в нахождении множества значений

свойств вершин $SXG = \{x_{i,m}(s_r)\}$, $\forall i \in \overline{1,I}$, $m \in \overline{1,M}$, $r \in \overline{1,J}$ и множества значений свойств ребер $SUG = \{u_m(s_r)\}$, $\forall\ m \in \overline{1,M}$, $r \in \overline{1,J}$.

По аналогии с задачей структурного синтеза параметры вершин и ребер находятся на основании данных ТЗ с использованием правил. Например, "Если вязкость среды > 500 Па·с, то тип мешалки – шнековая или ленточная".

Структура базы данных для представления N-ориентированного гиперграфа и решения задач структурного и параметрического синтеза представлена на рис. 5.1.

В представленной на рис. 5.1 структуре сложные названия таблиц использованы для лучшего восприятия. Таблицы Вершины,XD, Ребра,UD, Свойства_вершин,SX, Свойства_ребер,SU, Свойства,S, Тех.задание,TZ, Гиперграф,G, Значения_свойств_вершин,SXG и Значения_свойств_ребер,SUG соответствуют описанным выше множествам XD, UD, SX, SU, S, TZ, G, SXG и SUG.

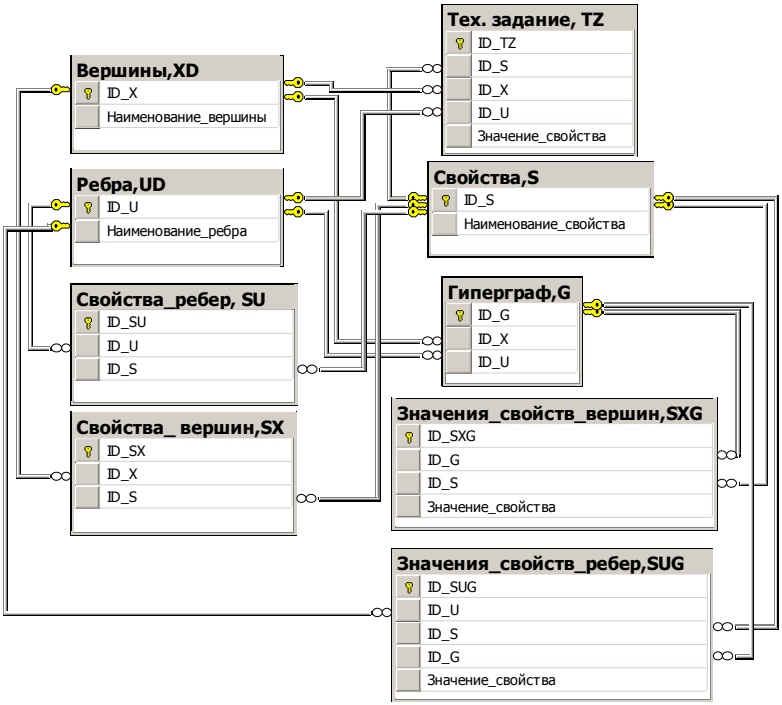


Рис. 5.1. Структура базы данных для решения задач структурного и параметрического синтеза

В структуре на рис. 5.1 отсутствует множество вершин *X* и множество ребер *U* гиперграфа. Эти множества можно получить в результате выполнения следующих запросов:

```
select Наименование_вершины from G, XD where G.ID_X=XD.ID_X
select Наименование_ребра  from G, UD  where G.ID_U=UD.ID_U
```

Здесь в запросах использованы короткие названия таблиц. Для написания запросов использован Transact-SQL (MS SQL Server – 2005).

Представленная структура позволяет создавать гиперграф, в котором ребра могут иметь одинаковые вершины с разными или одинаковыми свойствами. Кроме того, гиперграф может иметь одинаковые ребра с разными или одинаковыми вершинами и свойствами. Последнее соответствует случаю, когда один и тот же узел входит в изделие несколько раз. Например, узел штуцер, состоящий из патрубка и фланца, входит в емкостной аппарат много раз, но с разными свойствами как самого узла, так и его элементов. Например, свойство штуцера "назначение" может принимать значения: "технологический", "для ввода пара" и др. Для разных штуцеров будут разные свойства патрубка и фланца, например их геометрические размеры.

Как было указано выше, генерация структуры ТС осуществляется с помощью продукционных правил типа "Если А, то В". Продукционные правила будем хранить в виде текстовых строк SQL-запросов. Рассмотрим синтаксис правила на приведенном выше примере "Если необходимо перемешивать, то должна быть мешалка". Предположим, что существуют:

– в таблице *S* – строка ID_S=1, Наименование_свойства= "функция";

– в таблице TZ – строка ID_S=1, Значение_свойства= "перемешать" (ID_TZ может быть любым, ID_X и ID_U неопределены);

– в таблице XD – строка ID_X=5, Наименование_вершины= "мешалка".

Тогда текст правила, позволяющего выбрать мешалку, будет следующий:

```
if  exists (select * from TZ where ID_S=1
            and Значение_свойства="перемешать")
 select * from XD where ID_X=5
```
(5.1)

В тексте этого правила отсутствует указание, куда поместить результат выполнения. Это указание определяется классификацией правил, о которой будет сказано ниже. Кроме того, если придерживаться принципов нормализации базы данных, необходимо ввести таблицу Возможные_значения_свойств, в которой будет строка со значением "перемешать". Эта таблица, как и некоторые другие, для упрощения изложения опущена.

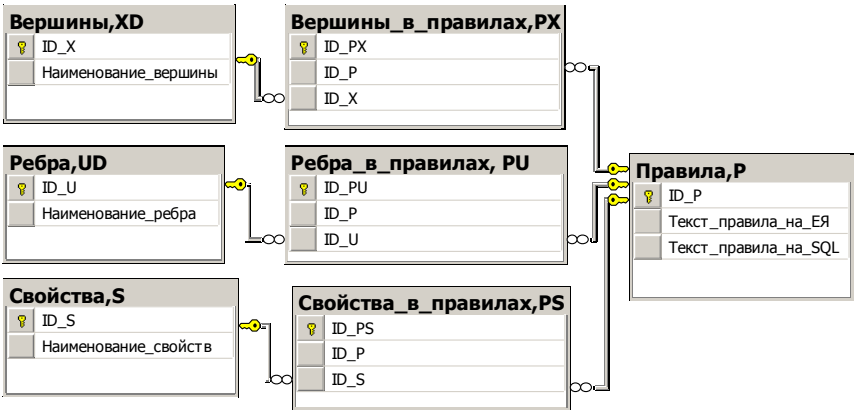


Рис. 5.2. Структура базы данных для хранения правил

Структура базы данных для хранения правил представлена на рис. 5.2. В основной таблице Правила,P тексты правил хранятся на естественном языке (поле Текст_правила_на_ЕЯ) и на языке SQL (поле Текст_правила_на_SQL). Таблицы Вершины_в_правилах,PX, Ребра_в_правилах,PU, Свойства_в_правилах,PS введены для обеспечения целостности базы, например, для исключения возможности удаления свойства из таблицы S, если оно присутствует в правиле.

5.2. СИНТАКСИС ПРАВИЛ И ИХ ОБРАБОТКА

Обработка правил заключается в многократном проходе по записям таблицы Правила,P и выполнении этих правил как SQL-запросов. При этом используется таблица TZ, а также таблицы, в которые помещаются результаты выполнения предыдущих правил.

Результаты работы правил могут быть классифицированы по следующим группам:

1) ребро вводится в граф;

2) ребро вводится в граф и задаются его свойства;

3) задаются свойства существующего ребра;

4) вершина вводится в граф (ребро не определено);

5) вершина вводится в граф (ребро не определено) и задаются ее свойства;

6) вершина вводится в определенное ребро;

7) вершина вводится в определенное ребро и задаются ее свойства;

8) задаются свойства вершины в определенном ребре;

9) задаются свойства вершины (ребро не определено).

Поясним группы 4 и 5. Приведенное выше правило (5.1) позволяет включить мешалку в структуру ТС, как элемент (как вершину), но не определяет, что мешалка должна входить в узел привод (в ребро), т.е. имеются правила, в которых говорится, что вершина будет входить в граф, но не говорится какому ребру она будет принадлежать. Следует ожидать, что дополнительно будет правило, в котором определяется, в какое ребро будет входить выбранная ранее вершина.

Гиперграф G, определение которого введено выше, представляет собой вариант конкретной ТС. При обработке правил получается не один вариант ТС, а семейство возможных вариантов, так как возможно существование правила "Если А, то В или С".

Обозначим:

$PG(PXG, PUG)$ – гиперграф, который получается в результате обработки правил;

$PXG = \{x_{pi}\} \subseteq XD,\ pi \in \overline{1, DI}$ – множество вершин гиперграфа;

$PUG = \{u_{pm}\} \subseteq U,\ pm \in \overline{1, DM}$ – множество ребер гиперграфа;

$SPXG = \{x_{pi}(s_r)\}$ – множество значений свойств вершин из PXG, $x_{pi}(s_r)$ – значение свойства s_r для вершины x_{pi};

$SPUG = \{u_{pm}(s_r)\}$ – множество значений свойств ребер из PUG, $u_{pm}(s_r)$ – значение свойства s_r для ребра u_{pm};

$SPG = \{x_{pi,pm}(s_r)\}$ – множество значений свойств вершин в ребрах (в гиперграфе PG), $x_{pi,pm}(s_r)$ – значение свойства s_r вершины x_{pi} в ребре u_{pm}.

Таким образом, результаты работы правил группы 1 записываются в *PUG*, группы 2 – в *PUG* и в *SPUG*, группы 3 – в *SPUG*, группы 4 – в *PXG*, группы 5 – в *PXG* и *SPXG*, группы 6 – в *PG*, группы 7 – в *PG* и в *SPG*, группы 8 – в *SPG*, группы 9 – в *SPXG*.

Примеры правил:

Правило 1. "Если необходимо перемешивать, то должна быть мешалка" (группа 4).

Правило 2. "Если вязкость среды > 500 Па·с, то тип мешалки – шнековая" (группа 9).

Правило 3. "Если есть мешалка, то должен быть привод" (группа 1).

Правило 4. "Привод состоит из мотор-редуктора, стойки и вала мешалки" (группа 6).

Если придерживаться терминологии, принятой в экспертных системах, то последнее правило можно назвать фактом. На наш взгляд, это дело

привычки, тем более, что в дальнейшем изложении это правило приобретает условие.

В терминах теории множеств представленные правила записываются так.

Правило 1.

$\exists функция, перемешать \in Tz \Rightarrow \exists мешалка \in PXG.$

Правило 2.

$\exists мешалка \in PXG \, и \, вязкость \, среды > 500 Па * с \Rightarrow \exists \, шнековая \in SPXG.$

Правило 3.

$\exists мешалка \in PXG \Rightarrow \exists привод \, и \, мешалка \in PG.$

Правило 4.

$\exists привод \in PG \Rightarrow \exists мотор\text{-}редуктор, стойка, вал \, мешалки \in PG.$

Структура базы данных для хранения результатов запросов представлена на рис. 5.3.

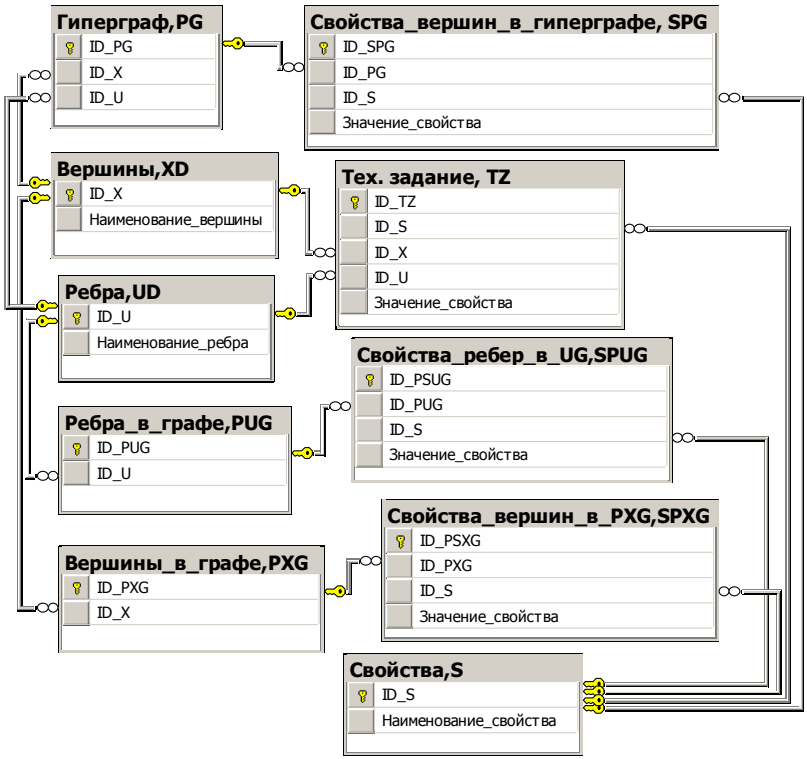


Рис. 5.3. Структура базы данных для хранения результатов запросов

Вершины, XD

ID_X	Наименование_ вершины
5	мешалка
6	стойка
7	вал_мешалки
8	мотор-редуктор

Свойства, S

ID_S	Наименование_свойства
1	функция
2	вязкость_среды
3	тип_мешалки

Ребра, UD

ID_U	Наименование_ ребра
1	привод

Техзадание, TZ

ID_TZ	ID_S	ID_X	ID_U	Значение_ свойства
1	1	null	null	перемешать
2	2	null	null	400

Рис. 5.4. Пример содержания таблиц
Вершины,XD, Ребра,UD, Свойства,S, Тех.задание,TZ

Рассмотрим синтаксис представления правил в формате SQL на следующем примере. Предположим, что в ТЗ написано, что функция ТС – перемешать жидкость вязкостью 400 Па·с. Содержание таблиц Вершины,XD, Ребра,UD, Свойства,S, Тех.задание,TZ представлено на рис. 5.4.

С учетом содержимого вышеприведенных таблиц, синтаксис правил в формате SQL будет следующим.

Правило 1. Если необходимо перемешивать, то должна быть мешалка.

$$\exists функция, перемешать \in Tz \Rightarrow \exists мешалка \in PXG.$$

```
if exists ( select  * from TZ where ID_S=1
          and Значение_свойства='перемешать' and ID_X is Null
          and ID_U is Null)
    begin
       insert PXG values (5)
       set @ret=1
    end
```

Правило 2. Если вязкость среды > 500 Па·с, то тип мешалки – шнековая.

$$\exists \textit{мешалка} \in PXG \textit{ и вязкость среды} > 500 \textit{Па} * c \Rightarrow \exists \textit{ шнековая} \in SPXG.$$

```
if exists (select * from  TZ where ID_S=2 and Значение_свойства>500
    and ID_X is Null and ID_U is Null) and exists
    (select * from PXG  where ID_X=5)
      begin
        declare @ID_PXG int;
        select @ID_PXG=ID_PXG from PXG where ID_X=5
        insert SPXG values (@ID_PXG, 3, 'шнековая')
        set @ret=1
      end
```

Правило 3. Если есть мешалка, то должен быть привод.

$$\exists \textit{мешалка} \in PXG \Rightarrow \exists \textit{привод и мешалка} \in PG.$$

```
if exists (select * from PXG where ID_X=5)
    begin
        insert PUG values (1)
        set @ret=1
    end
```

Правило 4. Привод состоит из мотор-редуктора, стойки и вала мешалки.

$$\exists \textit{привод} \in PG \Rightarrow \exists \textit{ мотор-редуктор, стойка, вал мешалки} \in PG.$$

```
if exists (select * from PUG where ID_U=1
    begin
          declare @ID_PUG int
          select @ID_PUG=ID_PUG from PUG where ID_U=1
          insert PG values (6,@ID_PUG)
          insert PG values (7,@ID_PUG)
          insert PG values (8,@ID_PUG)
          set @ret=1
    end
```

Приведенные правила в случае своего выполнения (выполняется условный оператор) возвращают значение @ret=1.

Выполнение правила осуществляется с помощью функции sp_executesql. Предположим, что запись с правилом 1 в таблице Правила,P имеет ID_P=1. Тогда выполнение этого правила осуществляется следующими операторами.

```
declare @str as nvarchar(500), @param nvarchar(100), @ret  int
select @str=Текст_правила_на_SQL from P where ID_P=1
set @param=N'@ret int output'
exec sp_executesql N'set @ret=0; '+@str, @param, @ret out
```

При этом переменная @ret будет иметь значения 1, если правило отработало и 0 в противном случае.

Алгоритм обработки правил заключается в многократной выборке и выполнении правил из таблицы Правила,P. Выборка правил прекращается, когда не выполняется ни одно правило. Программа, реализующая указанный алгоритм, представлена ниже.

```
declare @ret1 int, @str  nvarchar(1000),@param nvarchar(100),
        @ret  int,@str1  nvarchar(500)
set @param=N'@ret int output'
set @ret1=1
while @ret1=1
    begin
        set @ret=0
        declare cursor1 cursor for
                select  [Текст_правила_на_SQL] from P
        open cursor1
        fetch next from cursor1 into @str
        set @str1= N'set @ret=0; '+@str
        exec sp_executesql @str1, @param, @ret out
        if @ret=1 set @ret1=1
        while @@fetch_status=0
            begin
               set @str1= N'set @ret=0; '+@str
               exec sp_executesql @str1, @param, @ret out
               if @ret=1 set @ret1=1
               fetch next from cursor1 into @str
            end
            close cursor1
            deallocate cursor1
    end
```

В представленной программе отсутствует механизм исключения повторного выполнения правил. Разработать такой механизм достаточно просто. Например, можно записывать ID_P выполненных правил во временную таблицу и выполнять функцию sp_executesql только в том случае, если выполняемого правила нет в этой таблице.

Достоинством предложенного подхода является то, что он не требует для хранения и выполнения правил дополнительного программного обеспечения в виде оболочки экспертной системы. Современные предприятия уже имеют необходимое программное обеспечение и программистов нужной квалификации для его внедрения, так как реляционные базы данных в настоящее время являются основной программной средой для создания единого информационного пространства предприятия.

5.3. ОПРЕДЕЛЕНИЕ РАЗМЕРОВ ЭЛЕМЕНТОВ ТЕХНИЧЕСКИХ ОБЪЕКТОВ. ПАРАМЕТРИЧЕСКИЙ СИНТЕЗ

Размеры элементов – обязательные параметры, которые необходимо знать для изготовления рабочих чертежей деталей и сборочных единиц. Их число велико, однако на практике (при известной структуре ТО), зная значения небольшого числа определяющих размеров (ОР) элементов ТО, можно рассчитать все остальные. Например, для емкостного аппарата ОР – это диаметр и высота обечайки.

Размеры элементов ТО можно разделить на три типа:

а) определяющие размеры, которые зависят от параметров процессов, для которых предназначен проектируемый ТО;

б) размеры, которые зависят от размеров других элементов ТО (единичные параметры [14]), например, внутренний диаметр приварного встык фланца аппарата равен внутреннему диаметру корпуса;

в) размеры, которые зависят от других размеров этого же элемента (унитарные параметры [14]), например, все размеры стандартного фланца аппарата зависят от его внутреннего диаметра и давления в аппарате.

Размеры типа а) являются целью математического моделирования и оптимизации процессов, протекающих в ТО. Они очень важны, но их недостаточно для разработки рабочего проекта ТО. После их определения необходимо найти множество размеров типа б) и в). Далее рассматриваются размеры типа б) и в). В дальнейшем будем их называть а-размерами, б-размерами и в-размерами.

На рисунке 5.5 представлена структура базы данных, которая позволяет найти б- и в-размеры по известным а-размерам. Исходные данные представлены в таблицах Спецификация_вход и А-размеры, результат – в таблице Спецификация_выход.

В таблице Спецификация_вход содержится структура изделия, полученная в результате структурного синтеза. Элементы проектируемого ТО здесь определены в виде классов, например, известно, что обечайку и крышку соединяет фланец плоский приварной (класс), но размеры этого фланца неизвестны. Их определение – наша цель.

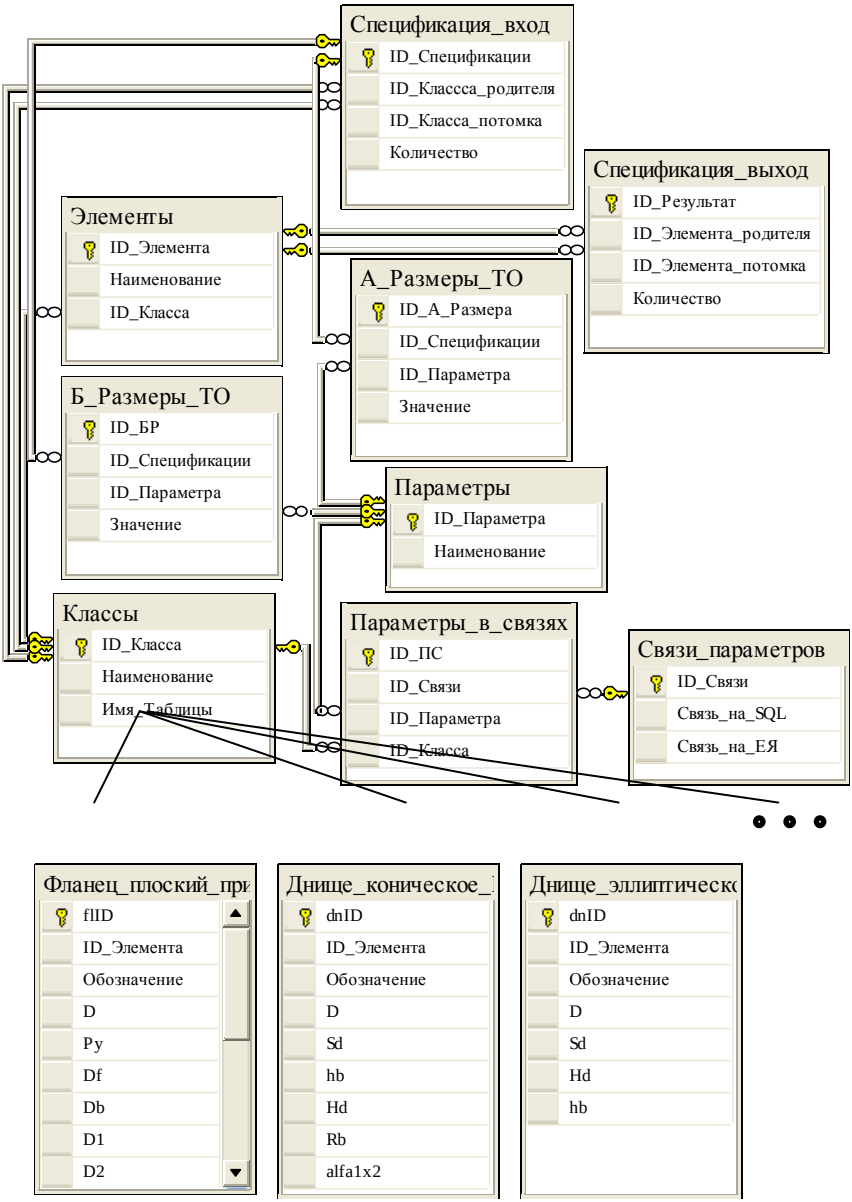


Рис. 5.5. Структура базы данных для расчета б- и в-размеров по известныи а-размерам

Связи между размерами элементов находятся в виде SQL-запросов в таблице Связи_параметров. Поле Связи_параметров.Текст_на_ЕЯ содержит описание связи на естественном языке. Таблица Параметры_связи необходима для поддержания целостности базы данных, чтобы нельзя было удалить параметр или класс, ссылка на который имеется в запросах в поле Связи_параметров.Связь_на_SQL. Каждой записи таблицы Классы соответствует таблица размеров элементов класса, представляющая собой нормативный документ, например, Фланцы плоские приварные ГОСТ. Таблица Параметры содержит список всех а- и б-размеров.

Заполнение таблицы Спецификация_выход осуществляется в два этапа:

– с использованием данных таблиц Спецификация_вход и А_Размеры_ТО в цикле выполняются запросы, содержащиеся в поле Связи_параметров.Связь_на_SQL. Результат помещается в таблицу Б_Размеры_ТО;

– по данным таблицы Б_Размеры_ТО в таблицах размеров класса находится элемент, удовлетворяющий условиям таблицы Б_Размеры_ТО и результат помещается в таблицу Спецификация_выход.

Следует отметить, что представленная ER-диаграмма отражает идею предлагаемого подхода и в реальной АИС она дополняется другими таблицами и полями.

Предложенная структура базы данных для определения размеров типовых элементов ТО, отличается тем, что связи между элементами описываются в виде SQL-запросов, хранимых в таблице, что позволяет использовать ее при проектировании различных типов ТО (емкостные аппараты, теплообменники и др.), не меняя структуры.

Эффективность и работоспособность описанного подхода структурного и параметрического синтеза технических систем доказана при создании и эксплуатации разработанной авторами системы автоматизированного проектирования химического оборудования.

6. ПРОЦЕДУРНЫЕ И ИНФОРМАЦИОННО-ЛОГИЧЕСКИЕ МОДЕЛИ ПЛАНИРОВАНИЯ ВЫПУСКА ПРОДУКЦИИ И РЕМОНТА ОБОРУДОВАНИЯ МНОГОАССОРТИМЕНТНЫХ ХИМИЧЕСКИХ ПРОИЗВОДСТВ

Традиционными объектами единого информационного пространства (ЕИП) для химических предприятий являются: номенклатура готовой продукции (ГП); поставщики сырья и потребители ГП; нормы расхода сырья; характеристики имеющегося оборудования и др. На этой информационной основе решаются задачи учета поступления сырья, отгрузки ГП, планирования выпуска ГП, составления графиков планово-предупредительных ремонтов (ППР) оборудования и др.

Планированию выпуска готовой продукции и составлению графиков ППР химическими предприятиями посвящено достаточно много публикаций [4]. В [10] рассматривается представление иерархических структур (в том числе и оборудования) в реляционной базе данных. Однако полного представления данных для решения задач планирования выпуска ГП и составления графиков ППР для многоассортиментных химических производств (МХП) в литературе не найдено.

В настоящем разделе для МХП предлагается:

– процедурная модель составления плана выпуска ГП;

– процедурная модель составления графиков ППР основного и вспомогательного оборудования;

– информационно-логическая модель данных, необходимых для решения задач планирования выпуска ГП и составления графиков ППР МХП.

6.1. ПРОЦЕДУРНАЯ МОДЕЛЬ СОСТАВЛЕНИЯ ПЛАНА ВЫПУСКА ГОТОВОЙ ПРОДУКЦИИ

Существуют многочисленные критерии оценки плана выпуска ГП: получение максимальной прибыли, максимальное удовлетворение заявок покупателей, максимальная загрузка оборудования и др. В современных условиях на МХП имеется избыток оборудования и недостаток заказов. Именно для этих условий и предлагается описываемая ниже модель, которая реализована в корпоративной информационной системе (КИС) ОАО "Пигмент", г. Тамбов. План выпуска ГП составляется на три месяца [11].

1) Менеджеры отдела реализации формируют планы продаж $PR^k = \{pr_i^k\}$ и нормативные остатки ГП на складе $NO^k = \{no_i^k\}$ в планируемых периодах, где i, $i = \overline{1, I}$ – номер продукта k, $k = 1, 2, 3$ – номер

месяца, pr_i^k – количество *i*-го продукта, которое предполагается продать в *k*-м месяце (для текущего месяца $k = 0$), no_i^k – нормативный остаток *i*-го продукта в *k*-м месяце. Нормативные остатки определяются экспертным путем и изменяются в зависимости от сезонных продаж, экономической ситуации и др.

2) Менеджеры отдела реализации формируют массив предполагаемых продаж продуктов на конец текущего месяца $PR^0 = \{pr_i^0\}$.

3) Плановый отдел или производственные подразделения формируют массив предполагаемого выпуска продуктов на конец текущего месяца, $W^0 = \{w_i^0\}$.

4) Автоматически рассчитывается предполагаемый план выпуска и остатки ГП на начало каждого планируемого месяца. Остатки на начало текущего месяца $O^0 = \{o_i^0\}$ известны.

$$O^k = \{o_i^k\}, \quad \text{где} \quad o_i^k = o_i^{k-1} + w_i^{k-1} - pr_i^{k-1},$$

$$W^k = \{w_i^k\}, \quad \text{где} \quad w_i^k = p_i^k - o_i^k + no_i^k, \quad \text{если} \quad w_i^k < 0, \quad \text{то} \quad w_i^k = 0,$$

$$k = 1, 2, 3.$$

5) Экспертная оценка плана и окончательное его утверждение. Здесь может быть рассчитана прибыль, загрузка оборудования и др.

6.2. ПРОЦЕДУРНАЯ МОДЕЛЬ СОСТАВЛЕНИЯ МЕСЯЧНОГО ГРАФИКА РЕМОНТА ОБОРУДОВАНИЯ

Оборудование делится на две категории: оборудование, месячный пробег которого зависит от количества выпущенного продукта и оборудование, месячный пробег которого постоянный (емкости для хранения, вентиляционное оборудование и др.). Как правило, к первой категории относится основное оборудование, ко второй – вспомогательное. Далее термины "основное" и "вспомогательное" оборудование применяются только в смысле переменного и постоянного месячного пробега. Предлагаемая модель реализована в КИС ОАО "Пигмент", г. Тамбов.

В конце месяца (текущий месяц) для следующего месяца (плановый месяц) в отделе главного механика выполняются следующие расчеты:

1) Для каждого продукта $i, i = \overline{1, I}$ плана рассчитывается число выпускаемых партий n_i

$$n_i = P_i / p_i, \text{ если } n_i \text{ не целое, то } n_i = \text{int}(P_i / p_i) + 1,$$

где i – номер продукта, P_i – план выпуска (кг) i-го продукта, p_i – размер партии (кг) i-го продукта, int – целая часть.

2) Для каждой единицы оборудования j, $j = \overline{1, J}$ рассчитывается время работы, необходимое для выпуска всех продуктов плана T_j :

– для основного оборудования $T_j = \sum_i (t_{ji} n_i + tpr_i) / k_{ji}$;

– для вспомогательного оборудования $T_j = Tx_j$,

где j – номер единицы оборудования, t_{ji} – время работы j-й единицы оборудования (час) при выпуске одной партии i-го продукта; k_{ji} – число взаимозаменяемых j-х единиц оборудования для выпуска i-го продукта; Tx_j – характеристика оборудования "время работы в месяц", tpr_i – общее время промывки при выпуске i-го продукта.

$$tpr_i = tpr1_i(\text{int}(n_i / npr1_i) + 1),$$

где $tpr1_i$ – время одной промывки при выпуске i-го продукта, $npr1_i$ – число партий i-го продукта, через которое надо делать промывку.

3) Для каждой единицы оборудования рассчитывается наработка на конец планируемого месяца после последнего ТО, ТР, КР, при этом наработка на начало планируемого месяца известна.

$$TtoK_j = TtoN_j + T_j,$$

$$TtrK_j = TtrN_j + T_j,$$

$$TkrK_j = TkrN_j + T_j,$$

где $TtoN_j$, $TtoK_j$ – наработка на начало и конец планируемого месяца после последнего ТО; $TtrN_j$, $TtrK_j$ – наработка на начало и конец планируемого месяца после последнего ТР; $TkrN_j$, $TkrK_j$ – наработка на начало и конец планируемого месяца после последнего КР.

4) Обнуление графика ремонта оборудования планируемого месяца

$$Dkr_j = 0, \quad Dtr_j = 0, \quad Dto_j = 0,$$

где Dkr_j – день проведения КР, Dkr_j – день проведения ТР, Dto_j – день проведения ТО.

5) Если $TkrK_j > Tkr_j$, где Tkr_j – межремонтный пробег КР,

$Dkr_j = 1$, если $TkrN_j > Tkr_j$, иначе $Dkr_j = (Tkr_j - TkrN_j)/(T_j 30)$, переход на пункт 8.

6) Если $TtrK_j > Ttr_j$, где Ttr_j – межремонтный пробег ТР,

$Dtr_j = 1$, если $TtrN_j > Ttr_j$, иначе $Dtr_j = (Ttr_j - TtrN_j)/(T_j 30)$, переход на пункт 8.

7) Если $TtoK_j > Tto_j$, где Tto_j – межремонтный пробег ТО,

$Dto_j = 1$, если $TtoN_j > Tto_j$, иначе $Dto_j = (Tto_j - TtoN_j)/(T_j 30)$.

В конце месяца механиками цехов выполняются следующие расчеты.

1) Ввод фактического дня ремонта ТО, ТР, КР $Dtof_j^l$, $Dtrf_j^l$, $Dkrf_j^l$ и фактического пробега оборудования $Tf_j{}^l$ (по факту на конец месяца, l – номер месяца).

2) Расчет состояния оборудования (наработка) на конец планируемого месяца (l – номер месяца), $TkrK_j{}^l$, $TtrK_j{}^l$, $TtoK_j{}^l$ – наработка на конец *l*-го месяца *j*-й единицы оборудования после последнего ТО, ТР, КР. Наработка на начало *l*-го месяца *j*-й единицы оборудования $TkrN_j{}^l$, $TtrN_j{}^l$, $TtoN_j{}^l$ после последнего КР, ТР, ТО известна.

Если $Dkrf_j{}^l > 0$, то $TkrK_j^l = Tf_j{}^l(30 - Dkrf_j)/30$, $TtrK_j^l = TkrK_j{}^l$, $TtoK_j^l = TkrK_j{}^l$; иначе $TkrK_j^l = TkrN_j{}^l + Tf_j{}^l$.

Если $Dtrf_j{}^l > 0$, то $TtrK_j^l = Tf_j{}^l(30 - Dtrf_j{}^l)/30$, $TtoK_j^l = TtrK_j{}^l$; иначе $TtrK_j{}^l = TtrN_j{}^l + Tf_j{}^l$.

Если $Dtof_j{}^l > 0$, то $TtoK_j^l = Tf_j{}^l(30 - Dtof_j{}^l)/30$, иначе $TtoK_j^l = TtoN_j{}^l + Tf_j{}^l$.

3) Формирование состояния оборудования на начало следующего месяца ($l+1$ – номер месяца).

$$TtoN_j^{l+1} = TtoK_j^l, \quad TtrN_j^{l+1} = TtrK_j^l, \quad TkrN_j^{l+1} = TkrK_j^l.$$

6.3. СТРУКТУРА БАЗЫ ДАННЫХ ДЛЯ ПЛАНИРОВАНИЯ ВЫПУСКА ГОТОВОЙ ПРОДУКЦИИ И СОСТАВЛЕНИЯ ГРАФИКОВ РЕМОНТА ОБОРУДОВАНИЯ

Основные элементы (сущности) базы данных: оборудование с характеристиками, план выпуска ГП, технологии выпуска каждой единицы ГП.

Оборудование с характеристиками. Это не только ремонтные характеристики (межремонтные циклы, время простоя при ремонте), но и все характеристики, интересующие все отделы предприятия, например для емкостного аппарата с мешалкой – это объем аппарата, тип мешалки, материал корпуса и др. Структура базы оборудования представлена на рис. 6.1.

Таблица "Характеристики" содержит перечень всех характеристик (свойств), которые встречаются в имеющемся оборудовании, например, объем аппарата, тип мешалки, межремонтный период КР и др. Для удобства использования эти характеристики можно классифицировать, например, выделить геометрические характеристики, ремонтные характеристики и др. (здесь классификация не показана).

Таблица "Значения_характеристик" содержит перечень возможных значений характеристики, например, для характеристики "тип мешалки" возможные значения "лопастная", "якорная", "рамная".

Таблицы "Типовое_оборудование" и "Значения_характеристик_типового_оборудования" позволяют осуществлять наследование при вводе оборудования завода (таблица "Оборудование") и его характеристик (таблица "Характеристики оборудования"), что значительно снижает трудозатраты поддержки базы. Полиморфизм достигается за счет ввода

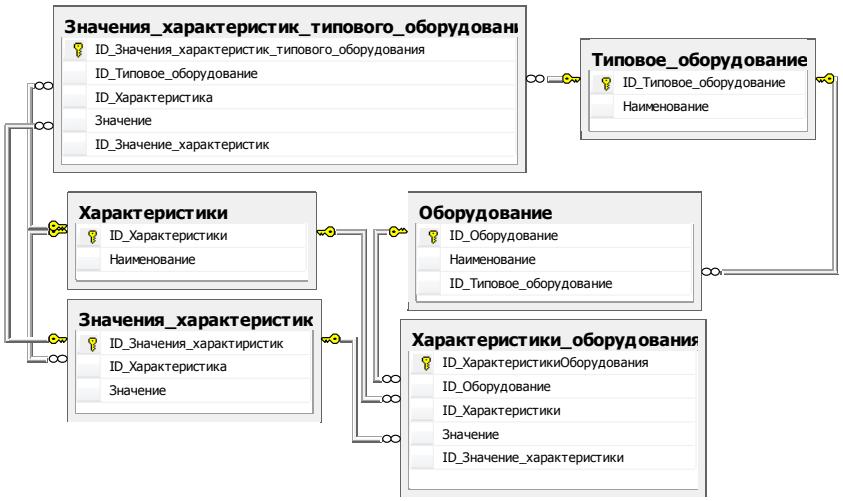


Рис. 6.1. Структура базы оборудования

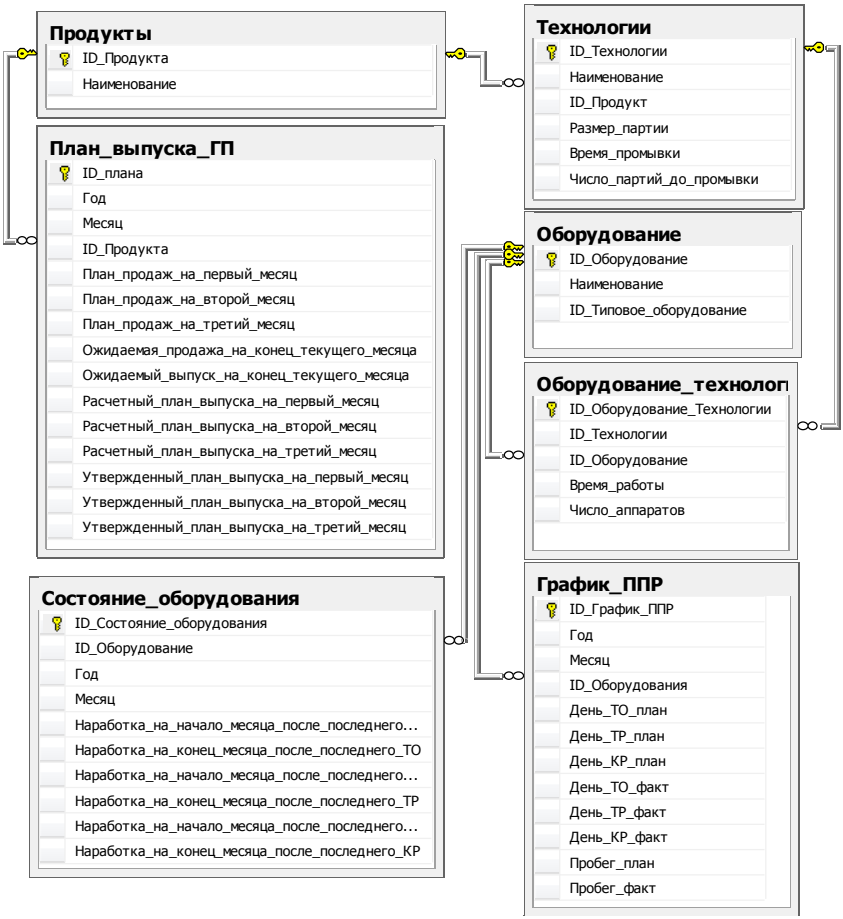


Рис. 6.2. Структура базы данных планирования выпуска ГП и составления графика ППР

для каждой единицы оборудования из таблицы "Оборудование" индивидуальных характеристик (таблица "Характеристики оборудования").

План выпуска и технологии выпуска каждой единицы ГП. Структура базы данных-диаграмма представлена на рис. 6.2.

Информация о технологиях выпуска хранится в таблицах "Технологии" и "Оборудование_технологии". Для одного продукта может быть задано несколько технологий. В таблице "Оборудование_технологии" находятся аппараты (ID_Оборудование) и время их работы для выпуска одной партии продукта (ID_Продукта) по заданной технологии (ID_Технологии).

7. ПРЕДСТАВЛЕНИЕ СТРУКТУРЫ ИЗДЕЛИЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ УПРАВЛЕНИЯ МАШИНОСТРОИТЕЛЬНЫМИ ПРЕДПРИЯТИЯМИ

Структура изделий, выпускаемых машиностроительными предприятиями, представлена в конструкторских спецификациях и является основной информацией, используемой практически всеми подзадачами информационной системы управления предприятием. Способы представления структуры изделия в едином информационном пространстве предприятия (ЕИПП) в значительной степени определяют работу таких модулей, как перспективное и оперативное планирование выпуска готовой продукции, материально-техническое снабжение, расчет плановой и фактической себестоимости и др.

Несмотря на то что ISO 10303 предлагает способы представления информации об изделии, реализация конкретных проектов требует решения практических задач по представлению структуры изделий в ЕИПП.

Разработке математического описания состава и свойств изделий посвящено значительное количество публикаций. В частности, в [31] предлагается использовать полихроматические множества для описания свойств изделий, способы структурного синтеза на основе различных графовых моделей представлены в [8], представление и проектирование семейства изделий рассматривается в [32]. Эти работы носят фундаментальных характер и не дают конкретных рекомендаций по представлению состава изделий в ЕИПП. В настоящем разделе предлагается структура представления информации об изделиях в ЕИПП машиностроительного предприятия в реляционной базе данных и в частности, представление групповой спецификации и спецификации изделий с переменной структурой.

7.1. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЕДИНОГО ИНФОРМАЦИОННОГО ПРОСТРАНСТВА МАШИНОСТРОИТЕЛЬНОГО ПРЕДПРИЯТИЯ

Информация о выпускаемых изделиях машиностроительного предприятия формируется на этапах конструкторской и технологической подготовки производства.

Результатом конструкторской подготовки является конструкторская документация, включающая структуру выпускаемых изделий (спецификация) и графическое представление элементов выпускаемых изделий (чертежи, 3D-модели).

Разработка технологической документации осуществляется в два этапа.

На первом этапе по конструкторской документации:

– для каждой детали определяется заготовка (лист, круг и др.), размер заготовки, маршрут изготовления детали (последовательность цехов или участков);

– для сборочных единиц задается маршрут изготовления;

– для покупных изделий задается признак "покупное".

Каждая деталь может иметь несколько возможных заготовок и несколько возможных маршрутов изготовления, каждая сборочная единица может иметь несколько возможных маршрутов сборки.

На втором этапе готовится вся технологическая документации в соответствии с действующими нормативными документами (расчет режимов резания, составление маршрутно-операционных карт и др.)

Первый этап выполняется оперативно и нужен для того, чтобы остальные службы предприятия (снабжение, комплектация) начали работу до появления полного комплекта технологических документов. Ниже речь пойдет только о первом этапе.

Основными элементами ЕИПП машиностроительного предприятия являются изделия (детали, сборочные единицы, стандартные, покупные) материалы, из которых изготавливаются изделия и подразделения. Структура базы данных этих элементов представлена рис. 7.1.

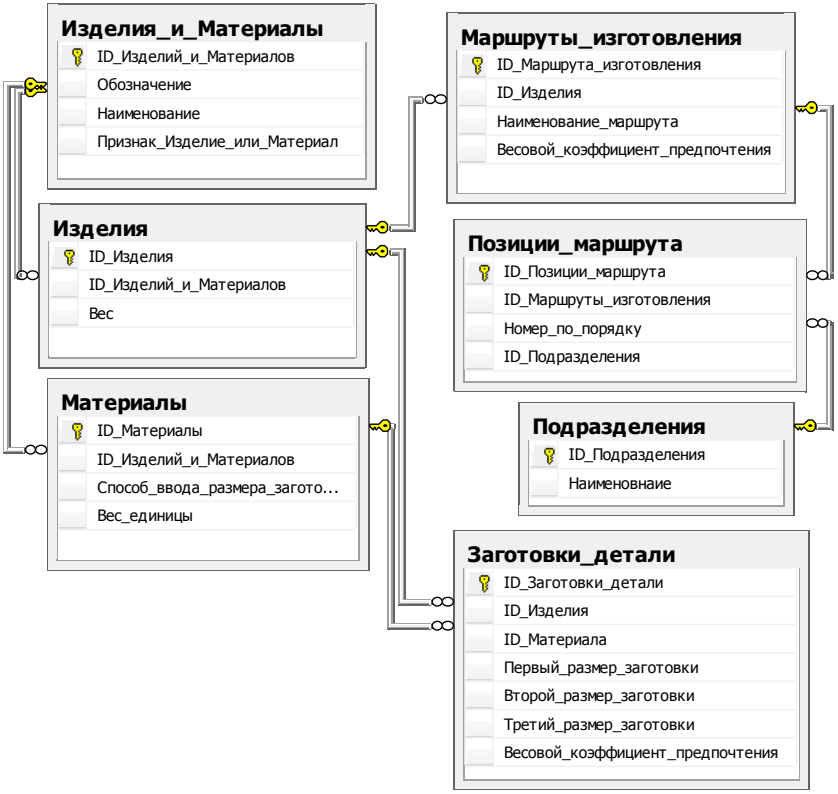


Рис. 7.1. Структура базы дагнных основных элементов ЕИПП

В таблице Изделия_и_материалы находятся как записи изделий (сборочные единицы, детали, стандартные изделия, покупные), так и записи о материалах, из которых эти изделия изготавливаются, в том числе и вспомогательные материалы (материалы на сварку, термообработку и др.). Объединение изделий и материалов в одной таблице позволяет для них иметь уникальный ID, что облегчает дальнейшее использование данных в задачах складского учета, расчета себестоимости и др., так как, например, в документах прихода не различаются изделия и материалы.

Далее таблица Изделия_и_материалы разбивается на таблицы Изделия и Материалы, которые и содержат основную информацию. Поле Способ_ввода_размера_заготовки в таблице Материалы определяет дальнейший интерфейс при вводе размера заготовки для детали. Например, заготовка из круга определяется длиной, заготовка из листа – длиной и шириной. Существуют следующие способы ввода: длина, длина-ширина, длина-ширина-высота, площадь, объем, масса, штуки.

Таблица Маршруты_изготовления содержит для каждой детали или сборочной единицы наименование маршрутов с весовыми коэффициентами предпочтения, например, если маршрут используется по умолчанию, то его весовой коэффициент предпочтения равен единице, для остальных маршрутов может быть ноль.

Таблица Позиции_маршрутов для каждого маршрута задает перечень подразделений, через которые проходит деталь или сборочная единица.

Таблица Заготовки_детали для каждой детали задает несколько возможных заготовок и их размеры.

7.2. СПОСОБЫ ПРЕДСТАВЛЕНИЯ СТРУКТУРЫ ИЗДЕЛИЙ МАШИНОСТРОИТЕЛЬНОГО ПРЕДПРИЯТИЯ

Классический способ представления спецификации. Структура базы данных, когда для каждой сборочной единицы составляется своя спецификация состоит, которая из таблицы изделий и таблицы спецификаций, рис. 7.2. Таблица изделий содержит сборочные единицы и детали как покупные, так и изготавливаемые на предприятии. В таблице Спецификации поле ID_изделия_родителя представляет изделие, для которого составлена спецификация (куда входит изделие потомок), Количество – определяет число входящих изделий.

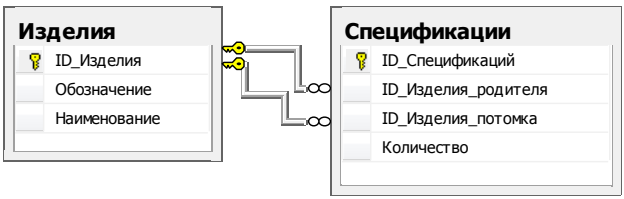


Рис. 7.2. Структура базы данных классических спецификаций

Групповая спецификация. Если номенклатура типоразмеров изделий, выпускаемых предприятием, большая и изделия сгруппированы по типам, причем изделия каждого типа имеют множество одинаковых деталей, то с целью уменьшения объема информации составляется групповая спецификация. В групповой спецификации имеется список деталей и сборочных единиц, входящих во все изделия группы (постоянные детали), и список деталей и сборочных единиц, входящих в отдельное изделие группы (переменные детали). Например, если в группу входит 10 изделий, каждое из которых содержит 100 деталей, причем 90 деталей входят во все изделия, то общее количество записей – 190. При составлении спецификации на каждое изделие общее количество записей в этом случае – 1000.

Структура базы данных для групповой спецификации представлена на рис. 7.3. Здесь таблица Групповые_спецификации содержит типы изделий, например, Насос НПЦ-32, Редуктор МРВ-2. Таблица Изделия_спецификаций содержит конкретные изделия, входящие в группу, заданную полем ID_Групповой_спецификации. Для насоса НПЦ-32 – это исполнения НПЦ.00.000, НПЦ.00.000-01, НПЦ.00.000-02, НПЦ.00.000-03 и т.д. Таблица Содержание_спецификации содержит перечень деталей и сборочных единиц, из которых состоят изделия из таблицы Изделия_спецификаций. Если поле ID_Изделия_родителя не определено или ноль, то деталь постоянная, в противном случае деталь принадлежит изделию, заданному этим полем.

Групповая спецификация с полем принадлежности. Рассмотренный способ составления групповой спецификации эффективен тогда, когда число изделий группы небольшое. Если, например, число изделий группы 100 и каждое изделие состоит из 100 деталей, и 90 деталей постоянных, то общее число записей групповой спецификации 1090. Между тем, оставшиеся переменные детали могут встречаться в нескольких изделиях. Например, для группы изделий, имеющих конструкторское обозначение 700.100.01, 700.100.02, …, 700.100.50, 700.200.01, 700.200.02, …, 700.200.50, групповая спецификация может иметь следующий вид (табл. 7.1).

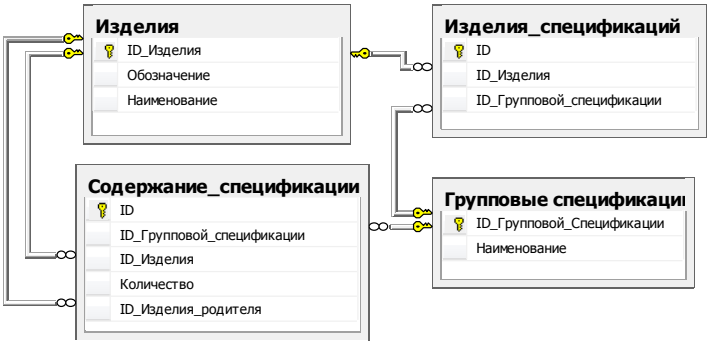


Рис. 7.3. Структура базы данных групповых спецификаций

7.1. Групповая спецификация

Обозначение	Наименование	Количество
Постоянные детали		
700.400.01	Шестерня	2
700.500.00	Хвостовик	1
…	…	…
Переменные детали		
Детали, входящие в изделия 700.100.01, 700.100.02, …, 700.100.10		
700.300.01	Крышка	1
700.310.01	Прокладка	2
…	…	…
Детали, входящие в изделия 700.100.11, 700.100.12, …, 700.100.20		
700.300.02	Крышка	1
700.310.02	Прокладка	2
…	…	…

В автоматизированной системе подготовки конструкторской документации предлагается в стандартную спецификацию добавить новую графу, которая будет определять принадлежность детали к тому или иному изделию. Запись в эту графу производится по следующим правилам.

Сначала каждой лексеме обозначения изделия даются имена. Например Z1, Z2 и т.д. Тогда обозначения изделий 700.100.01, 700.100.02, …, 700.100.50 записываются как Z1.Z2.Z3. Соответственно запись `0<Z3<11 и Z2=100` обозначает изделия, у которых второе поле обозначения – 100, а третье больше нуля и меньше 11, т.е. изделия 700.100.01, 700.100.02, …, 700.100.10. Запись `Z3=11 или Z3=15` обозначает изделия 700.100.11, 700.100.15, 700.200.11, 700.200.15. Таким образом, спецификация, представленная выше, примет вид (табл. 7.2).

7.2. Групповая спецификация с графой принадлежности

Обозначение	Наименование	Количество	Принадлежность
700.400.01	Шестерня	2	Постоянная
700.500.00	Хвостовик	1	Постоянная
…	…	…	…
700.300.01	Крышка	1	0<Z3<11 и Z2=100
700.310.01	Прокладка	2	0<Z3<11 и Z2=100
…	…	…	…
700.300.02	Крышка	1	10<Z3<21 и Z2=100
700.310.02	Прокладка	2	10<Z3<21 и Z2=100
…	…	…	…

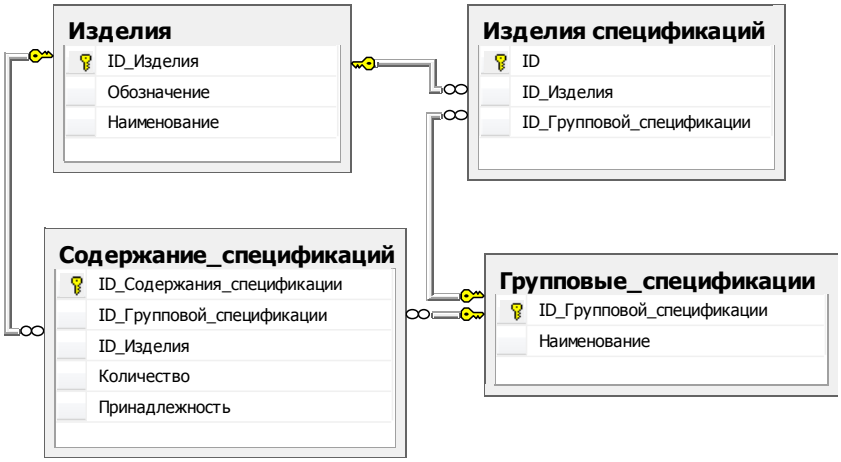


Рис. 7.4. Структура базы данных групповых спецификаций с полем принадлежности

На первый взгляд составление спецификации в таком виде может показаться трудоемким, однако эта спецификация не предназначена для ручного ввода. Ввод поля принадлежности в автоматизированной системе при наличии операции копирования или ввода по шаблону осуществляется нажатием всего одной клавиши. Структура базы данных в этом случае принимает вид, рис. 7.4.

Предложенное представление групповой спецификации включает в себя описанные выше первые два способа и позволяет значительно сократить объем вводимой информации и унифицировать программный код АСУП.

Спецификация изделия в автоматизированных системах управления предприятием используется в алгоритмах разузлования при нормировании материалов, планировании работы цехов и т.д. Недостатком предложенного способа задания спецификации является отсутствие ID_Изделия_Родителя, что, несомненно, увеличит время работы алгоритма разузлования.

Предложенный метод был использован в автоматизированной системе технологической подготовки производства ЗАО "Завод Тамбовполимермаш". Трехлетняя эксплуатация показала, что для редукторного производства (число деталей в изделиях порядка 100) замедление работы алгоритма разузлования не ощущается. База данных реализована в MS-SQL–2005, алгоритм – разузлования в виде хранимой процедуры.

Рис. 7.5. Структура базы данных групповых спецификаций с полем принадлежности и вторичной спецификацией

Для больших изделий (с числом деталей несколько тысяч) можно использовать дополнительно таблицу Вторичные_спецификации, рис. 7.5.

Записи таблицы Вторичные_спецификации создаются программно из таблиц Содержание_спецификаций и Изделия_спецификаций при вводе (удалении, редактировании) записей в таблицу Содержание_спецификаций. Таблица Вторичные_спецификации содержит ID_Изделия_родителя и алгоритм разузлования может быть применен к ней. Поле ID_Содержание_спецификаций позволяет автоматически поддерживать таблицу Вторичные_спецификации через таблицу Содержание_спецификаций.

Спецификация изделий с взаимозаменяемыми элементами [*18*]. Рассмотрим изделие а1, которое состоит из деталей а2 и а5 и двух сборочных единиц а3, а6 или а4, а7. Ниже представлены спецификации всех сборочных единиц.

Спецификация a1

Позиция	Обозначение	Количество
1	a2	1
2	a3 или a4	2
3	a5	3
4	a6 или a7	1

Спецификация a2

Позиция	Обозначение	Количество
1	a8	2
2	a9	2

Спецификация a4

Позиция	Обозначение	Количество
1	a10	2
2	a11	1

Спецификация a6

Позиция	Обозначение	Количество
1	a12 или a13	1
2	a14	2

Спецификация a7

Позиция	Обозначение	Количество
1	a15	2
2	a16	1

Спецификация a9

Позиция	Обозначение	Количество
1	a17 или a18	2
2	a19	1

На рисунке 7.6 приведено дерево изделия a1, составленное из спецификаций сборочных единиц, входящих в a1.

Взаимозаменяемость элементов определяется следующими правилами:

Правило 1. Если a3, то a6.

Правило 2. Если a4, то a7.

Правило 3. Если a17, то a12.

Правило 4. Если a18, то a13.

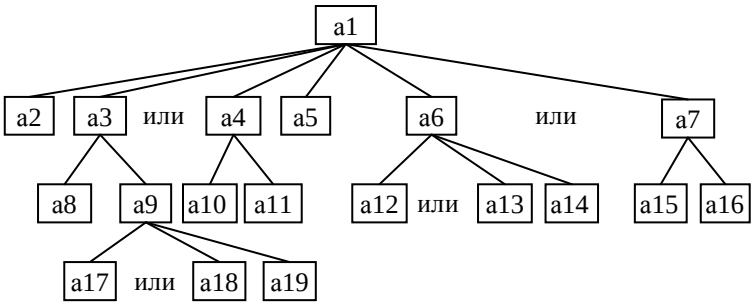
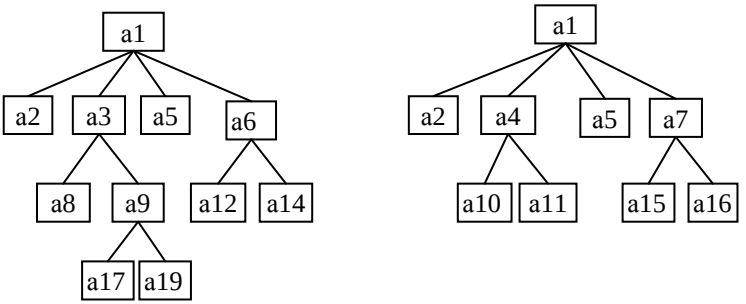


Рис. 7.6. Дерево изделия a1



а) *б*)

Рис. 7.7. Возможные варианты изделия а1

Для изготовления изделия а1 необходимо однозначно задать позиции с взаимозаменяемыми элементами. Исходными данными для составления такой спецификации является наличие или отсутствие элементов а3, а4, а17, а18 в конечном изделии. При этом возможны разные варианты. На рисунке 7.7. представлены деревья, полученные после применения правил 1 – 4 для следующих исходных данных:

– а3, а17 присутствуют в изделии а1, рис. 7.7, *а*;

– а4 присутствует в изделии а1, рис. 7.7, *б*.

Структура базы данных спецификации изделий с взаимозаменяемыми элементами представлена на рис. 7.8. Таблица Изделия содержит все сборочные единицы и детали как покупные, так и изготавливаемые на предприятии. В таблице Спецификации поле ID_Изделия_родителя представляет изделие, для которого составлена спецификация (куда входит изделие потомок). Для альтернативных позиций ID_Изделия_потомка будет равно нулю. В этом случае потомки определяет таблица ИЛИ_позиции, в которой для одной позиции спецификации задается несколько изделий потомков. В таблице Правила для альтернативных позиций спецификаций в поле Текст_на_SQL задается правило взаимозаменяемости элементов в формате SQL. Поле Текст_на_ЕЯ содержит правило на естественном языке, например "Если А17, то А12". Поля ID_Определяющего_элемента и ID_Определяемого_элемента необходимы для того, чтобы в дальнейшем выбирать для обработки только те правила, которые могут иметь отношение к раскрываемому изделию.

Алгоритм составления дерева изделия состоит из следующих шагов:

– шаг 1. Выбор корневого элемента из таблицы Изделия;

– шаг 2. Построение дерева изделия по таблице Спецификации без взаимозаменяемых элементов;

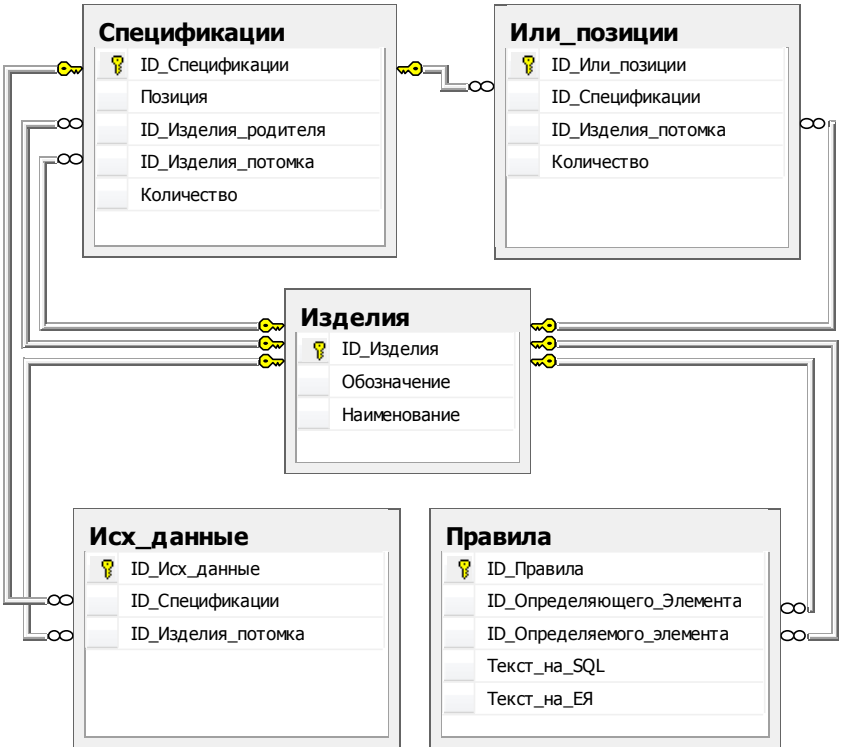


Рис. 7.8. Структура базы данных спецификации изделий с взаимозаменяемыми элементами

– шаг 3. Добавление в дерево взаимозаменяемых элементов из таблицы Или_потомки;

– шаг 4. Удаление из дерева записей с нулевым значением ID_Изделия_Потомка.

Ниже представлен листинг рекурсивной программы в формате Transact-SQL (релиз SQL фирмы Microsoft) построения дерева изделия (для изделия с ID_Изделия=4). Здесь Ключ_сортировки – символьная строка, полученная соединением ID всех родителей текущего изделия, разделенных точкой. Результат помещается в таблицу #Derevo. Поле Уровень позволяет контролировать глубину вложенности потомков для предотвращения зацикливании. Ниже представлен программный код реализующий предложенный алгоритм.

```
with tree (ID_Изделия, ID_Изделия_родителя, ID_Изделия_потомка,
        Количество, Уровень, Ключ_сортировки, ID_Спецификации)
  as
     /* Выбор корневого изделия из таблицы "Изделия" */
     (select  ID_Изделия, 0, ID_Изделия, 1, 0,
         cast (ID_Изделия as varchar(max)), 0
         from Изделия where ID_Изделия=1
     union all
     /* Построение дерева без взаимозаменямых элементов */
      select t.ID_Изделия, a.ID_Изделия_родителя,
            a.ID_Изделия_потомка, a.Количество,
            case a.ID_Изделия_потомка when 0
                 then t.уровень
                 else t.уровень+1
            end,
            case a.ID_Изделия_потомка when 0
               then t.Ключ_сортировки
               else t.Ключ_сортировки+'.'+
                     cast(a.ID_Изделия_потомка as varchar(5))
           end,
           a.ID_Спецификации
       from Спецификации as a
       inner join tree as t on
            a.ID_Изделия_родителя=t.ID_Изделия_потомка
     union all
     /* Добавление взаимозаменяемых элементов */
      select t.ID_Изделия, t.ID_Изделия_родителя,
             i.ID_Изделия_потомка, i.Количество, t.уровень+1,
             t.Ключ_сортировки+'.'+
                  cast(i.ID_Изделия_потомка as varchar(5)),
             t.ID_Спецификации
      from tree as t
      inner join  dbo.Или_позиции as i on
             t.ID_Спецификации=i.ID_Спецификации
      where t.ID_Изделия_потомка=0
     )
     select  ID_Изделия, ID_Изделия_родителя,
             ID_Изделия_потомка,  Количество, Уровень,
             Ключ_сортировки, ID_Спецификации into #Derevo
             from tree  where ID_Изделия=1
             order by  Ключ_сортировки
     /* Удаление записей с нулевым ID_Изделия_потомка */
     delete from #Derevo where  ID_Изделия_потомка=0
```

Ниже представлены таблицы описанных выше изделий a1, a2, a4, a6, a7, a9 и дерево изделия, полученное после выполнения программного.

Изделия

ID_Издел.	Обозн.
1	a1
2	a2
3	a3
4	a4
5	a5
6	a6
7	a7
8	a8
9	a9
10	a10
11	a11
12	a12
13	a13
14	a14
15	a15
16	a16
17	a17
18	a18
19	a19
20	a20

Спецификации

ID_Специф.	Позиция	ID_Изд._родителя	ID_Изд._потомка
1	1	1	2
2	2	1	0
3	3	1	5
4	4	1	0
5	1	3	8
6	2	3	0
7	1	4	10
8	2	4	11
9	1	6	0
10	2	6	14
11	1	7	15
12	2	7	16
13	1	9	0
14	2	9	19

Исх_данные

ID_Исх_Дан.	ID_Специф.	ID_Изд._потомка
1	2	3
2	13	17

Или_позиции

ID_Или позиции	ID_Специф.	ID_Изд._потомка
1	2	3
2	2	4
3	4	6
4	4	7
5	9	12
6	9	13
7	13	17
8	13	18

#Derevo

ID_Изделия_ родителя	ID_Изделия_ потомка	Уровень	Ключ_ сортировки	ID_Специ-фикации
0	1	0	1	0
1	2	1	1.2	1
1	3	1	1.3	2
3	8	2	1.3.8	5
3	9	2	1.3.9	6
9	17	3	1.3.9.17	13
9	18	3	1.3.9.18	13
9	19	3	1.3.9.19	14
1	4	1	1.4	2
4	10	2	1.4.10	7
4	11	2	1.4.11	8
1	5	1	1.5	3
1	6	1	1.6	4
6	12	2	1.6.12	9
6	13	2	1.6.13	9
6	14	2	1.6.14	10
1	7	1	1.7	4
7	15	2	1.7.15	11
7	16	2	1.7.16	12

Дальнейшая обработка таблицы #Derevo заключается в удалении альтернативных позиций по исходным данным, выполнении правил из таблицы Правила для каждой альтернатавной позиции.

```
 / * Удаление альтернативных элементов по исходным данным */
select Ключ_сортировки into #a from #derevo a, Исх_данные b
      where a.ID_Спецификации=b.ID_Спецификации and
      a.ID_Изделия_потомка<>b.ID_Изделия_потомка
delete from #derevo where Ключ_сортировки in
  select a.Ключ_сортировки from #derevo a
  inner join #a b on a.Ключ_сортировки Like b.Ключ_сортировки +'%')
```

Представления правил взаимозаменяемости элементов в формате SQL. Рассмотрим синтаксис правила на примере "Если a17, то a12".

В таком виде в реальной системе правило применять нельзя, так как в других изделиях (не а1) оно может не работать. В настоящем примере правило формулируется следующим образом: "Если в дереве изделия присутствует элемент а17, входящий в а9, и присутствует элемент а12, входящий в а6, то на позиции 1 изделия а6 надо оставить элемент а12". Оставить а12 – значит удалить с позиции 1 изделия а6 все кроме а12.

С учетом сказанного, текст правила на SQL будет следующим.

```
if exists (select * from #derevo where ID_Спецификации=13
            and ID_Изделия_Потомка=17 and
            Ключ_сортировки like '%.'+ltrim(str(9))+'.%' )
   and exists (select * from #derevo where ID_Спецификации=9
            and ID_Изделия_Потомка=12 and
            Ключ_сортировки like '%.'+ltrim(str(6))+'.%' )
   delete from #derevo where Ключ_сортировки in
          (select   a.Ключ_сортировки from #derevo a
            inner join (select Ключ_сортировки  from #derevo
            where ID_Спецификации=9 and
            ID_Изделия_потомка<>12) b on
                 a.Ключ_сортировки Like b.Ключ_сортировки +'%')
```

Аналогично можно составить тексты других правил.

ЗАКЛЮЧЕНИЕ

В последнее десятилетие непрерывно растет интерес к автоматизированным интеллектуальным информационным системам (АИИС) и их применению в различных областях человеческой деятельности. Как и любая автоматизированная информационная система АИИС имеет математическое и программное обеспечение. В настоящей работе автор уделил особое внимание использованию различных видов графов с ограничениями для математического описания объектов различной природы. В качестве среды для представления графовых структур в автоматизированных информационных системах используются реляционные базы данных, так как они в настоящее время наиболее распространены для хранения структурированной информации в системах различного назначения (управления, проектирования и др.).

В работе представлено математическое описание ограничений графовых структур и трансформация этого описания в описание на языке структурированных запросов SQL. Рассмотрены основные операции с графами в реляционных базах данных, которые выполняются с учетом доменной и ссылочной целостности базы данных. Описаны способы представления правил вида "Если А, то В" в реляционной базе данных и алгоритмы обработки этих правил.

Введенное в начале работы теоретическое описание N-ориентированного гиперграфа использовано в дальнейшем для разработки математического и программного обеспечения информационных систем, предназначенных для:

– размещения объектов различной природы в пространстве;

– структурного и параметрического синтеза технических объектов;

– планирования выпуска готовой продукции и ремонта оборудования в многоассортиментных химических производствах;

– описания структуры технических объектов, включая объекты с переменной структурой.

Данная монография – результат многолетней работы автора в области создания информационных систем автоматизированного управления и проектирования.

Автор надеется, что представленный в работе материал будет полезен студентам, аспирантам и специалистам, которые занимаются разработкой математического и программного обеспечения автоматизированных информационных систем.

СПИСОК ЛИТЕРАТУРЫ

1. Д.И. Батищев, Н.В. Старостин, А.В. Филимонов. Многоуровневая декомпозиция гиперграфовых структур [Текст]; Приложение к журналу Информационные технологии, 2008. – № 5. – 34 с.

2. Биряльцев Е.В., Гусенков А.М. Онтологии реляционных баз данных. Лингвистический аспект [Текст] // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2007» (Бекасово, 30 мая – 3 июня 2007 г.). – Изд-во РГГУ, 2007. – С. 50 – 53.

3. Божко А.Н., Толпаров А.Ч. Структурный синтез на элементах с ограниченной сочетаемостью [Электронный ресурс] // Наука и образование: электронное научное издание. – 2004. – № 5. – Режим доступа: http://technomag.edu.ru/doc/44191.html – Загл. с экрана.

4. Дмитриевский Б.С. Автоматизированные информационные системы управления инновационным наукоемким предприятием [Текст] М.: Машиномтроение-1, 2006. – 156 с.

5. Егоров С.Я., Немтинов В.А., Мокрозуб В.Г. и др. Автоматизированная информационная система поддержки проектных решений по компоновке промышленных объектов. Часть 1. Аналитические и процедурные модели [Текст] // Информационные технологии в проектировании и производстве. – 2009. – № 4. – С. – 3 – 11.

6. Егоров С.Я., Немтинов В.А., Мокрозуб В.Г. и др. Автоматизированная информационная система поддержки проектных решений по компоновке промышленных объектов. Часть 2. Структура и функционирование системы [Текст] // Информационные технологии в проектировании и производстве. – 2010. – № 1. – С. 33 – 39.

7. Иванова Г.С. Модели объектов задач структурного синтеза [Электронный ресурс] // Наука и образование: электронное научное издание. – 2006. – № 12. – Режим доступа: http://technomag.edu.ru/doc/62361.html – Загл. с экрана.

8. Иванова Г.С. Способы представления структурных моделей [Электронный ресурс] // Наука и образование; электронное научное издание. – 2007. – № 1. – Режим доступа: http://technomag.edu.ru/doc/62742.html – Загл. с экрана.

9. Курбатов С.С. Автоматизированное построение естественно-языкового интерфейса для реляционных баз данных [Текст] // Новости искусственного интеллекта. – 2002. – № 2. – С. 17 – 21.

10. Мещеряков С.В., Иванов В.М. Эффективные технологии создания информационных систем [Текст] – СПб.: Политехника, 2005. – 309 с.

11. Мокрозуб В.Г., Егоров С.Я., Немтинов В.А. Процедурные и информационно-логические модели ППР [Текст] // Информационные технологии в проектировании и производстве. – 2009. – № 2. – С. 72 – 76.

12. Мокрозуб, В.Г. Представление структуры изделий в реляционной базе данных [Текст] // Информационные технологии. – 2008. – № 11. – С. 11 – 13.

13. Мокрозуб, В.Г. Синтаксис запросов конечных пользователей к реляционной базе данных [Текст] // Прикладная информатика. – 2009. – № 3(21). – С. 95 – 99.

14. Мокрозуб В.Г., Немтинов В.А., Егоров С.Я. и др. Применение гиперграфов и реляционной базы данных для описания структуры радиотехнических систем [Текст] // Успехи современной радиоэлектроники. – 2009. – № 11. – С. 37 – 41.

15. Мокрозуб В.Г., Немтинова Ю.В., Морозов С.В. Организация диалога с пользователями в информационных системах менеджмента предприятия [Текст] // Вестник Тамбовского государственного технического университета. – 2010. – Т. 16, № 2. – С. 288 – 295.

16. Мокрозуб В.Г., Егоров С.Я., Немтинов В.А. Процедурные и информационно-логические модели планирования выпуска продукции и ремонтов технологического оборудования многоассортиментных производств [Текст] // Информационные технологии в проектировании и производстве. – 2009. – № 2. – С. 72 – 76.

17. Мокрозуб В.Г., Мариковская М.П., Красильников В.Е. Интеллектуальная автоматизированная система проектирования химического оборудования [Текст] // Системы управления и информационные технологии. – 2007. – № 4.2(30). – С. 264 – 267.

18. Мокрозуб В.Г., Сердюк А.И., Шамаев С.Ю. и др. Представление модели параметрического синтеза технического объекта в реляционной базе данных [Текст] // Вестник Тамбовского государственного технического университета. – 2011. – Т. 17, № 2. – С. 462 – 466.

19. Мокрозуб В.Г. Представление структуры изделий в информационных системах управления машиностроительными предприятиями [Текст] // Вестник компьютерных и информационных технологий. – 2009. – № 10(64). – С. 30 – 34.

20. Мокрозуб В.Г. Интеллектуальные информационные системы автоматизированного конструирования технологического оборудования.– Тамбов: Издательский дом ТГУ им. Г.Р. Державина, 2011. – 128 с.

21. Мокрозуб В.Г., Немтинов В.А., Мордвин А.С. и др. Применение N-ориентированных гиперграфов и реляционных баз данных для структурного и параметрического синтеза технических систем [Текст] // Прикладная информатика. – 2010. – № 4(28). – С. 115 – 122.

22. Мокрозуб В.Г.Основные операции над N-ориентированными гиперграфами в реляционной базе данных // Программные продукты и системы. – 2011. – № 1. – С 59 – 64.

23. Мокрозуб В.Г., Сердюк А.И., Каменев С.В. и др. Представление структуры технических объектов с взаимозаменяемыми элементами в виртуальных моделях [Текст] // Вестник Тамбовского государственного технического университета. – 2011. – Т. 17. – № 2. – С. 467 – 471.

24. Мокрозуб В.Г. Представление ориентированных ультра- и гиперграфов с ограничениями в реляционной базе данных [Текст] // Научно-техническая информация. Серия 2: информационные процессы и системы. – 2011. – № 3. – С. 17 – 24.

25. Мокрозуб В.Г. Таксономия в базе данных стандартных элементов технических объектов [Текст] // Информационные системы. – 2009. – № 11(159). – С. 18 – 22.

26. Овчинников В.А. Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем [Электронный ресурс] // Наука и образование. – 2009. – № 10. – Режим доступа: http://technomag.edu.ru/doc/132769.html Загл. с экрана.

27. Овчинников В.А. Математические модели объектов задач структурного синтеза [Электронный ресурс] // Наука и образование. – 2009. – № 3. – Режим доступа: http://technomag.edu.ru/doc/ 115712.html – Загл. с экрана.

28. Овчинников В.А. Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем (Ч. 2) [Электронный ресурс] // Наука и образование. – 2009. – № 11. – Режим доступа: http://technomag.edu.ru/doc/133223.html – Загл. с экрана

29. Павлов В.В. CALS-технологии в машиностроении (математические модели) [Текст]. – М.: ИЦ МГТУ Станкин, 2002. – 328 с.

30. Павлов В.В. Полихроматические графы в теории систем [Текст] // Информационные технологии. – 1998. – № 6. – С. 2 – 9.

31. Павлов В.В. Структурное моделирование в CALS-технологиях [Текст]; Отв. ред. Ю.М. Соломенцев; Ин-т конструкторско-технологической информатики РАН. – М.: Наука, 2006. – 307 с.

32. Третьяков В.М. Основы методики проектирования семейства изделий // Автоматизация и современные технологии. – 2004. – № 2. – С. 25 – 33.

ОГЛАВЛЕНИЕ