НОВЫЕ СРЕДСТВА
КИБЕРНЕТИКИ, ИНФОРМАТИКИ,
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И СИСТЕМНОГО АНАЛИЗА

Б.Е. ПАНЧЕНКО, В.Н. ГАЙДАБРУС

УДК 004.652

РЕЛЯЦИОННЫЙ КАРКАС И МОДЕЛЬ CASE-ОБОЛОЧКИ НОВОГО ТИПА

Ключевые слова: *реляционный каркас, каркасная модель данных, CASE-оболочка, предварительно настроенные запросы, унификация и минимизация интерфейса, OLAP в реальном времени.*

ВВЕДЕНИЕ

Всесторонний анализ многих типов предметных областей (ПрО) [1] позволил выявить в реляционной модели данных (РМД) возможность реализации двух разных подходов к удовлетворению пользовательских потребностей: механизм запросов к частично нормализованной схеме реляционной базы данных (БД), не выше нормальной формы Бойса–Кодда (НФБК), а возможно, и 3-й нормальной формы (3НФ) [2]; механизм типизации большинства запросов, унификации основных алгоритмов, связанных с ними, и компоновки этих алгоритмов в безаномальную схему БД [3].

Основной критерий применимости таких подходов — коэффициент прогнозирования развития ПрО и разнообразия запросов пользователей [1] (коэффициент запросов), т.е. отношение числа подтвержденных изменений к суммарному числу прогнозированных и спонтанных изменений за определенный период времени, например год.

Показательно, что удовлетворительными для второго подхода являются ПрО с коэффициентом запросов, равным единице. Эти области можно глубоко изучать, прогнозировать их развитие и моделировать на основе РМД. Приложения, моделирующие процессы в ПрО, могут иметь минимальный интерфейс для неподготовленного пользователя, позволяющий минимизировать (или практически исключить) использование механизма внешних запросов к БД. К таким ПрО относятся различные бизнес-приложения компаний и корпораций, развивающихся в соответствии с прогнозируемыми рыночными факторами.

Потребности пользователей ПрО с малым коэффициентом запросов, как, например, поисковые машины в Интернете, разнообразные социальные сети, Интернет-витрины данных, схемы БД которых зависят не от владельцев систем и реальных причинно-следственных связей в ПрО, а от хаотически обращающихся пользователей, могут эффективно моделироваться гибкими языковыми конструкциями.

ПОСТАНОВКА ЗАДАЧИ

Реляционный каркас позволяет представить любое приложение как среду управления данными с заданной целью [4]. Поскольку для их обработки необходимо использовать ту или иную модель, наиболее привлекательна для этого

именно РМД. Каркас — частный случай РМД. Одним из его важных свойств является возможность минимизировать объем запросов к БД, построенных на громоздких и вычислительно сложных операциях соединения [5].

Каркасная БД моделирует до 90% запросов без операции соединения и ее модификаций [4]. Это позволяет значительную часть данных обрабатывать по заранее сформированным индексным таблицам, тем самым существенно снижая объем вычислений, а также формализовать, унифицировать и интегрировать в приложение подавляющее большинство запросов пользователей. Такой подход дает возможность разработать универсальную перенастраиваемую оболочку, управляемую группой метаданных, а массивы метаданных, отражающие специфику разнообразных ПрО, создавать с помощью отдельной программы-инсталлятора.

Длительные исследования проблем автоматизации проектирования приложений БД [1] показывают, что при этом существенно сокращается совокупность операций по отслеживанию целостности БД [2, 5], которые необходимо использовать в такой перенастраиваемой системе (в дальнейшем — CASE-оболочке). Это происходит потому, что основным инструментом манипулирования данными является индексный поиск, так как все данные в каркасной схеме БД, моделирующей ПрО, строго структурированы унифицированным алгоритмом [3, 4].

Еще одной особенностью схемы БД, полученной на каркасной модели данных (КМД), является возможность моделирования любой связи между объектами со степенью $G{:}H$. Эта максимально возможная степень связи лежит в основе анализа любой ПрО. Связи с меньшей степенью моделируются как частный случай. На практике все отношения (реляционные кортежные таблицы), за исключением незначительной их части, построены на шунтированной декартовой зависимости (ДЗ) и ее модификациях — многозначной зависимости (МЗ) и зависимости проекции-соединения (ЗПС) [3]. Поэтому при проектировании инструментального средства и будущих приложений имеем единую схему каждого отношения, построенную на этимологии смысла сущности-объекта [4, 6], т.е. на строковой сумме атомарных атрибутов типа $A+B+C+D+\ldots+Z+\ldots$ Известно, что для такой строковой конструкции можно построить элементарные индексные деревья. Такой механизм проектирования может быть унифицирован вплоть до стандартизации.

CASE-ОБОЛОЧКА ГЕНЕРАЦИИ МЕТАДАННЫХ, УПРАВЛЯЮЩИХ ПРИЛОЖЕНИЕМ

Основная технологическая особенность КМД — пакет отношений, связанных между собой по индексированным ключевым атрибутам. Однако поскольку значительная часть каждого кортежа состоит из ключевых атрибутов (что является отличительным признаком каркасной БД), такими иерархическими связями пронизана вся проекция каркаса (совокупность актуальных отношений) на данную ПрО. Это означает, что CASE-оболочка, разработанная на базе КМД, должна быть генератором метаотношений, в каждом из которых унифицированным способом хранятся метаданные конкретной ПрО: имена пользовательских отношений и их атрибутов (далее — полей), группы индексных заголовков, другие специализированные данные. Принципиально важным отличительным элементом каркасной группы отношений является связь между кортежем из верхнего отношения и непустой группой кортежей из нижнего. Она осуществляется по индексированному общему ключевому атрибуту. По сути, именно к такому атомарному сочетанию сводится вся совокупность отношений.

Рассмотрим отношение, построенное в соответствии с КМД. В общем случае его схема сводится к схеме особого отношения $R(X_i, A_{ij})$ [3], где атрибуты и их подстрочные индексы имеют тот же смысл, что и в [3]. Связь верхнего кортежа и группы нижних кортежей осуществляется по индексированному общему

ключевому атрибуту X_i. Если A_{ij} — пустое множество и отношение не шунтировано, то оно имеет аномалию типа ДЗ (МЗ, ЗПС) или, что аналогично, моделирует связь, которая не актуальна в этой ПрО в данный момент времени.

Отметим, что такая связь может быть лишь временно не актуальна. В любой момент эксплуатации схемы БД, находящейся в текущем состоянии, может возникнуть потребность в актуализации некоторой связи, у которой появятся атрибуты, т.е. отношение станет безаномальным. Это означает, что CASE-оболочка должна предоставлять пользователю возможность «на всякий случай» формировать всю полноту отношений. Выбор определяется спецификой ПрО и проектировщиком.

Каждое отношение для любого пункта меню становится так называемым центром активности приложения, т.е. это отношение или группа отношений, в которые осуществляется ввод новых или редактирование уже существующих данных (что завершается аналогично) одним или группой уполномоченных пользователей. В момент завершения такой операции по принципу on-line-транзакции выполняются необходимые процедуры: обновление всех ключевых атрибутов актуальной совокупности отношений (связанных с фоновым центром активности «за экраном»), отслеживание целостности соответствующих по ключевым полям отношений, формирование всех необходимых итоговых полей, обновление журнала транзакций, перегруппировка блокировок, пользовательских буферов и т.п.

На описанном принципе разработано и всесторонне апробировано инструментальное средство — CASE-оболочка SWS (Server Workstation System) [7]. SWS поддерживает бесконечно много центров активности. Однако в одном конкретном пункте меню интерфейса пользователя синтезируемого приложения представлен единственный центр активности. Пункт меню предоставляет пользователю его полномочия — на ввод новых данных, их обновление, просмотр и вывод на печать и т.д. Центр активности — формализованный унифицированный запрос к серверу БД, который проектировщик подготовил после обследования ПрО.

Известно, что ввод или обновление данных является ответственной процедурой. Рассмотрим ее более подробно. По факту завершения редактирования текущего поля одного из отношений приложения, синтезированного на CASE-оболочке SWS, осуществляется совокупность унифицированных и дополнительно задекларированных пользователем действий, причем только по принципу on-line-транзакций. В нее входят: обновление участвующих в связи совокупностей отношений ключевых полей (отслеживание их целостности), генерация новых кортежей во всех группах фоновых отношений, обновление индексов, формирование задекларированных в CASE-оболочке суммирований, других хранимых пользовательских процедур или триггеров и т.п.

В такой процедуре центр активности — одно отношение схемы БД или совокупность отношений, связанных между собой в одно. Все остальные совокупности отношений распределяются на две группы — головные отношения или подчиненные. Это означает, что любому связанному по достаточной совокупности ключевых полей кортежу головного отношения соответствует текущая группа кортежей активного (находящегося в данный момент в редактируемом пользователем состоянии) подчиненного отношения. В зависимости от специфики ПрО к любому из подчиненных отношений связь с центром активности может осуществляться как со степенью $1{:}H$, так и $G{:}H$. К паре «отношение-отношение» в итоге сводится любой фрагмент ПрО, поэтому унификация каскадных процедур навигации не является сложной задачей.

Приведем краткое описание основных принципов синтеза приложений БД, используемых в CASE-оболочке SWS.

ОСНОВНЫЕ КОМПОНЕНТЫ CASE-ОБОЛОЧКИ SWS И СХЕМА МЕТА-БД

CASE-оболочка — инструментальная система SWS — включает два модуля: инсталлятор, посредством которого формируется мета-БД таблиц-сценариев работы приложения, и непосредственно CASE-оболочку, интерпретирующую таблицы мета-БД.

Как отмечалось в [4], схема каркасной БД отличается от диаграммы Чена [8] отсутствием дуг — их применение, как правило, нецелесообразно. Нецелесообразно также отмечать на схеме обязательность, вхождение и степень каждой связи сущности-объекта [4]. Каркасная схема инсталлятора CASE-оболочки SWS, приведенная на рис. 1, представляет собой традиционную иерархию меню, уже на первом шаге проектирования приложения обеспечивающую выбор принципа работы со списками проектируемых отношений: локальные отношения, общие отношения или слияние схем БД разных ПрО.

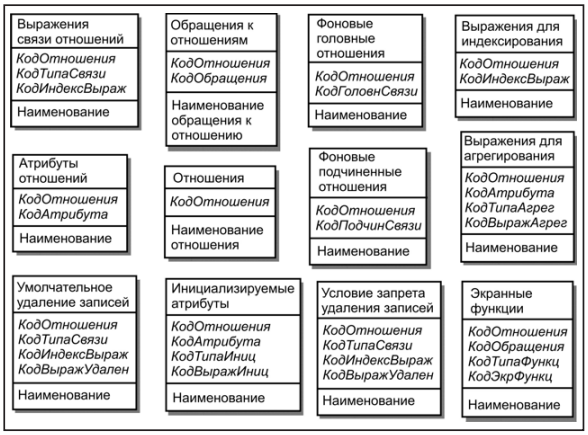

Рис. 1. Каркасная схема SWS-инсталлятора

В режиме «локальные отношения» производится настройка всей группы проектируемых отношений БД, которые обслуживаются только некоторым модулем приложения локально. Такие отношения недоступны при настройке другого модуля создаваемой подсистемы, в отличие от отношений БД, описываемых в режиме инсталлятора «общие отношения».

Определенный при первом конфигурировании модуля приложения вид работы с отношениями можно изменять, активизировав элемент подменю «режим работы с отношениями». Перестройка вида работы возможна «снизу вверх». Также возможна полная переконфигурация приложения, вплоть до режима «без отношений БД». Но такой выбор будет сопровождаться отключением уже существующих связей.

Основная работа по синтезу комплексов приложений, выполняемая в среде инсталлятора CASE-оболочки, — создание и конфигурирование схемы БД и интерфейса пользователя — управляющей среды «меню-подменю». Синтез схемы БД обеспечивается инсталлятором после выбора режима работы с отношениями — как монопольными, так и общими или по смешанному типу. Сценарий функционирования приложения составляется пользователем посредством раздела «Меню». Эти две установочные ветви (работа с отношениями приложения и работа с его меню) традиционно логически связаны — каждая из них в отдельности позволяет строить иерархические структуры для схемы БД и приложения. Кроме того, инсталлятор дает возможность задавать специальные перекрестные графовые связи между элементами деревьев «меню-подменю» приложения и элементами деревьев отношений БД (как монопольных, так и общих).

Два режима инсталлятора — синтез схемы БД и описание конфигурации приложения — наиболее трудоемкие. Их тесная функциональная взаимосвязь требует частого перехода из одного режима в другой. С этой целью система поддерживает подсказки взаимных данных.

Индексы. Важным механизмом поддержки целостности каркасной БД является индексный поиск. Колонка «индексы» используется наиболее часто. Она

предназначена для заполнения списков выражений индексации отношений, что приводит к созданию соответствующего числа индексных таблиц, позволяющих отслеживать структурные связи, а также выполнять необходимые быстрые поиски совокупностей записей в группах отношений.

Пример выражения индексации на языке типа dBase: $X + DTOS(Y)$. В этом выражении, кроме операции конкатенации, применена функция $DTOS(Y)$, позволяющая преобразовать значение атрибута отношения (в дальнейшем — поля) Y типа Date в строку вида ‘ГГГГММДД’. Здесь кавычки традиционно для многих языков программирования обозначают строковый тип данных. Такое преобразование дает возможность отсортировать записи отношения в хронологическом порядке.

Результирующая маска индексированного поиска, определяющая позиционное содержимое индексной таблицы для текущей записи, может, например, иметь вид ‘AAA20130321’, где строка ‘AAA’ находится в поле X текущей записи, а поле Y содержит значение даты — 21 марта 2013 года.

Таблица 1. Бинарное отношение

Поле X	Поле Y
AAA	01.07.13
AAA	02.07.13
AAA	03.07.13
AAB	01.07.13
AAB	02.07.13
AAB	03.07.13
AAC	01.07.13
AAC	02.07.13
AAC	03.07.13

Если в определенном подменю приложения включить обращение к отношению с выбором по порядку выражения индексации, то приложение, активизированное в подменю, выводит на экран данные, представленные в табл. 1.

Из таблицы видно, что выражение индексации дает возможность получить упорядочение записей по «пакетам» и «подпакетам». Помимо возможности быстрых поисков, такая концепция позволяет получить необходимые «видимые» вырезки этих пакетов из отношений БД и тем самым заменить механизм отработки запросов к БД on-line-визуализацией данных на экране, а также свести потребность в синтезе запросов к БД к минимуму. Кроме того, выражения индексации включаемых активных индексных таблиц позволяют организовывать связи по ведущим ключам. Таким образом, в SWS отслеживается специфика функционирования реляционных СУБД.

Еще одной важной особенностью SWS является отслеживание формата индексации отношения: $X \to (X + Y) \to (X + Y + Z)$. Она совпадает с концепцией предикатной этимологии смысла сущностей-объектов произвольной ПрО, обоснованной в рамках КМД [4, 6]. Приведенный абстрактный пример расшифровывается следующим образом. Постановка задачи требует трехзвенной табличной иерархии. Первое отношение индексировано по ключевому полю X, второе — по конкатенации ключевых полей $X + Y$, а третье — по конкатенации ключей $X + Y + Z$. Это позволило представить для обработки реляционные данные в иерархически взаимосвязанных отношениях, связанных только по уникальным ключам. Причем записи иерархически старших отношений «ведут» пакеты записей подчиненных (нижних, ведомых) отношений. Такая концепция системы SWS продиктована спецификой КМД, лежащей в ее основе. Практическое применение данного подхода более подробно описано далее при анализе «псевдофильтров» в «обращениях» к отношениям.

Обращения к отношениям. При последовательном описании окружения любого отношения БД на завершающем этапе осуществляется конфигурирование унифицированных запросов к фрагменту каркасной БД, озаглавленное в инсталляторе как «обращения». Приложения функционируют под управлением файлов мета-БД, т.е. сценариев установок, составленных в среде инсталлятора. Ветвь, отвечающая за интерфейс приложения, определяется в файле-сценарии

ИМЯ_МОДУЛЯ.MNU, который расположен в подкаталоге этого приложения. Ветвь, отвечающая за конфигурацию схемы БД, располагается в файле *ИМЯ_МОДУЛЯ.WRK* (для монопольных отношений модуля) и файле *ИМЯ_БАЗЫ_ДАННЫХ.WRK* (для общедоступных отношений), расположенных в назначенных при инсталляции подкаталогах.

Листья ветви «меню-подменю» приложения, озаглавленные в инсталляторе как «выводимые отношения», фиксируют графовые связи к листьям ветвей предварительно настраиваемых запросов к БД, озаглавленных в инсталляторе как «обращения». В определенных элементах подменю приложений происходят обращения к стандартно описываемым табличным формам ввода-вывода данных. Обращение заполняется для того, чтобы ввести в мета-БД приложения данные о том, как в определенном активизированном элементе подменю представить экранную таблицу ввода-вывода данных и их взаимосвязь.

Идентификатор обращения. Когда при составлении сценария интерфейса приложения в режиме основного меню инсталлятора осуществляется выбор описанного ранее обращения, инсталлятор предлагает таблицу выбора из существующего списка идентификаторов. Если идентификаторы несут определенную смысловую нагрузку, например «Ввод», «ПечатьПлатежныхПоручений» и т.п., выбор ветви обращения в инсталляторе произвести значительно легче. Таким образом, заполнение колонки «идентификатор обращения» — операция для обобщения и абстрагирования.

Выражение связи. В заголовке данной колонки инсталлятора (как и многих других), куда могут быть внесены выражения, запрашивается уточнение типа, возвращаемого при выполнении внесенного выражения. Для обеспечения межтабличной связи или получения «вырезки» из отношения необходимо записать такое выражение строкового типа, которое позволит задействовать необходимое активное выражение индексации.

На получение необходимого «пакета» записей в проектируемой форме ввода-вывода оказывают влияние три установочных фактора: запись нужного выражения индексации таблицы; выбор необходимого активного выражения индексации в режиме инсталлятора «открываемые отношения»; фильтрующее выражение с учетом выражения индексации активной индексной таблицы

В подменю приложения, где фиксируются открываемые отношения, предполагается к включению индексная строка, содержащая ранее записанное выражение индексации вида A_1. Выражение связи, называемое «псевдофильтром», записывается, например, как выражение строковой константы ‘00’. Тогда при активизации данного элемента подменю приложение сформирует таблицу «ввода-вывода» данных, где будут «видны» только те записи, в которых (согласно примеру) в поле A_1 содержатся данные, начинающиеся с символьной цепочки вида ‘00’, такие как ‘00AAA’.

Таким образом, псевдофильтр — унифицированный на уровне создаваемого модуля приложения запрос к данной таблице, построенный в соответствии с механизмами КМД и спроектированный не пользователем при эксплуатации, а разработчиком предварительно. В этом принципиальное отличие инструментальной CASE-оболочки SWS от аналогичных систем.

Концепция построения взаимосвязанных отношений в CASE-оболочке SWS, продиктованная КМД, требует определенного порядка внесения в метаданные выражений связи. Если предполагается необходимость иерархической структуры из двух отношений — R_1 и R_2, у которых ключевые индексные выражения имеют вид X и $X+Y$, то это означает, что запись верхнего отношения R_1 с ключевым полем X (содержащим определенное уникальное значение) должна отслеживать связь со всеми записями нижнего (ведомого отношения R_2), каждая из которых также содержит соответствующее значение в своем ключевом поле X.

В обращении к верхнему отношению R_1 (с выражением индексации X) выражение связи может отсутствовать, а в соответствующем для общей таблицы обращении к нижнему отношению R_2 (с выражением индексации $X+Y$) следует записать выражение связи вида $R_1 \to X$. Нотация $R_1 \to X$ в некоторых языках программирования читается как «алиас» поля X из отношения R_1. Это означает, что текущее значение поля X необходимо взять из отношения R_1.

Таблица 2. Взаимосвязь двух отношений — R_1 и R_2

Номер комплекта	**Наименование комплекта**
006	Комплект № 6

Номер комплекта	**Номер комплектующих**	**Наименование комплектующих**
006	01	Первая комплектующая комплекта № 6
006	02	Вторая комплектующая комплекта № 6
006	03	Третья комплектующая комплекта № 6
006	04	Четвертая комплектующая комплекта № 6
006	05	Пятая комплектующая комплекта № 6

В данном примере связь осуществляется из другого отношения, поэтому выражение связи указывается символом алиаса $R_1 \to$. Сортировка, требование уникальности значений или предполагаемая подобная связь с третьим отношением структуры в нижнем отношении R_2 определяется включением в выражение индексации поля Y.

При использовании сценария из примера настроенный модуль приложения выведет на экран монитора или на печать табл. 2.

Первое отношение, приведенное в табл. 2, имеет схему $R_1(X, A)$, является ведущим и упорядочено по значению ключевого поля $R_1 \to X$ (Номер комплекта). Второе поле отношения $R_1 \to A$ содержит наименования комплектов. В таблице приведено абстрактное наименование. В промышленных БД такое наименование, как правило, имеет значительное число символов.

Второе, подчиненное отношение имеет схему $R_2(X, Y, B)$, является ведомым и упорядочено по значениям ключевых полей $R_2 \to X$ (Номер комплекта) и $R_2 \to Y$ (Номер комплектующих). Третье поле отношения $R_2 \to B$ содержит наименования некоторых комплектующих, каждое из которых является частью соответствующих комплектов из ведущего отношения R_1.

Из примера видно, что выводимая на экран текущая запись из верхнего отношения R_1 со значением ключевого поля 006 фильтрует из подчиненного отношения R_2 только те записи, которые принадлежат данному коду. Связующее поле $R_2 \to X$ в таких таблицах на экран или печать, как правило, не выводится, так как служит лишь для связи с верхним отношением. Во всей совокупности записей нижнего отношения, связанных с единственной записью верхнего отношения, значение связующего поля является тождественным (в приведенном примере 006).

Поля $R_1 \to A$ и $R_2 \to B$ — информативные, они являются традиционной смысловой расшифровкой кодов. Конечный пользователь (оператор) работает в основном с кодами, организуя рабочую последовательность вида «код–данные». Экранные таблицы и выходные документы модуль формирует с использованием кодов записи и смысловых полей.

Таким образом, выражение связи применяется для организации структуры таблиц данных типа «дерево». Каждое отношение, как очевидно из теории КМД, — это в общем случае связь. Поэтому система SWS поддерживает иерар-

хическую совокупность отношений-связей, т.е. актуальный шунтированный фрагмент каркаса, моделирующий функционирование конкретной ПрО.

Фоновые головные отношения. В CASE-оболочке SWS проектировщику предоставляется возможность построения каркасной схемы БД, когда дерево связанных отношений БД строится «от листьев к корню» в соответствии с шунтированным фрагментом каркаса, моделирующим функционирование данной ПрО. Для этого в корневом окне редактирования обращения к отношению для одинаковых значений ключевых полей (в соответствующих записях совокупности отношений) обеспечивается формирование уникального кортежа в каждом фоновом (скрытом за экраном) отношении. При использовании в одном из элементов меню приложения такого обращения формируется on-line-сценарий обработки группы отношений построением иерархической цепочки «снизу вверх».

Другие обращения для других элементов подменю могут обеспечить возврат на экран и печать таких структурированных данных. Чтобы запросить отношение R_1 как фоновое головное для отношения R_2, необходимо следующее. В колонке инсталлятора «фоновые головные отношения», обеспечивающей окно редактирования списка фоновых головных отношений, записывается имя отношения R_1, в котором ключ X должен быть уникальным и ведущим для пакета соответствующих кортежей в отношении R_2. Выражение индексации отношения R_1 — указание имени ключевого поля X. Выражение связи из текущего отношения записывается тоже как X. В колонке «инициализируемые поля» вызывается редактор списка полей, и для поля X отношения R_1 инициализирующим будет также поле X, но взятое из отношения R_2.

В завершение построения в одном из подменю создаваемого приложения как открываемые заказываются оба отношения — R_1 и R_2 . Но как выводимое на экран (в мониторинговую таблицу) заказывается только отношение R_2. В дальнейшем вся созданная таким образом конструкция функционирует в определенном подменю приложения следующим образом: на экране приложения показывается таблица для просмотра или редактирования отношения R_2, где в произвольном порядке можно вносить неуникальные значения для поля X, которое для R_2 не является полным ключом . Ввод каждого нового значения X приводит к созданию новой записи в фоновом отношении R_1. Ключ X в этом отношении получает соответствующее уникальное значение.

Если при наборе в поле X отношения R_2 вводится значение, для которого уже создана ведущая запись в фоновом головном отношении R_1, новая запись в нем не добавляется. При этом происходит внутреннее позиционирование на запись с соответствующим ключом. Таким образом, создается схема БД, строго соответствующая фрагментам реляционного каркаса — совокупности групп данных по разделам ПрО. Такую схему легко применить для on-line-накопления расчетных данных.

Существует еще один способ использования фоновых головных отношений. Ранее описанные отношения-справочники не имеют жесткой индексированной связи с текущим отношением приложения. Поэтому для динамического связывания такое отношение-справочник следует объявить как фоновое головное отношение, указав соответствующее выражение связи с ним. Появляется возможность в колонках текущего отношения для определенного кода выводить на экран или печать расширенную информацию (например, наименование, иные атрибуты), но взятую из другого отношения — некоторого связанного отношения-справочника, что реализуется заказом непосредственного вывода соответствующих полей с указанием справочного отношения.

При создании такого фрагмента сценария работы приложения инициализируемые поля фонового головного отношения и выражения для их инициализации заказывать не нужно, поскольку соответствующие справочные записи должны уже существовать. Но если пользователь приложения имеет право обновлять этот справочник самостоятельно, выражения инициализации заказываются обязательно. В такой связи набор кортежей в справочном отношении первичен.

Фоновые подчиненные отношения. Когда в обращении приложения, сформированного унифицированными соглашениями SWS и транслирующего некоторый унифицированный запрос, предполагается редактирование какого-либо отношения, для которого существуют иерархически подчиненные отношения, не выводимые на экран, они должны объявляться в инсталляторе как фоновые подчиненные.

Смысл такого конфигурирования заключается в том, что при изменении какого-либо ключевого значения в ведущем (активном) отношении все соответствующие значения в подчиненных, ведомых отношениях также будут синхронно заменены с сохранением корректной ключевой связи между отношениями (целостности данных). Порядок заполнения колонок в окне инсталляционного редактора тот же, что и для фоновых головных отношений. Для подменю приложения, естественно, такие фоновые подчиненные отношения должны объявляться еще и как открываемые.

Анализ ПрО проектирования инструментального средства позволил сформировать список основных унифицированных каскадных функций. К ним относятся: инициализация и редактирование ключевых полей, проверка выполнения условий обязательности ввода или запрета на ввод, проверка условия целостности вводимого значения (минимум-максимум, положительное, уникальное и т.п.), установка полю статуса поискового, установка полю статуса выбираемого значения, установка связи для выбора значения в ином отношении (по полю, с фильтром, с показом поля, какое выражение связи по индексу и т.п.). К этой же группе относится и проверка некоторых дополнительных условий: замена значения одного поля при вводе и редактировании текущего (список заменяемых полей), генерация «сжатых» отчетов, каскадное удаление и т.п. Очевидно, что данный список можно пополнять.

OLAP — ПОДХОД НА ОСНОВЕ КАРКАСНОЙ МОДЕЛИ ДАННЫХ

При обслуживании современных БД основным вопросом является скорость реакции системы. Такие бизнес-приложения, как ретроспективный анализ деятельности компании, анализ рынка, стратегическое планирование и прогнозирование, другие проблемы, характеризуются необходимостью извлекать множество записей из очень больших наборов данных и вычислять на их основе с максимально возможной скоростью разные итоговые показатели. Предоставление поддержки таким приложениям — основное назначение всех OLAP-инструментов [9].

Международный комитет по стандартизации и международная электротехническая комиссия ISO/IEC в 2001 г. расширили известный стандарт SQL-подобных языков пакетом специфических OLAP-функций. Как основные решения стандарт [10] задекларировал функции GROUPING SETS, CUBE BY и ROLLUP BY. Каждая из них позволяет получать произвольные выборки данных для ответов на запросы пользователей. Однако основной недостатк постзапросного подхода — малоэффективное распараллеливание вычислительного процесса работы с терабайтами данных и, как следствие, невысокая итоговая производительность системы в целом. Подавляющее большинство запросов не случайны, а предопределены спецификой ПрО, но механизмы учета этого фактора используются не в полной мере.

OLAP и механизм on-line-транзакций. Альтернативной является методика использования реального времени, предложенная в [11] (в 1994 г.), когда в фоновых отношениях все необходимые итоги и подытоги формируются по факту ввода данных. При этом процесс распараллеливания осуществляется автоматически специализированной буферизацией транзакций. За счет механизма индексирования скорость формирования фоновых таблиц — максимально возможная в рамках используемой операционной среды. Поскольку ключевые поля фоновых отношений строятся на основании этимологии сущностей-объектов ПрО, а значит, соответствующей полной совокупности связей, вероятность появления непредусмотренных запросов при дальнейшей эксплуатации хранилища очень мала.

Данная методика позволяет значительно усовершенствовать известный механизм унификации часто повторяющихся запросов и on-line-формирование фоновых OLAP-итогов осуществлять на автоматизированно расширяющейся совокупности каркасных фоновых отношений, учитывающих новые запросы.

Важное свойство такой OLAP-методики — более естественный механизм отслеживания целостности данных, когда единицей атомарности является каскад данных, сформированный по всей совокупности отношений. Протоколирование таких транзакций значительно упрощается, и механизмы восстановления потерянных групп при сбоях в системе сводятся к простому копированию.

В работе [12] показано, что при агрегировании значений шунтирующих атрибутов «нижних» каркасных отношений и фиксации единственного агрегированного значения в качестве шунтирующего атрибута в одном кортеже «верхнего» каркасного отношения появление в этом кортеже дополнительных, не обусловленных ограничениями на домены и ключи [3], функциональных зависимостей (ФЗ) между атрибутами невозможно.

Для монотонного агрегирования в [12] данное утверждение доказано в виде леммы. Унарный атрибут S каркасного отношения со схемой $R(X, S)$, где X является ключом, полученный произвольной монотонной функцией $F(R_i(X_i, A_{ij}))$ от произвольной совокупности A_{ij} неключевых атрибутов иных каркасных отношений R_i, каждое из которых удовлетворяет только особым ограничениям [3], не является детерминантом отношения R.

Из этого утверждения следует, что использование в одном кортеже нескольких агрегированных атрибутов от разных аргументов агрегирования не приводит отношения к денормализации, так как никаких «паразитных» ФЗ между агрегатами не возникает. Однако в случае формирования нескольких агрегированных атрибутов от одного аргумента агрегирования в одном кортеже происходит денормализация отношения. В таком отношении появятся транзитивные ФЗ в неключевых шунтирующих агрегированных атрибутах.

Для выполнения основных OLAP-функций в SWS использован механизм пошаговых параллельных транзакций, построенный на каскадных функциях [13, 14], аналогичных перечню стандартизованных функций [10].

Рассмотрим классический OLAP-подход (асинхронный). OLAP-схема создается с использованием операции соединения отношений. В центре схемы хранилища данных (ХД) «снежинка» [15] находятся отношения-«факты», содержащие ключевые данные, по которым формируются запросы. Множественные отношения с «измерениями» присоединены к «фактам». Эти отношения показывают, как можно анализировать агрегированные реляционные данные. Число возможных агрегирований определяется числом первоначальных иерархических отображений. При использовании такого механизма OLAP пользователь вынужден тратить достаточно много времени на ожидание ответа на запрос — от нескольких минут до нескольких часов. Но чаще всего специфика работы современной ПрО требует мгновенного анализа.

Иной результат дает OLAP-подход на основе реляционного каркаса и режима реального времени (синхронный OLAP), при котором скорость получения данных сравнима со скоростью обращения к одной таблице с поиском по индексированным полям — менее 0,1 с.

OLAP-реализация в CASE-средстве SWS. Функционирование модуля CASE-оболочки SWS [7] организовано таким образом, что все необходимые расчетные данные обрабатываются в режиме реального времени — в тот момент, когда выполняется ввод или редактирование данных. Для реализации такого алгоритма существует, как минимум, три раздела инсталлятора: заполнение списка суммирующих полей в корне редактирования обращения; заполнение списка заменяемых полей в «настройках экрана»; заказ функциональных выражений вида $A * B$ в колонке «поле–функция–переменная» редактора экранных настроек. Как показала практика внедрений, пользователи превалирующее большинство расчетных данных формируют посредством заполнения списка суммирующих полей.

Алгоритм унифицированной процедуры следующий.

1. Суммирование числовых значений осуществляется только в поля либо текущего отношения, для которого описывается обращение, либо иерархически верхних отношений, объявленных как фоновые головные или подключенных иным способом. В поля кортежей иерархически подчиненных отношений производить суммирование бессмысленно.

2. При каждом вводе в любое поле (редактировании любого поля) текущего кортежа активизируется описываемая накапливающая инициализация суммирующих полей.

3. Если данные в суммирующих полях существуют, то на первом шаге производится арифметическое вычитание указанных в этих колонках выражений с их аргументами до момента возбуждения (ввода в поле, редактирования поля). На втором шаге выполняется арифметическое прибавление указанных в этих колонках выражений, но с учетом измененных аргументов (после момента возбуждения). Такая процедура позволяет динамически поддерживать целостность вычисляемых данных как при вводе, так и при редактировании полей с существующими данными.

4. Суммирование производится по порядку очередности списка сверху вниз — каждая последующая строка может использовать выражения с учетом суммирующих полей с просчитанными данными из строк выше.

Численный анализ OLAP-данных в режиме реального времени. Рассмотрим подробнее элементы синхронного формирования OLAP-данных на примере каркасной схемы БД , которая моделирует ведение журналов начислений по абонентам из ПрО «Горгаз» [6]. Имеем следующие сущности-объекты:

ИТОГ ЗА ГОД (*НомГода*, Сумма, Прим, …) — атомарная сущность-объект;

ИТОГ ЗА КВАРТАЛ (*НомГода*, *НомКварт*, Сумма, Прим, …) — слабая сущность-объект;

ИТОГ ЗА МЕСЯЦ (*НомГода*, *НомМес*, Сумма, Прим, …) — слабая сущность-объект;

ИТОГ ЗА ДЕНЬ (*НомГода*, *НомМес*, *НомДня*, Сумма, Прим, …) — слабая сущность-объект;

ИТОГ ЗА ДЕНЬ ПО ОТДЕЛУ (*НомГода*, *НомМес*, *НомДня*, *КодОтдела*, Сумма, Прим, …) — слабая сущность-объект;

ВЫРУЧКА СОТРУДНИКА ОТДЕЛА ЗА ДЕНЬ (*НомГода*, *НомМес*, *НомДня*, *КодОтдела*, *КодСотрудн*, Сумма, Прим, …) — слабая сущность-объект;

СОТРУДНИКИ В ОТДЕЛАХ (*КодОтдела*, *КодСотрудн*, Прим, …) — слабая сущность-объект;

АБОНЕНТЫ ПО СОТРУДНИКАМ (*КодАбонента*, *КодСотрудн*, Прим, …) — составная сущность-объект (справочник);

МЕСЯЦЫ В КВАРТАЛАХ (*НомМесяца*, *НомКварт*, Прим, …) — составная сущность-объект (справочник);

ОПЛАТА АБОНЕНТОВ (*НомГода*, *НомМес*, *НомДня*, *КодАбонента*, Сумма, НомСчета, …) — составная сущность-объект.

В описании ПрО упрощенно исключена зависимость от времени атрибутов некоторых сущностей-объектов, а также отношений, которые их моделируют. Это упрощение не влияет на результаты эксперимента.

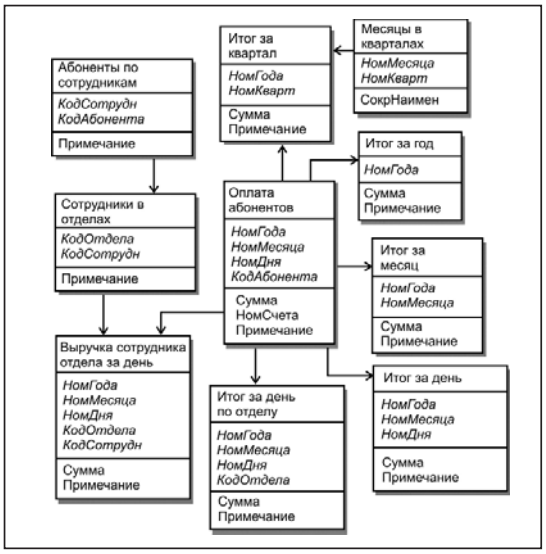

Рис. 2. Каркасная схема БД фрагмента ПрО «Горгаз»

В приведенной на рис. 2 каркасной схеме БД фрагмента ПрО «Горгаз» из [6] для большей наглядности дальнейших выводов показаны дуги. Но в данном случае они играют роль не связей между сущностями-объектами, как в диаграмме Чена [8], а указателей целостности данных и направления суммирования соответствующих числовых данных (итогов и подытогов). Процедура их формирования следующая: при введении оператором данных об оплате некоторым абонентом определенной суммы за определенный счет в поле Сумма отношения *ОПЛАТА АБОНЕНТОВ* срабатывает унифицированная каскадная функция, которая пересчитывает итоговые суммы во всех фоновых головных отношениях. Эта функция формирует элементы OLAP-данных синхронно, т.е. в реальном времени.

Проведен численный анализ скорости доступа к данным при постобработке и в on-line-режиме. Заметное уменьшение времени получения результатов в режиме реального времени — более 10% — наблюдается уже при значении суммарного числа записей в БД около 10^5.

Исследуемый показатель — задержка времени на выполнение арифметических операций в фоновых отношениях. При тестировании схемы моделировалась следующая ситуация: на протяжении года 200 пользователей ежедневно вносили по 1000 записей в общее отношение, что составило 73 000 000 записей. При этом в режиме продолжения сформировалось 100 фоновых итоговых отношений. На начальной и в завершающей фазе усредненное время выполнения функции (время задержки системы) не превышает 0,1 секунды на каждом рабочем месте, что несущественно. Результат полностью согласуется с методикой индексного поиска.

При дальнейшей корректировке основных характеристик системы — увеличении количества хранимых данных, числа пользователей, а также формирующихся в реальном времени фоновых отношений эффективность параллельной обработки значительно повышается. И хотя при этом наблюдается незначительное увеличение времени задержки на ввод данных на каждом рабочем месте, такие показатели не критичны.

На рис. 3 приведен график роста времени задержки (в наносекундах) на выполнение процедуры фонового суммирования от числа формируемых отношений (z) для 1 ядра (кривая *1*) и 24 ядер (кривая *2*) от 100 пользователей. Каждое

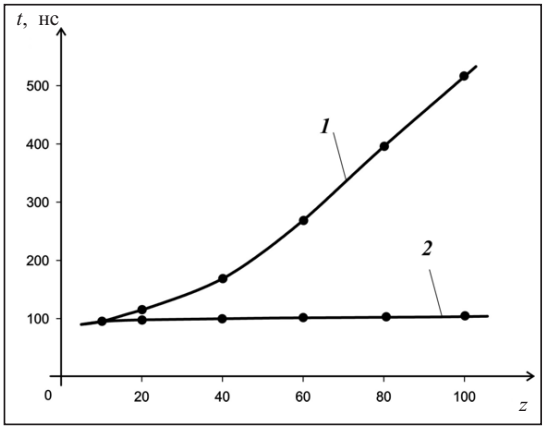

Рис. 3. График зависимости времени от числа фоновых отношений

отношение формируется до 10^7 кортежей. Видно, что распараллеливание процесса формирования агрегированных отношений значительно ускоряет выполнение фоновых процедур. Однако даже без распараллеливания сервер AMD OPTERON X12 с двумя процессорами по 12 ядер (3,26 GHz на ядро), с сервером данных PostgreSQL 8.4 и операционной системой UBUNTU SERVER 12.04 выполнил в описанном эксперименте транзакцию для 100 таблиц за 400 нс — более чем удовлетворительное время. Остальные затраты времени приходятся на работу сети.

Заметим, что приведенная на рис. 2 схема БД подобна известной [15] схеме ХД «снежинка». Однако отличием является возможность построения косвенных связей между отношениями по неполному соответствию ключей (на рис. 2 это связь между отношением *ОПЛАТА АБОНЕНТОВ* и *ВЫРУЧКА СОТРУДНИКА ОТДЕЛА ЗА ДЕНЬ*), формирования параллельных веток табличных деревьев, моделирования рекурсивных связей сущностей-объектов и поддержка режима реального времени. В [12] показано, что «снежинка» — частный случай реляционного каркаса; отмечено также, что схемы БД, подобные приведенной на рис. 2, могут использоваться не только как ХД, поддерживая при этом лишь операции извлечения кортежей, но и как OLTP-базы. Дугами на рис. 2 показаны противоположные [15] потоки данных, что тоже является важным отличием.

ИСТОРИЯ ПРОМЫШЛЕННОЙ ЭКСПЛУАТАЦИИ CASE-ОБОЛОЧКИ SWS

Проект CASE-оболочки SWS был инициирован авторами в Украине в 1988 г. Начало промышленных испытаний и широкого внедрения инструментального средства, разработанного в соответствии с КМД, зафиксировано в специализированной национальной и российской прессе: «Деловые новости», № 17 (80), К., 1995; ComputerWorld Киев, № 6 (29), К., 1995; «Монитор», № 8, М., 1994; «Компьютерра», № 6 (86), М, 1995.

Приведем список наиболее показательных задач, решаемых приложениями БД, разработанными и внедренными с помощью SWS сторонними пользователями в соответствующих организациях за период с 1993 по 1998 гг. Среди пользователей инструментального средства телевизионный технический центр «Останкино» [16], редакция газеты «Аргументы и факты» [17], Московская мебельная фабрика [18], Российское федеральное дорожно-строительное управление, шахта «Комсомолец Донбасса», Хмельницкая городская поликлиника, Дунайское речное пароходство [19], войсковая часть, машиностроительный завод и т.д.

Спектр ПрО, которые моделируются внедренными приложениями, очень широк. Это начисления по всем типам заработной платы, учет задолженности и начисления отпусков, система подготовки и обработки документов общего финансового учета (касса, банк, подотчетные лица, расчеты с дебиторами и кредиторами, валютные операции и т.п.), отчетность аппарата управления и сбыт готовой продукции (склад, счета, накладные, расчеты по сбыту), основные фонды и планирование производства (расчет себестоимости, рентабельности, ценообра-

зование), снабжение (взаиморасчеты с поставщиками, материалы, комплектующие) и производство (рабочее место технолога, движение сырья, материалов, деталей), кадры и ремонтные работы, учет травм и происшествий и т.п.

Среди внедренных приложений разработаны и уникальные системы: картотека первого отдела, розничное и подписное распространение прессы, анализ качества добытого угля, учет выработки тепла по котельным города и эксплуатация котельных (оборудование и теплотрассы), начисление денежного довольствия офицеров и прапорщиков, учет прививок, регистрация талонов амбулаторных больных и анализ заболеваемости и т.п.

У авторов имеются акты внедрения перечисленных систем, подтверждающие длительную актуальность промышленных приложений БД, которые были разработаны и внедрены пользователями. Особое значение имеет акт внедрения, фиксирующий текущий период актуальности приложений на шахте «Комсомолец Донбасса» (декабрь 2009 года), промышленная эксплуатация которых была начата в 1998 г.

ЗАКЛЮЧЕНИЕ

Описанный подход к автоматизации проектирования и кодирования приложений БД принципиально отличается от известных [20, 21] тем, что вместо избыточного синтезатора листингов в CASE-оболочке SWS использована строгая унификация схем БД и типизация функций; вместо привязки схемы БД к специфике ПрО посредством ручного кодирования ER-диаграммы [8] применен типовой шаблон, значительно упрощающий декомпозицию ПрО.

Разработанная и всесторонне тестированная CASE-оболочка SWS подтвердила гипотезу о том, что эффективное решение проблемы унификации, типизации и минимизации инструментального средства, которое масштабируется метаданными и позволяет синтезировать приложения БД, моделирующие функционирование произвольных ПрО, обеспечивает совокупность каркасных отношений. Созданный малогабаритный, но мощный CASE-генератор каркасных метаданных, управляющих универсальной оболочкой, практически полностью исключил потребность в ресурсоемких операциях соединения в большинстве запросов к БД и существенно упростил настройку приложений в прикладной разработке.

В рамках настоящего исследования базовые гипотезы КМД о единственности фактора этимологии и о сходимости алгоритма последовательных приближений на каркасе-шаблоне [4] не являются строго доказанными утверждениями. Однако они широко применяются авторами и пользователями на практике. В настоящей работе приведены доказательства справедливости этих утверждений, полученные экспериментальным путем. Исследовано 15 разных ПрО [16–19], в том числе не связанных непосредственно с БД [22–24]. Проанализированы приложения и схемы БД, разработанные сторонними пользователями как самостоятельно, так и при непосредственном участии авторов. Решенные задачи подтверждают справедливость основных теоретических утверждений, доказанных в [3], — о подобии каркасных схем БД для разных ПрО, модифицируемости систем без останова приложений, а также об отсутствии аномалий.

Данный подход особенно важен для ПрО, где ограничения часто модифицируются, так как позволяет существенно минимизировать затраты на разработку и сопровождение приложений.

СПИСОК ЛИТЕРАТУРЫ

1. Перевозчикова О.Л., Тульчинский В.Г., Коломиец А.В. и др. Высокопродуктивные методы анализа и спецификации пространств атрибутов предметной области для организации вычислений: (Отчет о НИР) / ИК им. В.М. Глушкова НАНУ. — № 0107U000800 ВФ.145.09.11. — Киев, 2011. — 378 с.

2. Codd E.F. The relational model for database management: version 2. — New York: Addison-Wesley Publ. Co, 1990. — 538 p.
3. Панченко Б.Е. Каркасное проектирование доменно-ключевой схемы базы данных произвольной предметной области // Кибернетика и системный анализ. — 2012. — № 3. — С. 174–187.
4. Панченко Б.Е. Алгоритм синтеза реляционного каркаса. Неформальное описание // Проблемы управления и информатики. — 2013. — № 1. — С. 83–103.
5. Мейер Д. Теория реляционных баз данных. — М.: Мир, 1987. — 608 с.
6. Панченко Б.Е. Численный анализ каркасной схемы реляционной базы данных // Компьютер. математика. — 2012. — № 2. — С. 116–126.
7. Панченко Б.Е., Гайдабрус В.Н., Церковицкий С.Л. CASE-генератор прикладных сетевых информационных комплексов — инструментальная система SWS 1.0 (CASE-генератор SWS 1.0) // Российское агентство по правовой охране программ для ЭВМ, баз данных и топологии интегральных микросхем; Свидетельство об официальной регистрации программы для ЭВМ № 940165 от 14.04.1994. — М., 1994. — С. 1–2.
8. Chen P.P. The entity-relationship model: toward a unified view of data // ACM Trans. Database Systems. — 1976. — **1**, N 1. — P. 9–36.
9. Федоров А., Елманова Н. Введение в OLAP. — М.: Диалог-МИФИ, 2002. — 268 с.
10. International Standart 9075, Database Language SQL, AMENDMENT 1: On-Line Analytical Processing (SQL/OLAP), ISO/IEC, 2001.
11. Панченко Б.Е., Гайдабрус В.Н., Церковицкий С.Л. Сетевые вычислительные комплексы // Компьютеры плюс программы. — 1994. — Спец. вып. — С. 30–37.
12. Панченко Б.Е. Хранилища данных на реляционном каркасе // Управляющие системы и машины. — 2013. — № 1. — С. 71–84.
13. Пат. № 63036 UA. Способ расположения данных в компьютерном хранилище, обеспечивающий модифицируемость его структуры / Б.Е. Панченко // Пром. власність. — 2004. — № 1. — С. 3.134. — (Заявл. 15.11.2001).
14. Пат. № 92248 UA. Способ обобщенного размещения данных с учетом модифицируемости структуры хранилища / Б.Е. Панченко // Пром. власність. — 2010. — № 19. — С. 3.131–3.132.
15. Kimball R. The data warehouse toolkit: Practical techniques for building dimensional data warehouses. — New York: John Willey & Sons, 1996. — 491 p.
16. Петренко С.И. Успех украинской SWS в «Останкино» // Украинские телевизионные новости, НТКУ. — 1994. — http://www.youtube.com/watch?v=1ePq6eQP2qc.
17. Петренко С.И. SWS в «Аргументах и фактах» // Украинские телевизионные новости, НТКУ. — 1994. — http://www.youtube.com/watch?v=42ZQTjteAoE.
18. Дейниченко Р.А. Московская мебельная фабрика «Интерьер» и украинская SWS // Компьютер-Х. — 1998. — Вып. 137. — http://www.youtube.com/watch?v=zVZfmLDt5GY.
19. Петренко С.И. SWS и пароходство в Измаиле // Компьютер-Х. — 1995. — Вып. 6. — http://www.youtube.com/watch?v=PcXBzLkPczo.
20. Колянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). — М.: Лори, 1996. — 360 с.
21. Вендров А.М. CASE-технологии: Современные методы и средства проектирования информационных систем. — М.: Финансы и статистика, 1998. — 176 с.
22. Панченко Б.Е., Назаренко А.М. Каркасный анализ предметной области: стационарные динамические задачи теории упругости для изотропных сред с произвольными неоднородностями // Кибернетика и системный анализ. — 2013. — № 1. — С. 172–187.
23. Панченко Б.Е., Кривомаз Л.С. Многокамерная прямая трансляция как метод эффективного дистанционного обучения // Инновационное развитие общества в условиях кросс-культурных взаимодействий: Тез. докл. 2-й Междунар. конф., 27–30 апр. 2009, Сумы. — Сумы, 2009. — **3**. — С. 18.
24. Панченко Б.Е., Печенюк Д.А. Каркасный анализ предметной области невычислительного характера — способ коммутации видеосигналов // Вестн. Черкас. ун-та. Сер. «Прикладная математика. Информатика». — 2012. — № 231. — С. 25–37.

Поступила 15.11.2012