УДК 621.7, 681.31

# О МЕТОДЕ СОЗДАНИЯ ОБЪЕКТНО-ОРИЕТИРОВАННЫХ БАЗ ДАННЫХ САПР

**Кочуров В.А., Павловский М. С.**
*Белорусский национальный технический университет*
*Минск, Беларусь*

Обеспечение надежного управления всем объемом разнородных данных, которые порождаются, хранятся и используются в различных информационных системах, осуществляющих информационную поддержку продукции в течение ее жизненного цикла, является актуальной задачей для современных предприятий.

В отличие от бумажного документооборота и простейших форм электронного документооборота, основанного на использовании электронных образов бумажных документов, в рамках CALS (Continuous Acquisition and Life Cycle Support) используются интегрированные информационные модели продукции и процессов — объекты, не имеющие прямых аналогов в традиционном бумажном документообороте: конструкторские данные, технологические данные, производственные данные, данные о качестве изделия, логистические данные и эксплуатационные данные об изделии.

Для организации этой информации был разработан и получил широкое применение международный стандарт ISO 10303 STEP (ГОСТ Р ИСО 10303), отличительной методологической особенностью которого является объектная ориентированность, т.е. он оперирует понятиями информационных объектов. Для работы с такими объектами служат специальные объектно-ориентированные системы управления базами данных (ООБД), однако применение последних приводит к ряду проблем.

Во-первых, ООБД являются пока немногочисленными, причем не существует ни одной некоммерческой ООБД. Кроме того, очень много информации уже накоплено в реляционных СУБД и конвертация этой информации для представления ее посредством ООБД будет очень дорога и нерентабельна. В тоже время для реализации объектной модели средствами реляционных СУБД не требуется прилагать сверхусилий.

На рис.1 представлена, разработанная на кафедре САПР БНТУ логическая схема реляционной базы данных, в которую можно отобразить произвольную совокупность информационных объектов.

Предполагается, что база данных отображает определенную предметную область, состоящую из перечислимого множества объектов, причем

объекты могут быть простыми (неделимыми) объектами или объектами – контейнерами, содержащими в себе другие объекты. Выполним построение базы данных посредством следующих шагов:

· присвоим каждому объекту уникальный идентификатор Id, исполняющего роль уникального системного идентификатора, присвоим идентификатор Name — содержательное имя, с которым будет оперировать конечный пользователь и необязательный идентификатор Description — пользовательское описание объекта. Занесем эти данные в таблицу объектов Object и определим в качестве первичного ключа поле Id, а на поле Name наложим ограничение Unique;

· каждый объект этой предметной области характеризуется определенным набором атрибутов. Разобьем все объекты на классы (типы) по общности состава атрибутов. Присвоим каждому классу уникальный идентификатор Id и неформальное имя для конечного пользователя Name. Занесем эти данные в таблицу классов Class и зададим первичный ключ по полю Id, а ограничение Unique — по полю Name;

· каждый атрибут характеризуется именем, содержанием, а также типом значений: числовым, символьным и т.п. Присвоим каждому атрибуту уникальный идентификатор в качестве системного имени Id, неформальное имя для конечного пользователя Description, обязательное условное обозначение Symbol и указание базового типа значений — символьный, числовой и т.п — в поле Kind. Занесем эти данные в таблицу атрибутов Attribute. Накладываем ограничение Unique на поля Kind, Symbol, Description.

· каждый класс может содержать некоторое количество атрибутов, и каждый атрибут может содержаться в более чем одном классе. Данное замечание позволяет сделать вывод о том, что классы и атрибуты находятся в отношении n:m. Это требует создания еще одной таблицы Structure для установления связи между классами и атрибутами, что позволит произвести в дальнейшем нормализацию. В качестве уникального идентификатора создадим поле a Id, поле Att_Id, которое будет являться внешним ключом по отношению к полю Id таблицы атрибутов и которое может выполнять роль имени поля таблицы в реляционной модели; поле Class_Id, которое будет являться внешним ключом по отношению к полю Id таблицы классов и которое может выполнять роль имени таблицы в реляционной модели; поле In_Order - порядковый номер атрибута в составе данного класса.

· отношение между именем атрибута и его значениями не всегда является равенством для информационных объектов САПР. Во многих случаях ими могут служить такие предикаты, как «находится в интервале», «следовать за или предшествовать», «находиться в отношении наложения» (напри-

мер, для обрабатываемых поверхностей) и т.п. Поэтому введем в наша базу данных таблицу Predicate с двумя полями Id и Name. Первое из полей будет содержать уникальный системный идентификатор предиката, а второе — его общепринятое обозначение;

· создадим таблицу Domain_Set, представляющую домен значений, которые могут принадлежать атрибутам, с полями Id и Name. Первое из них будет содержать уникальный системный идентификатор значения домена, а второе — собственно значение. Символьные значения могут быть переменной длины и представлять сложные лексемы. Создадим первичный индекс по полю Id и ограничение Unique по полю Name. Данная таблица содержит все значения атрибутов в символьном виде, даже если они обладают типом, отличным от строкового – числовым, вещественным, дата-время и т.д. Соответствующие преобразования и проверки осуществляются на клиенте. Возможна реализация модели с несколькими таблицами для хранения значений атрибутов — для каждого типа своя таблица. Но этот подход несколько усложняет реализацию, а потому он здесь не рассматривается.

· наконец, создадим последнюю таблицу в логической схеме нашей базы данных — таблицу связей (фактов) Relation содержащую поля: Id, Str_Id, Obj_Id, Pred_Id, Dom_Id, Element.

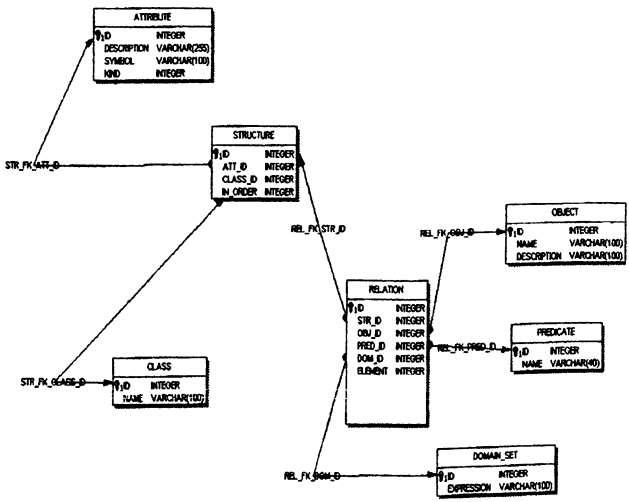


*Рис. 1. Логическая схема базы данных*

Поле Id является уникальным идентификатором в качестве системного имени.

Поле Str_Id «Идентификатор структуры» является внешним ключом по отношению к полю Id таблицы структур (Structure). Однозначно идентифицирует параметры атрибута в составе класса.

Поле Obj_Id «Идентификатор объекта» является внешним ключом по отношению к полю Id таблицы объектов (Object) и однозначно идентифицирует объект.

Поле Pred_Id «Идентификатор предиката» содержит ключ записи таблицы предикатов, идентифицирующий значение последнего.

Поле Dom_Id «Идентификатор значения» содержит ключ записи домена, содержащего данное значение.

Поле Element «Идентификатор элемента» содержит номер элемента (экземпляра типа элемента) в составе объекта — контейнера, хотя может применяться и любая другая уникальная в пределах одного объекта идентификация. Значение этого поля может исполнять роль физического номера строки таблицы в реляционной модели. Символьное представление нулевого значения соответствует атрибутам, характеризующих объект как единое целое.

Для таблицы связей Relation создаются следующие индексы:

· первичный индекс создается по полю Id и обеспечивает уникальность каждого элемента данных;

· уникальный индекс создается по полям Str_Id, Obj_Id, Pred_Id, Dom_Id, Element для обеспечения логической целостности данных, хранящихся в таблице Relation;

Повторяющееся значение в поле Obj_Id определяет всю группу значений признаков, относящихся к одному объекту. Повторяющееся значение в поле Class_Id таблицы Structure определяет все наборы данных одного структурного типа. Значение поля Att_Id таблицы Structure, с одной стороны позволяет проинтерпретировать любое данное, а с другой — выделить определенное свойство, присущее различным типам структурных объектов. Совместное задание Obj_Id, Str_Id позволяет выделить набор значений одноименного свойства всех экземпляров определенного элемента в составе заданного объекта.

На крупных предприятиях имеется множество отделов, организованных по принципу общности назначения работы — «Отдел главного технолога», «Отдел главного конструктора» и т.д. Представляется нецелесообразным ведение единой БД для всех отделов, т.к. это может значительно снизить скорость работы. Предлагается создание отдельных БД для каждого отдела. При необходимости получить информацию из БД другого отдела, клиент может

подключиться к любой другой БД этого этой структуры (при наличии у него прав для этого). Для централизации данного процесса предполагается создать дополнительный программный сервер, который будет знать о месторасположении всех работающих БД и краткое описание их содержимого.

Для осуществления обмена информации между различными БД, построенных на данной технологии, предполагается использовать формат XML.

К настоящему моменту на кафедре САПР БНТУ создано две реализации изложенного подхода: первая реализация выполнена в виде COM-сервера в средеVisual FoxPro, а вторая в форме трехзвенной информационной системы с базой данных в Interbase и клиент-серверной частью в Delphi.

УДК 519.63

## ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ НАПРЯЖЕННОГО СОСТОЯНИЯ МАССИВА ГОРНЫХ ПОРОД С ВЫРАБОТКОЙ

**Журавков М.А., Напрасникова Ю.В.**
*Белорусский государственный университет*
*Минск, Беларусь*

При проведении горных работ в окрестности выработки развивается такое напряженно-деформированное состояние (НДС) породы, при котором могут возникнуть трещины или произойти обрушение кровли выработки [1]. Поскольку при возникновении трещиноватости в область выработки могут проникать грунтовые воды, а обрушение кровли увеличивает опасность эксплуатации выработки, то необходимо уметь рассчитывать НДС в окрестности выработки.

В данной работе предлагается построение расчетной схемы и решение задачи в упругой постановке. На основе этой задачи с помощью существующих алгоритмов можно решить задачи, более приближенные к реальным, например, учесть реологические эффекты.

Постановка задачи

В настоящее время основным месторождением по добыче калийных солей в Беларуси является Старобинское месторождение.

Проанализировав геологический профиль этого месторождения, были выделены участки, имеющие одинаковую структуру. Поскольку выработка протяженная, то можно ввести предположение о плоском деформируемом состоянии и составить соответствующую расчетную схему.