ЛЕКЦИЯ №6 НОРМАЛИЗАЦИЯ СХЕМ БАЗЫ ДАННЫХ

6.1 Проблемы при построении БД 1
6.2 Нормальные формы. 2
6.3 Нормализация и функциональные зависимости 2
6.4 Первая нормальная форма 4
6.5 Вторая нормальная форма 4
6.6 Третья нормальная форма. 4
6.7 Нормальная форма Бойса-Кодда. 5
6.8 Четвертая нормальная форма. 7
6.9 Пятая нормальная форма. 8
6.10 Доменно-ключевая нормальная форма. 8
6.11 Разработка базы данных 9

6.1 Проблемы при построении БД

Прежде чем мы приступим к обсуждению методов проектирования хороших схем баз данных, давайте посмотрим, почему некоторые схемы могут оказаться неадекватными. В частности, обратимся к схеме отношения

ПОСТАВЩИКИ (НАЗВ_ПОСТ, АДРЕС_ПОСТ, ТОВАР, ЦЕНА)

НАЗВ_ПОСТ	АДРЕС_ПОСТ	ТОВАР	ЦЕНА
АГАТА-ИМПЕКС	ул Ата-тюрк 208	Компьютер P IV	2 000 000
НУРОН	ул Навои 158	Монитор LCD 17"	254 000
TS-TECHNOLOGY	ул Янгиюльская 64	Компьютер P IV	1 800 000
НУРОН	ул Навои 158	Клавиатура	25 000
SHARIFA-T	Чиланзар-6 64	Мышь	15 000

Рисунок 1

В связи с этой схемой возникает несколько проблем:

1. *Избыточность.* Адрес поставщика повторяется для каждого поставляемого товара.
2. *Потенциальная противоречивость (аномалии обновления).* Вследствие избыточности мы можем обновлять адрес поставщика в одном кортеже, оставляя его неизменным в другом. Таким образом, может оказаться, что для некоторых поставщиков нет единого адреса. Однако интуитивно мы чувствуем, что он должен быть.
3. *Аномалии включения.* В базу данных не может быть записан адрес поставщика, если он в настоящее время не поставляет по меньшей мере один товар. Можно, конечно, поместить неопределенные значения в компоненты ТОВАР и ЦЕНА кортежа для этого поставщика. Но если он начнет поставлять некоторый товар, не забудем ли мы удалить кортеж с неопределенными значениями? Хуже того, ТОВАР и НАЗВ_ПОСТ образуют ключ данного отношения, и поиск кортежей с неопределенными значениями в ключе может быть затруднительным или невозможным.
4. *Аномалии удаления.* Обратная проблема возникает при необходимости удаления всех товаров, поставляемых данным поставщиком, вследствие чего мы непреднамеренно утрачиваем его адрес.

Возникает вопрос: «Как найти хорошую замену для плохой схемы отношений?»

6.2 Нормальные формы.

Отношения можно классифицировать по типам аномалий модификации, которым они подвержены. В 1970-х годах теоретики реляционных баз данных постепенно сокращали количество этих типов. Кто-то находил аномалию, классифицировал ее и думал, как предотвратить ее возникновение. Каждый раз, когда это происходило, критерии построения отношений совершенствовались. Эти классы отношений и способы предотвращения аномалий называются нормальными формами (normal forms). В зависимости от своей структуры, отношение может быть в первой, во второй или в какой-либо другой нормальной форме.

В своей работе, последовавшей за эпохальной статьей 1970 г., Кодд и другие определили первую, вторую и третью нормальные формы (1НФ, 2НФ и ЗНФ). Позднее была введена нормальная форма Бойса-Кодда (НФБК), а затем были определены четвертая и пятая нормальные формы. Как показывает рис. 5.6, эти нормальные формы являются вложенными. То есть отношение во второй нормальной форме является также отношением в первой нормальной форме, а отношение в 5НФ (пятая нормальная форма) находится одновременно в 4НФ, НФБК, ЗНФ, 2НФ и 1НФ.

Эти нормальные формы помогали, но у них было и серьезное ограничение. Не было теории, гарантирующей, что какая-либо из этих форм устранит все аномалии: каждая форма могла устранить только определенные их виды. Эта ситуация разрешилась в 1981 г., когда Р. Фагин (R. Fagin) ввел новую нормальную форму, которую он назвал домеино-ключевой нормальной формой, или ДКНФ (domain/key normal form, DK/NF). В своей важной статье Фагин показал, что отношение в ДКНФ свободно от всех аномалий модификации, независимо от их типа1. Он также показал, что любое отношение, свободное от аномалий модификации, должно находиться в ДКНФ.

До введения ДКНФ теоретикам реляционных баз данных приходилось продолжать поиск все новых и новых аномалий и нормальных форм. Доказательство Фагина упростило ситуацию. Если мы можем привести отношение к ДКНФ, можем быть уверены, что в нем не будет аномалий модификации. Вся загвоздка в том, как привести отношение к ДКНФ.

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

В основе процесса проектирования лежит метод нормализации, декомпозиция отношения, находящегося в предыдущей нормальной форме, в два или более отношения, удовлетворяющих требованиям следующей нормальной формы.

6.3 Нормализация и функциональные зависимости

Нормализация — это процесс преобразования отношения, имеющего некоторые недостатки, в отношение, которое этих недостатков не имеет. Что еще более важно, нормализацию можно использовать как критерий для определения желательности и правильности отношений. Вопрос о том, что такое хорошо структурированное отношение, был предметом многочисленных теоретических исследований. Термин нормализация обязан своим появлением одному из пионеров технологии баз данных, Э. Ф. Кодду (E. F. Codd), который определил различные нормальные формы (normal forms) отношений.

Наиболее важные на практике нормальные формы отношений основываются на фундаментальном в теории реляционных баз данных понятии *функциональной зависимости*. Для дальнейшего изложения нам потребуются несколько определений.

.

Функциональная зависимость ***(functional dependency)*** — это связь между атрибутами. Предположим, что если мы знаем значение одного атрибута, то можем вычислить (или найти) значение другого атрибута. Например, если нам известен номер счета клиента, то мы можем определить состояние его счета. В таком случае мы можем сказать, что атрибут СостояниеСчетаКлиента функционально зависит от атрибута НомерСчетаКлиента.

Говоря более общим языком, атрибут Y функционально зависит от атрибута X, если значение X определяет значение Y. Другими словами, если нам известно значение X, мы можем определить значение Y.

Уравнения выражают функциональные зависимости. Например, если мы знаем цену и количество приобретенного товара, мы можем определить стоимость покупки по следующей формуле:

Стоимость = Цена x Количество

В этом случае мы могли бы сказать, что атрибут Стоимость функционально зависит от атрибутов Цена и Количество.

Функциональные зависимости между атрибутами в отношении обычно не выражаются уравнениями. Пусть, например, каждому студенту присвоен уникальный идентификационный номер, и у каждого студента есть одна и только одна специальность. Имея номер студента, мы можем узнать его специальность, поэтому атрибут Специальность функционально зависит от атрибута НомерСтудента. Или рассмотрим компьютеры в вычислительной лаборатории. Каждый компьютер имеет конкретный размер основной памяти, поэтому атрибут ОбъемПамяти функционально зависит от атрибута СерийныйНомерКомпьютера.

В отличие от случая с уравнением, такие функциональные зависимости нельзя разрешить при помощи арифметики; вместо этого они хранятся в базе данных. Фактически, можно утверждать, что базу данных стоит иметь только ради хранения и выдачи функциональных зависимостей.

Функциональные зависимости обозначаются следующим образом:

НомерСтудента > Специальность

СерийныйНомерКомпьютера > ОбъемПамяти

Первое выражение читается так: «атрибут НомерСтудента функционально определяет атрибут Специальность», «атрибут НомерСтудента определяет атрибут Специальность» или «атрибут Специальность зависит от атрибута НомерСтудента». Атрибуты по правую сторону от стрелки называются *детерминантами* ***(determinants)***.

В функциональные зависимости могут быть вовлечены группы атрибутов. Рассмотрим отношение ОЦЕНКИ (НомерСтудента,Дисциплина, Оценка). Комбинация номера студента и дисциплины определяет оценку. Такая функциональная зависимость записывается следующим образом:

(НомерСтудента**,** Дисциплина**)** > Оценка

Заметьте, что для определения оценки требуется как номер студента, так и дисциплина. Мы не можем разделить эту функциональную зависимость, поскольку ни номер студента, ни дисциплина не определяют оценку сами по себе.

Ключ **(key)** — это группа из одного или более атрибутов, которая уникальным образом идентифицирует строку. Каждое отношение имеет минимум один ключ. Это утверждение должно быть верным, поскольку ни одно отношение не может иметь одинаковых строк, и, следовательно, в крайнем случае, ключ будет состоять из всех атрибутов отношения

6.4 Первая нормальная форма

О любой таблице данных, удовлетворяющей определению отношения, говорят, что она находится в первой нормальной форме (first normal form, 1NF). Вспомните, что для того, чтобы таблица была отношением, должно выполняться следующее: ячейки таблицы должны содержать одиночные значения и в качестве значений не допускаются ни повторяющиеся группы, ни массивы. Все записи в одном столбце (атрибуте) должны иметь один и тот же тип. Каждый столбец должен иметь уникальное имя, но порядок следования столбцов в таблице несуществен. Наконец, в таблице не может быть двух одинаковых строк, и порядок следования строк несуществен.

Отношение находиться в 1НФ, если все его атрибуты являются простыми, т.е. имеют единственное значение.

Условия первой нормальной формы:

- должны отсутствовать повторяющиеся записи;
- должны отсутствовать повторяющиеся атрибуты
- каждый атрибут должен быть неделим.

6.5 Вторая нормальная форма

Отношение находится во второй нормальной форме, если все его неключевые атрибуты зависят от всего ключа. В соответствии с этим определением, если отношение имеет в качестве ключа одиночный атрибут, то оно автоматически находится во второй нормальной форме.

Поскольку ключ является одиночным атрибутом, то по умолчанию каждый неключевой атрибут зависит от всего ключа, и частичных зависимостей быть не может. Таким образом, вторая нормальная форма представляет интерес только для тех отношений, которые имеют композитные ключи.

Отношение находиться во 2НФ, если оно удовлетворяет следующим условиям:

- выполняется условие 1НФ;
- первичный ключ однозначно определяет запись;
- все поля записи зависят от первичного ключа;
- первичный ключ имеет минимальную форму (отсутствует избыточность).

6.6 Третья нормальная форма.

Отношения во второй нормальной форме также могут иметь аномалии. Рассмотрим отношение ПРОЖИВАНИЕ на рис. 2. Ключом здесь является НомерСтудента, и имеются функциональные зависимости НомерСтудента > Общежитие и Общежитие > Плата. Эти зависимости возникают потому, что каждый студент живет только в одном общежитии, и каждое общежитие взимает со всех проживающих в нем студентов одинаковую плату. Например, каждый живущий в общежитии Рэндольф-Холл платит $3200 за квартал.

Функциональные зависимости:Общежитие → Плата
НомерСтудента → Общежитие → Плата

НомерСтудента	Общежитие	Плата
100	Рэндольф	3200
150	Ингерсол	3100
200	Рэндольф	3200
250	Питкин	3100
300	Рэндольф	3200

Рисунок 2

Поскольку НомерСтудента определяет атрибут Общежитие, а Общежитие определяет атрибут Плата, то косвенным образом НомерСтудента > Плата. Такая структура функциональных зависимостей называется транзитивной зависимостью (transitive dependence), поскольку атрибут НомерСтудента определяет атрибут Плата через атрибут Общежитие.

Условия 3НФ:

- должны выполняться условия 2НФ;
- внутри каждой сущности должны отсутствовать транзитивные связи.

Ключом отношения ПРОЖИВАНИЕ является НомерСтудента, который является одиночным атрибутом, и, следовательно, отношение находится во второй нормальной форме (и Общежитие, и Плата определяются атрибутом НомерСтудента). Несмотря на это, отношение ПРОЖИВАНИЕ имеет аномалии, обусловленные транзитивной зависимостью.

Что произойдет, если мы удалим вторую строку отношения на рис. 2? Мы потеряем не только тот факт, что студент № 150 живет в Ингерсолл-Холле, но и тот факт, что проживание в этом общежитии стоит $3100. Это аномалия удаления. А как мы можем записать тот факт, что плата за проживание в Кэрригг-Холле составляет $3500? Никак, пока туда не решит вселиться хотя бы один студент. Это аномалия вставки.

НомерСтудента	Общежитие
100	Рэндольф
150	Ингерсол
200	Рэндольф
250	Питкин
300	Рэндольф

Общежитие	Плата
Рэндольф	3200
Ингерсол	3100
Рэндольф	3200
Питкин	3100
Рэндольф	3200

Рисунок 3

Чтобы удалить аномалии из отношения во второй нормальной форме, необходимо устранить транзитивную зависимость рис. 3. Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и не имеет транзитивных зависимостей.

6.7 Нормальная форма Бойса-Кодда.

Функциональные
зависимости: Преподаватель → Специальность

НомерСтудента	Специальность	Преподаватель
100	Математика	Коши
150	Психология	Юнг
200	Математика	Риман
250	Математика	Коши
300	Рэндольф	Перлс
300	Психология	Риман

Рисунок 4

Поскольку студенты могут специализироваться в нескольких областях, атрибут НомерСтудента не определяет атрибут Специальность. Более того, так как студент может иметь несколько консультантов, НомерСтудента не определяет и атрибут Преподаватель. Таким образом, НомерСтудента сам по себе не может быть ключом.

Комбинация (НомерСтудента, Специальность) определяет атрибут Преподаватель, а комбинация (НомерСтудента, Преподаватель) определяет атрибут Специальность. Следовательно, любая из этих комбинаций может быть ключом. Два или более атрибута или группы атрибутов, которые могут быть ключом, называются ключами-кандидатами (candidate keys). Тот из ключей-кандидатов, который выбирается в качестве ключа, называется первичным ключом (primary key).

Кроме ключей-кандидатов, есть еще одна функциональная зависимость, которую следует рассмотреть: атрибут Преподаватель определяет атрибут Специальность (любой из преподавателей является консультантом только по одному предмету; следовательно, зная имя преподавателя, мы можем определить специальность). Таким образом, Преподаватель является детерминантом.

По определению, отношение КОНСУЛЬТАНТ находится в первой нормальной форме. Оно также находится во второй нормальной форме, поскольку не имеет неключевых атрибутов (каждый из атрибутов является частью минимум одного ключа). Наконец, это отношение находится в третьей нормальной форме, так как не имеет транзитивных зависимостей. Тем не менее, несмотря на все это, отношение имеет аномалии модификации.

Пусть студент с номером 300 отчисляется из университета. Если мы удалим строку с информацией о студенте с номером 300, мы потеряем тот факт, что Перле является консультантом по психологии. Это аномалия удаления. Далее, как мы можем записать в базу тот факт, что Кейнс является консультантом по экономике? Никак, пока не появится хотя бы один студент, специализирующийся на экономике. Это аномалия удаления.

Ситуации, подобные только что описанной, приводят нас к определению нормальной формы Бойса-Кодда (Boyce-Codd normal form, BK/NF): отношение находится в НФБК, если каждый детерминант является ключом-кандидатом. Отношение КОНСУЛЬТАНТ не находится в НФБК, поскольку детерминант Преподаватель не является ключом-кандидатом.

НомерСтудента	Преподаватель
100	Коши
150	Юнг
200	Риман
250	Коши
300	Перлс
300	Риман

Консультант	Предмет
Коши	Математика
Юнг	Психология
Риман	Математика
Перлс	Психология

Рисунок 5

Как и в других примерах, отношение КОНСУЛЬТАНТ можно разбить на два отношения, не имеющие аномалий. Например, отношения СТУДЕНТ-КОНСУЛЬТАНТ (НомерСтудента, Преподаватель) и КОНСУЛЬТАНТ-ПРЕДМЕТ (Преподаватель, Специальность) не имеют аномалий рис. 5.

6.8 Четвертая нормальная форма.

Многозначные зависимости: НомерСтудента →→ Специальность
НомерСтудента →→ Секция

НомерСтудента	Специальность	Секция
100	Музыка	Плавание
100	Бухгалтерский учет	Плавание
100	Музыка	Теннис
100	Бухгалтерский учет	Теннис
150	Математика	Оздоровительный бег

Рисунок 6 – Отношение с многозначными зависимостями

Рассмотрим отношение СТУДЕНТ на рис. 6, которое отображает связи между студентами, специальностями и секциями. Предположим, что студенты могут иметь несколько специальностей и заниматься в нескольких различных секциях. В таком случае единственным ключом является комбинация (НомерСтудента, Специальность, Секция). Например, студентка с номером 100 специализируется на музыке и бухгалтерском учете и, кроме того, посещает секции плавания и тенниса, а студент с номером 150 специализируется только на математике и занимается бегом.

Какова связь между атрибутами НомерСтудента и Специальность? Это не функциональная зависимость, поскольку у студента может быть несколько специальностей. Одному и тому же значению атрибута НомерСтудента может соответствовать много значений атрибута Специальность. Помимо того, одному и тому же значению атрибута НомерСтудента может соответствовать много значений атрибута Секция.

Такая зависимость атрибутов называется многозначной зависимостью (multivalue dependency). Многозначные зависимости приводят к аномалиям модификации. Для начала обратите внимание на избыточность данных на рис. 6. Студентке с номером 100 посвящено четыре записи, в каждой из которых указана одна из ее специализаций и одна из посещаемых ею секций. Если бы те же данные хранились в меньшем количестве строк (скажем, было бы две строки — одна для музыки и плавания, а другая для бухгалтерского учета и тенниса), это дезориентировало бы пользователей. Получалось бы, что студентка с номером 100 плавает только тогда, когда специализируется на музыке, а в теннис играет только тогда, когда специализируется на бухгалтерском учете. Но такая интерпретация нелогична. Специальности и секции совеершенно независимы друг от друга. Поэтому, чтобы избежать таких неверных заключений, мы храним все сочетания специальностей и секций.

НомерСтудента	Специальность	Секция
100	Музыка	Лыжи
100	Музыка	Плавание
100	Бухгалтерский учет	Плавание
100	Музыка	Теннис
100	Бухгалтерский учет	Теннис
150	Математика	Оздоровительный бег

а

НомерСтудента	Специальность	Секция
100	Музыка	Лыжи
100	Бухгалтерский учет	Лыжи
100	Музыка	Плавание
100	Бухгалтерский учет	Плавание
100	Музыка	Теннис
100	Бухгалтерский учет	Теннис
150	Математика	Оздоровительный бег

б

Рисунок 7

Допустим, что студентка с номером 100 решила записаться в секцию лыж, и поэтому мы добавляем в таблицу строку [100, Музыка, Лыжи], как показано на рис. 7 а данный момент из отношения можно сделать вывод, что студентка 100 занимается лыжами только как музыкант, но не как бухгалтер. Чтобы данные имели согласованный характер, мы должны добавить столько строк, сколько имеется специальностей, и в каждой из них указать секцию лыж. Таким образом, мы должны добавить строку [100, Бухгалтерский учет, Лыжи], как показано на рис. 7 б. Это аномалия обновления: требуется слишком много модификаций, чтобы внести одно простое изменение.

Вообще говоря, многозначная зависимость существует, когда отношение имеет минимум три атрибута, причем два из них являются многозначными, а их значения зависят только от третьего атрибута. Другими словами, в отношении R (A, B, C) существует многозначная зависимость, если A многозначным образом определяет B и C, а сами B и C не зависят друг от друга. Как мы видели из предыдущего примера, НомерСтудента многозначно определяет атрибуты Специальность и Секция, но сами Специальность и Секция не зависят друг от друга.

Чтобы устранить эти аномалии, мы должны избавиться от многозначной зависимости. Мы сделаем это, создав два отношения, в каждом из которых будут храниться данные только по одному многозначному атрибуту. Результирующие отношения не будут иметь аномалий. Это отношения СТУДЕНТ-СПЕЦИАЛЬНОСТЬ (НомерСтудента, Специальность) и СТУДЕНТ-СЕКЦИЯ (НомерСтудента, Секция), приведенные на рис 8.

НомерСтудента	Специальность
100	Музыка
100	Бухгалтерский учет
150	Математика

НомерСтудента	Секция
100	Лыжи
100	Плавание
100	Теннис
150	Оздоровительный бег

Рисунок 8 – Устранение многозначной зависимости

Отношение находится в четвертой нормальной форме, если оно находится в НФБК и не имеет многозначных зависимостей.

6.9 Пятая нормальная форма.

Пятая нормальная форма (fifth normal form, 5NF) связана с зависимостями, которые имеют несколько неопределенный характер. Речь здесь идет об отношениях, которые ожно разделить на несколько более мелких отношений, как мы это делали выше, но затем невозможно восстановить.

Условия, при которых возникает эта ситуация, не имеют ясной, интуитивной интерпретации. Нам неизвестно, каковы следствия таких зависимостей; мы не знаем даже, есть ли у них какие-либо практические следствия.

6.10 Доменно-ключевая нормальная форма.

В 1981 г. Фагин опубликовал важную статью, в которой он определил доменно-ключевую нормальную форму (domain/key normal form, DKNFI. Он показал, что отношение в ДКНФ не имеет аномалий модификации и, более того, любое отношение, не имеющее аномалий модификации, должно находиться в ДКНФ.

Это открытие положило конец введению нормальных форм, и теперь в нормальных формах более высокого порядка нет необходимости — по крайней мере, для устранения аномалий модификации.

6.11 Разработка базы данных

В основу проектирования БД должны быть положены представления конечных пользователей конкретной организации — концептуальные требования к системе. Именно конечный пользователь в своей работе принимает решения с учетом получаемой в результате доступа к базе данных информации. От оперативности и качества этой информации будет зависеть эффективность работы организации. Данные, помещаемые в базу данных, также предоставляет конечный пользователь.

В результате анализа поставленной заказчиком задачи и обработки требований конечных пользователей составляется концептуальная модель. Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среде хранения.

При разработке логической модели базы данных, прежде всего необходимо решить, какая модель данных наиболее подходит для отображения конкретной концептуальной модели предметной области. Коммерческие системы управления базами данных поддерживают одну из известных моделей данных или некоторую их комбинацию. Почти что все популярные системы для персональных компьютеров поддерживают реляционную модель данных.

Отображение концептуальной модели данных на реляционную модель производится относительно просто. Каждый прямоугольник концептуальной модели отображается в одно отношение, которое отражает представление пользователя в удобном для него табличном формате. Построение реляционной базы данных основано на нормализации отношений.

Нормализация отношений – это процесс построения оптимальной структуры таблиц и связей в реляционной базе данных. В процессе нормализации элементы данных группируются в таблицы, представляющие объекты и их взаимосвязи. Теория нормализации основана на том, что создается определенный набор таблиц, обладающих лучшими свойствами при включении, модификации и удалении данных, чем все остальные наборы таблиц, с помощью которых могут быть представлены те же данные.

Здесь также решается вопрос ликвидации избыточности информации. То есть концептуальные требования, используемые несколькими структурными подразделениями, сводятся в одну таблицу с одновременным добавлением ключей для перехода к другим таблицам (для других структурных подразделений). Таким образом добиваются существенного сокращения требуемых объемов памяти. На этом этапе также решается вопрос о том, какие таблицы будут справочниками, т.е. информация в этих таблицах не изменяется или изменяется очень медленно.

Следует иметь в виду, что чрезмерное увеличение количества таблиц приводит к потере общей идеи создания БД и сама БД становится трудной для понимания и управления.

Оптимальное количество таблиц для БД объема предприятия должна быть не более 50. Всего существует 5 нормальных форм. На практике, как правило, при создании приложения БД в объеме предприятия используется первые 3 нормальные формы.

Рассмотрим последовательность проектирования базы данных, которые должны обеспечить необходимую независимость данных и выполнение эксплуатационных требований (пожеланий пользователей).

Этап 1. Определение сущностей

Этап 2. Определение взаимосвязей между сущностями

Определить для включенных в модель сущностей взаимосвязи в соответствии с ранее описанными рекомендациями. Представить на этом этапе информационную модель задачи в виде диаграммы «Сущность – Связь».

Этап 3. Задание первичных и альтернативных ключей, определение атрибутов сущностей

Для каждой сущности нужно определить атрибуты, которые будут храниться в базе данных. При этом необходимо учитывать тот факт, что при переходе от логической к физической модели данных, может произойти усечение числа объектов. На самом деле, как правило, значительное число данных, необходимых пользователю, может быть достаточно легко подсчитано в момент вывода информации. В то же время, в связи с изменением алгоритмов расчета или исходных величин, некоторые расчетные показатели приходится записывать в базу данных, чтобы гарантированно обеспечить фиксацию их значений.

Атрибуты, включаемые в состав БД для рассматриваемой модели, нужно привести в таблице под названием «Атрибуты и первичные ключи сущностей информационной модели». Информационную модель после третьего этапа проектирования привести на схеме «Взаимосвязи между атрибутами сущностей».

Этап 4. Приведение модели к требуемому уровню нормальной формы

В процессе нормализации элементы данных группируются в таблицы. Введение нормализации отношений при разработке информационной модели обеспечивает использование минимального объема физической памяти, то есть записанной на каком-либо носителе БД и ее максимальное быстродействие, что впрямую отражается на качестве функционирования информационной системы. Нормализация информационной модели выполняется в несколько этапов.

Данные, представленные в виде двумерной таблицы, являются первой нормальной формой реляционной модели данных.

Первый этап нормализации заключается в образовании двумерной таблицы, содержащей все необходимые атрибуты информационной модели, и в выделении ключевых атрибутов. Очевидно, что полученная весьма внушительная таблица будет содержать очень разнородную информацию. В этом случае будут наблюдаться аномалии включения, обновления и удаления данных, так как при выполнении этих действий придется уделить внимание данным (вводить или заботиться о том, чтобы они не были стерты), которые не имеют к текущим действиям никакого отношения.

Привести таблицы, в которых описаны сущности и атрибуты первой нормальной формы.

Отношение задано во второй нормальной форме, если оно является отношением в первой нормальной форме и каждый атрибут, не являющийся первичным атрибутом в этом отношении, полностью зависит от любого возможного ключа этого отношения.

Если все возможные ключи отношения содержат по одному атрибуту, то это отношение задано во второй нормальной форме, так как в этом случае все атрибуты, не являющиеся первичными, полностью зависят от возможных ключей. Если ключи состоят более чем из одного атрибута, отношение, заданное в первой нормальной форме, может не быть отношением во второй нормальной форме. Приведение отношений ко второй нормальной форме заключается в обеспечении полной функциональной зависимости всех атрибутов от ключа за счет разбиения таблицы на несколько, в которых все имеющиеся атрибуты будут иметь полную функциональную зависимость от ключа этой таблицы. В процессе приведения модели ко второй нормальной форме в основном исключаются аномалии дублирования данных.

Обосновать приведение своей базы данных ко второй нормальной форме. Привести таблицы, в которых описаны сущности и атрибуты второй нормальной формы.

.

Отношение задано в третьей нормальной форме, если оно задано во второй нормальной форме и каждый атрибут этого отношения, не являющийся первичным, не транзитивно зависит от каждого возможного ключа этого отношения.

Транзитивная зависимость выявляет дублирование данных в одном отношении.

Преобразование в третью нормальную форму происходит за счет разделения исходного отношения на два.

Основные правила, которым нужно следовать при проектировании базы данных:

- Исключать повторяющиеся группы — для каждого набора связанных атрибутов создать отдельную таблицу и снабдить ее первичным ключом. Выполнение этого правила автоматически приведет ко второй нормальной форме.
- Исключать избыточные данные — если атрибут зависит только от части составного ключа, переместить атрибут в отдельную таблицу. Это правило помогает избежать потери одних данных при удалении каких-то других. Везде, где возможно использование идентификаторов вместо описания, выносить в отдельную таблицу список идентификаторов с пояснениями к ним.
- Исключать столбцы, которые не зависят от ключа — если атрибуты не вносят свою лепту в описание ключа, переместить их в отдельную таблицу.

Рекомендуется использовать вместо естественных атрибутов коды в следующих случаях:

- В предметной области может наблюдаться синомия, то есть естественный атрибут отношения не обладает свойством уникальности. Например, среди сотрудников фирмы могут быть однофамильцы или даже полные тезки. В этом случае решить проблему помогает уникальный табельный номер.
- Если отношение участвует во многих связях, то для их отображения создается несколько таблиц, в каждой из которых повторяется идентификатор отношения. Для того чтобы не использовать во всех таблицах длинный естественный атрибут объекта, можно применять более короткий код. Это также будет способствовать повышению быстродействия вашей системы.
- Если естественный атрибут может изменяться во времени (например, фамилия), то это может вызвать очень большие сложности при эксплуатации системы. Использование неизменяемого кода (табельного номера) позволит избежать этих сложностей.

Обосновать приведение своей базы данных к третьей нормальной форме. Привести измененные таблицы, в которых описаны сущности и атрибуты третьей нормальной формы. Привести диаграмму взаимосвязи между атрибутами сущностей после нормализации модели.

Этап 5.Физическое описание модели

На этом этапе нужно составить проекты таблиц, которые будут в дальнейшем реализовываться в конкретной СУБД. Структуру разработанной базы данных отразить в таблице.

На этапе физического проектирования обеспечить безошибочность и точность информации, хранящейся в базе данных. Это называется обеспечением целостности базы данных.

Обеспечением целостности базы данных называется система мер, направленных на поддержание правильности данных в базе в любой момент времени.

В СУБД целостность данных обеспечивается набором специальных предложений, называемых ограничениями целостности.

Ограничения целостности — это набор определенных правил, которые устанавливают допустимость данных и связей между ними.

Ограничения целостности в большинстве случаев определяются особенностями предметной области. Ограничения целостности могут относиться к разным объектам базы данных: атрибутам (полям), записям, отношениям, связям между ними и т. п. Для полей могут использоваться следующие виды ограничений:

- Тип и формат поля автоматически допускают ввод только данных определенного типа. Выбор типа поля Date в формате ДД.ММ.ГГ позволит пользователю ввести только шесть чисел. При этом первая пара цифр не сможет превысить в лучшем случае значения 31, а вторая — 12.
- Задание диапазона значений, как правило, используется для числовых полей. Диапазон допустимых значений может быть ограничен с двух сторон (закрытый диапазон), а может с какой-то одной: верхней или нижней (открытый диапазон).
- Недопустимость пустого поля позволяет избежать появления в БД «ничейных» записей, в которых пропущены какие-либо обязательные атрибуты.
- Задание списка значений позволяет избежать излишнего разнообразия данных, если его можно ограничить. Например, для указания типа кузова мы можем ограничить фантазию пользователя только общепринятыми названиями: седан, кабриолет и т. д.
- Проверка на уникальность значения какого-то поля позволяет избежать записей-дубликатов. Вряд ли будет удобно в справочнике клиентов иметь несколько записей для одного и того же лица.

На пятом этапе также предусматриваются меры по обеспечению ссылочной целостности, то есть установление между таблицами не противоречивых взаимосвязей. Установление не противоречивых взаимосвязей и обеспечение достоверности в данных в любой момент времени является главной и самой трудоемкой задачей.

Для реализации ограничений целостности, имеющих отношение к записи, таблицам или связям между ними, в СУБД могут использовтся триггеры.