29. Базы данных. Нормальные формы. Нормальная форма Бойса-Кодда. Многофункциональная зависимость. Денормализация.

1. Базы данных.

Базой данных (БД) называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов - атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов).

Строки таблицы называются записями. Все записи таблицы имеют одинаковую структуру - они состоят из полей (элементов данных), в которых хранятся атрибуты объекта (рис. 1). Каждое поле записи содержит одну характеристику объекта и представляет собой заданный тип данных (например, текстовая строка, число, дата). Для идентификации записей используется первичный ключ. Первичным ключом называется набор полей таблицы, комбинация значений которых однозначно определяет каждую запись в таблице.

Рис. 1. Названия объектов в таблице

1.2. Этапы проектирования реляционной базы данных

Основная причина сложности проектирования базы данных заключается в том, что объекты реального мира и взаимосвязи между ними вовсе не обязаны иметь и, как правило, не имеют структуры, согласованной с реляционной моделью данных. Разработчик при проектировании должен придумать представление для реальных объектов и их связей в терминах таблиц, полей, атрибутов, записей и т. п., то есть в терминах абстракций реляционной модели данных. Поэтому в данном контексте термин «проектирование» можно понимать и как процесс, результатом которого является проект, и как процесс, результатом которого является проекция.

Разработка эффективной базы данных состоит из нескольких этапов. Процесс разработки БД начинается с анализа требований. Проектировщик на этом этапе разработки должен найти ответы на следующие вопросы: какие элементы данных должны храниться, кто и как будет к ним обращаться.

На втором этапе создается логическая структура БД. Для этого определяют, как данные будут сгруппированы логически. Структура БД на этом этапе выражается в терминах прикладных объектов и отношений между ними.

На заключительном (третьем) этапе логическая структура БД преобразуется в физическую с учетом аспектов производительности. Элементы данных на этом этапе получают атрибуты и определяются как столбцы в таблицах выбранной для реализации БД СУБД.

Рассмотрим применение концепции реляционных баз данных на практике. Представим себе деятельность туристической фирмы. Очевидно, что для ее работы необходимо хранить и отслеживать определенный набор информации о клиентах данной турфирмы (туристах), о предлагаемых им турах, об оформлении и оплате путевок. Это можно делать в обычной бумажной тетради, но со временем поиск нужных записей и финансовая отчетность будут представлять собой довольно рутинную, длительную работу.

1.2.2. Логическая модель

ER-диаграммы

Общим способом представления логической модели БД является построение ER-диаграмм (Entity-Relationship - сущность-связь). В этой модели сущность определяется как дискретный объект, для которого сохраняются элементы данных, а связь описывает отношение между двумя объектами.

В примере менеджера турфирмы имеются 5 основных объектов:

- Туристы
- Туры
- Путевки
- Сезоны
- Оплаты

Отношения между этими объектами могут быть определены простыми терминами:

- Каждый турист может купить одну или несколько (много) путевок.
- Каждой путевке соответствует ее оплата (оплат может быть и несколько, если путевка, например, продана в кредит).
- Каждый тур может иметь несколько сезонов.
- Путевка продается на один сезон одного тура.

Эти объекты и отношения могут быть представлены ER-диаграммой, как показано на рис2.

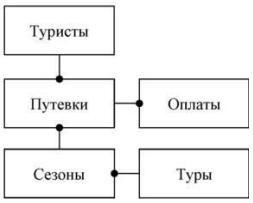


Рис. 2. ER-диаграмма для приложения БД менеджера турфирмы

Объекты, атрибуты и ключи

Далее модель развивается путем определения атрибутов для каждого объекта. Атрибуты объекта - это элементы данных, относящиеся к определенному объекту, которые должны сохраняться. Анализируем составленный словарь данных, выделяем в нем объекты и их атрибуты, расширяем словарь при необходимости.

Реляционная модель характеризуется использованием ключей и отношений. Существует отличие в контексте реляционной базы данных терминов relation (отношение) и relationship (схема данных). Отношение рассматривается как неупорядоченная, двумерная таблица с несвязанными строками. Схема данных формируется между отношениями (таблицами) через общие атрибуты, которые являются ключами.

Существует несколько типов ключей, и они иногда отличаются только с точки зрения их взаимосвязи с другими атрибутами и отношениями. Первичный ключ уникально идентифицирует строку в отношении (таблице), и каждое отношение может иметь только один первичный ключ, даже если больше чем один атрибут является уникальным. В некоторых случаях требуется более одного атрибута для идентификации строк в отношении. Совокупность этих атрибутов называется составным ключом. В других случаях первичный ключ должен быть специально создан (сгенерирован). Например, в отношение «Туристы» имеет смысл добавить уникальный идентификатор туриста (код туриста) в виде первичного ключа этого отношения для организации связей с другими отношениями БД.

Другой тип ключа, называемый внешним ключом, существует только в терминах схемы данных между двумя

отношениями. Внешний ключ в отношении - это атрибут, который является первичным ключом (или частью первичного ключа) в другом отношении. Это - распределенный атрибут, который формирует схему данных между двумя отношениями в БД.

Первичный ключ (сокращенно PK - primary key) - столбец, значения которого во всех строках различны. Первичные ключи могут быть логическими (естественными) и суррогатными (искусственными).

2. Нормальные формы.

Процесс нормализации состоит в пошаговом построении БД в нормальной форме (НФ).

1. Первая нормальная форма (1НФ) очень проста. Все таблицы БД должны удовлетворять единственному требованию - каждая ячейка в таблицах должна содержать атомарное значение, другими словами, хранимое значение в рамках предметной области приложения БД не должно иметь внутренней структуры, элементы которой могут потребоваться приложению.
2. Вторая нормальная форма (2НФ) создается тогда, когда удалены все частичные зависимости из отношений БД. Если в отношениях не имеется никаких составных ключей, то этот уровень нормализации легко достигается.
3. Третья нормальная форма (3НФ) БД требует удаления всех транзитивных зависимостей.
4. Четвертая нормальная форма (4НФ) создается при удалении всех многозначных зависимостей.

БД нашего примера находится в 1НФ, так как все поля таблиц БД атомарные по своему содержанию. Наша БД также находится и во 2НФ, так как мы искусственно ввели в каждую таблицу уникальные коды для каждого объекта (Код Туриста, Код Путевки и т. д.), за счет чего и добились 2НФ для каждой из таблиц БД и всей базы данных в целом. Осталось разобраться с третьей и четвертой нормальными формами.

Обратите внимание, что они существуют только относительно различных видов зависимостей атрибутов БД. Есть зависимости - нужно стоить НФ БД, нет зависимостей - БД и так находится в НФ. Но последний вариант практически не встречается в реальных приложениях.

3. Нормальная форма Бойса-Кодда.

Нормальная форма Бойса — Кодда (BCNF) — одна из возможных нормальных форм таблицы реляционной базы данных. Это модификация третьей нормальной формы

Определение

Отношение находится в BCNF, если оно находится в третьей нормальной форме и в нём отсутствуют функциональные зависимости атрибутов первичного ключа от неключевых атрибутов. Ситуация, когда отношение будет находиться в 3NF, но не в BCNF, возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут. На практике такая ситуация встречается достаточно редко, для всех прочих отношений 3NF и BCNF эквивалентны.

Первый пример

Пример приведения таблицы к нормальной форме Бойса — Кодда

Исходная таблица:

Номер клиента	Дата собеседования	Время собеседования	Номер комнаты	Номер сотрудника
C345	13.10.03	13.00	103	A138
C355	13.10.03	13.05	103	A136
C368	13.09.03	13.00	102	A154
C366	13.09.03	13.30	105	A207

В результате приведения к форме Бойса—Кодда получаются две таблицы:

Номер клиента	Дата собеседования	Время собеседования	Номер Сотрудника
C345	13.10.03	13.00	A138
C355	13.10.03	13.05	A136
C368	13.09.03	13.00	A154
C366	13.09.03	13.30	A207

Дата собеседования	Номер сотрудника	Номер комнаты
13.10.03	A138	103
13.10.03	A136	103
13.09.03	A154	102
13.09.03	A207	105

4. Многофункциональная зависимость.

Для примера предположим, что для каждого туриста должны храниться несколько контактных телефонов (домашний, рабочий, сотовый и пр., что весьма характерно на практике), а не один, как в примере. Получаем многозначную зависимость ключа - «Код туриста» и атрибутов «Тип телефона» и «Телефон», в этой ситуации ключ перестает быть ключом. Что делать? Проблема решается также путем разбиения схемы отношения на 2 новые схемы. Одна из них должна представлять информацию о телефонах (отношение «Телефоны»), а вторая о туристах (отношение «Туристы»), которые связываются по полю «Код туриста». «Код туриста» в отношении «Туристы» будет первичным ключом, а в отношении «Телефоны» - внешним.

Одни и те же данные могут группироваться в таблицы (дела) разными методами. Группировка атрибутов в отношениях должна быть рациональной (т. е. дублирование данных д.б. наименьшим) и упрощающей процедуры их обработки и обновления. Устранение избыточности данных является одной из важных задач проектирования баз данных и обеспечивается нормализацией.

Нормализация таблиц (отношений) — это формальный аппарат ограничений на формирование таблиц (отношений), который позволяет убрать дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудовые затраты на ведение (ввод, корректировку) базы данных. Процесс нормализации заключается в разложении (декомпозиции) начальных отношений БД на более обыкновенные дела. Любая ступень этого процесса приводит схему отношений в поочередные обычные формы. Для каждой ступени нормализации имеются наборы ограничений, которым должны удовлетворять дела БД. Нормализация позволяет удалить из таблиц базы сверхизбыточную неключевую информацию.

Сначала вспомним некие понятия:

Обычный атрибут — это атрибут, значения которого неразделимы. Другими словами, в таблице нет полей типа ФИО либо Адресок — они разложены на поля Фамилия, Имя, Отчество в первом случае и на поля Индекс, Город и т. д. во 2-м.

Непростой (составной) атрибут выходит методом соединения нескольких атомарных атрибутов, по другому его именуют вектором либо агрегатом данных.

Определение многофункциональной зависимости: Пусть Хи Y атрибуты некого дела. Если в хоть какой момент времени произвольному значению X соответствует единственное значение Y, то Y функционально находится в зависимости от X (X→Y)

Если ключ является составным, то хоть какой атрибут должен зависеть от ключа в целом, но не может находиться в многофункциональной зависимости от какой-нибудь части составного ключа, т.е. многофункциональная зависимость имеет вид (X1, X2, ..., X)→Y.

Многофункциональная зависимость может быть полной либо неполной.

Неполной зависимостью именуется зависимость неключевого атрибута от части составного ключа.

Полной многофункциональной зависимостью именуется зависимость неключевого атрибута от всего составного ключа, а не от его частей.

Определение транзитивной многофункциональной зависимости: Пусть X, Y, Z— три атрибута некого дела. При этом X→Y и Y→Z, но оборотное соответствие отсутствует,

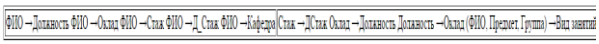
другими словами Y не находится в зависимости от Z, а X не находится в зависимости от Y. Тогда молвят, что Z транзитивно находится в зависимости от X.

Определение неоднозначной зависимости: Пусть X и Y атрибуты некого дела. Атрибут Y неоднозначно находится в зависимости от атрибута X, если. каждому значению X соответствует огромное количество значений Y, не связанных с другими атрибутами из дела. Неоднозначные зависимости могут носить нрав «один ко многим» (1:M), «многие к одному» (M:1) либо «многие ко многим» (M:M), обозначаемые соответственно: X=>Y, Y<=X и XY. К примеру, педагог ведет несколько предметов, а каждый предмет может вестись несколькими педагогами, тогда имеет место зависимость ФИО Предмет.

Разглядим последующий пример: Представим, что для учебной части факультета создается БД о педагогах, которая включает последующие атрибуты: ФИО - фамилия и инициалы педагога (совпадения фамилий и инициалов исключаются). Должность - должность, занимаемая педагогом. Оклад- оклад педагога. Загрузка... Стаж - преподавательский стаж. Д_Стаж - прибавка за стаж. Кафедра - номер кафедры, на которой считается педагог. Предмет - заглавие предмета (дисциплины), читаемого педагогом. Группа - номер группы, в какой педагог проводит занятия. Вид занятия - вид занятий, проводимых педагогом в учебной группе. Начальное отношение

ФИО	Должность	Оклад	Стаж	Д_Стаж	Кафедра	Предмет	Группа	Вид занятия
Иванов И. М.	Педагог					БД	АС-21	Практика
Иванов И. М.	Педагог					ОС	АС-22	Практика
Петров М. И.	Ст. педагог					БД	АС-21	Лекция
Петров М. И.	Ст. педагог					Архитектура	АС-21	Практика
Сидоров Н. Г.	Педагог					ОС	АС-22	Лекция
Сидоров Н. Г.	Педагог					Архитектура	АС-21	Лекция
Егоров В. В.	Педагог					Философия	ПС-22	Лекция

Итак, выделим в нашем отношении все виды зависимостей: многофункциональные (полные и неполные), неоднозначные, транзитивные. Выявление зависимостей меж атрибутами нужно для приведения данных к обычным формам, т.е. нормализации данных. Многофункциональные зависимости: ФИО→Кафедра, ФИО→Должность, Должность→ Оклад, ФИО→Предмет. Также в нашем отношении ключ является составным и состоит из атрибутов (ФИО, Предмет, Группа). Неполная многофункциональная зависимость: (ФИО, Предмет, Группа)→Должность, т.к. атрибут Должность находится в многофункциональной зависимости от атрибута ФИО, являющегося частью ключа. Полная многофункциональная зависимость: (ФИО, Предмет, Группа)→Вид занятия. Транзитивная зависимость: ФИО→Должность→Оклад, ФИО→Стаж→Д_Стаж. Таким макаром, выявились последующие зависимости, в базе выделения которых лежало условие, что один педагог в одной группе может проводить только один вид занятий (лекции либо практические занятия):



(ФИО →Должность ФИО →Оклад ФИО →Стаж ФИО →Д_Стаж ФИО →Кафедра Стаж →ДСтаж Оклад →Должность Должность →Оклад (ФИО, Предмет, Группа) →Вид занятий)

К выделению этих многофункциональных зависимостей приводят последующие суждения. Фамилия, Имя и Отчество у педагогов факультета уникальны. Каждому педагогу совершенно точно соответствует его стаж, т. е. имеет место многофункциональная зависимость ФИО →Стаж. Оборотное утверждение ошибочно, потому что однообразный стаж может быть у различных педагогов. Каждый педагог имеет определенную добавку за стаж, т. е. имеет место многофункциональная зависимость ФИО→Д_Стаж, но оборотная многофункциональная зависимость отсутствует, потому что одну и ту же прибавку могут иметь несколько педагогов. Каждый педагог имеет определенную должность (преп., ст. преп., доцент, доктор), но одну и ту же должность могут иметь несколько педагогов, т.е. имеет место многофункциональная зависимость ФИО→Должность, а оборотная многофункциональная зависимость отсутствует. Каждый педагог является сотрудником одной и только одной кафедры. Потому имеет место многофункциональная зависимость ФИО→Кафедра. С другой стороны, на каждой кафедре много педагогов, потому оборотной многофункциональной зависимости нет. Каждому педагогу соответствует определенный оклад, который схож для всех преподавателей с схожими должностями, что учитывается зависимостями ФИО→Оклад и Должность→Оклад. Нет схожих окладов для различных должностей, потому имеет место многофункциональная зависимость Оклад→Должность. Один и тот же педагог в одной группе по различным предметам может проводить различные виды занятий. Определение вида занятий, которые

проводит педагог, нереально без указания предмета и группы, потому имеет место многофункциональная зависимость (ФИО, Предмет, Группа) →Вид Занятия. Не были выделены зависимости меж атрибутами ФИО, Предмет и Группа, так как они образуют составной ключ и не учитываются в процессе нормализации начального дела. Дальше разглядим процесс нормализации. Как было сказано выше, процесс проектирования БД с внедрением способа обычных форм заключается в поочередном переводе отношений из первой обычной формы в обычные формы более высочайшего порядка по определенным правилам. Любая последующая обычная форма ограничивает определенный тип многофункциональных зависимостей, избавляет надлежащие аномалии при выполнении операций над отношениями БД и сохраняет характеристики предыдущих обычных форм. Выделяют последующую последовательность обычных форм: 1-ая обычная форма (1НФ); 2-ая обычная форма (2НФ); 3-я обычная форма (3НФ).

5. **Денормализация**.

Денормализация - это вовсе не незаконченная нормализация. Денормализация - процесс творческий и вряд ли формализуемый.

Для чего нужна денормализация

Денормализацию базы проводят обычно для повышения производительности, реже - для облегчения жизни программистам, разрабатывающим приложения, работающие с этой базой данных.

Рассмотрим некоторые распространенные ситуации, в которых денормализация может оказаться полезна:

Большое количество соединений таблиц

В запросах к полностью нормализованной базе нередко приходится соединять до десятка, а то и больше, таблиц. А каждое соединение - операция весьма ресурсоемкая. Как следствие, такие запросы очень аппетитно кушают ресурсы сервера и выполняются медленно.

В такой ситуации поможет денормализация путем сокращения количества таблиц. Лучше объединять в одну несколько таблиц, имеющих небольшой размер, содержащих редко изменяемую (как часто говорят, условно-постоянную, или нормативно-справочную) информацию, причем информацию, по смыслу тесно связанную между собой.

Расчетные значения

Зачастую медленно выполняются и потребляют много ресурсов запросы, в которых производятся какие-то сложные вычисления, особенно при использовании группировок и агрегатных функций (Sum, Max и т.п.). Иногда имеет смысл добавить в таблицу 1-2 дополнительных столбца, содержащих часто используемые (и сложно вычисляемые) расчетные данные.

Длинные поля

Если у нас в базе данных есть большие таблицы, содержащие длинные поля (Blob, Long и т.п.), то серьезно ускорить выполнение запросов к такой таблице мы сможем, если вынесем длинные поля в отдельную таблицу. Хотим мы, скажем, создать в базе каталог фотографий, в том числе хранить в blob-полях и сами фотографии (профессионального качества, с высоким разрешением, и соответствующего размера). С точки зрения нормализации абсолютно правильной будет такая структура таблицы:

* ID фотографии
* ID автора
* ID модели фотоаппарата
* сама фотография (blob-поле).

Естественно, есть еще отдельные таблицы, содержащие сведения об авторах (ключевое поле - ID автора) и о моделях фотоаппаратов (ключевое поле - ID модели). А сейчас представим, сколько времени будет работать запрос, подсчитывающий количество фотографий, сделанных каким-либо автором...

Правильным решением (хотя и нарушающим принципы нормализации) в такой ситуации будет создать еще одну таблицу, состоящую всего из двух полей - ID фотографии и blob-поле с самой фотографией. Тогда выборки из основной таблицы (в которой огромного blob-поля сейчас уже нет) будут идти моментально, ну а когда захотим посмотреть саму фотографию - что ж, подождем...

Как правильно проводить денормализацию

Проводя денормализацию, мы неизбежно создаем в базе данных избыточные, дублирующиеся данные. Поэтому перед разработчиками сразу возникает задача обеспечить непротиворечивость (а чаще - идентичность) дублирующихся данных.

В Oracle это делается достаточно просто - через механизм триггеров. К примеру, при добавлении вычисляемого поля на каждый из столбцов, от которых вычисляемое поле зависит, вешается триггер, вызывающий единую (это важно!) хранимую процедуру, которая и записывает нужные данные в вычисляемое поле. Надо только не пропустить ни один из столбцов, от которых зависит вычисляемое поле.

Подведем итоги. При денормализации важно сохранить баланс между повышением скорости работы базы и увеличением риска появления противоречивых данных, между облегчением жизни программистам, пишущим Select'ы, и усложнением задачи тех, кто обеспечивает наполнение базы и обновление данных. Поэтому проводить денормализацию базы надо очень аккуратно, очень выборочно, только там, где без этого никак не обойтись.