УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Том 151, кн. 3 Физико-математические науки *2009*

УДК 629.11.012.553

СПЕЦИАЛИЗИРОВАННЫЕ МОДЕЛИ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЙ БАЗ ДАННЫХ НА ОСНОВЕ КОМБИНАЦИИ СРЕДСТВ UML И CSP-OZ

Т.М. Бендума, А.И. Еникеев

Аннотация

В работе предлагается один из подходов к построению специализированной модели для разработки приложений баз данных, основанный на соединении средств описания диаграмм UML с формальным аппаратом теории CSP-OZ. Предлагаемый подход демонстрируется на реляционной модели данных.

Ключевые слова: формальные методы, реляционная модель данных, теория взаимодействующих процессов CSP, язык Z, формальный аппарат теории CSP-OZ.

Введение

В большинстве современных разработок различных приложений баз данных используются полуформальные методы (OMT, UML и т. д.) [1], основанные главным образом на графических системах обозначения (диаграмме классов, диаграмме состояний и т. п.), которые дают интуитивное представление о разрабатываемой системе. Эти методы представляют бесспорные преимущества для моделирования, создавая идеальную поддержку для общения между различными участниками системы, но в этих методах отсутствуют средства анализа и проверки спецификаций на основе формального доказательства свойств, получаемых при моделировании. Формальные методы (B, VDM, Z, CSP, CCS и т. д.) [2–4], основанные на строгом математическом подходе, предоставляют возможность адекватной концептуализации и позволяют не только создавать точные конструкции, но и анализировать их. Формальные методы можно разделить на две группы: ориентированные на описание статических аспектов (B, Z, VDM и т. д.) и ориентированные на описание динамических аспектов (CSP, CCS и т. д.). Актуальным представляется подход, основанный на комбинации статических аспектов с динамическими. Одной из наиболее интересных разработок в этом направлении является CSP-OZ [5], представляющая собой формализованный аппарат на основе языка Объект-Z (объектно-ориентированное расширение языка Z) для описания статических структурных аспектов систем и теории взаимодействующих систем CSP для спецификации динамического поведения.

В работе предлагается один из подходов к построению специализированной модели для разработки приложений баз данных, основанный на соединении средств описания диаграмм UML [1] с формальным аппаратом CSP-OZ. Предлагаемый подход демонстрируется на реляционной модели данных. Такое соединение дает интуитивное и визуальное представление графических обозначений, с одной стороны, а с другой – точность и возможность анализировать и доказывать полученные спецификации на основе формальных методов. Кроме этого рассматривается методика реализации спецификаций теории CSP-OZ средствами SQL.

1. Основные понятия теории CSP-OZ

Теория CSP-OZ представляет собой формализованный аппарат, построенный на основе теории взаимодействующих систем CSP для спецификации динамического поведения и языка Объект-Z для описания статических структурных аспектов систем.

1.1. CSP (теория взаимодействующих процессов). Теория CSP используется для описания динамических аспектов (поведения) систем на основе событийно-управляемого подхода [4, 6, 7]. Центральными понятиями теории являются синхронное взаимодействие через каналы между различными процессами, параллельная композиция и механизм сокрытия внутренних точек взаимодействия.

В CSP процессы определяются как:

$$\mathrm{Pr} ::= \mathrm{Fail} \,|\, \mathrm{Skip} \,|\, c \to P \,|\, P/s \,|\, P\backslash A \,|\, P;Q \,|\, P \Box Q \,|\, P \sqcap Q \,|\, P \triangle Q \,|\, X \,|\, P \,|||\, Q \,|\, P \underset{A}{\|} Q.$$

где Fail – пустой процесс, который «ничего не делает», процесс Skip также «ничего не делает», но, в отличие от процесса Fail, завершает свою работу успешно; $(c \to P)$ трактуется как процесс активации события c, после которого запускается процесс P; (P/s) (P после s) – поведение процесса P после выполнения следа s, представляющего собой последовательность событий; $(P\backslash A)$ – сокрытие событий, принадлежащих множеству A, например:

$$(a \to b \to P)\backslash\{a, c\} = (b \to P);$$

$(P;Q)$ – операция последовательной композиции процессов P и Q (сначала выполняется процесс P, а затем процесс Q при условии нормального завершения процесса P); $(P \Box Q)$ – операция альтернативной композиции процессов P и Q (трактуется как P или Q); процесс $(P \sqcap Q)$ представляет собой внутренний недетерминированный выбор, который может вести себя либо как P, либо как Q без влияния окружающей среды (например, пользователя или другого процесса) в отличие от альтернативной композиции $P \Box Q$, где поведение процесса полностью определяется взаимодействием с окружающей средой; $(P \triangle Q)$ – это тоже вид последовательной композиции, но, в отличие от процесса $P : Q$, процесс $P \triangle Q$ ведет себя как P до тех пор, пока выполнение P не будет заблокировано, после чего процесс P останавливается и $P \triangle Q$ ведет себя как Q (*interrupt*); обычно процесс Q в этом случае – это процесс перезагрузки или восстановления после блокировки; $(\mu\, X : A.F(X))$ – обозначение рекурсивного определения процесса X, где A – алфавит процесса X, то есть множество событий процесса X (предполагается, что $X = F(X)$ имеет единственное решение в алфавите A, например:

$$\text{Часы} = (\text{тик} \to \text{Часы});$$

$|||$ обозначает параллельную композицию без синхронизации, а $\underset{B}{\|}$ – параллельную композицию с синхронизацией на всех событиях множества B, то есть события из множества B представляют общие точки взаимодействия параллельных процессов. Для передачи данных от одного процесса к другому в теории CSP используются каналы передачи данных. Вводятся две операции: $c!m$ – операция передачи сообщения m через канал c и $c?x$ – операция приема сообщения из канала c в переменную x. В теории CSP рассматривается случай синхронного взаимодействия процессов, который предусматривает одновременное выполнение операций $c!m$ и $c?x$. Итак, одновременное выполнение этих операций позволяет трактовать

его как единое неделимое событие взаимодействия процессов. Последнее определяется следующим соотношением:

$$(c!m \to P)||(c?x \to F(x)) = (c \to (P||F(m))),$$

где событие c определяет точку взаимодействия параллельных процессов.

Таким образом, все события, являющиеся общими для параллельно выполняемых процессов, определяют точки взаимодействия этих процессов. Средствами CSP можно определить широкий спектр процессов, составляющих основу модели поведения.

1.2. Z и Объект-Z. Z является формальным языком спецификаций, используемым для описания и моделирования вычислительных систем [3, 8]. Спецификация в Z обычно включает в себя схемы состояний и операций. Схемы состояний группируют переменные и определяют отношения между их значениями. Схемы операций определяют отношения между значениями переменных 'до' и 'после' выполнения операции (см. рис. 1).

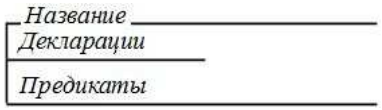

Рис. 1. Представления схем операций

Схемы операций отличаются от других схем тем, что они имеют в своем составе еще и $\triangle$ – список для выделения переменных, значения которых могут измениться при применении операций.

Объект-Z – это объектно-ориентированное расширение Z, основным элементом которого является схема класса. Класс включает в себя определение локальных типов и констант, одну схему состояния и связанную с ней схему инициализации состояния со всеми схемами операций [9–11] (см. рис. 2).

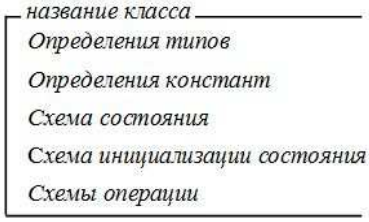

Рис. 2. Структура класса OZ

1.3. CSP-OZ. Теория CSP-OZ [5, 12–14] построена на основе комбинирования теории Объект-Z с теорией CSP. Спецификация CSP-OZ описывает систему как множество взаимодействующих объектов, каждый из которых задается описанием структуры и поведения. Взаимодействие между объектами выполняется через каналы аналогично взаимодействию процессов в теории CSP. Класс CSP-OZ имеет следующую структуру, изображенную на рис. 3.

Описание каналов определяет *интерфейс* класса. Описание задается в виде *Channel* c: $[p1 : ty1; \ldots; pn : tyn]$, где c – название канала, $p1, \ldots, pn$ – названия параметров, $ty1, \ldots, tyn$ – типы параметров.

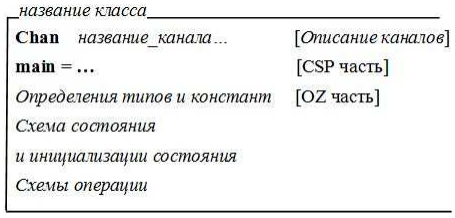

Рис. 3. Основная структура CSP-OZ класса

CSP-часть представлена в виде *CSP-название* = *CSP-процесс*. Каналы каждого процесса в CSP-части должны принадлежать множеству каналов, объявленных в интерфейсе. В CSP-части должен быть определен процесс **main** – стартовый процесс, с которого начинается работа. Этот процесс используется для определения семантики CSP-части.

В Z-часть включаются определения типов и констант, схема состояния и схема инициализации состояния (то же самое, как в Объект-Z). Схемы и имена переменных, используемые в Z-части, должны быть связаны с названиями, которые используются в CSP-части.

Для того чтобы связать схемы операций Z с поведением, используются специальные ключевые слова *enable* и *effect*, где *enable* определяет условия, при которых можно применить операцию, *effect* – соответствующий переход из одного состояния в другое.

2. Реляционная модель данных

Реляционная модель данных впервые была разработана Э.Ф. Коддом в 1970 г. [15]. Наиболее строгое и полное изложение теории реляционных баз данных (реляционной модели данных) можно найти в книге К.Дж. Дейта [16]. Согласно Дейту, в реляционной модели рассматриваются три аспекта: структура данных, подержание целостности данных (ограничения) и манипулирование данными (операции).

2.1. Структура данных. В основе реляционной модели данных [16–18] лежит понятие отношения, которое задается списком своих элементов и перечислением их значений. Реляционная база данных RDB – это реализация реляционной модели (модели данных) на физическом уровне и совокупность отношений, содержащих всю информацию, которая должна храниться в базе данных.

Отношение можно представить в виде двумерной таблицы (см. рис. 4), каждая строка таблицы носит название кортежа отношения, а каждый столбец таблицы называется атрибутом. Каждый атрибут определяет тип представляемых им данных, который вместе с областью его значений называется доменом.

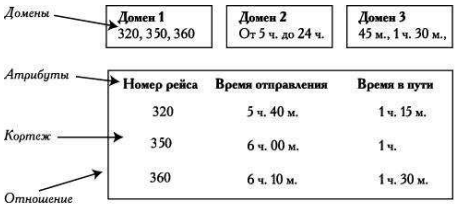

Рис. 4. Пример представления отношения

2.2. Механизм поддержания целостности данных. Механизм поддержания целостности данных основывается на системе ограничений, обеспечивающей целостность данных в реляционной базе данных [16]. По области действия ограничения делятся на ограничения домена, ограничения атрибута, ограничения кортежа, ограничения отношения и ограничения базы данных. Поддержание целостности данных можно определить как на отдельных атрибутах таблицы, так и на множестве таблиц. В SQL (норма 92) ограничения задаются конструкцией **CHECK**, определяющей соответствующий предикат. Ограничения, которые можно применять на атрибутах таблицы, определяются следующими ключевыми словами:

- **NOT NULL** – определяется на атрибуте, это ограничение позволяет указать, что значение данного атрибута не должно быть пустым для каждого кортежа таблицы;
- **PRIMARY KEY** – в отношении каждый кортеж должен обладать свойством уникальности. На самом деле, свойством уникальности в пределах отношения могут обладать отдельные атрибуты кортежей или группы атрибутов. Такие уникальные атрибуты удобно использовать для идентификации кортежей. Эти атрибуты называются первичными ключами и определяются ключевым словом *PRIMARY KEY*;
- Целостность ссылочных данных (внешний ключ): Внешний ключ – это атрибут, который появляется в отношении T2 и как первичный ключ – в другом связанном отношении T1 (T1 и T2 не обязательно различны). Внешний ключ определяет связь между разными отношениями.

Ниже предлагается метод соединения средств UML с теорией CSP-OZ, который демонстрируется на реляционной модели данных.

3. Метод соединения средств UML с теорией CSP-OZ

Метод, предлагаемый здесь, включает в себя следующие представленные ниже этапы (см. рис. 5):

а) моделирование в UML: данные и средства обработки данных представляются диаграммами UML, специализированными для описания информационных систем (диаграммы классов, диаграммы состояния и диаграммы сотрудничества);

б) генерация спецификаций реляционной модели из диаграмм UML, определенных на предыдущем этапе, в спецификацию CSP-OZ. Диаграммы UML не имеют строгой формализации семантики, поэтому цель этого этапа (перевод в CSP-OZ) заключается в соответствующей формализации. Полученная спецификация CSP-OZ состоит из трех главных компонентов: всех данных, представляющих состояние системы; всех базовых операций; всех каналов интерфейса, использующих базовые операции и обеспечивающих взаимодействие системы с другими компонентами (программами или другими системами);

в) создание кода SQL: на этом этапе создается код, соответствующий полученной спецификации CSP-OZ. Этот этап кодирования позволяет определить схему базы данных и связанные базовые операции в SQL (вставка, удаление, и т. д.), с одной стороны, а с другой – определить классы на языке программирования, позволяющие эффективную реализацию средств обработки в базе данных.

3.1. Перевод диаграмм UML в CSP-OZ. Здесь рассматривается методика преобразования диаграмм UML в соответствующие спецификации в CSP-OZ.

Рис. 5. Иллюстрация метода

Методика демонстрируется с помощью примеров, представляемых на основе реляционной модели данных.

3.1.1. Алгоритм преобразования диаграммы классов в реляционную схему. Процесс преобразования построен на основе классического алгоритма перевода из концептуальной модели данных в реляционную модель [19]. Он состоит из следующих основных этапов:

- все классы в диаграмме классов преобразуются в таблицы;
- отношения «многие к одному» (и «один к одному») становятся внешними ключами, то есть образуется копия уникального идентификатора класса на конце связи «один», и соответствующие столбцы составляют внешний ключ таблицы, соответствующей классу на конце связи «многие»;
- для построения отношения «многие ко многим» между классами A и B создается дополнительная таблица, имеющая, помимо ее собственных атрибутов, еще два столбца, один из которых содержит уникальные идентификаторы класса A, а другой – уникальные идентификаторы класса B.

На рис. 6. приведен пример диаграммы классов «Университет».

На языке CSP-OZ представление диаграммы имеет вид:

PROF $\widehat{=}$ [ProfID : PROFID; ProfName : STRING; **PrDepNum :**DEPNUM].
DEPT $\widehat{=}$ [DepNum : DEPNUM; **ChairDep :** PROFID; NbPrf: NBPRF].
CRS $\widehat{=}$ [CrsNum : CRSNUM; **CrDepNum:**DEPNUM].
PRF-CRS $\widehat{=}$
[ProfID :PROFID;**CrsNum :**CRSNUM; Schedule: SCHEDULE; Term: TERM].

3.1.2. Статический аспект (диаграмма классов). Для получения формального описания структуры базы данных из диаграммы классов необходимо определить реляционную модель в CSP-OZ на основе нижеследующих этапов.

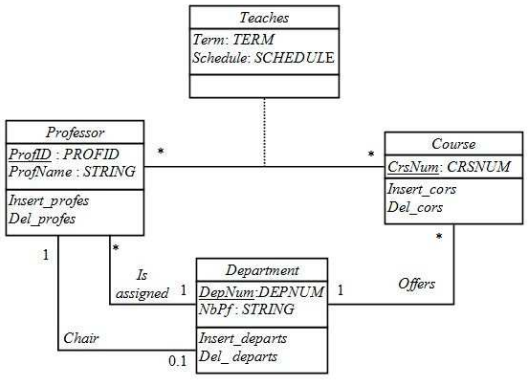

Рис. 6. Диаграмма классов «Университет»

Определение доменов в Z. В реляционной модели домен каждого атрибута должен быть атомарным (неделимым):

STUDENT_ID = = Integer

STUDENT_Name = = String AGE = = { a: Integer – 15<a <80}. SEX == Male | Female.

Определение атрибутов и кортежей. Все атрибуты перечисляются как переменные в части декларации схемы кортежа (строки), а все статические ограничения атрибута (если они есть) – как постоянные предикаты в части предикатов схемы *REL* (см. рис. 7).

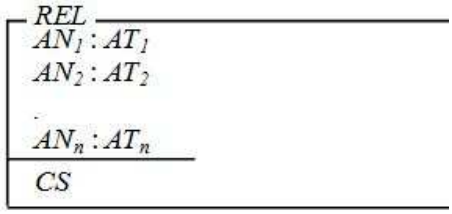

Рис. 7. Схема кортежа в Z

На рис. 8 приведен пример схемы кортежа «профессора» в Z.

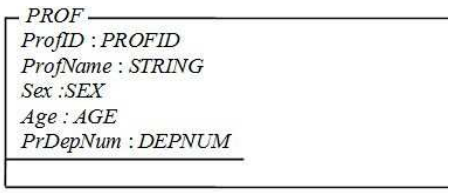

Рис. 8. Схема кортежа «профессора» в Z

Определение отношений (таблицы). В части декларации схемы отношения определяется конечное множество всех кортежей этого отношения, а в части предикатов – *первичный ключ* (с помощью предиката уникальности) и все атрибуты, которые не могут иметь свойство *Null-Значение* (с помощью конструкции *REQUIRED*) (см. рис. 9).

На рис. 10 приведен пример схемы отношения «профессор кафедры» в Z.

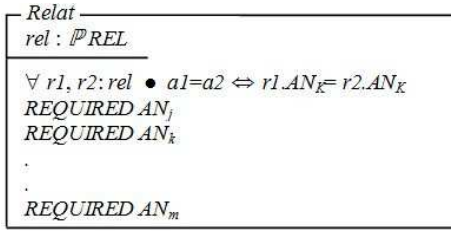

Рис. 9. Схема отношения в Z

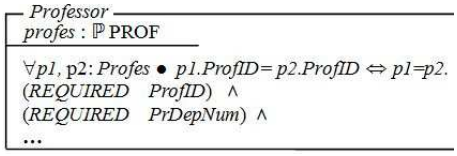

Рис. 10. Схема отношения «профессор кафедры» в Z

Определение схемы состояния базы данных. Схема группирует все определения базы данных, включая все схемы отношений в части декларации, а в части предикатов определяются все внешние ключи (с помощью предиката) и динамические ограничения (если они есть) (см. рис. 11).

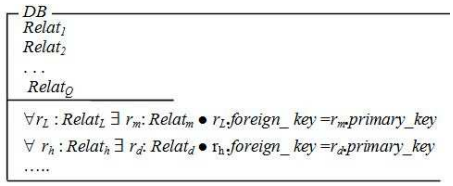

Рис. 11. Схема состояния базы данных в Z

На рис. 12 приведен пример схемы состояния базы данных «Университет» в Z.

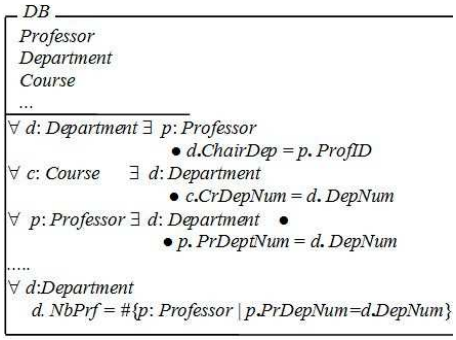

Рис. 12. Схема состояния базы данных «Университет» в Z

Отметим, что последний предикат определяет, что число профессоров кафедры (NbPrf) является производным атрибутом.

Определение схемы инициализации базы данных. Схема инициализации состояния базы данных определяет начальное состояние базы данных, показывая,

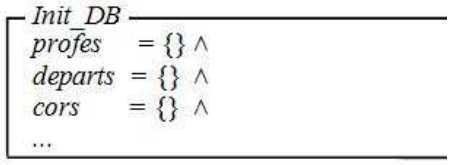

Рис. 13. Схема инициализации состояния DB в Z

Рис. 14. Схема операции вставки в Z

что все ее отношения пустые. На рис. 13 приведен пример схемы инициализации состояния DB в Z.

Определение базовых операций. Список базовых операций диаграмм классов UML включает в себя такие операции, как вставка и удаление.

Вставка. В декларативной части схемы операции вставки (*Insert_rel*) объявляются новые переменные *Relat* и *new_rel*, где *Relat* – это отношение (таблица), к которой хотим добавить новый кортеж *new_rel*. Пусть $\triangle$ обозначает, что значение отношения или таблицы *Relat* изменится при выполнении операции. В части предикатов определяется следующее состояние отношения *Relat* с помощью операции объединения в Z (см. рис. 14).

На рис. 15 приведен пример схемы операции вставки в Z.

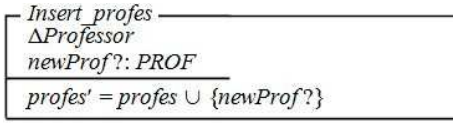

Рис. 15. Схема операции вставки в Z (пример)

Удаление. При выполнении операций удаления (а также модификации) базы данных применяются стратегии поддержания ссылочной целостности, представленные в книге Дейта [16]:

- CASCADE (КАСКАДНОЕ УДАЛЕНИЕ) – выполняет удаление кортежа, а также каскадное удаление всех тех кортежей в дочернем отношении, которые ссылаются на удаляемый кортеж (см. рис. 16);

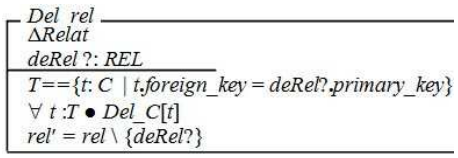

Рис. 16. Схема каскадного удаления в Z

- RESTRICT (ОГРАНИЧЕННОЕ УДАЛЕНИЕ) – не разрешает удаление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на удаляемый кортеж (см. рис. 17);

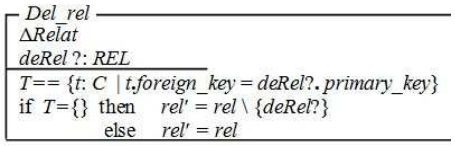

Рис. 17. Схема ограничения операции удаления в Z

- SET NULL (УСТАНОВИТЬ В NULL) – выполняет удаление кортежа, а во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, меняет значения внешних ключей на null-значение (см. рис. 18);

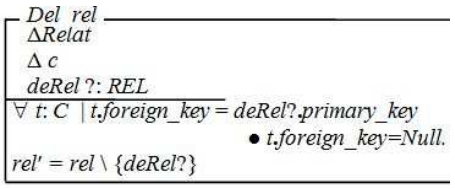

Рис. 18. Схема удаления с установкой в Null в Z

Примеры операций удаления приведены на рис. 19, 20.

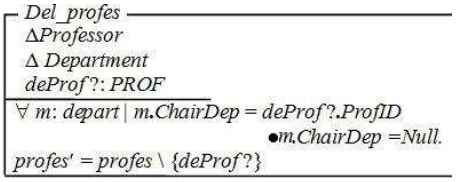

Рис. 19. Схема удаления «профессора» в Z

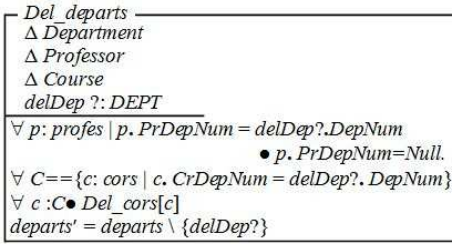

Рис. 20. Схема удаления «кафедры» в Z

Определение класса базы данных в Объект-Z. Класс базы данных включает в себя схему состояния базы данных, схему инициализации базы данных и все схемы базовых операций (см. рис. 21).

Пример класса DB-OZ для базы данных «Университет» приведен на рис. 22.

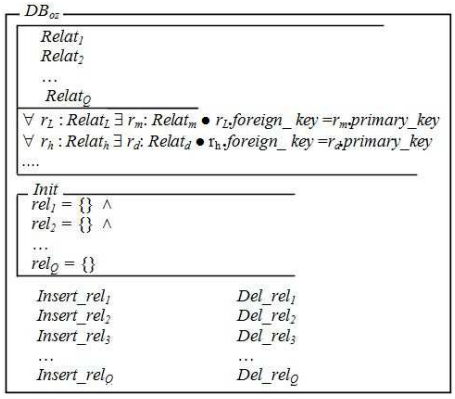

Рис. 21. Класс DB-OZ для базы данных

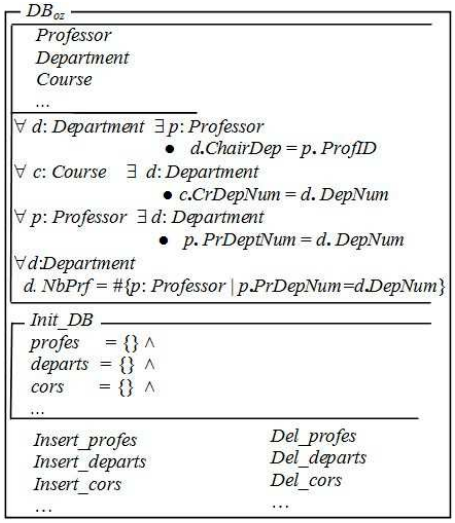

Рис. 22. Класс DB-OZ для базы данных «Университет»

3.2. Динамический аспект. Динамический аспект определяется диаграммами состояния и сотрудничества. В CSP-OZ поведение описывается с помощью CSP-части. Процесс **main** – это стартовый процесс, с которого начинается работа. Для каждой базовой операции *op* определяются схемы *enable_ op* и *effect_ op* (*enable_ op* описывает условия выполнения операции *op*, а *effect_ op* – соответствующий переход из одного состояния в другое) и для этих операций определяются каналы, через которые одни классы могут взаимодействовать с другими (см. рис. 23).

Пример класса RDB для базы данных «Университет» приведен на рис. 24.

3.3. Перевод на SQL. Полученная спецификация CSP-OZ описывает оба аспекта баз данных (структуру и поведение). Если реализацию спецификации CSP-OZ, соответствующей данным и базовым операциям, можно выполнить, используя только язык SQL, то перевод поведения (порядок выполнения, взаимодействие, манипуляции с таблицами и т. д.) требует использования языка программирования более высокого уровня (JAVA, DELPHI, C и т. д.).

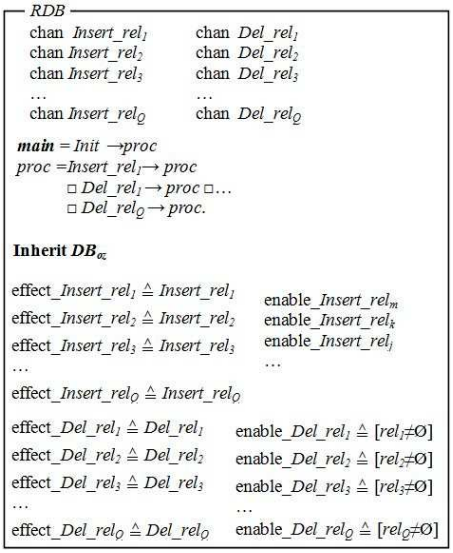

Рис. 23. Класс RDB

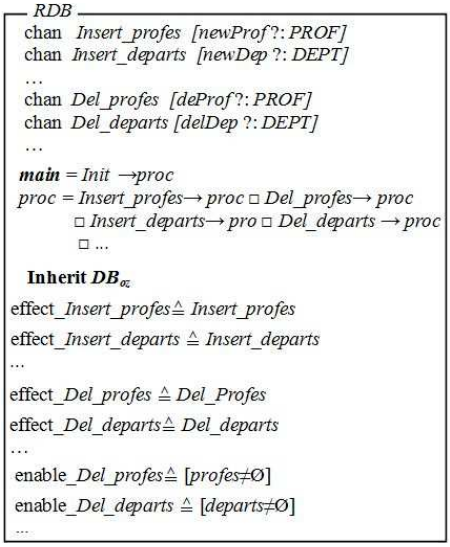

Рис. 24. Класс RDB для базы данных «Университет»

Ниже рассматривается перевод (реализация) на SQL и приводятся основные этапы этого перевода.

Статический аспект. Каждый тип данных CSP-OZ переводится в соответствующие типы SQL. Например, типы INTEGER и STRING переводятся в INT и CHARACTER STRING(n) в SQL, где n – максимальная длина строки.

Каждая схема отношения в CSP-OZ переводится в таблицу в SQL.

Предикаты первичного ключа, внешнего ключа и конструкция REQUIRED переводятся в **PRIMARY KEY, REFERENCES и NOT NULL** соответственно.

Статические и динамические ограничения переводятся с помощью конструкции CHECK.

Динамический аспект. Каждая операция вставки *insert_rel*(*NewRel*) пере-

водится в SQL как **INSERT INTO** *relat* **VALUES** NewRel. Каждая операция удаления *Del_rel*(*DeRel*) переводится в SQL как **DELETE FROM** *relat* **WHERE**.

Создаются классы на выбранном языке программирования (например, на JAVA, DELPHI, Visual FoxPro и т. п.), позволяющие производить манипуляцию с таблицами, созданными на предыдущем этапе.

Создаются классы, соответствующие спецификациям CSP-OZ, полученным из диаграмм сотрудничества и состояний (для описания поведения и взаимодействия).

Заключение

Основной целью представленной нами модели является создание специализированной интегрированной среды разработки, ориентированной на построение различных приложений в области так называемых информационно-расчетных задач. К информационно-расчетным задачам относятся задачи компьютерной бухгалтерии, делопроизводства, статистики и т. п. В настоящее время на основе СУБД Visual FoxPro и MS SQL проводятся работы по разработке и реализации упомянутой выше интегрированной среды.

Summary

T.M. Benduma, A.I. Enikeev. Specialized Models for the Development of Database Applications on the Basis of UML and CSP-OZ Tools Combination.

The paper presents an approach to building a specialized object-oriented model for the development of database applications. The approach is based on the combination of the UML diagram tools with the CSP-OZ formal methods. The approach is demonstrated on the relational data model.

Key words: formal methods, relational data model, communicating sequential processes CSP, language Z, CSP-OZ formal method.

Литература

1. *Буч Г., Якобсон А., Рамбо Дж.* UML. Классика CS. – СПб.: Питер, 2006. – 736 с.
2. *Abrial J.R.* The B-Book: Assigning Programs to Meanings. – Cambridge: Cambridge Univ. Press , 1996. – 813 p.
3. *Spivey J.M.* The Z Notation: A Reference Manual.– Prentice-Hall, 1992. –150 p.
4. *Хоар Ч.* Взаимодействующие последовательные процессы. – М: Мир, 1989. – 264 с.
5. *Fischer C.* Combination and Implementation of Processes and Data: From CSP-OZ to Java: Ph.D. Thesis, Bericht Nr. 2/2000. – Oldenburg: University of Oldenburg, 2000. – 353 p.
6. *Brookes S.D., Hoare C.A.R., Roscoe A.W.* A theory of communicating sequential processes // J. ACM. – 1984.– No 31. – P. 560–599.
7. *Roscoe A.W.* The Theory and Practice of Concurrency. – Prentice-Hall, 2005. – 605 p.
8. *Woodcock J., Davies J.* Using Z-Specification, Refinement, and Proof. – Prentice-Hall, 1996. – 386 p.
9. *Smith G.* The Object-Z Specification Language. Advances in Formal Methods. – Kluwer Acad. Publ., 2000. – 160 p.
10. *Duke R., Rose G., Smith G.* Object-Z: A specification language advocated for the description of standards // Computer Standards and Interfaces. – 1995. – No 17. – P. 511–533.

11. *Smith G.* A fully abstract semantics of classes for Object-Z // Formal Aspects of Computing. – 1995. – No 7. – P. 289–313.

12. *Fischer C., Smith G.* Combining CSP and Object-Z: Finite or infinite trace-semantics? // Proc. FORTE/PSTV'97, Chapmann & Hall. – 1997. – P. 503–518.

13. *Fischer C.* CSP-OZ: A combination of Object-Z and CSP // Bowmann H., Derrick J. (eds.) Formal Methods for Open Object-Based Distributed Systems (FMOODS'97).– Chapman & Hall, 1997. – V. 2. – P. 423–438.

14. *Fischer C., Wehrheim H.* Model-checking CSP-OZ specifications with FDR // Araki K., Galloway A., Taguchi K. (Eds.) Integrated Formal Methods. – Springer, 1999. – P. 315–334.

15. *Codd E.F.* A relational model of data for large shared data banks // Communications of the ACM. – 1970. – V. 13, No 6. – P. 377–387.

16. *Дейт К.* Введение в системы баз данных. – М.: Изд. дом «Вильяме», 2005. – 1328 с.

17. *Кузнецов С.Д.* Введение в системы управления базами данных // СУБД. – 1995. – № 1–4; 1996. – № 1–5.

18. *Джексон Г.* Проектирование реляционных баз данных для использования с микро-ЭВМ. – М.: Мир, 1991. – 252 с.

19. *Batini C., Ceri S., Navathe S.B.* Conceptual Database Design: An Entity Relationship Approach. – Redwood City, Calif.: Benjamin/Cummings Pub. Co., 1992. – 470 p.

Поступила в редакцию
29.04.09

Бендума Тахар Мохаммет – аспирант кафедры теоретической кибернетики Казанского государственного университета.

E-mail: *b.taher@mail.ru*

Еникеев Арслан Ильясович – кандидат физико-математических наук, доцент кафедры теоретической кибернетики Казанского государственного университета.

E-mail: *a_ eniki@inbox.ru*