Обзор семантических ограничений для моделей баз данных

Б. Тальхайм

Моделирование семантики - одна из наиболее трудных задач в проектировании баз данных. Для этой цели в различных моделях используются ограничения целостности, которые выражают ограничения, накладывемые предметной областью, определяют связи между компонентами и описывают поведение базы данных. Их использование зависит от разнообразия используемой в модели системы типов данных. Реляционная модель использует простую систему типов и имеет большое количество ограничений целостности. Семантические модели используют более богатые типовые системы, которые выражают различные виды ограничений целостности. В то же время, теория ограничений целостности является более общей. Объектно-ориентированные модели позволяют использовать как простые, так и подобные применяющимся в семантических моделях типовые системы. В данный момент теория ограничений целостности все еще находится в стадии развития. Этот обзор является попыткой дать общий подход к изучению ограничений целостности[1].

1 Введение

Целью данного обзора является дать общее систематическое введение в теорию ограничений целостности в базах данных. Первым шагом в построении теории баз данных является строгое определение модели данных. Без этого теория не может быть использована для проектирования, анализа и применения схем, протоколов доступа и собственно баз данных. *Модель базы данных* - это набор положений, математически строго определяющих структурные свойства хранящихся

[1]Этот обзор является продолжением [87], а также сокращенной и дополненной версией [89].

в базе данных объектов, их поведение. При аксиоматическом подходе она задается описанием своей структуры и операторов. Свойства структур данных задаются аксиомами, которые являются формальными утверждениями, достаточно простыми, чтобы быть самоочевидными. Семантика также задается аксиомами, или *ограничениями целостности*, которые описывают допустимые состояния и последовательности состояний базы данных.

Реляционная модель является одной из важнейших моделей баз данных. Основное преимущество этой модели — ее однородность. Все данные представляются в виде таблиц, в которых строки имеют один и тот же формат. Каждая строка в такой таблице описывает один из объектов существующей в реальном мире связи. Преимуществами реляционной модели являются простая и удобная модель данных, возможность обеспечить логическую и физическую независимость без обращения к средствам доступа к данным, предоставление пользователям языков высокого уровня, которые могут быть использованы без специальной подготовки, оптимизация доступа к базе данных, обеспечение целостности и конфиденциальности, общий подход к проектированию схем и баз данных.

Эти возможности основаны на теории, ядром которой является теория ограничений целостности и зависимостей в реляционной модели. Ограничения целостности могут быть использованы в качестве языка описания семантики базы данных. Они описывают различные типы связей между данными. Поскольку такие связи очень разнообразны, в литературе рассматривается множество (более сотни) классов зависимостей. Изучая их свойства, можно установить, как взаимосвязаны различные типы зависимостей. Эти свойства могут использоваться как правила вывода, позволяющие получить новые зависимости из имеющихся, построить замыкание множества зависимостей. Таким образом, мы можем проверить, являются ли два множества зависимостей эквивалентными, выявить избыточность множества зависимостей. Решение этих задач является важным шагом на пути к автоматическому проектированию схем баз данных.

В настоящий момент имеется как минимум шесть областей, в которых используется теория зависимостей:

1. нормализация для более эффективного хранения, поиска и модификации данных;
2. разложение базы данных на эквивалентные подбазы;
3. использование зависимостей в дедуктивных базах данных для вывода новых отношений из уже существующих;

4. проверка зависимостей для разработки более совершенного и удобного языка проектирования баз данных;
5. автоматическая проверка непротиворечивости;
6. преобразование запросов в более эффективную форму.

Эти направления в настоящее время требуют дополнительных исследований. Например, нормализация до сих пор неправильно понимается и трактуется в литературе по базам данных.

Ограничения целостности рассматриваются, как правило, только для реляционной модели. Более сложные модели используют более богатые системы типов данных, которые позволяют непосредственно описывать структурные ограничения целостности. В общем случае модель базы данных, как и модели абстрактных данных, может быть задана описанием структуры, операций и семантики. В моделях баз данных операции задаются как в общем виде, так и конкретно. Основные операции, такие как *Insert* (вставка), *Delete* (удаление) и *Update* (обновление) определяются структурой модели.

Теория, построенная для реляционной модели, может быть перенесена на большинство других моделей. В настоящее время теории различных моделей баз данных находятся на разных уровнях развития. Более того, для некоторых моделей, таких как сетевая и иерархическая, теория не развита вообще.

Данная работа построена следующим образом. Во втором разделе дано краткое введение в моделирование баз данных и их структур, алгебру баз данных и теорию ограничений целостности вообще. Третий раздел посвящен структурным и алгебраическим ограничениям. Рассматривается идентификация объектов в базах данных и понятие ключа, изучаются алгебраические ограничения и обсуждаются классы ограничений. В четвертом разделе обсуждаются классы зависимостей в различных моделях. Используя общий подход, вводятся и формально интерпретируются ограничения целостности в реляционной, семантической, объектно-ориентированной и дедуктивной моделях.

Удивительно, что результаты, полученные в теоретических исследованиях и особенно по основам семантики, почти не известны исследователям баз данных. Существует множество работ, повторяющих результаты, полученные для реляционной модели. Следующие книги и обзорные статьи могут быть рекомендованы для более глубокого ознакомления с моделированием семантики в базах данных [5, 23, 63, 72, 84, 85, 90, 101], основами теории (дедуктивных) баз данных [38, 43, 44, 68, 91], логическими основами [8, 40, 41], теорией типов и спецификаций [69, 97] и системами баз данных и баз знаний

[13, 29, 32, 35, 36, 57, 58, 60]. Библиографии в этих книгах содержат ссылки на более ранние исследования. Библиография в [55] содержит ссылки на все основные источники до 1980 года. Из-за ограничений на объем полная библиография (она содержит более 3000 работ) не может быть включена в данный обзор. Теоремы, используемые в изложении, взяты из [84, 85]. Кроме того, каждый раздел завершается дополнительными библиографическими ссылками.

2 Моделирование баз данных

2.1 Механизмы обобщения

Модель базы данных содержит описание ее структуры, семантики, операций и поведения во времени. Строительным блоком, общим для всех моделей, является небольшой набор механизмов обобщения и абстракции. Они помогают проектировщику и пользователю понимать, классифицировать, моделировать и использовать знания. Используя методы обобщения, проектировщик может классифицировать сущности реального мира и моделировать его как множество объектов и отношений между ними. В соответствии с [66] мы выделяем три основных типа обобщений: *Обобщение понятий* (*классификация / реализация*, *объединение / декомпозиция*, *обобщение / конкретизация*), *Обобщение локализации* (*именование, параметризация, шаблоны*) и *Обобщение реализации* (*инкапсуляция, скрытие, обзор*).

2.2 Основы алгебры для баз данных

В любой базе данных или системе баз данных имеется три уровня. Первый - это среда данных, называемая схемой данных, которая задается системой (или предполагаемым множеством) доступных отдельных данных. Второй уровень - это схема базы данных. Третий - это сама база данных, отражающая состояние конкретных данных и системы знаний для рассматриваемого приложения.

Второй уровень по-разному интерпретируется в различных моделях баз данных. Тем не менее, имеются общие признаки, в особенности для конструкторов типов. Для их обсуждения мы рассмотрим общий подход к определению типа. В большинстве моделей характерно определение операций в соответствии со структурой типа. В некоторых объектно-ориентированных моделях вместо таких операций используются операции, определяемые пользователем.

2.2.1 Системы типов

Конструктор типов - это функция, порождающая новые типы из уже существующих. Конструктор может быть дополнен:

- *переключателем* для выбора (типа *Select*) и *функциями модификации* (типа *Insert*, *Delete* и *Update*), отображением из множества значений нового типа в значения его типов–компонент или другого нового типа,
- *критериями корректности* и правилами проверки,
- правилами по *умолчанию*,
- одной или несколькими *пользовательскими реализациями*,
- *физической реализацией* или ее свойствами.

Кроме того, функции могут быть определены на каждом типе и использоваться в других конструкторах типов. Множество построенных типов и функций может рассматриваться в качестве сигнатуры логического языка, который может быть применен для представления семантической информации и определения производных данных и знаний.

Конструкторы типов определяют системы типов над схемой данных, иными словами, наборы построенных множеств данных. В некоторых моделях баз данных конструкторы типов основаны на семантике указателей.

Обобщенные операции на типовых системах могут быть заданы с помощью *структурной рекурсии*. Пусть даны типы T, T' и коллекционный тип C^T на T (например, множества значений типа T, списки, наборы), операции типа обобщенного объединения $\cup_{C^T}$, обобщенного пересечения $\cap_{C^T}$, обобщенный пустой элемент $\emptyset_{C^T}$. Пусть, далее, h_0 - элемент на T' и заданы две функции

$$h_1 : T \to T' \qquad\qquad h_2 : T' \times T' \to T'.$$

Определим структурную рекурсию при 'вставочном' представлении R^t на T следующим образом

$\mathrm{srec}_{h_0,h_1,h_2}(\emptyset_{C^T}) = h_0$

$\mathrm{srec}_{h_0,h_1,h_2}(\{\!|s|\!\}) = h_1(s)$ для одноэлементного $\{\!|s|\!\}$

$\mathrm{srec}_{h_0,h_1,h_2}(\{\!|s|\!\} \cup_{C^T} R^t) = h_2(h_1(s), \mathrm{srec}_{h_0,h_1,h_2}(R^t))$ тогда и только тогда, когда $\{\!|s|\!\} \cap_{C^T} R^t = \emptyset_{C^T}$.

Этот тип структурной рекурсии может быть расширен до структурной рекурсии для 'представления объединения'

$\mathrm{srec}_{h_0,h_1,h_2}(\emptyset_{C^T}) = h_0$

$\mathrm{srec}_{h_0,h_1,h_2}(\{\!|s|\!\}) = h_1(s)$ для одноэлементного $\{\!|s|\!\}$

$\mathrm{srec}_{h_0,h_1,h_2}(R_1^t \cup_{C^T} R_2^t) = h_2(\mathrm{srec}_{h_0,h_1,h_2}(R_1^t), \mathrm{srec}_{h_0,h_1,h_2}(R_2^t))$ тогда и только тогда, когда $R_1^t \cap_{C^T} R_2^t = \emptyset_{C^T}$.

Первый тип рекурсии связан с последовательной обработкой наборов, второй - с параллельной в стиле 'разделяй и властвуй'. Последний тип является более выразительным.

Структурная рекурсия может быть ограничена функциями $h_0 = \perp$ и функцией объединения $\cup_{T'}$ для нулевого элемента $\perp$ в T'. В этом случае структурная рекурсия всегда корректно определена. Для первого и второго типов структурной рекурсии проблема проверки корректности в общем случае неразрешима. Ограниченная структурная рекурсия задается функцией h_1. Она может рассматриваться как *расширение* функции h_1:

$$ext(h_1)(R^t) = \mathrm{srec}_{\perp,h_1,\cup_{T'}}(R^t).$$

Как показано в [24] операция ext эквивалентна *композиции* (comprehension) [95]. Включения покрывают различные типы коллекций, такие как индексированные структуры, массивы, 'или'-множества (or-sets), деревья и графы.

'Обобщенный собиратель', названный в [16] 'помпой' ('pump'), может быть представлен структурной рекурсией с $T' = \mathbb{N}$. Примерами таких операций является sum, каторая начинается с 0 и использует $+$ в качестве функции h_2: $pump = \mathrm{srec}_{0,h_1,+} = ext(h_1)$. Операции сборки реляционной алгебры являются частными случаями операций такого типа.

Другим примером является параметрический итератор map, который 'реструктурирует' каждый элемент или множество элементов на T. Тип T' в этом случае является коллекционным. Например, берется один функциональный параметр, допустим h_1, и затем при применении к множеству $R^t \subseteq T^t$, дает результат $\{h_1(\{s\}) \mid s \in R^t\}$, то есть $\mathrm{srec}_{\emptyset,h_1,\cup} = ext(h_1)$.

Операция *вложения* основана на отношении эквивалентности для одного или нескольких атрибутов, которое делит элементы T^t на классы по общим значениям и сопоставляет каждому такому классу множество значений h_2 на его элементах.

Операция filter (фильтр) основана на разбиении h_1 и определяется для $h_2 = \cup$ и $T' = \{T\}$, то есть $\mathrm{srec}_{\emptyset,h_1,\cup} = ext(h_1)$. Это может быть задано формулой α из L_T с одной свободной переменной x (то есть $\alpha = \alpha(x)$, $\mathrm{filter} = \mathrm{filter}_\alpha$)

$$h_1(\{s\}) = \begin{cases} \{s\} & \text{если } \models \alpha(s) \\ h_0 & \text{если } \not\models \alpha(s) \end{cases}$$

Выражение SQL типа `Select... From... Where...` является

выражением вида *map*(filter(...)). Конструкция `Group By...` является частным случаем выражения *map*.

Структурная рекурсия на коллекционных типах вместе с каноническими операциями дает мощный инструмент программирования структур баз данных. Если ограничить включения 'простыми' отношениями, они будут выражать в точности те запросы, которые представлены в реляционной алгебре. Добавляя в синтаксис включения конечное множество операций сборки, таких как *sum, count, max, min* и огранивая ввод 'простыми' отношениями, мы получим язык такой же выразительности, как SQL.

Небходимо заметить, что выразительность структурной рекурсии также ограничена. В ней не могут быть описаны недетерминированные программы, порожденные кортежами (или объектами). Опеределение структурной рекурсии для 'представления объединения' выражает также свойство отделимости. Поэтому программы, которые нельзя определить на объединении непересекающихся частей, не представимы. Однако, большинство используемых в базах данных операций и все операции SQL основаны на разделении. Понятие *смешанного комбинатора* (traversal combinator) [37] является более общим. Оно охватывает большое семейство сохраняющих тип примитивных функций. В это семейство входят большинство функций, задаваемых последовательностью рекурсивных программ, при каждом вызове которых уменьшается только один параметр. Однако такое ограничение исключает такие функции, как структурная эквивалентность, упорядочивание, поскольку они требуют параллельной обработки своих входов. Стандартный смешанный комбинатор обобщает это понятие комбинаторами, берущими функции как входы, а в качестве выхода возвращающие новую функцию, осуществив необходимую редукцию.

Другой очень удобной конструкцией является *именование*. Каждое понятие, каждый класс понятий имеет имя. Эти имена могут быть использованы для определения новых типов.

Некоторые типы могут быть использованы в других только при определенных ограничениях. Такие ограничения могут быть заданы предусловиями использования элементов данного типа и постусловиями принятия результата функции этого типа.

Множество типов может быть организовано в иерархию так, чтобы связанные понятия были в отношении имеет_тип (is-a-relationship). Типы могут иметь множество супертипов.

Множество понятий с общим содержанием называется *классом* и может быть сгруппировано в понятия-классы, называемые *сущностями*.

Необходимо подчеркнуть, что слово 'класс' может использоваться как минимум в трех различных значениях. Класс определяет: содержание понятия, обобщения понятия, или его представление. Мы будем использовать второе значение.

Модели различаются также определениями типов. Определение типов может быть рекурсивным или строго иерархическим. *Рекурсивные определения типов*, как например в функциональной или объектно-ориентированной моделях, требуют дополнительных критериев корректности. В противном случае, количество реализаций типов бесконечно. Определения типов могут повторно использовать части других типов или связывать понятия различных типов. Такие понятия как общие атрибуты методов требуют введения наследования и совместного доступа. Наследование является одним из главных механизмов в развитых системах баз данных. Его следует отличать от иерархической организации.

2.2.2 Реляционная модель.

Прежде всего рассмотрим реляционную модель баз данных. Для этой модели построена развитая теория, которая может использоваться в большинстве остальных моделей.

Схема данных $DD = (U, \underline{D}, dom)$ задается:
конечным множеством U *элементарных атрибутов*,
множеством $\underline{D} = \{D_1, D_2, ...\}$ (возможно пересекающихся) *доменов*(алфавитов) и
функцией $dom : U \longrightarrow \underline{D}$, которая ставит в соответствие каждому элементарному атрибуту его домен. Теперь определим *реляционную схему* $R((B_1, ..., B_n), \Sigma)$ как набор элементарных атрибутов и множество (локальных) ограничений целостности (они будут рассмотрены в следующем разделе; пока мы можем полагать $\Sigma = \emptyset$, т.е. $R = ((B_1, ..., B_n), \emptyset))$. Множество атрибутов R обозначается $attr(R) = \{B_1, ..., B_n\}$.

Для некоторого множества $X \subseteq U$ кортеж t на X определяется как функция, ставящая в соответствие каждому $A \in X$ некоторое значение из $dom(A)$. Конечное множество R^t кортежей на $(B_1, ..., B_n)$ является отношением на реляционной схеме R (или допустимой реализацией этой схемы), если Σ справедливо в R^t.

Множество всех допустимых реализаций схемы R обозначается $SAT(R)$.
Эти определения могут быть распространены на наборы различных реляционных схем, называемые *схемами баз данных* DS, т.е.

$DS = ((R_1, ..., R_n), \Psi)$, где Ψ – множество (глобальных) ограничений целостности на DS. Аналогично, могут быть введены допустимые реализации баз данных на DS.

Операции *Insert, Delete* и *Update* на DS определяются соответственно как вставка, удаление кортежей и их модификация при выполнении определенных условий. Эти операции расширяются до реляционной алгебры, которая основана на алгебре множеств. Реляционная алгебра может использоваться для поддержки процедурных языков, она также содержит декларативную часть — *реляционное исчисление*.

Реляционная модель основана на семантике множеств. Однако часто описание моделей вводят через *понятие мультимножества*. SQL в этом смысле также основан на семантике мультимножеств. Это смешение ведет к сложностям в интерпретации значений операций над базами данных.

2.2.3 Другие модели

Различные модели баз данных могут быть охарактеризованы частотой их использования в приложениях. Существует множество различных оценок, например, в [74] использование систем в промышленных приложениях оценивается следующим образом:

	файловая	реляционная	сетевая	иерархическая	другие
1990	42.9	24.7	4.2	23.1	4.7
1998	21.7	53.5	6.2	13.3	5.3

Эти числа не стоит переоценивать, однако они показывают важность дальнейшего развития теории реляционной модели.

Типы в различных моделях

Попробуем развить данный подход в применении к другим моделям баз данных. В большинстве моделей различаются мета-данные (например, типы), классы, и элементы этих классов, конкретные данные. Схема базы данных также может быть рассмотрена как база данных [16]. Большинство моделей имеют рассмотренную выше трехуровневую архитектуру, то есть прежде всего описывают структуру данных, затем схему базы данных и, наконец, определяют множество возможных реализаций. Например, классическая реляционная модель разделяет схему данных, схему реляционной базы данных и множество допустимых реляционных баз данных.

Для множества имен $Names$ мы можем построить множество вложенных атрибутов $NestAttr$, положив $\{\emptyset\} \cup U \subseteq NestAttr$ и рекурсивно применяя следующее правило:
Если $X_1, ..., X_n, Y$ – различные элементы $NestAttr$ и $X \in Names$, то $X(X_1, ..., X_n)$ – вложенный атрибут в $NestAttr$, значениями которого являются кортежи, и $X\{Y\}$ – (множественно-значный) вложенный атрибут в $NestAttr$.

Соответственно определяются домены вложенных атрибутов.

Подобный подход может быть применен к различным моделям. Их описание зависит от допустимости рекурсивных определений и повторного использования структур.

Вложенная реляционная схема определяется как упорядоченный набор (вложенных) атрибутов в совокупности с множеством ограничений целостности.

Объектная схема задается множествами (вложенных) атрибутов и ограничений целостности. Схема связей определяется последовательностью объектных схем и множествами (вложенных) атрибутов и ограничений целостности. Расширенная ER-модель[2] [85] позволяет ввести иерархию связей: схема связей порядка i определяется последовательностью схем связей порядка менее i и множеством (вложенных) атрибутов в совокупности с множеством ограничений целостности, при этом схемы объектов рассматриваются как схемы связей порядка 0. ER-модель также может быть определена как объектная схема и схема связей, основанная на ссылках. В этом случае ограничения целостности имеют другой смысл. Однако, поскольку вторая интерпретация этой модели может быть преобразована в первую, мы будем использовать семантику множеств для ER-моделей.

Сетевая модель использует списки кортежей и ссылок. Она является примером нерекурсивных моделей. Часто полагают, что сетевая модель состоит из множеств кортежей и связей между ними. Однако, эта интерпретация не используется в существующих системах. Такое определение сетевой модели не позволяет рассматривать ER-модель как обобщение сетевой.

Основу функциональной модели составляют объекты и функции. Объекты рассматриваются как отдельные понятия, функции отображают объекты в другие объекты или в данные. Функциями представляются связи между объектами, атрибуты и операции.

Основное понятие объектно-ориентированных моделей —

[2]ER-модель (entity-relationship model) — модель, описывающая некоторые объекты (сущности) и связи между этими объектами. (прим. перев.)

идентификатор. В этих моделях используется другая алгебра операторов. В частности, неинъективная функция *map* используется на уровне идентификаторов. Примером такой операции является проекция. Если два кортежа t, t' проектируются в один и тот же кортеж t'', то значение t'' ассоциируется с обоими идентификаторами. Другие понятия, вводимые для объектно-ориентированных баз данных в соответствии с [1] - это замещение, перегрузка и затем связывание, вычислительная полнота, расширяемость, устойчивость, перераспределение памяти, множественное наследование, проверка типов и их вывод, транзакции, версии. Для прояснения смысла этих понятий требуются дополнительные исследования.

Скрипт - расширение понятия фрейма, использующее функции с пред- и постусловиями.

Выше теория баз данных излагалась с позиции теории моделей. В дедуктивных базах данных используется подход теории доказательств. Дедуктивные базы данных являются важным случаем *баз знаний*. Такие базы данных могут рассматриваться как множества фактов или исходных формул, а ответы на запросы получаются путем доказательства теорем, основанных на фактах. В этом случае предполагаются справедливыми следующие допущения.

Допущение *полноты доменов* постулирует, что домены включают все возможные значения соответствующих атрибутов и только их.

Допущение *единственности имен* требует, чтобы объекты с разными именами были различными.

Допущение *замкнутости мира* предполагает, что предметная область состоит только из элементов базы данных.

Такой подход дает метод обработки запросов, корректный при наличии неопределенных значений, а также метод контроля соблюдения ограничений целостности и может быть использован для концептуального моделирования семантики предметной области. Поскольку все реляционные выражения представимы в виде формул, а транзитивное замыкание представимо в виде формулы, но не в реляционной алгебре, данное исчисление предоставляет более мощный аппарат обработки запросов, чем реляционное. Так как запросы, ограничения целостности, факты и операции описываются на одном и том же языке, такой подход является достаточно общим. Кроме того, поскольку логика может быть использована в качестве языка представления знаний (или модели знаний), такой подход позволяет осуществлять непосредственный переход от баз данных к базам знаний.

Дедуктивная база данных описывается набором формул логики предикатов первого порядка. Формулы могут быть представлены в

сколемовской форме

$\gamma((\alpha_1 \wedge \alpha_2 \wedge .... \wedge \alpha_n) \rightarrow (\beta_1 \vee \beta_2 \vee ... \vee \beta_m))$

где α_i, β_j – атомарные формулы (возможно замкнутые, то есть использующие только константы).

Дедуктивная база данных — это тройка DDB = (*Факты, Правила, Ограничения_целостности*). Эта модель может быть расширена до моделей, использующих формулы как факты и логики высших порядков для описания ограничений целостности. Логики высоких порядков [70] могут использоваться и для описания мета-данных.

2.3 Ограничения целостности

Поскольку при представлении базы данных в виде набора кортежей теряется семантика, она должна быть описана отдельно. Описания семантики баз данных называют ограничениями целостности. Они определяют, какие базы данных имеют смысл в данном приложении, а какие нет. Логика и алгебра используются для описания ограничений в базах данных, для определения внутренних и выводимых из них связей. Наиболее изученный вид ограничений целостности - зависимости.

В реляционных моделях особый интерес вызывают ограничения целостности, называемые зависимостями данных. За последние два десятилетия было изучено множество классов зависимостей (обзор большинства известных классов зависимостей в реляционной модели можно найти в [84]).

Прежде чем описывать конкретные классы, рассмотрим общую структуру. Зависимости можно разделить на

- *статические ограничения целостности* (представляющие семантику всех возможных состояний базы данных);
- *динамические ограничения целостности* (описывающие поведение базы данных во времени, то есть корректность последовательностей ее состояний).

Статические ограничения целостности могут быть разделены на следующие классы:

1. структурные ограничения, то есть ограничения, используемые при проектировании базы данных и накладываемые на ее структуру, например, включения, исключения, функциональные зависимости;
2. семантические ограничения, то есть ограничения целостности, которые также используются при проектировании базы данных и накладываются на ее семантику, например, функциональные, многозначные зависимости;

3. ограничения реализации, то есть ограничения, которые используются для представления базы данных, например, включения, объединения, зависимости, порожденные кортежами;
4. ограничения проектирования, то есть ограничения, которые используются для построения удобного интерфейса [84], например, общие функциональные и обобщенные функциональные зависимости.

Можно показать, что эти типы ограничений могут быть использованы и для динамических ограничений целостности. Динамические ограничения используются для поддержания структуры базы данных. В данный момент не существует общего подхода к использованию динамических ограничений целостности. По своему назначению они могут быть разделены на:

1. ограничения перехода, т.е. ограничения, которые описывают операции над базой данных, изменение ее состояний, например, пред- и постусловия, накладываемые на операции изменения данных;
2. временные формулы, т.е. ограничения на последовательности состояний [76].

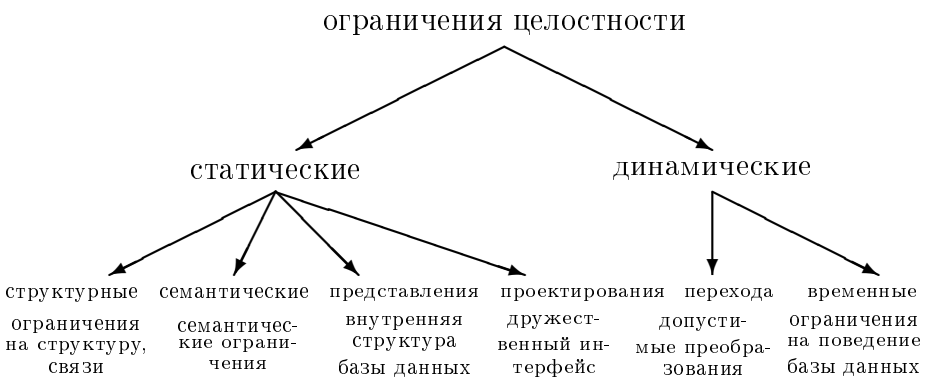

Рис. 1: Классификация ограничений целостности.

Данная классификация включает как скрытые, так и явные ограничения. Разница между этими типами ограничений зависит от используемой модели. В реляционной модели все ограничения целостности представлены вместе. Это приводит к сложностям в их классификации. В [17] предложена классификация зависимостей по их роли. Преимущество рассмотрения всех возможных зависимостей состоит в том, что требуется только одна общая процедура вывода. Однако для некоторых типов ограничений целостности,

таких как функциональные зависимости и зависимости включения, аксиоматизация возможна только в рамках логики предикатов первого порядка. Но это достаточно сложно. Кроме того, смешение ведет к несоответствию типов ограничений, особенно в проектировании реляционных баз данных, когда зависимостями пытаются описать как структурные, так и семантические ограничения. При ER подходе структурные ограничения моделируются внутренними ограничениями, такими как зависимости включения, которые заданы в структуре схемы. Большинство расширений ER-модели поддерживает различные типы функциональных зависимостей, такие как один-к-одному, один-ко-многим и кардинальные ограничения. Преимущество этих ограничений состоит в том, что они легко описываются при проектировании базы данных.

Мы можем ограничить использование ограничений целостности следующими отображениями:

1. $f_1 : \{DesignDep\} \hookrightarrow \{StructInt\} \bigcup \{SemanticInt\}$,
2. $f_2 : \{StructInt\} \hookrightarrow \{RepresentInt\}$,
3. $f_3 : \{SemanticInt\} \hookrightarrow \{RepresentInt\}$,
4. $f_4 : \{StructInt\} \bigcup \{SemanticInt\} \bigcup \{RepresentInt\} \hookrightarrow \{DynamicInt\}$.

Соответственно, процесс проектирования может быть рассмотрен как ряд последовательных преобразований схемы [51]. На первом шаге определяется структура схемы и ограничения проектирования (*проект схемы*). При помощи функции f_1 строится новая схема, имеющая ту же структуру, а также структурные и семантические ограничения (*концептуальная схема*). С помощью f_2 и f_3 эта схема преобразуется в более эффективную (*схема базы данных*). Далее с помощью f_4 определяются соответствующие операции вставки, удаления и изменения данных. Эти операции обеспечивают согласованность состояний базы данных с точки зрения динамических ограничений целостности (*схема управления*).

В описание ограничений целостности должны быть включены условия их применения. Они могут быть справедливы только в определенной подсхеме (модуле). Они могут быть частичными или иметь исключения [14]. Они должны поддерживаться на протяжении всего времени функционирования базы данных и вызывать некоторые действия в случае нарушения. Кроме того они могут быть рассмотрены как действующие правила. Поэтому полное описание ограничений целостности должно включать также и внешнее окружение:

Ограничение целостности φ
[`Локализация:` <название модуля>]

[Значимость: <условие справедливости>]
[Исключения: <условия исключений>]
[Активизация: (<условия активизации>, <события активизации>)]
[Последующие действия: <условный оператор>].

Существуют также другие подходы к классификации ограничений целостности.

- Ограничения целостности можно классифицировать по области их применения: один кортеж, два кортежа, несколько отношений и т.д.;
- Ограничения целостности можно различать по их логической форме.
- Ограничения целостности можно разделить по их инвариантности, например, относительно переименования.

Далее мы можем различать специальные классы ограничений целостности по области определения. Например, семантические ограничения могут зависеть от области применения. Они могут представлять законы природы или принадлежать к общезначимым знаниям.

В данной работе мы используем теоретико-модельную интерпретацию формул. Для дедуктивных баз данных может оказаться полезным рассматривать ограничения целостности с точки зрения теории доказательств. В этом случае мы различаем выполняющиеся и выводимые ограничения целостности. Различные теоретико-доказательственные подходы могут быть представлены формулами модальной логики.

Библиографические замечания

Алгебраические основы данной теории изложены в известных книгах [23, 32, 63, 72, 92] и в обзоре [56]. Развитие реляционной модели обсуждается в [13, 35, 39, 53, 77]. Распределенные схемы [30] могут быть определены как схемы баз данных с множеством ограничений, наложенных на размещение данных в сети. Общий подход к ограничениям целостности введен в [84, 85] и расширен в [14].

3 Алгебраические и структурные ограничения

3.1 Идентификация и ключи

Понятие ключа является центральной идеей в концепции идентификации, различимости объектов.

Это понятие может быть перенесено на объектно-ориентированные базы данных (ООБД). Объект представляется в базе данных своим идентификатором, значениями, методами и ссылками на другие объекты. Данные представляются в соответствии со структурой типов данной модели. Как и в [16], мы различаем структурное представление и поведение объекта. 4Объектный уровень используется для представления структуры объекта. Динамические свойства объектов описываются на уровне методов.

Идентификация является одной из основных концепций ООБД. Без нее невозможно строго определить такие понятия как наследование, совместный доступ и взаимодействие объектов. Для определения тождества объектов используются идентификаторы, которые могут рассматриваться как обобщение классического понятия ключа. Однако в отличие от ключа, идентификаторы обычно скрыты от пользователя и используются только для идентификации объектов. Идентификация объекта возможна также по его взаимодействию с другими объектами [19]. Идентификаторы можно считать неизменными. Однако, с точки зрения системно-ориентированного подхода, изменение индентификаторов не должно влиять на поведение базы данных.

Для пользователя абстрактный идентификатор объекта не имеет никакого смысла, поэтому требуется другой подход к этой проблеме. Единственность идентификации объекта в классе приводит к понятию *(слабой) идентифицируемости по значению* [80], и слабая индентифицируемость по значению может быть использована для выбора объектов, которые не существуют сами по себе, но зависят от других объектов. Корректное определение общих операций модификации требует введения более сильноего понятия *представимости значением.*

Идентификация неопределяемых объектов может основываться на их типе значения. В этом случае следует отказаться от индентификаторов объектов и заменить их значениями 'ссылочных' объектов. Такой тип представления значением задается однозначно с точностью до изоморфизма. Объект называется слабо представимым по значению,

если он может быть идентифицирован значениями ссылочных объектов и также ссылками от других объектов. Такой тип слабо представимых по значению объектов определяется единственным образом при условии, что он существует. Из представления по значению вытекает слабое представление. Более того, каждый класс слабо представимый по значению является также слабо идентифицируемым по значению и наоборот. (Слабое) представление по значению вычислимо. Объект однозначно идентифицируем [19], если его орбита в $G(\mathcal{D})$ тривиальна. Класс обладает свойством однозначной идентифицируемости, если все реализации объектов однозначно идентифицируемы. Класс C обладает свойством однозначной идентификации тогда и только тогда, когда C является слабо идентифицируемым по значению [19, 80].

3.2 Алгебраические ограничения

Формальная система описания различных классов ограничений целостности может быть построена на основе выражений реляционной алгебры. Реляционные выражения строятся из имен реляционных схем и операторов, и они могут рассматриваться как запросы к базе данных. Например, для отношений $R_1^t, ..., R_m^t$ на $R_1, ..., R_m$ соединение $\bowtie_{i=1}^m R_i^t$ интерпретируется как отношение $(((R_1^t \bowtie R_2^t) \bowtie R_3^t) ... \bowtie R_n^t)$. Пусть $\{X_1, ..., X_m\}$ – покрытие множества атрибутов $\{B_1, ..., B_n\}$. Будем говорить, что отношение R^t на $\{B_1, ..., B_n\}$ имеет свойство *соединения без потерь* (может быть разложено без потери информации на $R^t[X_1], ..., R^t[X_m]$) если $\bowtie_{i=1}^m R^t[X_i] = R^t$.

Выражения реляционной алгебры сравнимы друг с другом по отношениям включения и эквивалентности. Задача определения включения или эквивалентности реляционных выражений алгоритмически неразрешима. Эти отношения могут быть использованы для детализации ограничений целостности (как, например, свойство соединения без потерь). Реляционная алгебра может рассматриваться как цилиндрическая [101].

Основываясь на таком алгебраическом языке, можно определить алгебраические ограничения целостности. Алгебраические зависимости могут рассматриваться в качестве общего подхода к теории зависимостей в реляционных базах данных. Эти зависимости вводятся для расширенной схемы, содержащей бесконечное количество копий имен реляционных схем. Этот класс ограничений эквивалентен рассматриваемому далее классу BV-зависимостей.

Пусть R – реляционная схема, e_1, e_2 – алгебраические выражения на R, определенные на подмножествах атрибутов X и Y из R, тогда

справедливы следующие алгебраические зависимости:

$$(e_1[W])[V] = e_1[V] \qquad V \subseteq W \subseteq X, \qquad\qquad e_1[X] = e_1,$$

$$e_1 \bowtie e_1[W] = e_1 \qquad W \subseteq X, \qquad\qquad (e_1 \bowtie e_2)[X] \subseteq e_1,$$

$$(e_1 \bowtie e_2)[V \cup W] \subseteq (e_1 \bowtie e_2[W])[V \cup W] \qquad V \subseteq X,\, W \subseteq Y,$$

$$(e_1 \bowtie e_2)[V \cup W] = (e_1 \bowtie e_2[W])[V \cup W] \quad V \subseteq X, W \subseteq Y, X \cap Y \subseteq W,$$

$$(e_1 \bowtie e_2)[W] = (e_1[X \cap (Y \cup W)] \bowtie e_2[Y \cap (X \cup W)])[W],$$

$$e_1[W \cup V] \subseteq (e_1[V] \bowtie e_1[W]) \qquad V, W \subseteq X.$$

Кроме того, операция соединения ассоциативна и коммутативна.

Один из классов алгебраических зависимостей составляют зависимости включения. Обобщением зависимостей включения являются *недетерминированные зависимости включения*. Это обобщение важно в моделях, допускающих кластеризацию. *Обобщенная зависимость включения* задается выражением вида

$R_1[X_1] \cap ... \cap R_n[X_n] \subseteq S_1[Y_1] \cup ... \cup S_m[Y_m]$

для совместимых последовательностей атрибутов X_i, Y_j и выполняется в базе данных $(..., R_i^t, ..., S_j^t, ...)$, если $R_1^t[X_1] \cap ... \cap R_n^t[X_n] \subseteq S_1^t[Y_1] \cup ... \cup S_m^t[Y_m]$.

При $n = m = 1$ обобщенная зависимость включения называется зависимостью включения.

Особым классом алгебраических ограничений целостности являются основанные на ключе зависимости включения, то есть зависимости вида $R[X] \subseteq S[Y]$ где Y является ключом S. Такие зависимости называются *ссылочными ограничениями целостности*.

Другим важным классом алгебраических ограничений целостности является класс зависимостей исключения [84]. Пусть R, S – схемы, $R.A_1, ..., R.A_n,\ S.B_1, ..., S.B_n$ – имена атрибутов этих схем. *Зависимость исключения* – это выражение вида

$R[R.A_1, ..., R.A_n] \parallel S[S.B_1, ..., S.B_n]$,

она имеет место в базе данных $(..., R^t, ..., S^t, ...)$, если для всех $r \in R^t,\ s \in S^t \qquad r\,|_{A_1,...,A_n} \neq s\,|_{B_1,...,B_n}$.

Класс зависимостей включения и исключения аксиоматизируем (см., например, [84]).

Аксиомы:

$$XY \subseteq X \qquad\qquad X \subseteq X \cup Y$$

Правила вывода:

$$\frac{X_1...X_n \subseteq Y \cup W_1 \cup ... \cup W_m \qquad Y Z_1...Z_n \subseteq V_1 \cup ... \cup V_l}{X_1...X_n Z_1...Z_n \subseteq V_1 \cup ... \cup V_l \cup W_1 \cup ... \cup W_m} \quad l \neq 0$$

$$\frac{X_1...X_n \subseteq Y \cup W_1 \cup ... \cup W_m \qquad YZ_1...Z_n \parallel V}{X_1...X_nZ_1...Z_nV \subseteq W_1 \cup ... \cup W_m} \qquad m \neq 0$$

$$\frac{X_1...X_n \subseteq Y, \qquad YZ_1...Z_n \parallel V}{X_1...X_nZ_1...Z_n \parallel V}$$

$$\frac{X \parallel X}{X \parallel Y} \qquad \frac{X \parallel X}{X \subseteq Y} \qquad \frac{X_1...X_n \parallel X_0}{X_0X_1...X_{n-1} \parallel X_n}$$

$$\frac{X_1...X_n \subseteq \zeta}{X_{\pi(1)}...X_{\pi(n)} \subseteq \zeta} \quad \pi$$

$$\frac{X_1...X_n \parallel Z}{X_{\pi(1)}...X_{\pi(n)} \parallel Z} \quad \pi$$

Проблема выводимости для (обобщенных) зависимостей включения и зависимостей исключения является NP-полной.

Алгебраические ограничения вида $(R[X_1] \bowtie ... \bowtie R[X_n])[X] \subseteq R[X]$ для $X \subseteq X_1 \cup ... \cup X_n \subseteq attr(R)$ называются *проецирующими зависимостями соединения* и обозначаются $(X_1, ..., X_n)[X]$. Если $(R[X_1] \bowtie ... \bowtie R[X_n])[X] \supseteq R[X]$, то проецирующая зависимость соединения эквивалентна $(R[X_1] \bowtie ... \bowtie R[X_n])[X] = R[X]$. А если $X = X_1 \cup ... \cup X_n$ то проецирующая зависимость соединения называется *X-зависимостью соединения*. Если $attr(R) \neq X_1 \cup ... \cup X_n$, то проецирующая зависимость соединения называется вложенной, иначе – тотальной. *Зависимости соединения* являются тотальными X-зависимостями соединения. Бинарные ($n = 2$) зависимости соединения эквивалентны многозначным зависимостям. Бинарные X-зависимости соединения также называют вложенными многозначными зависимостями. Множество бинарных зависимостей соединения аксиоматизируемо, однако множество всех зависимостей соединения не аксиоматизируемо. Зависимости соединения являются шаблонными зависимостями. Шаблонные зависимости аксиоматизируемы.

Для $X, Y, Z \subseteq attr(R)$, $Y \cap Z = \emptyset$ проецирующая зависимость соединения $(R[X \cup Y] \bowtie R[X \cup Z])[Y \cup Z] \subseteq R[Y \cup Z]$ называется *транзитивной зависимостью* и обозначается $X(Y, Z)$. Транзитивные зависимости аксиоматизируются следующей системой:
Аксиомы:

$$XY(Y, Z)$$

Правила вывода:

$$\frac{X(Y,Z), Y(Z,T)}{X(Z,T)} \qquad \frac{X(Y,Z), X(T,Z), Z(T, Y \cup Z)}{X(Y \cup T, Z)}$$

$$\frac{X(Y,Z)}{X(Z,Y)} \qquad \frac{X(Y \cup T,Z)}{X(Y,Z)} \qquad \frac{X(Y,Z)}{X \cup T(Y,Z)}$$

Расширенная транзитивная зависимость определяется следующим соотношением

$$(\overset{p}{\underset{i=1}{\bowtie}} \overset{q}{\underset{j=1}{\bowtie}} R[X_i \cup Y_j])[\bigcup_{j=1}^{q} Y_j] \subseteq R[\bigcup_{j=1}^{q} Y_j]$$

для попарно непересекающихся множеств X_i и Y_j.

Библиографические замечания

Более подробно алгебраический подход к зависимостям описан в [84, 101].

4 Логические ограничения

4.1 Зависимости в реляционных моделях

В реляционной модели зависимости составляют достаточно широкий класс ограничений целостности. Введем формальные определения.

Пусть $R = ((B_1, ..., B_n), \emptyset)$ – реляционная схема над схемой данных DD, а X, Y – подмножества $attr(R) = \{B_1, ..., B_n\}$. Отношение $R^t \in SAT(R)$ удовлетворяет *функциональной зависимости* $X \to Y$, если для любых кортежей t, t' в R^t из $t[X] = t'[X]$ следует $t[Y] = t'[Y]$. Будем говорить, что отношение удовлетворяет множеству функциональных зависимостей, если оно удовлетворяет всем зависимостям из этого множества. Подмножество атрибутов X будем называть *ключом* R^t, если зависимость $X \to attr(R)$ имеет место в R^t. Минимальные такие подмножества будем называть *минимальными ключами*.

Пусть $X, Y_1, ..., Y_m$ – покрытие $attr(R)$, причем множества $Y_1, ..., Y_m$ попарно не пересекаются. *Иерархическая зависимость* $X \to\!\!\to Y_1 \mid Y_2 \mid ... \mid Y_m$ имеет место в R^t, если для любых кортежей $t_1, t_2, ..., t_m$ из R^t, совпадающих на X, найдется кортеж t из R^t, такой что $t[X \cup Y_i] = t_i[X \cup Y_i]$ для всех $i\,(1 \le i \le m)$.

При $m = 2$ иерархическая зависимость называется *многозначной зависимостью*.

Функциональные зависимости являются частным случаем зависимостей, порожденных равенствами:

$$\forall(x_{1,1}, ..., x_{m,n})$$
$$(P_R(x_{1,1}, ..., x_{1,n}) \wedge ... \wedge P_R(x_{m,1}, ..., x_{m,n}) \wedge F(x_{1,1}, ..., x_{m,n}) \to \to G(x_{1,1}, ..., x_{m,n})),$$

где $F(x_{1,1}, ..., x_{m,n}), G(x_{1,1}, ..., x_{m,n})$ – конъюнкции равенств вида

$x_{i,j} = x_{i',j'}$, а P – ассоциированный с R предикатный символ. Зависимости, порожденные равенствами, имеют ряд полезных свойств. Например, если отношение удовлетворяет такой зависимости, то ей удовлетворяет и любое его подмножество. Если допустить использование в качестве F и G любых формул с равенством и положить $m = 2$, получим *обобщенную функциональную зависимость*.

Многозначные и иерархические зависимости являются примерами зависимостей, порожденных кортежами:

$$\forall(x_{1,1},...,x_{m,n})\exists(y_{i_1},...,y_{i_k})(P_R(x_{1,1},...,x_{1,n}) \wedge ... \wedge P_R(x_{m,1},...,x_{m,n}) \wedge$$

$$\wedge F(x_{1,1},...,x_{m,n}) \rightarrow P_R(y_1,...,y_n) \wedge H(y_1,...,y_n,x_{1,1},...,x_{m,n})),$$

где $F(x_{1,1},...,x_{m,n})$ – конъюнкция равенств вида $x_{i,j} = x_{i',j'}$,
$H(y_1,...,y_n,x_{1,1},...,x_{m,n})$ – конъюнкция равенств вида $y_j = x_{i',j'}$, $y_{i_1},...,y_{i_k}$ – переменные, не входящие в H.
Зависимость, порожденная кортежами, называется *полной*, если все y_i входят в равенства в H.

Типизированные зависимости, порожденные равенствами или кортежами, содержат только равенства, в которых $j = j'$.

Зависимости соединения могут быть представлены как полные типизированные зависимости, порожденные кортежами, у которых в формулу F входят только равенства с переменными, входящими в H. Зависимости соединения используются для декомпозиции отношений. Иерархические зависимости являются зависимостями соединения.

Произвольному множеству реляционных схем $\mathcal{R} = \{R_1,\ldots,R_n\}$ сопоставим множество предикатных символов $\mathcal{P}_\mathcal{R} = \{P_{R_1},...,P_{R_n}\}$, которые вместе с константами $C_{DD} = \{c_{d,D} \mid d \in D, D \in \underline{D}\}$ схемы данных DD составляют сигнатуру логики предикатов первого порядка $L = L(\mathcal{P}_\mathcal{R}, C_{DD})$. Этот логический язык может быть использован для описания *ограничений базы данных* на $\mathcal{R}$ и C_{DD}. Ограничения базы данных, записанные на этом языке, можно свести к формулам, не содержащим констант схемы данных. Пусть DD и DD' – схемы данных. Формулу α из $L(\mathcal{P}_\mathcal{R}, C_{DD}) \cup L(\mathcal{P}_\mathcal{R}, C_{DD'})$ будем называть независимой от данных, если для любых баз данных $DB^t(R_1^t,...,R_n^t)$ над DD и $DB'^t = (S_1^t,...,S_n^t)$ над DD', таких что $R_i^t = S_i^t$ для любого i, DB^t удовлетворяет α тогда и только тогда, когда DB'^t удовлетворяет α. Базу данных $DB^t = (R_1^t,...,R_n^t)$, отношения которой имеют не более одного элемента, будем называть тривиальной. Независимую от данных формулу, которая справедлива в любой тривиальной базе данных, будем называть *зависимостью*. Множество зависимостей рекурсивно перечислимо только для $n = 1$. Задача определения, является ли

данная формула зависимостью, алгоритмически неразрешима, поэтому представляет интерес изучение отдельных классов зависимостей.

Формулу будем называть *одно-реляционной* (uni-relational), если она содержит только один предикат, *типизированной*, если переменные не используются в двух различных аргументах одного предиката, и в равенствах используются только те переменные, которые входят в один аргумент предиката.

Обобщенной вложенной импликативной зависимостью будем называть формулу вида

$$\alpha = \forall y_1 ... \forall y_k \exists z_1 ... \exists z_l ((\beta_1 \wedge ... \wedge \beta_p) \longrightarrow (\gamma_1 \wedge ... \wedge \gamma_q)),$$

где все β_i, γ_j являются атомарными формулами, по меньшей мере одна из β_i является предикатом $P_R(\bar{x})$, множество переменных, входящих в β_i совпадает с множеством переменных, входящих в предикат β_i и является в точности $\{y_1, ..., y_k\}$, множество переменных, входящих в γ_j содержит $\{z_1, ..., z_l\}$ и является подмножеством $\{z_1, ..., z_l, y_1, ..., y_k\}$. Обобщенные вложенные импликативные зависимости могут быть классифицированы следующим образом:

	Ограничения на					
Зависимость	l	p	q	β_i	γ_j	α
включения		1	1	предикаты	предикаты	
BV-зависимость	0					одно-реляционная
порожденная кортежами	0			предикаты	предикаты	одно-реляционная
вложенная шаблонная			1	предикаты	предикаты	одно-реляционная многотиповая
шаблонная	0		1	предикаты	предикаты	одно-реляционная типизированная
порожденная равенствами	0			предикаты	равенства	одно-реляционная типизированная
функциональная	0	2		предикаты	равенства	одно-реляционная типизированная
тотальная BV-зависимость	0					одно-реляционная
вложенная зависимость, порожденная кортежами				предикаты	предикаты	одно-реляционная

Изложенная ниже процедура прогонки (chase) может быть использована для проверки, выводимо ли данное множество тотальных обобщенных импликативных зависимостей (с $q = 1$ и $l = 0$) из другого. Данная процедура доказательства может быть записана в виде списка.

Процедура состоит в последовательной генерации новых строк. Для шаблонных зависимостей такая процедура называется прогонкой списка.

Пусть даны множество тотальных обобщенных импликативных зависимостей Σ и тотальная обобщенная импликативная зависимость $\alpha = \forall((P_{i1}(\bar{x}_{i1}) \wedge ... \wedge P_{im}(\bar{x}_{im})) \rightarrow \beta)$.
Индуктивно определим следующее замыкание:
$Cl_0(\Sigma, \alpha) = \{P_{ik}(\bar{x}_{ik}) \mid 1 \leq k \leq m\}$
$Cl_{k+1}(\Sigma, \alpha)$ получается из $CL_k(\Sigma, \alpha)$ применением следующей процедуры унификации:

для зависимости $\forall(P_{l1}(\bar{u}_1) \wedge ... \wedge P_{lp}(\bar{u}_p)) \rightarrow y_i = y_j$ из Σ и
подстановки ς, такой что $P_{ls}(\varsigma(\bar{u}_s)) \in Cl_k(\Sigma, \alpha)$ для $1 \leq s \leq p$
идентифицируем $\varsigma(y_i)$ и $\varsigma(y_j)$ в $Cl_k(\Sigma, \alpha)$;

$Cl_{k+1}(\Sigma, \alpha)\{P_{l0}(\varsigma(\bar{u}_0)) \mid \forall(P_{l1}(\bar{u}_1) \wedge ... \wedge P_{lp}(\bar{u}_p)) \rightarrow P_{l0}(\bar{u}_0) \in \Sigma,$
$\varsigma-$ подстановка, такая что $P_{ls}(\varsigma(\bar{u}_s)) \in Cl_k(\Sigma, \alpha)$ для $1 \leq s \leq p\} \cup$
$\cup \tilde{C}l_{k+1}(\Sigma, \alpha);$

$Cl(\Sigma, \alpha) = \bigcup\limits_{k=0}^{\infty} Cl_k(\Sigma, \alpha)$.

Поскольку количество атомарных формул, составленных из α, конечно, $Cl(\Sigma, \alpha)$ может быть построено за конечное время, то есть существует k, такое что $Cl_k(\Sigma, \alpha) = Cl(\Sigma, \alpha)$. Можно показать, что $\Sigma \models \alpha$ тогда и только тогда, когда $\beta \in Cl(\Sigma, \alpha)$, или y_i и y_j унифицированы в $Cl(\Sigma, \alpha)$, $\beta = (y_i = y_j)$.

Зависимостью замыкания $X@Y$ будем называть одно-реляционную формулу на R, имеющую следующий вид

$$\forall x_1 ... \forall x_n \forall y_1 ... \forall y_n \exists z_1 ... \exists z_n((P_R(x_1, ..., x_n) \wedge P_R(u_1, ..., u_n)) \longrightarrow P_R(v_1, ..., v_n))$$

где для $1 \leq i \leq n$

$$u_i = \begin{cases} x_j, & \text{если для некоторого } k \ A_j = C_k \text{ и } B_k = A_i \\ y_i & \text{иначе} \end{cases}$$

$$v_i = \begin{cases} x_i, \text{если } A_i = B_k \text{ для некоторого } k \\ y_i, \text{если для некоторого } k \ A_i = C_k \text{ и ни для какого } l \ B_l = A_i \\ z_i \ \text{иначе} \end{cases}$$

для последовательностей $X = B_1, ..., B_m, Y = C_1, ..., C_m$ атрибутов из $attr(R) = \{A_1, ..., A_n\}$, где все атрибуты в последовательностях различны.

При выполнении этого ограничения отношение R^t совпадает со своим транзитивным замыканием на X и Y. Зависимости замыкания коммутативны, и на этом основана аксиоматизация

этого класса. Однако, в совокупности функциональные зависимости и зависимости замыкания не аксиоматизируемы. В определении обобщенной зависимости замыкания снимается требование различия атрибутов в последовательностях.

Для многозначных зависимостей могут быть даны следующие четыре эквивалентных определения:
1. Многозначная зависимость $X \rightarrow\rightarrow Y \mid Z$ имеет место в R^t.
2. Имеет место алгебраическое тождество $R^t = R^t[X \cup Y] \bowtie R^t[X \cup Z]$.
3. Для любого кортежа t из R^t выполнено $(\sigma_{t[X]}(R^t))[Y] = (\sigma_{t[X \cup Z]}(R^t))[Y]$.
4. Отношение R^t может быть представлено как вложенное отношение $\nu(\nu(R^t, Y, A), Z, B)$, где $\nu(S^t, V, C)$ отображает отношение S^t над U на вложенное отношение над $U \setminus (V), C\{V\}$.

Эквивалентность первых двух определений хорошо известна. Третье условие означает что Y-значения X-ограниченных кортежей не зависят от их Z-значений. Это условие описывает суть многозначных зависимостей. Последнее определение связывает многозначные зависимости с горизонтальной декомпозицией отношений и с представимостью их вложенными отношениями.

Эти ограничения могут быть обобщены на схемы баз данных.
Зависимость включения является важным примером ограничения, действующего на несколько отношений. Пусть R, R' – реляционные схемы, X и X' – последовательности атрибутов из $attr(R)$ и $attr(R')$ соответственно, имеющие одинаковую длину. *Зависимость включения* $R[X] \subseteq R'[X']$ имеет место для отношений R^t, R'^t на R, R' соответственно, если $R^t[X] \subseteq R'^t[X']$. Если длины X и Y равны 1, то такие зависимости называются унарными.

Большинство работ по теории зависимостей посвящено различным аспектам проблемы конечной выводимости, то есть проблемы определения, выводима ли логически зависимость α из данного множества зависимостей Σ (обозначается $\Sigma \models_{fin} \alpha$, то есть все отношения r, в которых имеет место Σ, удовлетворяют α). Обобщенное свойство выводимости (обозначается $\models$) определено на всех (конечных и бесконечных) множествах кортежей над данной схемой. Реляционные базы данных содержат только конечные отношения. Свойства конечной и обобщенной выводимости могут совпадать или не совпадать на множествах зависимостей. Например, они совпадают на зависимостях включения, полных зависимостях, порожденных кортежами или равенствами. Однако, эти понятия не совпадают для функциональных зависимостей и зависимостей включения. Проблема выводимости может быть как неразрешимой, так и (P- или NP-) разрешимой. Обе

проблемы выводимости неразрешимы для типизированных полных зависимостей, порожденных кортежами. Для зависимостей включения эти проблемы эквивалентны и P-пространственно (PSpace) полные. Однако, для зависимостей включения и функциональных зависимостей эти проблемы неразрешимы, а для функциональных зависимостей и унарных зависимостей включения они разрешимы за полиномиальное время. Для функциональных зависимостей и зависимостей соединения проблемы выводимости разрешимы за экспоненциальное время. Задача проверки, выводима ли из зависимости соединения и множества функциональных зависимостей другая зависимость соединения, является NP-полной, а задача проверки, выводима ли зависимость соединения из множества многозначных зависимостей, NP-сложна. Если проблемы выводимости разрешимы, можно говорить об аксиоматизации соответствующего класса зависимостей. Для множества зависимостей соединения не существует гильбертовой аксиоматизации [73], но существует аксиоматизация по Генцену. Класс функциональных и многозначных зависимостей имеет следующую полную гильбертову аксиоматизацию.

Аксиомы

$$X \cup Y \ \to\ Y \ ; \qquad X \cup Y \ \to\!\to\ Y \mid Z$$

Правила вывода

$$\frac{X \ \to\ Y}{X \cup V \cup W \ \to\ Y \cup V} \qquad \frac{X \ \to\ Y \ , \ Y \ \to\ Z}{X \ \to\ Z}$$

$$\frac{X \ \to\!\to\ Y \mid Z}{X \ \to\!\to\ Z \mid Y} \qquad \frac{X \cup Y \cup Z \ \to\!\to\ V \mid W \cup U \ , \ X \ \to\!\to\ Y \cup V \cup W \mid Z \cup U}{X \cup Y \ \to\!\to\ V \mid Z \cup W \cup U}$$

$$\frac{X \ \to\ Y}{X \ \to\!\to\ Y \mid Z} \ Z = attr(R) - (X \cup Y) \qquad \frac{X \ \to\!\to\ Y \mid V \ , \ Z \ \to\ W}{X \ \to\ W} \ W \subseteq Y, \ Y \cap Z = \emptyset.$$

Проблема выводимости может быть решена путем описания зависимостей на другом языке, для которого проблема выводимости разрешима. Одним из таких примеров может послужить класс всех функциональных и многозначных зависимостей. Этот класс может быть представлен формулами пропозициональной логики, что позволяет получить простое решение проблемы выводимости.

Сопоставим каждому атрибуту A из $attr(R)$ высказывание p_A, а подмножеству $X = \{C_1, ..., C_k\}$ – формулу $\tau(X) = p_{C_1} \wedge ... \wedge p_{C_k}$. Далее, $\tau(X \to Y) = \tau(X) \to \tau(Y)$ и
$\tau(X \ \to\!\to\ Y_1 \mid Y_2 \mid ... \mid Y_m) \quad \tau(X) \ \to\!\to\ (\tau(Y_1) \vee \tau(Y_2) \vee ... \vee \tau(Y_m))$.
Для множества $\{\alpha_1, ..., \alpha_k, \beta\}$ функциональных и иерархических зависимостей верно

$\{\alpha_1, ..., \alpha_k\} \models \beta$ тогда и только тогда, когда
$\{\tau(\alpha_1), ..., \tau(\alpha_k)\} \models \tau(\beta)$ тогда и только тогда, когда
$(\tau(\alpha_1) \wedge ... \wedge \tau(\alpha_k)) \rightarrow \tau(\beta) = 1$.

Другим примером класса, имеющего аксиоматизацию и представимого пропозициональной логикой, является класс обобщенных функциональных зависимостей.

Теория зависимостей в реляционной модели очень развита. В [84] приведено 95 классов статических зависимостей. Большинство классов используется для вертикальной декомпозиции. Однако, особенно для распределенных баз данных, не меньшее значение имеет горизонтальная декомпозиция. Горизонтальная декомпозиция хорошо подходит для описания исключений.

Для горизонтальной декомпозиции могут быть использованы условные функциональные зависимости. *Условная функциональная зависимость* $X \rightarrow Y \supset\!\!\rightarrow X \rightarrow Z$ имеет место в R^t, если для любого $t \in R^t$, если $X \rightarrow Y$ выполнено в $\sigma_{t[X]}(R^t)$, то там же выполнено и $X \rightarrow Z$. Если отношение удовлетворяет условной функциональной зависимости, то оно может быть разбито на два подотношения, таких что в одном будут выполнены обе функциональные зависимости, а второе будут составлять исключения.

Ограничения объединения выражают тот факт, что отношение может быть разбито на два подотношения, таких что исходное отношение может быть представлено в виде суммы проекции первого подотношения и второго.

Афункциональное ограничение $X \not\!\!\rightarrow Y$ имеет место в R^t, если для любого кортежа t из R^t, в R^t найдется кортеж t', совпадающий с t на X и отличающийся на Y. Если в R^t имеет место афункциональное ограничение, то R^t может быть разбито на R_1^t и R_2^t, такие что R^t есть их объединение, в R_1^t имеет место функциональное ограничение $X \rightarrow Y$, а в R_2^t – афункциональное. Обобщением афункциональных ограничений являются *(p, q)-ограничения.* R^t удовлетворяет (p, q)-ограничению, если для любого кортежа t из R^t $p \leq |\{t' \in R^t \mid t[X] = t'[X]\}| \leq q$. Если (1,3)-ограничение $X \rightarrow_{(p,q)} Y$ имеет место в R^t, то R^t может быть разбито на R_1^t, R_2^t, R_3^t, такие что функциональная зависимость выполняется во всех R_i^t, $1 \leq i \leq 3$.

Исключающее функциональное ограничение$X \not\rightarrow Y$ показывает, что функциональная зависимость $X \rightarrow Y$ неверна. Эти ограничения используются, например, на этапе проектирования базы данных.

Функциональные зависимости могут быть обобщены до *межтабличных функциональных зависимостей.* Они определяют, когда одно из отношений базы данных удовлетворяет обычной

функциональной зависимости. Пусть схема базы данных состоит из схем $R_1, ..., R_n$ над $attr(R_1), ..., attr(R_n)$ и множества функциональных зависимостей F на $attr(R_1) \cup ... \cup attr(R_n)$. Отношение R^t на $attr(R_1) \cup ... \cup attr(R_n)$ является слабым универсальным отношением для базы данных $(R_1^t, ..., R_n^t)$ над $R_1, ..., R_n$, если $R_i^t \subseteq R^t[attr(R_i)]$ для всех i. База данных $(R_1^t, ..., R_n^t)$ глобально удовлетворяет F, если для него существует слабое универсальное отношение.

Вышеперечисленные свойства могут быть рассмотрены также и для отношений с неопределенными значениями, то есть отношений, кортежи которых имеют для некоторых атрибутов неопределенные значения. В этом случае, например, понятие ключа расширяется до понятия семейства ключей. Пусть R – реляционная схема с множеством атрибутов $attr(R)$, $\mathcal{K}$ – множество подмножеств $attr(R)$. Будем называть $\mathcal{K}$ *семейством ключей* R^t, если для любой пары кортежей t, t' из R^t существует элемент $K \in \mathcal{K}$, такой что оба кортежа полностью определены на K и $t[K] \neq t'[K]$. Существующие алгоритмы и подходы могут быть перенесены на семейства ключей.

Ограничения совместного существования $X \Rightarrow Y_1, Y_2, ..., Y_n$ показывают, что если кортеж полностью определен на X, то он полностью определен на Y_i для некоторого i. Существует аксиоматизация этого класса ограничений. Они могут быть представлены монотонными булевыми функциями.

Зависимости могут быть расширены на отношения, содержащие неопределенные значения. Два кортежа t, t' будем называть строго эквивалентными по отношению к X (обозначается $t \approx_X t'$), если они определены и совпадают на X. Они слабо эквивалентны на X (обозначается $t \sim_X t'$), если они совпадают для каждого атрибута $A \in X$ при условии, что они определены на A. Эти кортежи эквивалентны на X (обозначается $t \simeq_X t'$), если они либо оба не определены, либо равны на каждом $A \in X$. Теперь можно определить различные способы проверки функциональной зависимости $X \rightarrow Y$ в отношении R^t с неопределенными значениями. Мы отметим некоторые из них.

- Отношение R^t 1-удовлетворяет функциональной зависимости $X \rightarrow Y$, если все пары строго X-эквивалентных кортежей Y-эквивалентны.
- Отношение R^t 2-удовлетворяет функциональной зависимости $X \rightarrow Y$, если все пары строго X-эквивалентных кортежей строго Y-эквивалентны.
- Отношение R^t 3-удовлетворяет функциональной зависимости $X \rightarrow Y$, если все пары слабо X-эквивалентных кортежей слабо Y-

эквивалентны.

- Отношение R^t 4-удовлетворяет функциональной зависимости $X \to Y$, если все пары строго X-эквивалентных кортежей слабо Y-эквивалентны.
- Отношение R^t 5-удовлетворяет функциональной зависимости $X \to Y$, если все пары слабо X-эквивалентных кортежей строго Y-эквивалентны.

Свойство 3 влечет свойство 4. Свойство 1 влечет свойство 4. Свойство 2 влечет свойство 1. Свойство 5 влечет свойство 2. Приведенная выше аксиоматизация функциональных зависимостей применима к свойствам 2 и 3. Аксиома расширения $X \cup Y \to Y$ не верна для свойств 4 и 5. Правило транзитивности не верно для свойства 1, то есть если отношение 1-удовлетворяет $X \to Y$ и $Y \to Z$, то оно не обязательно 1-удовлетворяет $X \to Z$. Ключ K будем называть *достоверным ключом* R^t, если R^t 5-удовлетворяет $K \to attr(R)$. Ключ называется *возможным*, если R^t 2-удовлетворяет $K \to attr(R)$.

Аналогично на случай отношений с неопределенными значениями могут быть обобщены многозначные зависимости, зависимости соединения и другие типы зависимостей.

Следует выделить различные типы неопределенных значений, отражающие причину неопределенности: применимо ли данное свойство к объекту, находится ли оно в процессе изменения, доступно ли оно, хранится ли данное значение, выведено ли оно из неполной или противоречивой информации, не является ли оно секретным. В [84] описаны контекстно-зависимые семантические неопределенные значения.

В литературе предложено множество подходов к моделированию динамики в базах данных. В качестве примеров можно привести следующие:

1. активные базы данных и системы продукций;
2. различные механизмы описания поведения баз данных: условия срабатывания процедур, пред- и постусловия выполнения транзакций, язык описания доступа к базе данных;
3. программирование на основе временных логик, дедуктивные базы данных, различные применения временных логик к описанию динамики баз данных;
4. многозначные и модальные логики;
5. подход, базирующийся на сетях Петри.

Рассмотрим два из этих подходов.

Система продукций задается схемой базы данных, множеством правил, соответствующих этой схеме и механизмом управления, который определяет, как применяются правила и как они изменяют базу данных. Продукционные правила определены на основе (конечного) множества *Ops* операций над базами данных (например, $Insert(R_5, (a_1, ..., a_n))$ – оператор вставки кортежа $(a_1, ..., a_n)$ в отношение R_5). Пусть ';' – оператор следования, то есть $o_1; o_2$ означает, что эти операторы применяются согласно некоторой стратегии разрешения конфликтов. Тогда продукция будет иметь вид

$$\alpha \longrightarrow o_1; o_2; ...; o_n$$

где α – формула, задающая некоторое условие. Управляющий механизм решает, как продукции применяются в цикле распознавание–действие, и как в результате определяется новое состояние базы данных. Продукции могут применяться как параллельно, с использованием механизма разрешения конфликтов, так и последовательно, с использованием некоторой стратегии выбора.

Для описания поведения базы данных используются также временные формулы. Временная логика является расширением логики предикатов добавлением специальных операторов (*next, after,* $\forall_{future}$, $\forall_{past}$, $\exists_{future}$,$\exists_{past}$), связывающих состояния базы данных в последовательность допустимых состояний, например

$$(\delta_{start}, \delta_1, ..., \delta_{current}, \delta_{c+1}, ...)$$

для линейной дискретной модели времени. *Ограничения перехода* описывают допустимые изменения состояний базы данных сужением множества допустимых переходов (δ_i, δ_{i+1}). Во временных логиках они могут быть описаны следующим образом: $\alpha \rightarrow next(\beta)$ где α, β – статические ограничения целостности. Временные ограничения целостности могут быть представлены графом переходов типа диаграммы конечного автомата.

Статическая зависимость $\alpha = (\gamma \rightarrow \theta)$ может быть выражена ограничениями перехода следующим образом: предполагается, что начальное состояние базы данных корректно, затем ограничение перехода $\alpha \rightarrow next(\alpha)$ выражает, что эта зависимость не исчезает, если она присутствует в начальном состоянии.

Алгебраические свойства зависимостей очень важны, особенно потому, что позволяют обобщить зависимости на другие модели баз данных. Для реляционного оператора o и формулы α отношение r,

удовлетворяющее α, будем называть (o, α)-инвариантным, если α верна в $o(r)$.

Если функциональная зависимость $X \rightarrow Y$ имеет место в r, то r является (o, α)-инвариантным для операций проекции, выбора, разности, пересечения с отношением того же типа, соединения с любым отношением, произведения с любым отношением, объединения с отношением r', таким что $r(X) \cap r'(X) = \emptyset$, и для операции суммы в ограниченном смысле.

Отношения не являются $(o, X \twoheadrightarrow Y \mid Z)$-инвариантными относительно проекции на множество атрибутов, содержащее X, выбора, различных операций соединения, пересечения, суммы и разности. В то же время они инвариантны относительно произведения и ограниченного объединения. Аналогичные результаты могут быть получены для зависимостей соединения.

Относительно левой части зависимости включения отношения $(o, R[X] \subseteq S[Y])$-инвариантны для операций выбора, соединения с другим отношением, пересечения, произведения, разности и проекции на множества, содержащие X. Относительно правой части отношения инвариантны к объединению, произведению, сумме.

Относительно операции дополнения ни одно из свойств инвариантности не имеет места.

4.2 Ограничения в семантических моделях

Смысл, вкладываемый в ограничения целостности, меняется при переходе от модели к модели. Например, зависимость включения $R[X] \subseteq S[Y]$ имеет для ER-схемы как минимум два значения. В связи имеет_тип она означает наследование ключа. Совместно со свойством ключа это ограничение определяет ссылочное ограничение целостности. Совместно с ограничениями мощности оно описывает свойство идентификации.

Модели, основанные на более богатых системах типов, имеют также большие множества неявных ограничений целостности. Например, если ER-модель основана на семантике множеств, то схема отношений базируется на схеме компонент, то есть для схемы отношений $R = (..., R', ..., attr(R), ...)$ зависимость включения $R[R'] \subseteq R'$ вытекает из определения.

Используя описанные выше свойства инвариантности, зависимости могут быть обобщены на семантические модели. Например, функциональные зависимости могут быть обобщены на функциональные зависимости путей в ER-модели. Пусть дана схема $ERDec = \{E_1, ..., E_n, R_1, ..., R_m\}$. Последовательность $p = P_1 - ... - P_k$ типов

из *ERDec* будем называть *ERDec-путем* (или просто путем), если для любого i, $1 \leq i < k$ либо P_i является компонентой P_{i+1}, либо P_i имеет компоненту P_{i+1}.

Для путей могут быть рассмотрены идентификаторы атрибутов [85]. Для пути имеют значение только те идентификаторы, которые в нем используются. Мы будем различать листья и корни пути. Пусть даны путь p, множество идентификаторов атрибутов этого пути $attr(p)$, $X, Y \subseteq attr(p)$. Введем *функциональную зависимость пути* $p : X \to Y$. Выполнение этой зависимости определим как выполнение соответствующей функциональной зависимости в отношении p^t, которое индуцируется на пути p отношением $ERDec^t$. Поскольку запись атрибутов пути зависит от его направления, мы можем ввести операцию обращения пути. Множество функциональных зависимостей путей аксиоматизируется следующим исчислением.

Аксиомы

$$X \supseteq Y : \quad p : X \to Y \qquad\qquad X\ p' \star Y : \quad p : p' \to X$$

Правила вывода

$$\frac{p : X \to Y}{p : X \bigcup Z \to Y \bigcup Z} \qquad \frac{p : X \to Y\ , \quad p : Y \to Z}{p : X \to Z}$$

$$\frac{p : X \to Y}{p' \star p : p' \star X \to p' \star Y} \qquad \frac{p : X \to Y}{(p : X \to Y)^{-1}}$$

Можно показать, что эта система является непротиворечивой и полной для выведения функциональных зависимостей путей. Доказательство основано на свойствах инвариантности функциональных зависимостей.

Ограничения мощности изучаются на протяжении достаточно большого периода времени. Это очень сильные ограничения. Правильное использование ограничений этого класса требует глубоких познаний в теории. Существуют различные интерпретации этих ограничений:

- По-разному определяются сами ограничения. Или они задают ограничения мощности через 'видимость' объекта (сколько объектов можно 'увидеть' с помощью схемы отношений из некоторого объекта), или они вводят отношения мощности, используя реализацию схем отношений и мощность этих реализованных множеств при определенных ограничениях.
- Значения по умолчанию, принимаемые в случае отсутствия описания, также различны. В некоторых случаях значения по умолчанию недопустимы.

- Существуют различные графические представления ограничений мощности для бинарных схем отношений.
- Делаются различные попытки обобщения графических представлений для схем отношений более высокой арности.
- Ограничения вхождения (на минимальное количество вхождений) интерпретируется либо как достижимая нижняя граница, либо как строгое условие.
- Значения ограничений мощности зависят от семантики, от того, основана ли схема отношений на интерпретации множеств или указателей.

Эти различия, а также различия в терминологии (ограничения мощности, например, называют ограничениями сложности, относительной мощности, связности, степени, единственности, иногда эти ограничения описываются ограничениями других типов, например, ключами) показывают необходимость унификации определений и формальной их интерпретации. Для $R = (R_1, ..., R_k, attr(R))$, где R_i – схема отношений или объектов для любого $i, 1 \leq i \leq k$, *ограничение мощности*

$$comp(R, R_i) = (m, n)$$

определяет, что в любом состоянии базы данных элемент e из R_i^t встречается в R^t не менее m и не более n раз.

Для бинарной схемы отношений $R = (R_1, R_2, attr(R))$ между двумя схемами объектов или отношений R_1, R_2 традиционно вводятся специальные ограничения мощности: Один-к-Одному, Один-ко-Многим, Многие-к-Одному, Многие-ко-Многим. Например, схема отношений многие-к-одному означает, что любой элемент из R_2^t связан с произвольным количеством элементов из R_1^t, любой элемент из R_1^t связан с не более чем одним элементом из R_2^t, то есть $comp(R, R_1) = (0, 1)$ или $comp(R, R_1) \leq (1, 1)$ и $comp(R, R_2) \in \{(0, m), (1, m)\}$. Такая запись может быть построена для любого отношения.

Наиболее общей формой обобщенных ограничений мощности является следующая:
Пусть дан тип отношения $R = (seq, attr(R))$ для последовательности типов компонент и интервал I натуральных чисел с нулем. Пусть seq_1 – подпоследовательность seq, а seq_2 – непустая подпоследовательность seq_1 и $SEQ_2 = seq_{2,0}, seq_{2,1}, ..., seq_{2,n}$ – разбиение seq_2 на подпоследовательности или пустая последовательность. seq_1 будем называть корневым выражением, seq_2 – компонентным выражением.
Обобщенное ограничение мощности $comp^{seq_{2,1},...,seq_{2,n}}(R[seq_1], seq_2) = I$

определяет, что элементы R^t_{jh} из $seq_{2,0} = R_{j1}...R_{jk},\ 1 \le h \le k$ и элементы из $R^t\mid_{seq_{2,i}},\ 1 \le i \le n$ встречаются в проекции $R^t\mid_{seq_1}$ i раз, $i \in I$.

Используя эти обозначения, можно унифицировать остальные обобщения. Например, для $R = (E, F, G, H, \emptyset)$
$comp^{\lambda}(R[EFGH], EF) = \{0, 1, 2\}$ эквивалентно $comp(R, EF) = (0, 2)$,
$comp^{E,\,F}(R[EFG], EF) = \{1, 2, 3\} \longleftrightarrow comp^{*}(R[EFG], EF) = (1, 3)$,
$comp^{EF}(R[EFG], EF) = \{1, 2, 3, 4\} \longleftrightarrow comp^{+}(R[EFG], EF) = (1, 4)$,
$comp^{E}(R[EFGH], E) = \{1\} \longleftrightarrow comp^{*}(R, E) = (1, 1)$ и $comp^{+}(R, E) = (1, 1)$.
$comp^{EG,\,F}(R[EFGH], EFG) = \{0, 1, 2, 3\}$ не может быть представлено в другой форме.
Пустая последовательность обозначается λ и может опускаться.

	seq_1	$seq_{2,0}$	$seq_{2,1}$	$seq_{2,i}, (i \ge 2)$
обобщенное ограничение мощности	seq_R	seq_2	λ	λ
$^{+}$-ограничение мощности	seq_R	λ	R'_1	$R'_i, (i \le m)$
*-ограничение мощности	seq_R	λ	seq_2	λ
обобщенное проецирующее ограничение мощности	$\sqsubset seq_R$	seq_2	λ	λ
проецирующее $^{+}$-ограничение мощности	$\sqsubset seq_R$	λ	R'_1	$R'_i, (i \le m)$
проецирующее *-ограничение мощности	$\sqsubset seq_R$	λ	seq_2	λ

Приведенная таблица иллюстрирует эту связь для обобщенного ограничения мощности $comp^{seq_{2,1},...,seq_{2,n}}(R[seq_1], seq_2) = I$
на $R = (seq_R, attr(R))$ и интервала $I = \{l, l+1, \ldots, p\} = (l, p)$.

Основываясь на этих понятиях, мы можем обобщить ограничения мощности на произвольные типы конструкторов [90] и другие модели баз данных.

Ограничения мощности не аксиоматизируемы. То же верно и для обобщенных ограничений мощности. Однако, они имеют ряд полезных свойств.

Функции мощности монотонно убывают как для последовательностей компонент, так и для корневых выражений. Нижняя граница достигается на корневых выражениях.

Пусть задан тип отношения $R = (R_1, ..., R_k, attr(R))$. Ограничение мощности $comp(R, R'_1...R'_m) \le (1, 1)$ имеет место тогда и только тогда,

когда в R имеет место функциональная зависимость $R'_1...R'_m \longrightarrow R_1, ..., R_k$.

Ограничение $comp(R, R') \geq (1, 1)$ эквивалентно зависимости включения $R' \subseteq R[R']$.

$comp^*(R, R'_1...R'_m) = (1, 1)$ тогда и только тогда, когда $R'_1...R'_m \longrightarrow R_1, ..., R_k$ имеет место в R.

Ограничение мощности $comp(R, R') = (0, .)$ может использоваться в качестве значения по умолчанию, поскольку не является ограничением.

Ограничения мощности не являются зависимостями. Зависимостью является формула, выполняющаяся во всех базах данных, содержащих только отношения с единственным элементом. Вообще говоря, реализацией ER-схемы с множеством ограничений мощности является пустая база данных. Однако, если множество ограничений мощности неудачно описано, пустое множество может быть единственной конечной реализацией этой схемы. Если ER-схема является иерархической и использует только ограничения мощности (без обобщенных ограничений мощности), она имеет также непустые конечные реализации.

ER-схему $\underline{S}$ с множеством ограничений мощности $\underline{C}$ будем называть совместной (строго реализуемой), если существует хотя бы одна конечная база данных $DB = (r_1, ..., r_k)$ в $SAT(\underline{S}, \underline{C})$, в которой все r_i непусты.
Это свойство не является тривиальным.

Можно показать, что ER-схема совместна тогда и только тогда, когда каждый цикл имеет вес не менее 1. Веса вычисляются как отношение максимума мощности по направлению ориентированного цикла к минимуму в направлении, противоположном ориентации цикла (в случае, если $comp(R, R_i) = (0, m)$ для некоторого m, вес полагается равным ∞).

Кроме того, веса могут использоваться для упрощения схемы. Если цикл имеет только смежные мощности из множества $\{(0, 1), (0, .), (1, 1), (1, .)\}$, то из того, что вес цикла равен 1, следует, что все смежные мощности равны $(1, 1)$.

4.3 Ограничения в объектно-ориентированных моделях

Существует несколько подходов к объектно-ориентированным моделям.

1. Объектно-ориентированные модели строятся на основе моделей, достаточно удобных для исследования явлений. Одной из таких моделей является графическая модель, использованная в [19].

2. Объектно-ориентированные модели рассматриваются как обобщения семантических моделей. Добавляется один класс – класс идентификаторов объектов (например, [81]). В большинстве работ объектные модели основываются на семантических.

Оба подхода по сути эквивалентны. Вообще говоря, объект описывается своей структурой и поведением. Из объектов могут формироваться более сложные объекты. Имеется некоторое свойство идентичности, которое сохраняется при определенных изменениях. На практике это свойство кодируется идентификаторами, которые могут различаться в различных приложениях и системах. Объекты взаимодействуют друг с другом. Они могут классифицироваться по схемам и объединяться в классы объектов.

Базис графа объекта $B = (AV, Labels)$ определяется (конечным) множеством атомарных значений AV и (конечным) множеством меток $Labels$, $AV \cap Labels = \emptyset$.

Графом объекта G на $(AV, Labels)$ называется пара $(Nodes, Edges)$, где $Nodes$ – объединение конечных непересекающихся множеств атомарных элементов: множества AV атомарных значений, множества CV сложных значений и множества объектов O. $Edges$ является отношением $Edges \subseteq (O \cup CV) \times Labels \times (AV \cup CV \cup O)$.

$Edges$ может быть как множеством, так и набором или мультимножеством. В последнем случае некоторые элементы $Edges$ могут повторяться. Однако, интерпретация $Edges$ как набора приводит к сложностям при реализации модели. Поэтому мы будем представлять наборы множествами с дополнительной информацией о числе вхождений элементов.

Одним из главных принципов ООБД является принцип тождества объектов. Без возможности правильно различать объекты невозможно корректно интерпретировать такие понятия как наследование и взаимодействие: наследование использует понятие *того же* объекта, взаимодействие – *различных* объектов. Каждый объект должен быть однозначно идентифицируемым. Это достигается использованием идентификаторов объектов. Идентификаторы являются зависящими от реализации элементарными данными, единственной задачей которых является однозначно соответствовать объекту. Каждый объект имеет некоторое внутреннее состояние, которое может наблюдаться и использоваться только через методы данного объекта, его 'интерфейс'. Поэтому тождество определяется 'интерфейсом' объектов. В базах данных понятие тождества основано на понятии неразличимости. Два объекта, которые не могут быть различены, полагаются тождественными. Поэтому понятие тождества зависит

также и от языка, используемого для описания свойств объектов. Этот язык основывается на структурных конструкциях и может иметь также расширенную функциональность, например, подсчет и т.п.

Схема, построенная из структурного выражения класса S заменой каждой ссылки и каждой подссылки соответствующим идентификатором ID, называется *представлением типа* V_C класса C, схема $U_C = (ident : ID, value :: T_C)$ называется *схемой класса* C.

Объектная база данных над структурной схемой $\mathcal{S}$ сопоставляет каждой схеме класса C множество объектов, таких что выполняются следующие условия:
Единственность идентификаторов: Для каждого класса C^t идентификаторы уникальны.
Целостность по включению: Идентификаторы, использующиеся в подклассе C^t класса C'^t используются и в C'^t. Более того, если V_C – подсхема V'_C с функцией подсхемы $f : V_C \to V'_C$, то $(i, v) \in C^t$ влечет $(i, f(v)) \in C'^t$.
Ссылочная целостность: Если объект ссылается на объект m другого класса, последний должен существовать.
Локальная ссылочная целостность: Если два объекта ссылаются на один и тот же объект из подкласса, то эти два объета совпадают.

Основываясь на этих предположениях, мы можем переформулировать определения некоторых типов ограничений целостности. *Ограничением целостности* над схемой $\mathcal{S} = \{C_1, \ldots, C_n\}$ будем называть формулу из построенной для этой схемы логики.
Для схемы класса C, *ограничение единственности* на C требует, чтобы объекты, имеющие одинаковые представления были тождественными.
Функциональное ограничение на C – это ограничение вида $X \to Y$ для подструктур X, Y структуры C.
Ограничение включения на C_1 и C_2 – это ограничение вида $C_1[X] \subseteq C_2[Y]$ для подструктур X, Y структур C_1 и C_2 соответственно.
Ограничение исключения на C_1, C_2 имеет вид $C_1[X] \| C_2[Y]$ для подструктур X, Y структур C_1 и C_2 соответственно.
Аналогичным образом мы можем обобщить теорию зависимостей других моделей.

Ограничения неравенства определяются на уровне класса. Пусть дана схема представления значений V_C схемы класса C, X, Y – подструктуры V_C ($X \sqsubseteq V_C$; $\sqsubseteq, \sqcup, \sqcap$ – операции и предикаты, определенные на подструктурах). *Ограничение неравенства* $C(X \rightleftharpoons \| \; Y)$ имеет место в классе C^t, если не существует двух объектов в C^t, которые совпадают как на X-значениях, так и на Y-значениях. Очевидно, что X, Y могут быть использованы для идентификации

объектов в C^t. Поэтому это объектно-ориентированное понятие близко к понятию ключевых множеств. Можно определить систему формальных правил вывода, например

$$\frac{C(X \nparallel Y)}{C(Y \nparallel X)} \qquad \frac{C(Y \nparallel X)}{C(Y \sqcup Z \nparallel X)} Z \sqsubseteq V_C \qquad \frac{C(X \nparallel Y, Z \to X)}{C(Z \nparallel Y)}$$

Кроме того, могут быть добавлены правила, описывающие семантику наследования для подклассов и отображений, например, префиксные расширения или расширенные пути.

Ограничения целостности в объектно-ориентированных моделях находятся в стадии исследования. Некоторые проблемы до сих пор не разрешены. Одной из них является несоответствие между классами и подклассами, поскольку логические языки, основанные на логике предикатов первого порядка, не могут описывать наследование [61].

4.4 Ограничения целостности в других моделях

Большинство ограничений, введенных в других моделях, являются обобщениями уже введенных в реляционной модели ограничений. Единственное исключение составляет множество зависимостей, используемых в моделях, содержащих списки ссылок для моделирования свойств множества. Эти ограничения имеют вид:

$$y_1 = (\bar{x}), y_2 = (\bar{x}) \in R^t \;\Rightarrow\; y_1 = y_2 \; .$$

Зависимости, порожденные элементом, являются обобщением зависимостей, порожденных кортежами. Эти ограничения необходимы в том случае, когда система типов модели гораздо богаче, чем используемая в обычной реляционной модели. Например, мы можем положить для некоторого отображения, что если существует некоторое множество, то существует некоторый элемент в структуре, который удовлетворяет некоторому условию. Более формально, пусть T, T_1 – типы, $f : T \to T_1$. *Обобщенное ограничение, порожденное элементом* (Ω, ω) задается множеством $\Omega \cup \{\omega\}$, определенным на T_1. Ограничение (Ω, ω) имеет место в отношении R^t на T, если из существования S^t – подмножества R^t, такого что $f(S^t) = \Omega$ следует, что существует элемент r в R^t, такой что $f(r) = \omega$.

Аналогично, обобщение ограничения, порожденного равенствами, можно определить следующим образом: пусть даны типы $T_{i,j} = T_{j,i}$, $1 \leq i, j \leq m$, $T'_{i,j} = T'_{j,i}$, $1 \leq i, j \leq m$ и функции $f_{i,j} : T \to T_{i,j}$, функции $f'_{i,j} : T \to T'_{i,j}$. Положим $F = \{f_{i,j} \mid 1 \leq i, j \leq m\}$, $F' = \{f'_{i,j} \mid 1 \leq i, j \leq m\}$. Тогда класс C^t на T удовлетворяет *обобщенному ограничению, порожденному равенствами* (F, F'), если для

любых m элементов $r_1, ..., r_m$ из C^t $\bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^{m} f_{i,j}(r_i) = f_{j,i}(r_j)$ влечет $\bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^{m} f'_{i,j}(r_i) = f'_{j,i}(r_j)$.

Если функции $f, f_{i,j}, f'_{i,j}$ построены на подструктурах T, и $T_1, T_{i,j}, T'_{i,j}$ являются подструктурами T, то процедура прогонки может быть обобщена на обощенные ограничения, порожденные элементами и равенствами. В остальных случаях аксиоматизация зависит от свойств $f, f_{i,j}, f'_{i,j}$, особенно от некоторого варианта обобщенной инъективности.

Обычно объект o, существование которого зависит от другого объекта o', не может быть добавлен к базе данных до того, как добавлен o', а o' не может быть удален до того, как удален o. Таким образом, o блокирует удаление o', а o' блокирует добавление o. Далее, значение ограничений включения может быть уточнено рассмотрением случаев, когда используемые элементы удалены из S^t, использованием вместо них неопределенных значений. Это приводит к следующему разделению [64]:

1. *Блокирующее ограничение существования* есть зависимость включения $R[X] \subseteq S[Y]$, которая запрещает добавление в R^t прежде, чем соответствующие элементы будут добавлены в S^t и удаление из S^t, если при этом нарушится ограничение.

2. *Триггерное ограничение существования* есть зависимость включения $R[X] \subseteq S[Y]$, требующая, что если элемент вставляется в R^t, то соответствующий элемент должен быть вставлен в S^t, если он там отсутствует, при удалении элемента из S^t, из R^t удаляются все элементы, существование которых зависит только от удаляемых элементов.

Ограничения целостности накладываются и на правила вывода в содержательных (intensional) частях дедуктивных баз данных. Они могут использоваться для перевода в программы на Прологе и для улучшения представления логических программ. Например, функциональные зависимости могут быть использованы для автоматической вставки 'заплаток' (cuts) в логические программы. Можно показать, что проверка ограничений в логических моделях не намного сложнее, чем в реляционной. Сложность данных (размер $SAT(DS)$) обычно логарифмическая, сложность выражений (количество всех ограничений, применимых к базе данных) – полиномиальная.

Обслуживание дедуктивных баз данных включает проверку выполнения ограничений целостности. На практике это может быть невыполнимо. Поэтому множества ограничений редуцируются или упрощаются. Для каждой общей операции выбираются 'подозрительные'

(infected) ограничения целостности, модифицируются и добавляются в предусловия соответствующих операций. Излишние и фиктивные (phantom) предусловия могут быть опущены. Ограничения целостности используются также для оптимизации запросов пользователя.

Библиографические замечания

Теория зависимостей обсуждается в [4, 9, 63, 72, 84, 93]. Обобщение этой теории на другие модели развивается в [7, 49, 50, 52, 85] для семантических моделей и в [3, 21, 42, 48, 59, 80] для объектно-ориентированных. Книга [101] поможет составить впечатление об алгебраическом подходе к зависимостям. Обзор по дедуктивным базам данных представлен в [31, 92]. Далее, в работах [22, 27, 28], цитированных в [45] и в [68], рассмотрены некоторые специальные вопросы. Обзор по динамическим ограничениям целостности дан в [62].

5 Заключение

Существует множество исследований по базам данных, интересных не только самих по себе, но и по их практическому применению. Эти работы можно сгруппировать следующим образом.

Построение систем. Конкретные примеры удачного применения теоретических результатов.

Методологические и концептуальные подходы. Методологические замечания, описывающие этапы, шаги, технику и рекомендации, основанные на теоретических результатах.

Улучшение производительности систем. Развитие и улучшение систем реляционных баз данных показывает применимость различных теорий.

Упрощение алгоритмов. Для больших баз данных очень важно построить алгоритмы с удовлетворительной функцией сложности по времени.

Рекомендации по дальнейшему ознакомлению с теорией

Обзор по нормализации дан в [72, 92, 99], горизонтальной нормализации – в [46, 72]. Новый подход к нормализации представлен в [23, 94]. Последняя работа показывает, что нормализация до сих пор

недостаточно изучена. Большинство книг по ER-модели может быть использовано в качестве источника по подходам к вертикальной нормализации. Работы [38, 43, 45, 86] обсуждают влияние логики на теорию баз данных. Вопросы сложности обсуждаются в [54, 84], результаты доказываются там же или в [15, 33, 34, 65]. Обзор по сложности алгоритмов может быть найден в [56, 63].

Список литературы

[1] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik. The object-oriented database system manifesto. Proc. Conf. DOOD 89, Kyoto 1989, 40 - 57.

[2] S. Abiteboul, R. Hull. IFO: a formal semantic database model. Proc. PODS 84, 3, 119–132.

[3] S. Abiteboul, P.C. Kanellakis. The two facets of object-oriented data models. IEEE Data Engineering Bulletin 14, 2, 1991, 3-7

[4] S. Abiteboul, V. Vianu. Transactions and integrity constraints. Proc. of Database Systems, 1985, 193-204.

[5] S. Abiteboul, R. Hull, and V. Vianu. Foundations of databases. Addison-Wesley, Paris 1995.

[6] H. Aít-Kaci. A glimpse of paradise. In J. W. Schmidt, A. A. Stognij (Eds.): Proc. Next Generation Information Systems Technology , Springer, LNCS, vol. 504, 1991, 17 - 25.

[7] S. Al-Fedaghi, B. Thalheim. The key concept in database models. Information systems, 1992.

[8] K.R. Apt. Logic programming. In Handbook of Theoretical Computer Science (ed. J. van Leeuwen), Vol. B, Formal Models and Semantics, Elsevier, Amsterdam, 1990, 493 – 574.

[9] P. Atzeni, V. De Antonellis. Relational database theory. Benjamin Cummings, Reading, 1993.

[10] F. Bancilhon, R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. Proc. ACM SIGMOD Conf, 1986, 16 – 52.

[11] F. Bancilhon, R. Ramakrishnan. Performance evaluation of data intensive logic programs. In Foundations of Deductive Databases and Logic Programming (ed. J. Minker), Morgan Kaufman, Los Altos, 1988, 439 – 518.

[12] C. Beeri, P.A. Bernstein, N. Goodman. A model for concurrency in nested transaction systems. Journal of the ACM, 36, 1989, 230 – 269.

[13] C. Batini, S. Ceri, S. Navathe. Conceptual database design, An entity-relationship approach. Benjamin Cummings, Redwood, 1992.

[14] A.P. Buchmann, R.S. Carrera, M.A. Vazquez-Galindo. A generalized constraint and exception handler for an object-oriented CAD-DBMS. IEEE Conf. 1986, 38–49.

[15] C. Beeri, M. Dowd, R. Fagin, R. Statman. On the structure of Armstrong relations for functional dependencies. Journal of ACM, Vol.31, No.1, January 1984, 30-46.

[16] C. Beeri. New data models and languages - the challenge. Proc. ACM PODS, 1992.

[17] C. Beeri, M. Kifer. An integrated approach to logical design of relational database schemes. ACM TODS, 11, 1986, 159–185.

[18] C. Beeri, T. Milo. Subtyping in OODB's. Proc. 10th ACM PODS 1991, 300-314.

[19] C. Beeri, B. Thalheim. Can I see your identification, please?, manuscript in preparation, 1993

[20] P.A. Bernstein, V. Hadzilacos, N. Goodman. Concurrency control and recovery in database systems. Addison-Wesley, Reading Mass. 1987.

[21] J. Biskup, P. Dublish. Objects in relational database schemes with functional, inclusion and exclusion dependencies. Proc. MFDBS - 91, LNCS 495, 1991, 276 – 290.

[22] N. Bidoit. Negation in rule-based database languages: A survey. Theoretical computer science 78, 1991, 3 - 83.

[23] J. Biskup. Foundations of information systems. Vieweg, Braunschweig, 1995 (in German).

[24] P. Buneman, L. Libkin, D. Suciu, V. Tannen, L. Wong. Comprehension syntax. SIGMOD Record, 23, 1, 1994, 87 – 96.

[25] M.L. Brodie, J. Mylopoulos, J.W. Schmidt. On conceptual modeling. Springer, Heidelberg, 1984.

[26] M.L. Brodie and J. Mylopoulos. On knowledge base management systems. Springer, Heidelberg, 1986.

[27] F. Bry. Query evaluation in recursive databases: Bottom-up and top-down reconciled. ECRC Report IR-KB-64, 1989.

[28] F. Bry. Intensional updates: Abduction via deduction. Proc. 7th Int. Conf. on Logic Programming, 1990.

[29] R.G.G. Cattell. Object data management: Object-oriented and extended relational database systems. Addison-Wesley, Reading, 1991.

[30] S. Ceri, G. Pelagatti. Distributed databases: Principles and systems. McGraw-Hill, New York, 1984.

[31] S. Ceri, G. Gottlob, A. Tanca. Logic Programming and databases. Springer 1991.

[32] C. Delobel, M. Adiba. Relational Database Systems. North-Holland, Amsterdam 1985.

[33] J. Demetrovics, G.O.H. Katona. Combinatorial problems of database models. Colloquia Mathematica Societatis Janos Bolyai 42, Algebra, Cominatorics and Logic in Computer Science, Győr (Hungary), 1983, 331-352.

[34] J. Demetrovics, L.O. Libkin, I.B. Muchnik. Functional dependencies and the semilattice of closed classes. Proc. MFDBS-89, LNCS 364, 1989, 136–147.

[35] R. Elmasri, S. H. Navathe. Fundamentals of database systems. Benjamin/Cummings Publ., Redwood City, 1989.

[36] R.A. Frost. Introduction to knowledge base systems. MacMillan, New York 1986.

[37] L. Fegaras, T. Sheard, D. Stemple. Uniform traversal combinators: Definition, use and properties. Proc. CADE-11, LNCS 607, Springer 1992, 118 – 162.

[38] H. Gallaire, J. Minker. Logic and databases. Plenum Press, New York, 1978.

[39] S.K. Gadia, C.-S. Yeung. A generalized model for a relational temporal database. Proc. ACM SIGMOD 1988, June 1988, Chicago, p. 251-259.

[40] M.R. Genesereth, N.J. Nilsson. Logical foundations of artificial intelligence. Morgan-Kaufman, los altos, 1988.

[41] M. Ginsberg. Nonmonotonic reasoning. Morgan-Kaufman, Los Altos, 1988.

[42] G. Gottlob, G. Kappel, M. Schrefl. Semantics of object-oriented data models - The evolving algebra approach. LNCS 504, Springer 1991, 144–160.

[43] H. Gallaire, J. Minker, J.-M. Nicolas. Advances in database theory, Vol. I, Plenum-Press, New York, 1981.

[44] H. Gallaire, J. Minker, J.-M. Nicolas. Advances in database theory, Vol. II, Plenum-Press, New York, 1983.

[45] H. Gallaire, J. Minker, J.M. Nicolas. Logic and databases: a deductive approach. Computing Surveys 16, June 1984, 153-185.

[46] S.J. Hegner. Decomposition of relational schemata into components defined by both projection and restriction. ACM SIGACT-SIGMOS-SIGART Sym. 1988, 174-183.

[47] A. Heuer, G. Saake. Datenbanken. Konzepte und Sprachen. Thomson, Bonn, 1995.

[48] A. Heuer. Equivalent schemes in semantic, nested relational, and relational database models. LNCS 364, Springer, 1989, 237–253.

[49] R. Hull, R. King. Semantic database modeling: Survey, applications, and research isues. ACM Computing Surveys 19, 3, 1987, 201 -260.

[50] R. Hull. Four Views of Complex Objects: A Sophisticate's Introduction. In Proc. Conf. on Nested Relations and Complex Objects in Databases (Eds.

: S. Abiteboul, P.C. Fischer, and H.J. Schek), Lecture Notes in Computer Science, 1989, 361, 87–116.

[51] T. Imielinski, W. Lipski Jr. A systematic approach to relational database theory. ICS PAS Reports 457, Warszawa, 1982.

[52] B.E. Jacobs. On database logic. Journal of ACM, 29, 2, 1982, 310 - 332.

[53] G. Jaeschke, H.J. Schek. Remarks on the algebra of nonfirst-normal-form relations. Proc. First ACM SIGACT-Sigmod Symposium on Principles of Database Systems, 1982, 124-138.

[54] G.O.H. Katona, J. Demetrovics. A survey of some combinatorial results concerning functional dependencies in relational databases. Annals of Mathematics and Artificial Intelligence, 6, 1992.

[55] Y. Kambayashi. Database, a bibiliography. Computer Science Press, Rockville, 1981.

[56] P.C. Kanellakis. Elements of relational database theory. In Handbook of Theoretical Computer Science (ed. J. van Leeuwen), Vol. B, Formal Models and Semantics, Elsevier, Amsterdam, 1990,

[57] L. Kerschberg (ed.). Expert database systems. Benjamin Cummings, Menlo-Park, 1987.

[58] W. Kim, F.H. Lochovsky (eds.). Object-oriented concepts, databases, and applications. Addison-Wesley, Reading, 1989.

[59] M. Kifer, G. Lausen. F-logic: A higher-order language for reasoning about objects, inheritance and schema. Proc. ACM SIGMOD Conf. 1989, 134 – 146.

[60] I. Kobayashi. An overview of database mangement technology. In Advances in Information System Science, ed. J.T. Tou, Vol. 9, Plenum Press, New York, 1985.

[61] M. Lenzerini, D. Nardi, M. Simi (eds.). Inheritance hierarchies in knowledge representation and programming languages. John Wiley, Chichester, 1991.

[62] U.W. Lipeck, B. Thalheim (eds.). Modelling Database Semantics, Springer Series Workshops in Computing, 1992.

[63] D. Maier. The theory of relational databases. Computer Science Press, Rockville, MD, 1983.

[64] V.M. Markowitz. Referential integrity revisited: An object-oriented perspective. Proc. VLDB 1990, 578 – 589.

[65] H. Mannila, K.-J. Ráihá. On the complexity of inferring functional dependencies. Discrete Applied Mathematics, 1992.

[66] F. Matthes, J.W. Schmidt. Towards database application systems: Types, kinds and other open invitations. Proc. Next Generation Information System Technology, LNCS 504, 1991, 185 – 211.

[67] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. Telos: Representing knowledge about information systems. ACM ToIS, 8, 4, 1990, 325 - 362.

[68] J. Minker (ed.). Foundations of deductive databases and logic programming. Morgan Kaufman, Los Altos, 1988.

[69] J.C. Mitchell. Type systems for programming languages. In Handbook of Theoretical Computer Science (ed. J. van Leeuwen), Vol. B, Formal Models and Semantics, Elsevier, Amsterdam, 1990, 365 – 458.

[70] J.F. Nilsson. Knowledge base property combinator logic. Information Processing 89 (ed. G.X. Ritter), Elsevier, Amsterdam, 1989, 661 – 666.

[71] C.H. Papadimitriou. The theory of database concurrency control. Computer Science Press, Rockville, 1986.

[72] J. Paredaens, P. De Bra, M. Gyssens, D. Van Gucht. The structure of the relational database model. Springer, Berlin, 1989.

[73] S.V. Petrov. Finite axiomatization of languages for representation of system properties: Axiomatization of dependencies. Information Sciences 47, 1989, 339-372.

[74] Plenum Studies, Database Technology. University Wúrzburg, 1991.

[75] R. Reiter. On formalizing database updates: Preliminary report. Proc. EDBT-92, LNCS 580, 1992, 10 - 20.

[76] G. Saake. Descriptive specification of database object behaviour, Data and Knowledge Engineering 6 (1991), 47 – 73

[77] H.J. Schek, M.H. Scholl. Evolution of data models. Proc. Database Systems of the 90s, LNCS 466, 1990, 135-153.

[78] J.W. Schmidt, C. Thanos (eds.). Foundations of knowledge base management. Springer, Heidelberg, 1989.

[79] D.–G. Shin, K.B. Irani. Fragmenting relations horizontally using a knowledge-based approach. IEEE Transaction on Software Engineering, 17, 9, 1991, 872–883.

[80] K.-D. Schewe, B. Thalheim, I. Wetzel, J.W. Schmidt. Extensible safe object-oriented design of database applications. University Rostock, Computer Science Department, Preprint CS-09-91, 1991.

[81] K.-D. Schewe, B. Thalheim, I. Wetzel. Foundations of object-oriented concepts. Technical Report, Computer Science Department, Hamburg University, FBI - HH - B - 157/92, Aug. 1992.

[82] B. Thalheim. Deductive basis of relations. Proc. MFSSSS 84, LNCS 215, 226-230.

[83] B. Thalheim. On the number of keys in relational and nested relational databases. Discrete Applied Mathematics, 38, 1992.

[84] B. Thalheim. Dependencies in Relational Databases. Leipzig, Teubner Verlag 1991.

[85] B. Thalheim. Foundations of entity-relationship modeling. Annals of Mathematics and Artificial Intelligence, 6, 1992.

[86] B. Thalheim. Fundamentals of cardinality constraints. Proc. Conf. on Entity-Relationship-Approaches, LNCS 645, 7–23, 1992.

[87] B. Thalheim. An overview on database theory. Datenbank-Rundbrief 10, November 1992, 2–13.

[88] B. Thalheim. Database design strategies. Advances in Database Systems (ed. J. Paredaens, L. Tenenbaum), Springer, CISM Courses 347, 1994, 267 - 285.

[89] A survey on database constraints. Preprint CS-08-94, Cottbus Technical University, Computer Science Institute.

[90] B. Thalheim. Fundamentals of Entity-Relationship models. Springer, Heidelberg 1996.

[91] A. Thayse (ed.). From modal logic to deductive databases. John Wiley, vol. 1: 1989, vol. 2: 1990.

[92] J. D. Ullman. Principles of database and knowledge-base systems. Computer Science Press, 1989.

[93] M.Y. Vardi. Fundamentals of dependency theory. In Trends in Theoretical Computer Science (ed. E. Bórger), Computer Science Press, Rockville, 171 – 224.

[94] M. Vincent. The semantic jusitification for normal forms in relational database design. PhD Thesis, Monash University, 1994.

[95] P. Wadler. List comprehensions. Chapter 7 in P. Jones, The implementation of functional programming languages. Prentice Hall, New York, 1987.

[96] P. Wegner. Concepts and paradigms of object-oriented programming. ACM SIGPLAN OOP Messenger, Vol. 1, No. 1, 1990, 7 - 87.

[97] M. Wirsing. Algebraic specification. In Handbook of Theoretical Computer Science (ed. J. van Leeuwen), Vol. B, Formal Models and Semantics, Elsevier, Amsterdam, 1990, 675 – 788.

[98] G. Wiederhold, M. Winslett, N. Naclerio. Layering an Engineering Information System. IEEE COMPCON, 34, 1989, 444–449.

[99] C.-C. Yang. Relational databases. Prentice Hall, Englewood Cliffs, 1986.

[100] L.-Y. Yuan, Z.M. Ozsoyoglu. Design of desirable relational database schemes. JCSS 1992, 45, 3, 1992, 435 – 470.

[101] M.S. Zalenko. Modeling semantics in data bases. Science, Moscov, 1989 (in Russian).

[102] C. Zaniolo (ed.). Data Engineering, 1987, 10, 4.