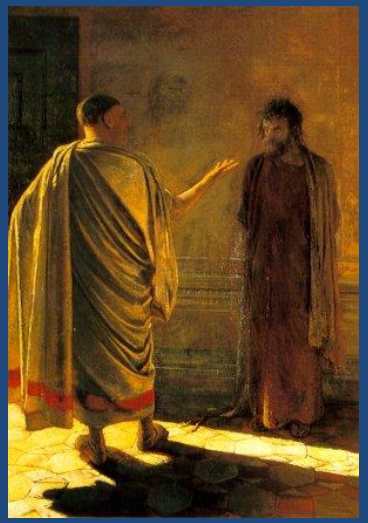

ТРЕТИЙ МАНИФЕСТ СИСТЕМ БАЗ ДАННЫХ

Ясность и истина не совпадают, но ясность – дополнение к истине.

Н. Бор

Пост-реляционные системы баз данных

Содержание

- Основные положения Третьего манифеста
- Первая "истинно реляционная" СУБД Dataphor

Третий манифест

- В отличие от первых двух манифестов
  - не содержит критики предыдущих манифестов
  - не опирается на существующие реализации СУБД
  - обновляется авторами (как технический отчет).
- Отвергает OO и SQL подходы первых двух манифестов.
- Представляет собой набор конструктивных предложений, следование которым может (по мнению авторов) привести к созданию СУБД следующего поколения, удовлетворяющим современным потребностям и базирующимся на чистой реляционной модели.

Крис Дейт
(р. 1941)

Хью Дарвен
(р. 19??)

Основные положения Третьего манифеста

- Поддержка ОО возможностей желательна, но эти возможности ортогональны реляционной модели.
  - Реляционная модель не нуждается в расширении или коррекции, чтобы можно было связать эти возможности с некоторым языком баз данных, способным представлять искомые основы.
  - Пусть такой язык существует и называется **D** (Date and Darwen's Database Dream :-).
- Язык **D** является предметом *предписаний*, *запретов* и *очень строгих суждений* двух типов:
  - RM (Relational Model) – следующих из сущности РМД, которые абсолютны и не могут быть предметом компромисса
  - OO (Other Orthogonal :-) – не относящихся непосредственно к РМД.

RM-предписания

1. Скалярные типы
2. Скалярные значения типизированы
3. Скалярные операции
4. Реальные и возможные представления
5. Раскрытие возможных представлений
6. Генератор типа TUPLE
7. Генератор типа RELATION
8. Равенство
9. Кортежи
10. Отношения
11. Скалярные переменные
12. Кортежные переменные
13. Переменные отношений (relvars)
14. Реальные и виртуальные relvars
15. Возможные ключи
16. Базы данных
17. Транзакции
18. Реляционная алгебра
19. Имена relvars, селекторы отношений и рекурсия
20. Операции над значениями отношений
21. Присваивания
22. Сравнения
23. Ограничения целостности
24. Предикаты над relvar и базами данных
25. Каталог
26. Разработка языка

RM-предписание 1

- *Скалярный тип* – это именованное множество скалярных значений.
- Язык **D** должен обеспечивать пользователям возможности определять собственные скалярные типы (*определяемые пользователями* скалярные типы); другие скалярные типы должны обеспечиваться системой (*встроенные* скалярные типы).
- Должна иметься возможность уничтожения определенных пользователями скалярных типов.
- Значениями заданного скалярного типа и переменными, значения которых должны принадлежать данному скалярному типу, можно оперировать только с помощью операций, определенных для этого типа.

RM-предписание 1

- Для каждого скалярного типа и каждого объявленного возможного представления его значений в состав этих операций должны входить:
  - Операция *selector*, служащая для выборки произвольного значения данного скалярного типа
  - Набор операций для раскрытия рассматриваемого возможного представления.
- В состав определяемых системой скалярных типов должен входить тип **truth value** (с ровно двумя значениями, *true* и *false*). Для этого типа должны прямо или косвенно определяться и поддерживаться все логические операции.

Комментарий к RM-предписанию 1

- Термины *домен* и *тип данных* интерпретируются как синонимичные и взаимозаменяемые. В том же самом смысле иногда используется термин *объектный класс*, но Д&Д не используют этот термин.
- Д&Д обобщенно называют значения домена *скалярными значениями*. Тем самым явно допускаются “скалярные” значения произвольной сложности. Например, при определенных обстоятельствах массив стеков списков и др. может рассматриваться как скалярное значение.

RM-предписание 2

- Все скалярные значения должны быть типизированы, т.е. должны позволять идентифицировать (уникальный) тип, которому они принадлежат.

RM-предписание 3

- *Скалярная операция* – это операция, которая возвращает скалярное значение или обновляет скалярную переменную.
- Язык **D** должен обеспечивать для пользователей возможности определения собственных скалярных операций (*определяемых пользователями* скалярных операций); другие такие операции (*встроенные* или *определяемые системой* скалярные операции) должны обеспечиваться системой.
- В общем случае должно быть возможно уничтожать определенные пользователями скалярные операции.

RM-предписание 4

- Пусть *T* – скалярный тип, а *v* – вид (в некотором контексте) некоторого значения этого типа. Тогда по определению у *v* имеется в точности одно **реальное представление** и одно или более **возможных представлений** (по крайней мере, одно, так как, очевидно, одно возможное представление всегда существует и совпадает с реальным представлением).
- Реальные представления, связанные с типом T, должны определяться средствами некоторого *языка определения структуры хранения* и не должны быть видимы в языке **D**. По крайней мере одно возможное представление (не обязательно совпадающее с реальным представлением), связанное с типом *T*, должно быть объявлено как часть определения *T* и, следовательно, должно быть видимо в языке **D.** Для каждого объявленного возможного представления *PR* типа *T* должна автоматически обеспечиваться операция **selector** *S* со следующими свойствами:
  - Для определенности предположим, что компоненты *PR* и параметры *S* представлены в виде упорядоченных списков. Тогда эти два списка должны содержать одно и то же число элементов, скажем *n*, и объявленные типы *i*-тых элементов списков (*i* = 1, 2, ..., *n*) должны быть одинаковы.
  - Каждое значение типа *T* должно производиться путем некоторого вызова *S*.
  - Каждый (успешный) вызов *S* должен производить некоторое значение типа *T*.

RM-предписание 5

- Пусть некоторое объявленное возможное представление *PR* для скалярного типа *T* определено в терминах компонентов *C1*, *C2*, ..., *Cn* (у каждого компонента имеются имя и объявленный тип). Пусть *v* – значение типа *T*, а *PR(v)* обозначает соответствующее возможное представление. Тогда *PR(v)* должно быть **раскрываемым** – иначе говоря, должен автоматически обеспечиваться набор операций только чтения и обновления, такой что:
  - Для всех таких значений *v* и для всех *i* (*i* = 1, 2, ..., *n*) можно “выбрать” (т.е. прочитать значение) компонента *Ci* из *PR(v)*.
  - Для любой переменной *V* объявленного типа *T* и для всех *i* (*i* = 1, 2, ..., *n*) можно обновить *V* таким образом, что если значениями *V* до и после обновления являются *v* и *v’* соответственно, то возможные представления *PR(v)* и *PR(v’)* отличаются самое большее в их компонентах *Ci*.
- Такой набор операций должен обеспечиваться для каждого возможного представления, объявленного в определении *T*.

RM-предписание 6

- Должен поддерживаться генератор типов **TUPLE**. То есть при наличии некоторого заголовка кортежа *H* должна быть возможность использовать **генерируемый тип** TUPLE {*H*} как основу определения (или, в случае значений, выборки):
  - значений и переменных этого генерируемого типа;
  - значений атрибутов кортежей и атрибутов заголовка кортежей этого генерируемого типа;
  - компонентов объявленных возможных представлений этого генерируемого типа.
- Генерируемый тип TUPLE {*H*} называется **типом кортежей**; имя этого типа – TUPLE {*H*}. Терминология *степени, атрибутов* и *заголовков*, вводимая в RM-предписании 9, должна применяться, с необходимыми изменениями, к этому типу кортежей, а также к значениям и переменным этого типа. Типы кортежей TUPLE {*H1*} и TUPLE {*H2*} одинаковы в том и только в том случае, когда *H1* = *H2*.
- В состав применимых операций должны входить аналоги операций реляционной алгебры RENAME, *project*, EXTEND и JOIN, операции присваивания кортежей и сравнения кортежей; операция выборки кортежей, операция извлечения из указанного кортежа значения указанного атрибута (степень этого кортежа должна равняться единице) и операции “вкладывания” и “выкладывания” кортежей.

RM-предписание 7

- Должен поддерживаться генератор типов **RELATION**. То есть при наличии некоторого заголовка отношения *H* должна иметься возможность использования **генерируемого типа** RELATION {*H*} как основы для определения (или, в случае значений, для выборки):
  - значений и переменных этого генерируемого типа;
  - значений атрибутов кортежей и атрибутов заголовка кортежей этого генерируемого типа;
  - компонентов объявленных возможных представлений этого генерируемого типа.
- Генерируемый тип RELATION {*H*} называется **типом отношения**, и имя этого типа – RELATION {*H*}. Терминология *степени, атрибутов* и *заголовков*, должна применяться, с необходимыми изменениями, к этому типу отношения, а также к значениям и переменным этого типа. Типы отношения RELATION {*H1*} и RELATION {*H2*} одинаковы в том и только в том случае, когда H1=H2.
- В состав применимых операций должны входить операции реляционной алгебры, операции реляционного присваивания и реляционного сравнения; операция выборки отношений, операция извлечения единственного кортежа из указанного отношения мощности один, и операции “вкладывания” и “выкладывания” отношений.

RM-предписание 8

- Для каждого типа должна поддерживаться операция сравнения **по равенству**, “=”.
- Пусть выражения *X1* и *X2* имеют значения *v1* и *v2* соответственно, причем *v 1* и *v 2* принадлежат одному и тому же типу *T* . Тогда операция *X1 = X2* вырабатывает значение *true* в том и только в том случае, если *v1* и *v2* в действительности являются одним и тем же элементом *T* (то есть тогда и только тогда, когда *X1* и *X2* имеют одно и то же значение).
- Кроме того, пусть *Op* – это операция с параметром *P* объявленного типа *T*. Тогда для всех таких операций *Op*, если значением сравнения *X1 = X2* является *true* , последствия двух (успешных) вызовов *Op*, которые отличаются только тем, что в одном из них аргументом, соответствующим *P*, является *X1*, а в другом – *X2*, должны быть неразличимы.

RM-предписание 9

- **Значение кортежа** *t* (для краткости **кортеж**) – это множество упорядоченных триплетов вида <*A*, *T*, *v*>, где:
  - *A* – имя **атрибута** кортежа *t*. Никакие два различных триплета в *t* не должны содержать одно и то же имя атрибута;
  - *T* – имя **типа** атрибута *A* кортежа *t*.
  - *v* – значение типа *T*, называемое **значением атрибута** *A* кортежа *t*.
- Мощность множества триплетов в *t*, или число атрибутов *t* называется **степенью** *t*.
- Множество упорядоченных пар <*A*, *T*>, получающихся путем удаления компонента *v* (значения) из каждого триплета, является **заголовком** *t*.
- Кортеж *t* называется **соответствующим** этому заголовку (принадлежит к соответствующему типу кортежа).
- **Степенью** заголовка является степень кортежа, а атрибутами и соответствующими типами заголовка являются **атрибуты** и соответствующие **типы** кортежа *t*. При заданном заголовке *H* должна быть доступна операция *selector* для выборки произвольного кортежа, соответствующего *H*.

RM-предписание 10

- **Значение отношения** *r* (для краткости – **отношение**) состоит из *заголовка* и *тела*, где:
  - **Заголовком** *r* является заголовок кортежа *H*.
  - Отношение *r* называется **соответствующим** этому заголовку (принадлежит к соответствующему типу отношения), а степенью *r* является **степень** этого заголовка.
  - Атрибутами и соответствующими типами *r* являются **атрибуты** и соответствующие **типы** *H*.
  - **Тело** *r* – это множество *B* кортежей, каждый из которых имеет заголовок *H*; мощность тела называется **мощностью** *r*.
- При заданном заголовке отношения *H* должна быть доступна операция *selector* для выборки произвольного отношения, соответствующего *H*.

RM-предписание 11

- **Скалярная переменная типа *T*** – это переменная, допустимыми значениями которой являются скаляры указанного скалярного типа *T*, **объявленного типа** этой переменной.
- Язык **D** должен обеспечивать для пользователей возможности определения скалярных переменных. При определении скалярной переменной должна производиться инициализация переменной некоторым значениям – явно указанным в операции определения переменной или не указанным явно, а определенным в реализации.

RM-предписание 12

- **Переменная кортежа типа TUPLE {*H*}** – это переменная, допустимыми значениями которой являются кортежи, соответствующие указанному заголовку кортежа *H*.
- **Объявленный тип** этой переменной кортежа есть TUPLE {*H*}.
- **Атрибутами** переменной кортежа являются атрибуты *H*, соответствующими типами – **объявленные типы** этих атрибутов, **степенью** переменной кортежа является степень *H*.
- Язык **D** должны обеспечивать для пользователей возможности определения переменных кортежей. При определении переменной кортежа должна производиться инициализация переменной некоторым значением – явно указанным в операции определения переменной или не указанным явно, а определенным в реализации.

RM-предписание 13

- **Переменная отношения (relation variable ,** для краткости – **relvar) типа RELATION {*H*}** – это переменная, допустимыми значениями которой являются отношения, соответствующие указанному заголовку отношения *H*.
- **Объявленный тип** relvar есть RELATION {*H*}.
- **Атрибутами** relvar являются атрибуты *H*, соответствующими типами – **объявленные типы** этих атрибутов, **степенью** relvar является степень *H*.
- Язык **D** должен обеспечивать для пользователей средства определения и уничтожения переменных relvar базы данных (для тех relvar, которые принадлежат базе данных, а не приложению – см. RM-предписание 16).
- Язык **D** должен также поддерживать возможности определения relvar на уровне приложений.

RM-предписание 14

- Переменные relvar базы данных могут быть *реальными* или *виртуальными*.
- **Виртуальная relvar** – это relvar базы данных, значением которой в любой момент времени является результат вычисления некоторого реляционного выражения, указываемого при определении этой relvar.
- **Реальная relvar** – это relvar базы данных, которая не является виртуальной. При определении реальной relvar должна производиться ее инициализация пустым отношением (т.е. отношением мощности нуль).

RM-предписания 15-17

- По определению у каждой relvar имеется по меньшей мере один **возможный ключ**. По меньшей мере один такой ключ должен быть определен при определении relvar, и не должна иметься возможность ликвидировать все возможные ключи данной relvar (кроме как ликвидировав саму relvar).
- **База данных** – это именованный контейнер relvar; содержимое базы данных в любой момент времени – это набор relvar базы данных. Операции, необходимые для определения и ликвидации баз данных, не должны являться частью языка **D** (другими словами, определение и ликвидация баз данных должна производиться “за пределами среды **D**”).
- Каждая **транзакция** должна взаимодействовать в точности с одной базой данных. Однако разные транзакции должны иметь возможность взаимодействия с разными базами данных, и разные базы данных не обязательно должны быть разъединенными. Кроме того, транзакции должны иметь возможность определять новые и уничтожать существующие relvar внутри соответствующей базы данных

RM-предписание 18

- В языке **D** должны поддерживаться обычные **операции реляционной алгебры**: RENAME, *restrict* (WHERE), *project*, EXTEND, JOIN, UNION, INTERSECT, MINUS, DIVIDEBY и SUMMIRIZE, а также кванторы всеобщности и существования.
- В языке **D** должен также обязательно поддерживаться механизм **вывода типов отношения**, благодаря чему заголовок результата вычисления произвольного реляционного выражения должен быть правильно определенным и известным как системе, так и пользователю.

RM-предписание 19

- Имена **relvar** и **вызовы селектора отношений** должны быть допустимыми реляционными выражениями.
- В реляционных выражениях должна допускаться **рекурсия**.

RM-предписание 20

- В **D** должны поддерживаться возможности для определения и уничтожения операций только чтения **со значениями-отношениями**.
- Отношение, являющееся результатом вызова такой операции, должно определяться некоторым реляционным выражением, указываемым при определении операции.
- В этом выражении должно допускаться наличие параметров в любом месте, где допускаются вызовы операций выборки.
- Вызовы таких операций внутри реляционных выражений должны допускаться в любом месте, где разрешаются вызовы селекторов отношений.

RM-предписание 21

- В **D** должно допускаться:
  - **присваивание** (значения) скалярного выражения скалярной переменной;
  - **присваивание** (значения) кортежного выражения переменной кортежа;
  - **присваивание** (значения) реляционного выражения relvar.
- В каждом случае типы источника и цели должны совпадать. В дополнение к этому в языке **D** должна поддерживаться **множественная** форма операции присваивания, в которой несколько отдельных операций присваивания выполняются параллельно как одна логическая операции.

RM -предписание 22

- В **D** должны поддерживаться **операции сравнения**:
  - Операции **сравнения скаляров** должны включать “=”, “¹ ” и, возможно, “<”, “>” и др. (в зависимости от данного скалярного типа);
  - Операции **сравнения кортежей** должны включать “=” и “<>” (и только);
  - Операции **сравнения отношений** должны включать “=”, “<>”, “является подмножеством” и др.;
  - Должна поддерживаться операция “in” для проверки вхождения кортежа в отношение.
- Во всех случаях, кроме “in”, операнды должны быть одного типа, а в случае “in” кортеж и отношение должны иметь одинаковые заголовки.

RM-предписание 23

- Выражение, при вычислении которого вырабатывается истинностное значение, называется **логическим выражением**.
- **Ограничением целостности** является логическое выражение, которое:
  - именовано;
  - является замкнутой WFF (Well Formed Formula – правильно построенной формулой) реляционного исчисления или ее логическим эквивалентом;
  - при вычислении должно вырабатывать значение *true*.
- Язык **D** должен обеспечивать возможности для определения и уничтожения ограничений целостности. Такие ограничения должны классифицироваться на ограничения **типа**, **атрибута**, **relvar** и **базы данных**, и в **D** должен поддерживаться механизм **вывода ограничений**, ассоциированный с этой схемой классификации (насколько это осуществимо).

RM-предписания 24-26

- Для каждой relvar имеется соответствующий **предикат relvar**, и для каждой базы данных имеется соответствующий **предикат базы данных**. Все такие предикаты должны удовлетворяться в границах оператора.
- Каждая база данных должна включать набор relvar, составляющих **каталог** этой базы данных. Должна существовать возможность производить присваивания для relvar каталога.
- **D** должен конструироваться в соответствии с установившимися принципами **правильной разработки языка**.

RM-запреты

1. Отсутствие упорядоченности атрибутов
2. Отсутствие упорядоченности кортежей
3. Отсутствие кортежей-дубликатов
4. Отсутствие неопределенных значений
5. Отсутствие ошибок логики неопределенных значений
6. Отсутствие конструкций физического уровня
7. Отсутствие операций уровня кортежей
8. Отсутствие составных атрибутов
9. Отсутствие возможности преодоления проверок доменов
10. Не SQL

ОО-предписания

1. Проверка типов во время компиляции
2. Поддержка наследования (простого и/или множественного) в соответствии с моделью Д&Д (25 IM-предписаний)
3. Вычислительная полнота языка **D**
4. Явное начало и завершение транзакций
5. Поддержка вложенных транзакций
6. Агрегатные операции для пустых множеств должны выдавать начальные значения операции

Первая "истинно реляционная" СУБД Dataphor

- Возможность определения пользовательских типов данных, равноправие этих типов со встроенными типами.
- Работа с отсутствующей информации без привлечения неопределенных значений и трехзначной логики.
- Обновляемые представления без ограничений, свойственных SQL.
- Разработан и реализован язык баз данных D4, претендующий на то, чтобы стать открытым стандартом, заменяющим SQL.
  - Основан на реляционной алгебре.
  - Обладает вычислительной полнотой.
  - Позволяет более точно, чем в SQL, сформулировать запросы к сложным данным.

OO-запреты

- Relvars – это не домены
  - Отношение – это не объектный класс.
- Кортеж не может содержать ОИД.
  - Отвергаются идея о том, что ОИД могли бы использоваться в “объектах” для совместного использования “подобъектов”; идея о том, что пользователи могли бы быть обязаны “разыменовывать” такие ОИД, чтобы получать значения.

Очень строгие RM-суждения

1. Системные ключи
   - Продуцирование системой уникальных значений потенциального ключа либо атрибута первичного ключа.
2. Внешние ключи
   - Возможность определения в языке **D** ограничений ссылочной целостности.
3. Вывод потенциальных ключей
   - Возможность вывода системой потенциальных ключей.
4. Ограничения переходов
   - Ограничения на допустимые изменения значения relvar или базы данных
5. Миграция от SQL
   - Следует обеспечить возможность реализации языка **SQL** средствами языка **D**, что обеспечит безболезненный переход к **D** для пользователей SQL.

Очень строгие ОО-суждения

1. Наследование типов необходимо поддерживать
2. Типы и операции не связаны
3. Генераторы типов коллекций
4. Преобразование значений типов коллекций в отношения и обратных операций
5. Одноуровневое хранение

Заключение

- Трудно рассчитывать, что ведущие производители SQL-ориентированных СУБД решатся радикально изменить свою технологию с переориентацией на Третий манифест.