Е. П. Емельченков, Н. А. Левин, В. И. Мунерман, Алгебраический подход к оптимизации разработки и эксплуатации систем управления базами данных, *Системы и средства информ.*, 2009,дополнительный выпуск, 114–137

УДК 681.3

# АЛГЕБРАИЧЕСКИЙ ПОДХОД К ОПТИМИЗАЦИИ РАЗРАБОТКИ И ЭКСПЛУАТАЦИИ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

*Е.П. Емельченков, Н.А. Левин, В.И. Мунерман*

В статье рассматривается автоматизация программирования задач обработки больших объемов информации в базах данных. Для решения проблем автоматизации предложен объектно-ориентированный подход. В основу положено строгое математическое определение объекта как универсальной алгебраической системы. Построены теоретико-множественная и многомерно-матричная модели обработки данных. Предложен специальный тип объектов — универсальная алгебраическая машина, при помощи которого доказаны изоморфизм моделей и возможность их параллельной реализации.

## Введение

Статья посвящается проблеме автоматизации программирования задач обработки больших объемов информации в базах данных. Разрабатывать СУБД, предназначенные для решения подобных задач, весьма непросто, особенно в настоящее время, когда параллельные вычислительные системы перешли в разряд типового серийного оборудования доступного любому потребителю.

Важная проблема, которую необходимо решать в этих условиях, — кроссплатформенность, т. е. независимость программного обеспечения от операционной среды, в которой оно должно функционировать. Программное обеспечение должно сохранять свою работоспособность при изменении количества и типов процессоров (или ядер) вычислительной системы. При этом доработка программ должна быть минимальной и не затрагивающей всю иерархию программных средств. Фактически переработке должны подвергаться только модули, непосредственно взаимодействующие с оборудованием.



Вторая проблема связана с тем, что возникает необходимость переработки программного обеспечения верхних уровней (уровней взаимодействия с пользователем). Это объясняется постоянно растущими требованиями пользователей к СУБД, имеющими целью наибольшее соответствие объектам и отношениям предметных областей. Вместе с доработкой системного программного обеспечения это вырастает в проблему, требующую решения задач автоматизации программирования.

И, наконец, третья проблема состоит в том, чтобы независимо от аппаратных средств СУБД максимально быстро обрабатывала запросы пользователя, независимо от того, как этот запрос сформулирован, т. е. описание процессов обработки данных должно быть приспособлено к решению задач оптимизации.

Для решения этих проблем авторы предлагают объектно-ориентированный подход. В основу положено строгое математическое определение объекта [1]. В статье строится специальный тип объектов — *универсальная алгебраическая машина*, при помощи которого можно решить сформулированные выше проблемы [2].

## 1. Основные понятия

В качестве основы объектно-ориентированных (алгебраических) моделей для обработки файлов использованы понятия универсальной алгебры и универсальной алгебраической системы, известные определения которых приведены ниже.

Определение 1. *Пусть $A$ — некоторое непустое множество. Частично определенная функция $y = F(x_1, \ldots, x_n)$, $(y, x_1, \ldots, x_n) \in A$, называется $n$-арной частичной операцией на $A$. Если функция всюду определена, говорят просто об $n$-арной операции.*

Определение 2. *Система $U_A = \langle A, \Omega \rangle$, состоящая из основного множества $A$ и определенной на нем совокупности частичных операций $\Omega = \{F_s^{n_s}\}$ $(s = 1, 2, \ldots)$, называется частичной универсальной алгеброй с сигнатурой $\Omega$.*

Определение 3. *Универсальные алгебры $U_A$ и $U_B$, в которых заданы соответственно сигнатуры $\Omega$ и $\Omega'$, называются однотипными, если можно установить такое взаимнооднозначное соответствие между сигнатурами $\Omega$ и $\Omega'$, при*

*котором любая операция $\omega \in \Omega$ и соответствующая ей операция $\omega' \in \Omega'$ будут $n$-арными с одним и тем же $n$.*

Пусть даны две однотипные универсальные алгебры $U_A = \langle A, \Omega \rangle$ и $U_B = \langle B, \Omega \rangle$ с основными множествами $A$ и $B$.

Определение 4. *Отображение $\varphi$: $A \to B$ называется гомоморфным отображением алгебры $U_A$ в алгебру $U_B$, если для любых элементов $a_1, \ldots, a_n \in A$ и произвольной $n$-арной операции $F \in \Omega$ выполняется соотношение $\varphi(F(a_1, \ldots, a_n)) = F(\varphi(a_1), \ldots, \varphi(a_n))$, где $\varphi(a_i) = b_i$ и $b_i \in B$ $(i = 1, \ldots, n)$. Алгебры $U_A$ и $U_B$ называются гомоморфными.*

*Если между основными множествами $A$ и $B$ устанавливается взаимно-однозначное соответствие, то отображение $\varphi$ называется изоморфным отображением, а алгебры $U_A$ и $U_B$ называются изоморфными.*

Определение 5. *Пусть $\pi(x_1, \ldots, x_n)$, $x_1, \ldots, x_n \in A$ — $n$-местный предикат, $\Pi = \{\pi_s^{n_s}\}$ $(s = 1, 2, \ldots)$ — сигнатура предикатов. Система $\mathbf{U}_A = \langle A; \Omega; \Pi \rangle$ называется универсальной алгебраической системой.*

Между алгебраическими системами также можно устанавливать гомоморфные и изоморфные отображения.

Случаи, когда можно обойтись единственным основным множеством, встречаются нечасто, поскольку объекты предметной области являются сложными системами, для описания которых используется много разнотипных параметров. Для моделирования таких предметных областей используются многоосновные алгебры.

Определение 6. *Система $U_M = \langle \mathbf{M}; \Omega \rangle$, состоящая из семейства основных множеств $\mathbf{M} = \{A_\alpha\}$ $(\alpha = 1, 2, \ldots)$ и сигнатуры $\Omega$ операций, определенных на семействе $\mathbf{M}$ так, что каждая $n$-арная операция из $\Omega$ является отображением декартова произведения $n$ множеств из семейства $\mathbf{M}$ в множество из того же семейства $A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to A_{\alpha_r}$ называется многоосновной алгеброй.*

Определение 7. *Система $\mathbf{U}_M = \langle \mathbf{M}; \Omega; \Pi \rangle$, где $\Pi$ — сигнатура $n$-местных предикатов $\pi : A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to \{0, 1\}$, называется многоосновной алгебраической системой.*

Многоосновные алгебры и алгебраические системы играют важную роль в программировании. Они служат основой для создания пользовательских типов данных в современных языках программирования, а ими являются некоторые встроенные в языки типы данных, например, строковый тип.



Очевидно, что хорошо известные и часто применяемые системы числовых матриц, нагруженных графов также можно рассматривать как многоосновные алгебраические системы.

На многоосновных алгебраических системах базируется современное теоретическое и практическое программирование. *Абстрактные типы данных* (они же *объекты* или *классы*), сокращенно АТД, определяются как многоосновные алгебраические системы [1].

Среди множества произвольных абстрактных типов данных выделяется и рассматривается один специфический тип, называемый в дальнейшем *универсальной алгебраической машиной* [2]. Эти АТД представляют собой двухосновные алгебраические системы вида $\mathbf{E} = \langle S, T; \Omega; \Pi \rangle$. Основу $S$ назовем *структурой*, а основу $T$ — *типом*. *Структура* представляет собой некоторую конструкцию, составленную из элементов *типа*. Типичные примеры такого рода структур — векторы, матрицы, графы. Выбор *типа* определяется особенностями решаемой задачи.

Достаточно очевидно следующее утверждение.

Утверждение 1. *Пусть $S$ — структура, а $T_1, \ldots, T_n$ — допустимые для этой структуры типы. Тогда $T_1, \ldots, T_n$ — однотипные алгебраические системы.*

Практическая особенность универсальных алгебраических машин состоит в том, что суть операций над элементами структуры $S$, не изменяется при изменении сути операций над элементами типа $T$. Более того, если типы $T_1, \ldots, T_n$ — гомоморфные универсальные алгебраические системы, то становится возможной отладка операций над структурой на наиболее простом типе данных. Отлаженная таким образом структура становится объектом, от которого можно порождать объекты, предназначенные для решения задач на сложных типах данных.

Это хорошо видно на примере алгебры матриц. Операции сложения и умножения матриц не зависят от того, какого типа элементы матриц. При реализации этих операций с помощью традиционных языков программирования программируются одинаковые циклы для числовых, логических и прочих типов элементов матриц. Отличия заключаются в одной или двух строках программы, в которых производятся операции над элементами [3]. Особенно ярко достоинства универсальных алгебраических

машин проявляются при параллельной обработке данных. Это объясняется тем, что при работе с параллельными вычислительными системами основная сложность состоит в программировании алгоритмов обработки структур, а не отдельных элементов, составляющих эти структуры. Следует отметить, что подобный подход рассматривался еще на заре вычислительной техники в работах Л. В. Канторовича [4] задолго до появления первых работ по объектно-ориентированному программированию.

Далее рассматриваются две изоморфные алгебраические системы, которые могут быть эффективно применены при разработке современных СУБД [3, 5].

## 2. Алгебра файлов

Алгебра файлов строится на основе теоретико-множественной модели обработки файлов [6, 7]. В языках программирования, начиная с COBOL'a, для отображения свойств объектов предметных областей используются **записи** — структурные типы данных, объединяющие под одним именем разнотипные переменные. Поэтому формализация начинается с записей.

Пусть $A = \{A_1, \ldots, A_p\}$ — некоторая конечная система конечных множеств, а $N = \{N_1, \ldots, N_p\}$ — конечное множество элементов, называемых *именами* множеств $A_1, \ldots, A_p$. Множества $A_1, \ldots, A_p$ могут состоять из элементов любой природы: чисел с фиксированной или плавающей точкой, строк, а также таких структур, как массивы, кортежи и тому подобные. На множествах $A_1, \ldots, A_p$ могут быть заданы операции и отношения, тогда $A_1, \ldots, A_p$ называются *типами*.

Определение 8. *Полем записи называется пара* $F = \langle N_i, A_i \rangle$ $(i = 1, \ldots, p)$*;* $N_i$ *— имя, а* $A_i$ *— множество значений поля.*

Определение 9. *Кортеж* $R = \{F_1, \ldots, F_p\}$ *называется записью типа* $R$.

Определение 10. *Кортеж вида* $R^* = \{\langle N_1, A_1^* \rangle, \ldots, \langle N_p, A_p^* \rangle\}$ $(A_i^* \in A_i,\ i = 1, \ldots, p)$ *называется экземпляром записи типа* $R$.

Замечание 1. В дальнейшем изложении термин *запись* будет отождествляться с термином *экземпляр записи* типа $R$ в тех случаях, когда это не будет вызывать неоднозначного толкования.



В дальнейшем, в ряде случаев из соображений удобства, записи будут отождествляться с синтаксической конструкцией Pascal (***record***), а экземпляры записей — со строками таблиц.

Определение 11. *Множество* $\mathbf{X}$ *экземпляров записей типа* $R$ *называется множеством записей типа* $R$ *или множеством однотипных записей.*

Пусть $K = \{K_1, \ldots, K_m\}$, $(m < p)$ — множество полей записи $R$, такое что $K_1 = F_{\alpha_1}, \ldots, K_m = F_{\alpha_m}$, причем все $A_{\alpha_i}$ $(i = 1, \ldots, m)$ — типы, на которых заданы отношения порядка.

Определение 12. *Конечное множество* $K = \{K_1, \ldots, K_m\}$ *называется множеством ключей, а его элементы ключами.*

Определение 13. *Кортеж* $K^* = \{K_1^*, \ldots, K_m^*\}$*, для элементов которого выполняется правило* $K_i^* \in A_{\alpha_i}$ $(i = 1, \ldots, m)$*, называется экземпляром множества ключей (*$K_i^*$ *называется экземпляром ключа).*

Очевидно, что любая совокупность экземпляров множества ключей может быть лексикографически упорядочена. Одновременно упорядочивается и множество однотипных записей $\mathbf{X}$, тип которых включает множество ключей $K$.

Определение 14. *Две однотипные записи называются эквивалентными, если они содержат одинаковые экземпляры множества ключей.*

Замечание 2. В языках программирования существует возможность задавать под одним и тем же именем записи с различной структурой. Это не нарушает однотипность записей в рассмотренном смысле, при условии, что каждый вариант структуры записи содержит все ключи из множества $K$.

Можно сказать, что задание множества ключей $K$ разбивает множество однотипных записей $\mathbf{X}$ (индуцирует разбиение $\mathbf{X}$) на группы (классы), содержащие записи с одинаковыми значениями ключей — эквивалентные записи. Эти классы называются *классами эквивалентности*. Они могут содержать разное количество записей.

Совокупность всех классов эквивалентности по отношению заданному множеством ключей образует *фактор-множество* множества однотипных записей $\mathbf{X}$.

В дальнейшем такое фактор-множество будет обозначаться $X_K$, составляющие его классы эквивалентности — $X_{K^*}$, или

$X_{K^*_{(1)}}, X_{K^*_{(2)}}$ .... Также будет полезно рассматривать экземпляры множества ключей, которым во множестве $\mathbf{X}$ не соответствует ни одной записи. В этом случае целесообразно считать, что таким экземплярам множества ключей соответствует ***универсальная неопределенная запись*** $\Theta$. Класс эквивалентности, соответствующий экземпляру множества ключей $K^*$ и состоящий из единственной записи $\Theta$, будет обозначаться $\Theta_{K^*}$.

Определение 15. *Пусть даны множество однотипных записей $X$ и множество ключей $K$. Файлом $X_K$ называется фактор-множество множества однотипных записей $X$ по отношению эквивалентности порожденному множеством $K$.*

Замечание 3. Структура, определенная выше, могла бы быть названа *отношением* или *таблицей*, но авторам кажется, что название *файл* больше соответствует сути работы, в которой обработка данных рассматривается на стыке логического и физического уровней.

Как и фактор-множество, файл обозначается $X_K$, а класс эквивалентности, соответствующий экземпляру множества ключей $K^*$, обозначается $X_{K^*}$.

При таком подходе файл не может быть неупорядоченным. Это в некоторой степени не соответствует общепринятому представлению о файлах как носителях произвольной информации. Но в дальнейшем рассматриваются только такие файлы, которые несут в себе структурированную информацию об объектах предметных областей и как носители этой информации используются на входе, выходе и в процессе решения задач, поставленных и формализованных на этих предметных областях. Процессы решения таких задач построены на основе набора операций обработки файлов, которые реализуются алгоритмами весьма чувствительными к упорядоченности входных файлов. Это особенно заметно при обработке больших объемов данных.

Замечание 4. Известно, что многие реляционные СУБД требуют упорядоченности данных, настойчиво предлагая пользователю задавать, так называемые первичные ключи для каждой таблицы.

Определение 16. *Если каждый класс эквивалентности файла $X_K$ содержит единственную запись, то файл $X_K$ называется строго упорядоченным, если же в каждом классе эквивалентности может быть более одной записи — нестрого упорядоченным.*



Это определение завершает построение теоретико-множественной модели файла — носителя информации об объектах предметных областей. В терминах этой модели легко задать теоретико-множественные описания операций над файлами.

**Сортировка.** В теоретико-множественной модели роль операции сортировки особенно высока. Именно ее выполнение приводит к построению из исходного множества однотипных записей $X$ файла $X_K$ (фактор-множества $X$ по заданному множеству ключей $K$). Практические соображения требуют, чтобы для любого множества $X$ можно было подобрать такое множество ключей $K$, по которому файл $X_K$ будет строго упорядоченным (это соответствует уникальности строк таблицы в реляционном подходе).

**Выборка.** Пусть даны файл $X_K$ и предикат $\pi(K)$, определенный на множестве ключей $K$. Операция выборки приводит к созданию файла $X^\pi_K$, удовлетворяющего следующим условиям:

$X^\pi_K \subseteq X_K$, т. е. файл $X^\pi_K$ есть подмножество файла $X_K$;

$\forall K^*(X_{K^*} \in X^\pi_K \wedge \pi(K^*))$, т. е. класс эквивалентности $X_{K^*}$ присутствует в файле $X^\pi_K$ тогда и только тогда, когда все значения ключей в экземпляре множества ключей $K^*$ превращают предикат $\pi(K)$ в истинное высказывание.

Таким образом, в результате выполнения операции выборки образуется подмножество исходного файла. Принадлежащие этому подмножеству записи содержат фиксированные экземпляры одного или нескольких ключей. Если фиксируются значения не всех ключей, то остальные ключи принимают все допустимые значения.

**Сжатие.** Пусть даны файлы $X_K$, нестрого упорядоченный по множеству ключей $K$, и $Y_K$, строго упорядоченный по множеству ключей $K$. Классы эквивалентности этих файлов связаны соотношением $Y_{K^*} = f(X_{K^*})$. Тогда считается, что файл $Y_K$ получен из файла $X_K$ в результате применения операции сжатия. В операциях сжатия каждому непустому классу эквивалентности $X_{K^*}$ исходного файла, нестрого упорядоченного по множеству ключей $K$, ставится в соответствие класс $Y_{K^*}$ выходного файла $Y_K$, строго упорядоченного по тому же множеству ключей. При этом единственная запись, принадлежащая $Y_{K^*}$, вычисляется как значение некоторой функции $f$, определенной на всех за-

писях класса эквивалентности $X_{K^*}$. В качестве такой функции могут быть использованы вычисление суммы или среднего значения, поиск наименьшего или наибольшего значения и тому подобные.

**Слияние строго упорядоченных файлов.** Пусть даны два файла $X_K$ и $Y_K$, строго упорядоченные по одному и тому же множеству ключей $K$. В результате слияния этих строго упорядоченных файлов образуется файл $Z_K$, классы эквивалентности задаются соотношением

$$Z_{K^*} = f(X_{K^*}, Y_{K^*}).$$

Функция $f(X_{K^*}, Y_{K^*})$, определенная на классах эквивалентности исходных файлов задает характер операции.

Если слияние строго упорядоченных файлов используется для реализации теоретико-множественной операции объединения файлов, то функция $f$ задается как

$$f(X_{K^*}, Y_{K^*}) = \begin{cases} Y_{K^*}, & \text{если } X_{K^*} = \Theta_{K^*}, \\ X_{K^*} & \text{в противном случае.} \end{cases}$$

Для реализации теоретико-множественной операции симметрической разности файлов задание функции $f$ будет таким:

$$f(X_{K^*}, Y_{K^*}) = \begin{cases} Y_{K^*}, & \text{если } X_{K^*} = \Theta_{K^*}, \\ X_{K^*}, & \text{если } Y_{K^*} = \Theta_{K^*}, \\ \Theta_{K^*}, & \text{если } X_{K^*} \neq \Theta_{K^*}, Y_{K^*} \neq \Theta_{K^*}. \end{cases}$$

Наконец, в сложных вычислительных задачах построение функции $f$ может быть следующим:

$$f(X_{K^*}, Y_{K^*}) = \begin{cases} g_1(Y_{K^*}), & \text{если } X_{K^*} = \Theta_{K^*}, \\ g_2(X_{K^*}), & \text{если } Y_{K^*} = \Theta_{K^*}, \\ g_3(X_{K^*}, Y_{K^*}), & \text{если } X_{K^*} \neq \Theta_{K^*}, Y_{K^*} \neq \Theta_{K^*}. \end{cases}$$

Одноместные функции $g_1$ и $g_2$ определены на записях файлов $X_K$ и $Y_K$ соответственно. Двухместная функция $g_3$ определена на парах записей, принадлежащих декартову произведению $X_K \times Y_K$. Эти функции реализуют формирование записи выходного файла с вычислением новых значений неключевых полей, из значений неключевых полей записей $X_{K^*}$ и $Y_{K^*}$. Значения ключей все три функции переносят в выходную запись $Z_{K^*}$ без изменений.



Пример 1. Распространенная операция корректировки одного файла при помощи другого задается следующим образом:

$$f(X_{K^*}, Y_{K^*}) = \begin{cases} X_{K^*}, \text{если } Y_{K^*} = \Theta_{K^*}, \\ Y_{K^*}, \text{ если } X_{K^*} = \Theta_{K^*}, \\ Y_{K^*}, \text{ если } X_{K^*} \neq \Theta_{K^*} \wedge Y_{K^*} \neq \Theta_{K^*} \wedge \pi(Y_{K^*}), \\ \Theta_{K^*}, \text{если } X_{K^*} \neq \Theta_{K^*} \wedge Y_{K^*} \neq \Theta_{K^*} \wedge \neg\pi(Y_{K^*}). \end{cases}$$

Предикат $\pi(Y_{K^*})$ служит признаком замены или удаления записи корректируемого файла.

**Слияние нестрого упорядоченных файлов.** Пусть $X_K$ и $Y_L$ — файлы, упорядоченные (возможно строго) по множествам ключей $K$ и $L$, причем выполняется условие $K \cap L \neq \varnothing$, и пусть $M$ — множество ключей, связанное с множествами $K$ и $L$ соотношениями

1. $M \subseteq K \cup L$,
2. $M \cap K \neq \varnothing$ и $M \cap L \neq \varnothing$.

Это означает, что множество ключей $M$ состоит из ключей, входящих в множества $K$и $L$, причем в $M$ содержится по крайней мере по одному ключу из каждого множества. Тогда по крайней мере один файл $X_M$ или $Y_M$ нестрого упорядочен по множеству ключей $M$. Если $K \not\subset L$ и $L \not\subset K$, то оба файла нестрого упорядочены по множеству ключей $M$. Слияние файлов производится по множеству ключей $M$. Пусть $M^*$ — фиксированный экземпляр множества ключей $M$, а $K^*$ и $L^*$ — такие экземпляры множеств ключей $K$ и $L$, что значения одноименных ключей в $M^*$, $K^*$ и $L^*$ совпадают. Тогда можно задать вычисление класса эквивалентности файла $Z_M$ по следующему правилу:

$$Z_{M^*} = \begin{cases} \Theta_{M^*}, & \text{если } X_{K^*} = \Theta_{K^*} \text{ или } Y_{L^*} = \Theta_{L^*}, \\ f(X_{K^*}, Y_{L^*}) & \text{в противном случае.} \end{cases}$$

Функция $f(X_{K^*}, Y_{L^*})$ определена на классах эквивалентности $X_{K^*}$ и $Y_{L^*}$, а ее значение — класс эквивалентности $Z_M$, состоящий из элементов, каждый из которых вычисляется из пары элементов, принадлежащей декартову произведению $X_{K^*} \times Y_{L^*}$.

Замечание 5. Операция *слияния нестрого упорядоченных файлов*, по сути, совпадает с реляционной операцией *Join*, которая тоже осуществляет слияние нестрого упорядоченных таблиц.

Пример 2. Для демонстрации применения операций обработки файлов хорошо подходит традиционная для автоматизированных систем управления предприятиями задача расчета стоимости производства изделий. Естественно, здесь она рассматривается в упрощенном виде. Исходные данные для этой задачи порождаются и первоначально используются в различных местах и различных задачах. Для решения рассматриваемой задачи они собираются в следующие исходные файлы:

1. *План производства* — запись состоит из полей
   1.1) *изделие–узел*,
   1.2) *количество*.
2. *Применяемость* — запись состоит из полей
   2.1) *изделие–узел*,
   2.2) *узел–деталь*,
   2.3) *количество*.
3. *Трудовые затраты* — запись состоит из полей
   3.1) *узел–деталь*,
   3.2) *цех*,
   3.3) *сумма*.
4. *Материальные затраты* — запись состоит из полей
   4.1) *узел–деталь*,
   4.2) *цех*,
   4.3) *сумма*.

Множества ключей у этих файлов следующие:

$K_{(\text{план производства})} = \{\textit{изделие–узел}\}$,
$K_{(\text{применяемость})} = \{\textit{изделие–узел}, \textit{узел–деталь}\}$,
$K_{(\text{трудовые затраты})} = \{\textit{узел–деталь}, \textit{цех}\}$,
$K_{(\text{материальные затраты})} = \{\textit{узел–деталь}, \textit{цех}\}$.

Эту систему исходных файлов можно представить в виде графа (рис. 1), вершины которого соответствуют файлам. Две вершины связаны ребром, если соответствующие им файлы имеют общие ключи. На рис. 1 множества общих ключей привязаны к соответствующим ребрам. Таким образом, системе исходных файлов соответствует связный граф, в котором существует, по крайней мере, один путь, проходящий через все вершины. Это очень важно, так как две вершины, связанные ребром могут быть операндами одной из операций слияния.

В результате решения задачи должен получиться файл *сводные затраты*, запись которого состоит из полей

1) *изделие–узел*,



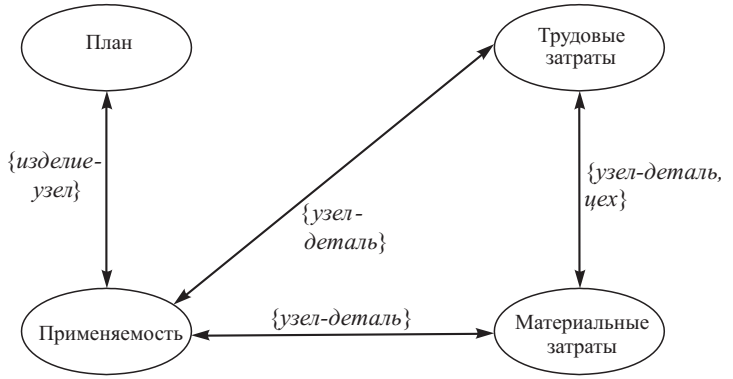


**Рис. 1.** Граф связей файлов в задаче расчета стоимости

2) *узел–деталь*,
3) *цех*,
4) *затраты*.

Этот файл содержит всю необходимую информацию для построения отчетов, которые позволят решать различные аналитические экономические задач.

Получение файла результата возможно посредством реализации следующего процесса:

1. Выполняется слияние строго упорядоченных файлов *трудовые затраты* и *материальные затраты*. В результате получается промежуточный файл *tmp*1 с той же структурой записи и строго упорядоченный по тому же множеству ключей. Алгоритм формирования записи выходного файла задан следующей таблицей:

| Функция | Выходной файл *tmp*1 | Входные файлы | |
|---|---|---|---|
| | Поле | Трудовые затраты | Материальные затраты |
| $g_1$ | *узел–деталь* | | *узел–деталь* |
| | *цех* | | *цех* |
| | *затраты* | | *сумма* |
| $g_2$ | *узел–деталь* | *узел–деталь* | |
| | *цех* | *цех* | |
| | *затраты* | *сумма* | |
| $g_3$ | *узел–деталь* | *узел–деталь* | |
| | *цех* | *цех* | |
| | *затраты* | *сумма* + *сумма* | |

2. Выполняется слияние нестрого упорядоченных файлов *применяемость* и *tmp*1. Файл *применяемость* переупорядочивается сортировкой по множеству ключей {*узел–деталь*, *изделие–узел*}. В результате получается выходной промежуточный файл *tmp*2 с записью $R = \{$*узел–деталь*, *изделие–узел*, *цех*, *затраты*$\}$, в которой поле *затраты* вычисляется по формуле:

$$\textit{затраты}_{(\textit{tmp}2)} = \textit{количество}_{(\textit{применяемость})} \times \textit{затраты}_{(\textit{tmp}1)}.$$

3. Выполняется слияние нестрого упорядоченных файлов *план производства* и *tmp*2, переупорядоченного по множеству ключей $K = \{$*узел–деталь*, *изделие–узел*, *цех*$\}$. В результате получается искомый выходной файл *сводные затраты*, поле *затраты* в записи которого вычисляется по формуле

$$\textit{затраты}_{(\textit{сводные затраты})} = \textit{количество}_{(\textit{план производства})} \times \\ \times\, \textit{затраты}_{(\textit{tmp}2)}.$$

Рассмотренный процесс приводит к получению требуемого результата. Он не идеален, так как требует выполнения двух сортировок, которые относятся к числу операций с большой вычислительной сложностью. Кроме того, возможно, что предварительное выполнение операции слияния файлов *план производства* и *применяемость* может уменьшить время последующих слияний. В дальнейшем будет рассмотрен способ повышения эффективности реализации процессов обработки файлов за счет распараллеливания слияния нестрого упорядоченных файлов.

Далее рассматривается теоретико-множественная формализация понятия процесса обработки файлов. Пусть $\mathbf{P}$ — некоторая задача, решаемая посредством обработки данных, хранящихся в исходных файлах.

Определение 17. *Совокупность* $S = \{X^{(1)}_{K_{(1)}}, \ldots, X^{(n)}_{K_{(n)}}\}$ *исходных файлов для задачи* $\mathbf{P}$ *называется системой исходных файлов, а файл* $X^{(0)}_{K_{(0)}}$*, получаемый в результате решения этой задачи, — результатом.*

Для операций слияния двух файлов будет использоваться обозначение: $\omega(X^1_{K_1}, X^2_{K_2})$. Тогда процесс решения задачи на системе исходных файлов $S$ можно задать выражением:

$$X^{(0)}_{K_{(0)}} = \omega(X^{(1)}_{K_{(1)}}, \omega(X^{(2)}_{K_{(2)}}, \omega(\ldots, X^{(n)}_{K_{(n)}})\ldots)). \qquad (1)$$



Очевидно, следует определить условия, при которых возможно существование такого процесса.

Необходимое условие. Ни одна из определенных операций над файлами не может создать выходной файл, содержащий больше ключей, чем оба исходных файла. То же самое верно и для операций с одним исходным файлом. Следовательно, независимо от последовательности операций, которая применяется к системе исходных файлов $S = \{X^{(1)}_{K_{(1)}}, \ldots, X^{(n)}_{K_{(n)}}\}$, выходной файл не может содержать ключей больше, чем все файлы системы $S$ вместе взятые. Из сказанного следует, что для осуществления процесса обработки системы исходных файлов $S = \{X^{(1)}_{K_{(1)}}, \ldots, X^{(n)}_{K_{(n)}}\}$ с получением выходного файла $X^0_{K_0}$ необходимо, чтобы выполнялось условие

$$K_{(0)} \subseteq \bigcup_{i=1}^{n} K_{(i)},$$

т. е. множество ключей выходного файла должно быть подмножеством объединения множеств ключей всех исходных файлов.

Достаточное условие. Система исходных файлов $S = \{X^{(1)}_{K_{(1)}}, \ldots, X^{(n)}_{K_{(n)}}\}$ называется связной, если для любой пары файлов $X^{(l)}_{K_{(l)}} \in S$ и $X^{(m)}_{K_{(m)}} \in S$ $(1 \leqslant l, m \leqslant n)$, существует упорядоченная последовательность $S'$ файлов из $S$, удовлетворяющая следующим условиям:

множества ключей любых двух соседних файлов из последовательности $S'$ имеют непустое пересечение;

множества ключей первого и последнего файлов последовательности $S'$ имеют непустое пересечение с множествами ключей файлов $X^l_{K_l}$ и $X^m_{K_m}$ соответственно.

Если система $S$ связна, то входящие в нее файлы всегда можно расположить в таком порядке, что любая пара соседних файлов будет иметь общие ключи. В этом случае становится возможным построение выражения по типу выражения (1), т. е. процесса обработки файлов. Следовательно, для существования процесса обработки файлов достаточно, чтобы система $S$ исходных файлов была связной.

Очевидно, что множество всех файлов $\mathbf{F}$, с определенной на нем сигнатурой операций $\Omega_F = \langle$*сортировка*, *выборка*, *сжатие*, *слияние строго упорядоченных файлов*, *слияние нестрого упо-*

*рядоченных файлов*$\rangle$, образует универсальную алгебру. Выражение (1) рассматривается как алгебраическое выражение в этой алгебре.

Сформулированные выше необходимое и достаточное условия завершают построение теоретико-множественной модели для обработки файлов.

## 3. Алгебра многомерных матриц

Алгебра многомерных матриц подробно рассмотрена в [7]. Рассмотрим некоторую совокупность элементов $T = a_{i_1 \dots i_p}$, над которыми определены две операции, одна из которых рассматривается как аддитивная, а вторая как мультипликативная.

Определение 18. *$p$-мерная матрица — это совокупность элементов $a_{i_1 \dots i_p}$, где индексы $i_1, \dots, i_p$ принимают значения от 1 до $n_\alpha$ $(\alpha = 1, \dots, p)$ соответственно.*

Таким образом, $p$-мерная матрица содержит $n_1 \times \dots \times n_p$ элементов. Обозначим многомерную матрицу $A = \|a_{i_1 \dots i_p}\|$. Для многомерных матриц определяются операции сложения и умножения.

Определение 19. *Суммой двух $p$-мерных матриц $A = \|a_{i_1 \dots i_p}\|$ и $B = \|b_{i_1 \dots i_p}\|$ с одинаковыми наборами индексов $i_1, \dots, i_p$ называется $p$-мерная матрица $C = \|c_{i_1 \dots i_p}\|$ с тем же набором индексов, элементы которой вычисляются по формуле $c_{i_1 \dots i_p} = a_{i_1 \dots i_p} + b_{i_1 \dots i_p}$.*

Пусть даны две матрицы $A = \|a_{i_1 \dots i_p}\|$ и $B = \|b_{i_1 \dots i_q}\|$, $p$- и $q$-мерные соответственно. Разобьем совокупности индексов $i_1, \dots, i_p$ и $i_1, \dots, i_q$ на четыре группы, содержащие соответственно $\varkappa$, $\lambda$, $\mu$ и $\nu$ индексов ($\varkappa$, $\lambda$, $\mu$, $\nu \geqslant 0$). Причем $\varkappa + \lambda + \mu = p$, а $\lambda + \mu + \nu = q$. Обозначим группы индексов как $l = (l_1, \dots, l_\varkappa)$, $s = (s_1, \dots, s_\lambda)$, $c = (c_1, \dots, c_\mu)$ и $m = (m_1, \dots, m_\nu)$. Представим матрицы $A$ и $B$ в виде $A = \|a_{lsc}\|$ и $B = \|b_{scm}\|$. Индексы групп $s$ и $c$ в матрицах $A$ и $B$ полностью совпадают.

Определение 20. *Матрица $C = \|c_{lsm}\|$, элементы которой вычисляются по формуле $c_{lsm} = \sum\limits_{(c)} a_{lsc} \times b_{scm}$, называется произведением матриц $A$ и $B$.*

Алгоритм реализации этой операции состоит в следующем:

перемножаются все пары элементов, у которых полностью совпадают значения индексов групп $s$ и $c$,



суммируются все произведения с одинаковыми значениями индексов группы $c$.

Произведение многомерных матриц называется $(\lambda, \mu)$-свернутым произведением и обозначается $\lambda, \mu(A \times B)$.

Из определения $(\lambda, \mu)$-свернутого произведения следует, что для любой пары многомерных матриц можно построить много различных произведений, подбирая различные значения $\lambda$ и $\mu$.

Пример 3. Рассмотрим $(\lambda, \mu)$-свернутое произведение на примере умножения двух трехмерных матриц.

Пусть дан набор индексов $i$, $j$, $k$, $l$, размерности которых $n_i$, $n_j$, $n_k$, $n_l$ соответственно. Определим трехмерные матрицы $A = \|a_{ijk}\|$ и $B = \|b_{jkl}\|$, тогда

четырехмерная матрица $C = \|C_{ijkl}\|$, элементы которой вычисляются по формуле $c_{ijkl} = a_{ijk} \times b_{jkl}$ для всех совпадающих значений индексов $j$ и $k$,

трехмерная матрица $C = \|C_{ikl}\|$, элементы которой вычисляются по формуле $c_{ikl} = \sum\limits_{j=1}^{n_j} a_{ijk} \times b_{jkl}$ и

двумерная матрица $C = \|C_{il}\|$, элементы которой вычисляются по формуле $c_{il} = \sum\limits_{j=1}^{n_j} \sum\limits_{k=1}^{n_k} a_{ijk} \times b_{jkl}$,

рассматриваются как различные формы произведения матриц $A$ и $B$.

Кроме рассмотренных бинарных операций, в [8] предлагаются три унарные операции: транспонирование, сечение и свертка. Перечисленные операции составляют сигнатуру алгебры многомерных матриц $\Omega_M = \langle$*транспонирование*, *сечение*, *свертка*, *сложение*, $(\lambda, \mu)$-*свернутое произведение*$\rangle$.

## 4. Изоморфизм алгебры файлов и алгебры многомерных матриц

Гомоморфизм этих алгебр достаточно очевиден.

Пусть файл $X_K$ строго упорядочен по множеству ключей $K$. Можно установить взаимно-однозначное соответствие между множеством значений ключа с номером $\alpha$ и конечным множеством натуральных чисел, заключенных между 1 и $n_\alpha$, где $n_\alpha$ — количество экземпляров данного ключа. Таким образом, устанавливается взаимно-однозначное соответствие между множеством ключей файла $X_K$ и совокупностью индексов $p$-мерной матрицы.

Каждому классу эквивалентности файла $X_K$, соответствующему фиксированному экземпляру множества ключей $K^*$, ставится в соответствие элемент $p$-мерной логической матрицы, индексы которого соответствуют $K^*$, а значение вычисляется по правилу

$$x_{i_1^* \ldots i_p^*} = \begin{cases} 0, & \text{если } X_{K^*} = \Theta, \\ 1 & \text{в противном случае.} \end{cases}$$

В результате каждому файлу $X_K$ ставится в соответствие многомерная логическая матрица $X = \|x_{i_1 \ldots i_p}\|$. При таком отображении нескольким файлам может соответствовать одна и та же логическая матрица. Над элементами такой матрицы определяются логические операции дизъюнкции (аддитивная) и конъюнкции (мультипликативная). Алгебры однотипные, так как сигнатуры $\Omega_F$ и $\Omega_M$ содержат одинаковое число операций. Кроме того, в обеих сигнатурах три унарные и две бинарные операции. В [6, 9] доказано, что операциям сортировки, выборки и сжатия файлов соответствуют операции транспонирования, сечения и свертки многомерных матриц, а операциям слияния строго упорядоченных файлов и слияния нестрого упорядоченных файлов соответствуют операции сложения и $(\lambda, \mu)$-свернутого произведения многомерных матриц.

Для доказательства изоморфизма используется механизм универсальных алгебраических машин. В [3, 5] показано, как может быть построена универсальная алгебраическая машина для обработки обычных плоских матриц, не зависящая от типов элементов. Аналогично может быть построена универсальная алгебраическая машина, реализующая алгебру файлов, независимо от типов записей. Структурно такая машина в языке Object Pascal задается как совокупность типов:

1. *Типы процедур обработки элементов матриц*:
   *TNull* = ***procedure of object***; (*обнуление*)
   *TAssign* = ***procedure of object***; (*присваивание*)
   *TAdd* = ***procedure of object***; (*Сложение*)
   *TMult* = ***procedure of object***; (*Умножение*)
2. *АТД — универсальная матричная машина*:
   *TUniversalMatrixMachine* = ***class*** (*TObject*)
   *Ri*, *Rj*, *Rk*: ***word***; (*регистры индексов*)
   (*указатели на процедуры-операции над элементами типа*:)
   *EAdd*: *TAdd*;
   *EMult*: *TMult*;
   *ENull*: *TNull*;
   *EAssign*: *TAssign*;
   (*инициализация операций над элементами типа*:)
   ***procedure*** *Init*(*_EAdd*: *TAdd*; *_EMult*: *TMult*;
   *_ENull*: *TNull*, *EAssign*: *TAssign*);
   (*операции над абстрактными матрицами*:)
   ***procedure*** *MatTransposition*; (*транспонирование*)
   ***procedure*** *MatAdd*; (*сложение*)
   ***procedure*** *MatMult*; (*умножение*)
   ***end***;

Подробно эти типы описаны в [3]. Кратко:

*TNull*, *TAssign*, *TAdd*, *TMult* – типы процедур для операции построения нейтрального элемента, присваивания, аддитивной и мультипликативной операций для элементов матриц, которые будут заданы в объекте-наследнике, реализующем алгебру матриц с конкретным типом элементов;

*EAdd*, *EMult*, *ENull*, *EAssign* – имена для вызова этих процедур в операциях над матрицами;

*MatTransposition*, *MatAdd*, *MatMult* – процедуры-реализации операций алгебры матриц (в АТД, реализующем алгебру многомерных матриц, добавляются процедуры *MatCut* для операции сечения и *MatConvolve* для операции свертки).

Далее рассматриваются два примера реализации операции сложения записей, которые могут быть использованы в операциях *слияния строго упорядоченных файлов* или *сложения матриц*.

Пример 4. Слияние файлов *Трудовые затраты* и *Материальные затраты* из примера 2. Записи этих файлов имеют следующие структуры в Object Pascal:

```
TWorkCost = record
    Subassembly_Detail: string [20]; (узел_деталь)
    Shop: string [3];                (цех)
    WorkCost: single;                (трудовые затраты)
end;
TMatCost = record
    Subassembly_Detail: string [20]; (узел_деталь)
    Shop: string [3];                (цех)
    MatCost: single;                 (материальные затраты)
end;
```

В результате слияния получается выходной файл *tmp*1 с записями типа:

*TCosts* = ***record***
*Subassembly_Detail*: *string* [20]; (*узел_деталь*)
*Shop*: *string* [3]; (*цех*)
*Costs*: *single*; (*затраты*)
***end***;.

Этим файлам соответствуют двухмерные матрицы $A$, $B$ и $C$ с неотрицательными действительными элементами, индексы $i$ и $j$ которых соответствуют ключевым полям *Subassembly_Detail* и *Shop*. Над элементами матриц задана аддитивная операция «сложение действительных чисел». Значения элементов матриц задаются правилом

$$x_{i^*j^*} = \begin{cases} 0, \text{ если } X_{Subassembly\_Detail^*,\ Shop^*} = \Theta, \\ WorkCost/MatCost/Cost \text{ в противном случае.} \end{cases}$$

В этом случае операция сложения элементов задается правилом

$$C[i,j] := A[i,j] + B[i,j].$$

Пример 5. Корректировка. В этой операции, рассмотренной в примере 1, используются два входных файла (матрицы): *Основной* и *Корректура*. Операция формирует выходной файл (матрицу) *Результат*.

Типы записей файлов *Основной* и *Результат* одинаковые. Они содержат ключевые поля $K_1, \ldots, K_p$ и вычисляемые поля $F_1, \ldots, F_q$. Соответствующие им $p$-мерные матрицы содержат структурные элементы (записи) $F^*$ типа $F = \{F_1, \ldots, F_q\}$. Существует специальный (нейтральный) элемент-константа вида $\mathrm{E} = \{\varepsilon_1, \ldots, \varepsilon_q\}$, где $\varepsilon_1, \ldots, \varepsilon_q$ — нейтральные элементы типов полей $F_1, \ldots, F_q$. Этим файлам соответствуют $p$-мерные матрицы $A$ и $C$ элементами типа $F$, индексы $i_1, \ldots, i_p$ которых соответствуют ключевым полям $K_1, \ldots, K_p$. Значения элементов матриц задаются правилом

$$x_{i_1^* \ldots i_p^*} = \begin{cases} \mathrm{E}, & \text{если } X_{K^*} = \Theta, \\ F^* & \text{в противном случае.} \end{cases}$$

Тип записи файла *Корректура*, кроме ключевых и вычисляемых полей, содержит поле $D$, принимающее значения из множества $\{0, 1, 2\}$, т. е. имеет тип $F^D = \{F, D\}$. Нейтральный элемент имеет вид $\mathrm{E}^D = \{\varepsilon_1, \ldots, \varepsilon_q, 0\}$. Этому файлу соответствует матрица $B$, значения элементов которой задаются правилом



$$x_{i_1^* \ldots i_p^*} = \begin{cases} \mathrm{E}^D, & \text{если } X_{K^*} = \Theta, \\ F^{D*} & \text{в противном случае.} \end{cases}$$

В этом случае операция сложения элементов задается правилом

***case*** $B[i_1, \ldots, i_p].F^D.D$ ***of***
(*сохранение записи файла Основной:*)
$0 : C[i_1, \ldots, i_p].F := A[i_1, \ldots, i_p].F$;
(*замена записи файла Основной на запись файла Корректура*:)
$1 : C[i_1, \ldots, i_p].F := B[i_1, \ldots, i_p].F^D.F$;
(*удаление записи файла Основной*:)
$2 : C[i_1, \ldots, i_p] := \mathrm{E}$;
***end***;

В обоих примерах множествам записей файлов ставились в соответствие алгебраические структуры с одной бинарной операцией, которая трактовалась как аддитивная, и нейтральным элементом — моноиды. Наличие в алгебраической структуре нейтрального элемента обязательно, поскольку матрица в отличие от файла содержит элементы для всех значений индексов. Универсальные неопределенные записи, которым соответствуют нейтральные элементы в матрицах, в физический файл не включаются. Для реализации умножения матриц следует строить алгебраические системы, которые должны включать две операции, трактуемые как аддитивная и мультипликативная. Эти системы должны быть, по крайней мере, моноидами по каждой из операций.

## 5. Параллельная реализация операций алгебры файлов

Возможность параллельной реализации операций алгебры файлов непосредственно следует из доказанного выше изоморфизма этой алгебры и алгебры многомерных матриц. Параллелизм алгебры матриц (в том числе и многомерных) давно известен и хорошо изучен. Кроме того, возможна непосредственная

реализация операций алгебры файлов на параллельных вычислительных системах.

Пример 6. Параллельная реализация операции слияния нестрого упорядоченных файлов. Пусть файлы $X_K$ и $Y_K$ нестрого упорядочены по множеству ключей $K$. И пусть файл $X_K$ состоит их классов эквивалентности, соответствующих экземплярам $K_1^*$, $K_3^*$, $K_4^*$ множества ключей $K$, а файл $Y_K$ — из классов эквивалентности $K_1^*$, $K_2^*$, $K_3^*$. Вычислительная система, ориентированная на реализацию операции слияния нестрого упорядоченных файлов $X_K$ и $Y_K$, может иметь архитектуру, подобную показанной на рис. 2.

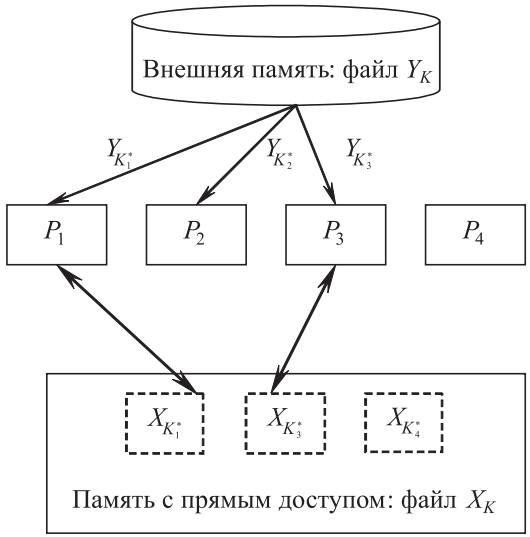


**Рис. 2.** Пример архитектуры параллельной вычислительной системы для реализации слияния нестрого упорядоченных файлов

На рис. 2 $P_1$, $P_2$, $P_3$, $P_4$ — группы, состоящие из одного или более процессоров, имеющих прямой доступ к памяти, в которой размещены классы эквивалентности файла $X_K$. При индексно-последовательной организации файла $Y_K$ все процессоры одной группы в каждый момент времени имеют доступ ко всем записям одного класса эквивалентности этого файла. Такая архитектура позволяет реализовать (при использовании в качестве внешней памяти RAD-массивов) несколько уровней параллелизма слияния нестрого упорядоченных файлов.



Алгебра файлов была также реализована на вычислительной системе с ассоциативным распределением ресурсов, а модель представления данных запатентована [10].

## Заключение

В предыдущих разделах было показано, что построение изоморфных моделей для обработки данных сводится к построению универсальных алгебраических машин с различными структурами и практически одинаковыми типами.

Операции алгебры матриц (операции структуры) могут быть реализованы по-разному. Для последовательной обработки плоских матриц — это простые программы, состоящие из двух или трех циклов. Для многомерных матриц — это гораздо более сложные программы, использующие динамические структуры данных, подобные связным спискам. В случае параллельной реализации это весьма сложные программы, требующие высокого уровня владения алгоритмизацией и программированием [11]. Поэтому их целесообразно разработать и отладить один раз, включив затем в специализированные библиотеки операционных систем. То же самое можно сказать и про операции алгебры файлов. Операции над элементами, как видно из примеров, напротив, весьма просты и сводятся к присваиваниям, возможно, незначительно осложненным условными операторами с простыми условиями. Поэтому они могут быть легко реализованы в пользовательских программах. Таким образом, легко решаются две проблемы: автоматизация программирования и кроссплатформенность. Проблема оптимизации процессов обработки данных требует отдельного исследования. Для решения этой проблемы необходимо, во-первых, оценить сложность операций над структурами. Например, известно, что при последовательной реализации сложность операций над квадратными матрицами имеет порядок $n^2$ для транспонирования и сложения и $n^2$ для умножения. Аналогичные оценки легко сделать и для многомерных матриц. Проблема оценки сложности операций алгебры файлов рассмотрена в [9]. Очевидно, что она не превосходит сложности операций над матрицами, поскольку файлы не содержат универсальных неопределенных записей, которым в матрицах соответствуют нейтральные элементы. Во-вторых, необходимо разработать алгоритмы преобразования алгебраических выраже-

ний с целью оптимизации этих выражений. Некоторые подходы к оптимизации выражений рассмотрены в [9, 12].

Таким образом, можно утверждать, что предложенный метод, основанный на алгебраическом и объектно-ориентированном моделировании, позволяет решить актуальные и сложные проблемы, возникающие при проектировании систем управления базами данных.

### Список литературы


1. *Глушков В. М.*, *Цейтлин Г. Е.*, *Ющенко Е. Л.* Алгебра. Языки. Программирование. — Киев: Наукова думка, 1989. — 376 с.
2. *Мунерман В. И.* Абстрактные алгебраические машины как технология программирования // Системы компьютерной математики и их приложения: Матер. Междунар. конф. — Смоленск: Изд-во СмолГУ, 2007. — Вып. 8. — С. 106–109.
3. *Емельченков Е. П.*, *Левин Н. А.*, *Мунерман В. И.* Математические модели для проектирования информационных систем // Системы и средства информатики. Спец. вып. Математические модели в информационных технологиях. — М.: Наука, 2005. — с. 210–225.
4. *Канторович Л. В.* Перспективы развития и использования электронных счетных машин // Математика, ее содержание, методы и значение. Т. 1. — М.: Из-во АН СССР, 1956. — С. 382–390.
5. *Емельченков Е. П.*, *Мунерман В. И.* Алгебраический подход к объектно-ориентированным базам данных // Математическая морфология: Электронный математический и медико-биологический журнал. — 2006. — Т. 5. — Вып. 4. — `http://www.smolensk.ru/user/sgma/MMORPH/N-12-html/borisov/munerman/munerman.htm`.
6. *Гендель Е. Г.*, *Мунерман В. И.* Применение алгебраических моделей для синтеза процессов обработки файлов // Управляющие системы и машины — Киев: Наук. думка, 1984. — Вып. 4. — С. 69–72.
7. *Соколов Н. П.* Введение в теорию многомерных матриц. — Киев: Наук. думка, 1972. — 175 с.
8. *Мунерман В. И.* Теоретико-множественная модель обработки данных // Системы компьютерной математики и их приложения: матер. междунар. конф. — Смоленск: Изд-во СмолГУ, 2008. — Вып. 9. — С. 114–116.
9. *Гендель Е. Г.*, *Мунерман В. И.*, *Шкляр Б. Ш.* Оптимизация процессов обработки данных на базе алгебраических моделей // Управляющие системы и машины. — Киев: Наук. думка, 1985. — Вып. 6. — С. 91–95.





10. Патент на полезную модель № 82355 Система представления данных в базе данных. Зарег. в Гос. реестре полезных моделей РФ 20 апреля 2009.
11. *Левин Н. А.*, *Мунерман В. И.*, *Сидоренкова С. В.* Реализация алгебры многомерных матриц в ассоциативной памяти систем // Системы и средства информатики. Вып. 14. — М.: Наука, 2004. — С. 86–89.
12. *Левин Н. А.*, *Мунерман В. И.* Использование алгебраических методов для повышения производительности вычислительных систем // Там же. — С. 84–86.