УДК 629.78.018

П.А. ЛУЧШЕВ

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

РЕЛЯЦИОННАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ ИСПЫТАНИЙ

Рассмотрены теоретические аспекты применения баз данных для хранения описания технологических процессов испытаний сложных технических систем. Приведена формальная модель баз данных, которая с учетом практических особенностей использована в описании технологических процессов испытаний системы энергоснабжения космических аппаратов.

испытания, технологический процесс испытаний, реляционная модель данных ТП испытаний, формализация ТП испытаний

Введение

Для создания программного обеспечения автоматизации испытаний сложных технических систем приходится сталкиваться с задачами разного плана. С одной стороны для обеспечения достаточной гибкости работы все данные, имеющиеся в такой системе, должны быть представлены как объекты произвольной структуры. С другой стороны для минимизации объемов данных и в частности уменьшения дублирования информации все данные, имеющиеся в такой системе, должны быть представлены в виде реляционных переменных.

Концепции реляционного и объектного подхода абсолютно не противоречат друг другу и не требуют каких-либо изменений для того, что бы использоваться в общей системе, обладающей всеми свойствами как объектных, так и реляционных систем. Основные свойства системы, объединяющей реляционный и объектный подходы, должны складываться из свойств, присущих каждой из этих систем в отдельности, а недостатки каждой модели неразрывно связаны с их преимуществами и фактически противоположны друг другу.

Общая постановка проблемы. Интенсивное развитие таких сложных технических систем как космические аппараты сталкивается с проблемой недостатка реальной информации об объектах испытаний ввиду значительного параллелизма выполняемых работ. В такой ситуации необходимы информационные средства, которые позволяют накапливать и использовать информацию без модификации информационных систем.

Очевидно, что подобные системы должны хранить информацию в базе данных и иметь программные средства, разработанные на основе объектно-ориентированного подхода для их эффективного применения.

Анализ публикаций. В ряде случаев информационные системы могут оказаться настолько связаны с обработкой данных, что с одной стороны использование внешней системы управления данными будет неизбежным, а другой стороны набор требований, предъявляемый задачей, может оказаться выше, чем у существующих СУБД [1].

Создание реляционной модели базы данных для описания ТП испытаний может решить вопрос о целесообразности создания или использования специализированной БД.

Постановка задачи. Для эффективного решения задач автоматизации ТП испытаний энергоустановок КА, обработки их результатов необходимо создать средства автоматизации разработки ТП испытаний СТС с использованием БД, которая

учитывает неопределенность исходной информации и обеспечивает переносимость и повторное использование сценариев ТП испытаний.

Результаты исследования

Блок информации любой сложности можно сохранить как набор записей различных таблиц. Эти рассуждения можно использовать и для объекта, данные о котором необходимо сохранить в реляционной базе данных. Из того факта, что информация об объекте с произвольной внутренней структурой может быть сохранена в реляционной системе, необходимо сделать следующий вывод: любому объекту можно поставить в соответствие идентифицируемый и осмысленный набор кортежей.

Для основных понятий структуры БД ТП испытаний воспользуемся ГОСТ 16504-81, в котором сказано, что испытание – это «экспериментальное определение (оценивание и (или) контроль) количественных и (или) качественных свойств объекта испытаний при его функционировании, при моделировании объекта и (или) воздействий».

Таким образом, представим испытание как отношение режимов, проверок и получаемых от системы сообщений:

$\mathbf{R_{test}}$ (**Modes**, **Checks**, **Messages**).

В этом случае домен **Modes** можно представить как отношение атомарных атрибутов имени *Name* ∈ Text, стартовой последовательности

$$StartSequenceID \in \mathrm{ID},$$

позиции $Position = \{x, y \in \mathbb{N}\}$,

размера $Size = \{w, h \in \mathbb{N}\}$

и текста $Label \in$ Text на экране:

Modes (*ModeID*, *Name*, *StartSequenceID*, *Position*, *Size*, *Label*).

Для проверок **Checks** отношение связывает имя *Name* проверки *CheckID* ∈ ID с последовательностью действий *SequenceID* ∈ ID и переменной *VarID* ∈ ID, которая будет хранить результат каждой проверки:

Checks (*CheckID*, *Name*, *SequenceID*, *VarID*).

Между собой проверки и режимы связаны отношением, определяющим состояние *CheckIsOn* ∈ Bool множества выполняемых на заданном режиме *ModeID* ∈ ID проверок *CheckID*:

ChecksOnMode (*ModeID*, *CheckID*, *CheckIsOn*).

Для выполнения перехода *TransitionID* ∈ ID с текущего *StartModeID* ∈ ID на новый режим *FinishModeID* ∈ ID используется отношение:

Transitions (*TransitionID*, *StartModeID*, *FinishModeID*, *SetCondID*, *Confirm*).

Это отношение определяет необходимость подтверждения *Confirm* ∈ Bool оператора на переход и связанно с таблицей условий:

ConditionTable (*CondID*, *SetCondID*, *VarID*, *Sign*, *Value*)

множеством условий *SetCondID* ∈ ID:

SetConditions (*SetCondID*, *Name*, *Inverse*),

где *SetCondID* задает набор условий, результат вычисления которых может инвертироваться в зависимости от значения атрибута *Inverse* ∈ Bool. Каждое условие представляет собой логическое выражение, в котором значение, хранимое в переменной *VarID*, сравнивается со значением *Value* с помощью логической операции, задаваемой в атрибуте

$$Sign = \{\leq, \geq, \neq, =, >, <\}.$$

Фактически как для управления режимами работы, так и для выполнения проверок необходимо выполнять команды управления аппаратурой **Command**. Выполнение одной команды *CommandID* ∈ ID заключается в записи определенных значений *Value* в переменные *VarID*, которые соответствуют портам ввода аппаратуры. Значения могут иметь различный тип и ограничения, что определяется атрибутом *ValueType* ∈.

С учетом необходимости выполнения некоторых команд в ручном режиме необходим атрибут, опре-

деляющий пункт меню *MenuPath* ∈ Text и дополнительную справочную информацию *Hint* ∈ Text:

Command (*CommandID*, *Name*, *VarID*, *ValueType*, *Value*, *MenuPath*, *Hint*).

Однако не всегда возможно управлять аппаратурой единичными командами, поэтому для логически завершенных действий необходимо отношение последовательностей **Sequences** состоящих из действий **Action** с определенным порядком *Order* ∈ ℕ.

Через интервал времени заданный атрибутом *PauseValue* ∈ ℕ выполнение действия проверяется набором условий, в зависимости от выполнения которого устанавливается результат выполнения *ResultIfTrue* ∈ ℤ или *ResultIfFalse* ∈ ℤ:

Sequence (*SequenceID*, *Name*, *Comment*),

Action (*ActionID*, *SequenceID*, *Order*, *PauseValue*, *MessageID*, *VarID*, *VarValue*, *SetCondID*, *ResultIfTrue*, *ResultIfFalse*, *CommandID*).

На основании выполнения последовательностей и вычисления набора условий для каждой проверки определяется ее результат *CheckResult* ∈ ℤ, который связан отношением с сообщением оператору *MessageID* ∈ ID

ChecksResult (*CheckID*, *CheckResult*, *MessageID*).

Каждое сообщение, в свою очередь, определятся отношением самого текста *Content* ∈ Text приоритета *Priority* = { n ∈ ℕ : n < 256 } и типа при выводе на экран *Type* ∈ и в отчет *TypeInReport* ∈:

Messages (*MessageID*, *Content*, *Priority*, *Type*, *TypeInReport*).

Для вывода некоторых сообщений необходимо указывать значения технологических параметров. Это требование реализуется отношением **ValuesInMessages**, которое определяет количество *Count* ∈ ℕ, порядок вывода *Order* ∈ ℕ в сообщении *MessageID* и его значение, хранящееся в *VarID*:

ValuesInMessages (*MessageID*, *Count*, *Order*, *VarID*).

Для обработки команд, действий и условий необходимы переменные, которые отображают состояние аппаратуры. Для этого используются отношения **Variables** и **VarsInProgSubscriber**, которые определяют программных потребителей информации, которые подписываются на получение уведомления при изменении значений в указанных переменных:

Variables (*VarID*, *Name*, *Description*, *Format*, *FormatToFile*, *AlwaysCallUpdate*, *HostVarID*) ,

VarsInProgSubscriber (*SubsID*, *Order*, *VarID*,).

Потребитель информации (подписчик) *SubsID* ∈ ID в зависимости от проводимых работ может иметь конфигурационные настройки, которые определены отношением:

ProgSubscriber (*SubsID*, *Order*, *VarID*),

Config (*ConfigID*, *ProgID*, *SectionName*, *ParamName*, *ParamValue*)

Если использовать допущение, что имена атрибутов отношения совпадают с именами соответствующих доменов, то можно определить следующие домены:

ModeID, *CheckID*, *SequenceID*, *StartSequenceID*, *VarID*, *StartModeID*, *FinishModeID*, *SetCondID*, *CondID*, *HostVarID*, *ProgID*, *ConfigID*, *SubsID*, *MessageID*, *CommandID*, *SequenceID* ∈ ID,

ID = {id ∈ ℕ : ∀ i, j (i ≠ j) $id_i \neq id_j$}

Text = {s ∈ String : Length(s) < 255}

Bool = {true, false}

Name, Label, Hint, MenuPath, Comment, Content, Description, Format, FormatToFile, SectionName, ParamName, ParamValue

AlwaysCallUpdate, CheckIsOn, *Inverse*, *Confirm* =

VarValue, Value = {r ∈ ℝ}

ValueType = {Fixed, FloatRange, IndexFromList, IntegerRange, ValueFromList}

Count, PauseValue, Order = { n ∈ ℕ }

CheckResult, ResultIfTrue, ResultIfFalse = {r ∈ ℤ}

Type = {Авария, Системная ошибка, Запрос от ЦПУ, Предупреждение, Информация, Действия оператора, Отладка}

TypeInReport = {Норма проверки, Предупреждение, Отказ аппаратуры, Игнорировать, Конец работы}.

Формальная модель базы данных для хранения технологического процесса испытаний КА показана на рис. 1.

Результаты применения

В результате разработки создано программное обеспечение для рабочего места инженера-технолога, обеспечивающее представление технологических процессов испытаний систем электроснабжения космических аппаратов в виде БД.

Эта разработка позволила сократить временные и материальные затраты на подготовку к проведению испытаний СЭС КА за счет замены трудоемкой ручной подготовки сценариев испытаний автоматизированным процессом их создания.

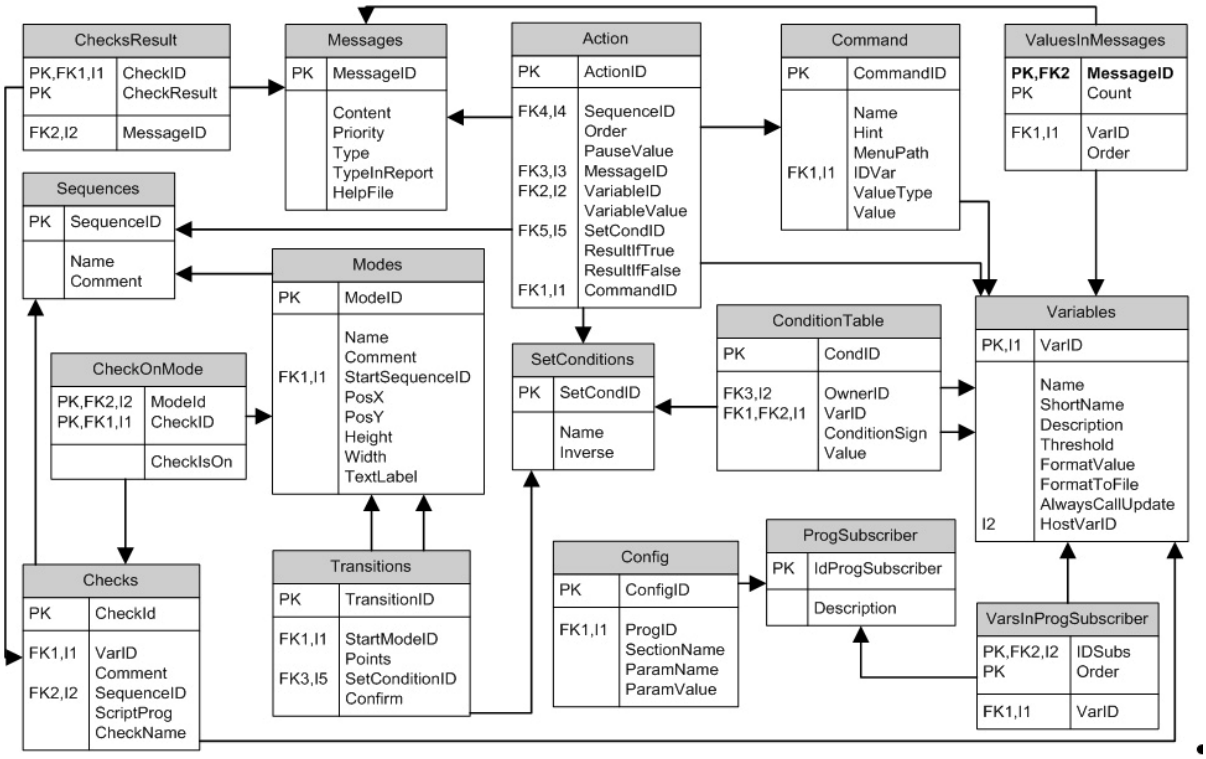

Рис. 1. Формальная модель БД для хранения ТП испытаний КА

Выводы

Эксплуатация разработанных программных средств обеспечивает сокращение временных и материальных затрат на подготовку испытаний СЭС КА за счет возможности формального представления технологических процессов испытания в визуально-графическом режиме инженером-технологом в БД без участия программиста, что в итоге приводит к увеличению информативности испытаний, повышению качества принимаемых решений по результатам испытаний.

Литература

1. Борисов В., Максимов В. СУБД для специализированных систем // Мир ПК. – 2007. – № 5. – С. 52-55.

Поступила в редакцию 22.01.2008

Рецензент: д-р техн. наук, проф. Э.Г. Петров, Харьковский национальный университет радиоэлектроники, Харьков.