Этапы проектирования реляционных баз данных.

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. Можно выделить следующие этапы проектирования.

1. Системный анализ и словесное описание информационных объектов предметной области.
2. Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некой инфологической, например, ER-модели.
3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели.
4. Физическое проектирование БД, то есть выбор способа размещения БД на внешних носителях и создание файла БД в какой-либо СУБД.

Первый этап, системный анализ, должен заканчиваться подробным описанием информации об объектах, формулировкой задач с кратким описанием алгоритмов их решения, описанием входных и выходных документов.

Между вторым и третьим этапами необходимо решить, с помощью какой СУБД будет реализовываться проект.

Для ER-модели существует алгоритм ее преобразования в реляционную модель данных:

1. Каждой сущности ставится в соответствие отношение реляционной модели данных.
2. Каждый атрибут сущности становится атрибутом соответствующего отношения. Для каждого атрибута задается тип данных и обязательность (или необязательность) данного атрибута.
3. Ключевой атрибут сущности становится первичным ключом соответствующего отношения.
4. Если в ER-модели присутствуют связи «один-ко-многим», то в каждое отношение, соответствующее подчиненной сущности, добавляется атрибут основной сущности, и этот атрибут становится внешним ключом. Для определения обязательного (необязательного) типа связи у атрибута, соответствующего внешнему ключу, устанавливается обязательность (необязательность).
5. Если в ER-модели присутствуют связи «один-к-одному», то необходимо учесть, что в реляционной модели такая связь может быть организована только между ключевыми атрибутами отношений, которые должны иметь одинаковые названия. Если в ER-модели присутствуют связи «многие-комногим», то для перехода к реляционной модели данных (где такие связи не поддерживаются) вводится дополнительное связующее отношение. Оно связано с каждым исходным связью «один-ко-многим», а его атрибутами являются первичные ключи связываемых отношений.

В результате выполнения даталогического проектирования должна быть разработана схема БД, то есть совокупность отношений, которые моделируют объекты БД и связи между ними. Кроме того, должны быть получены: описание концептуальной схемы БД в терминах выбранной СУБД, описание внешних моделей, описание правил и разработка процедур поддержки целостности БД.

Связь «многие ко многим»

Так как отношения многие ко многим могут скрыть другие бизнес правила или ограничения, они должны быть полностью исследованы на на уровне ключевой или атрибутивной модели.

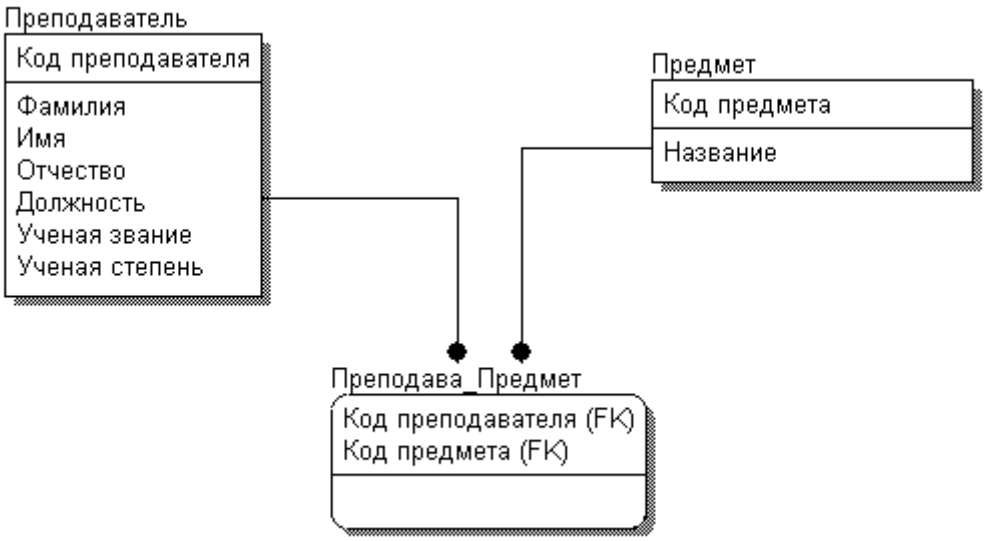
Это обусловлено тем, что отношение многие ко многим на ранних стадиях моделирования идентифицируется неправильно, на самом деле представляя два или несколько случаев отношений один-ко-многим между связанными сущностями.

Связь «многие ко многим» может быть преобразована к типу «один ко многим» либо вручную либо автоматически.

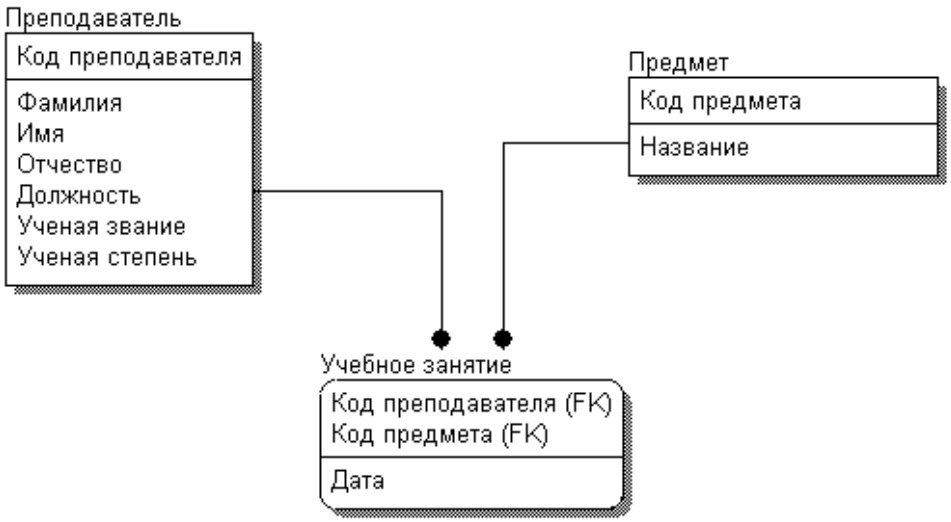
Автоматическое преобразование может быть выполнено при переходе на физический уровень. Для автоматического преобразования связи следует во вкладку General диалога Model, ModelProperty включить опцию Many-To-Many Relationships with Association Table. Преобразование связи включает создание новой таблицы и двух новых связей «один ко многим» от старых к новой таблице. При этом имя новой таблице присваивается автоматически как Имя1_Имя2**.** Автоматического решения проблемы связи многие-ко-многим не всегда оказывается достаточно.

Для принудительного преобразования связи «многие ко многим» на логическом уровне необходимо щелкнуть по связи правой кнопкой мыши и выбрать пункт меню Create Association Table. Many-To-Many Relationship Transform Wizard. Диалог Many-To-Many Relationship Transform Wizard предлагает 4 шага для преобразования связи. Для перехода к следующему шагу надо щелкнуть по кнопке Next (Далее). На втором и третьем шаге следует задать имя вновь создаваемой таблицы и имя преобразования.

На рисунке показан пример связи «многие ко многим». Преподаватель может преподать разные дисциплины, дисциплина может преподаваться разными преподавателями.



В примере таблица **Преподават_Предмет** имеет смысл учебного занятия, поэтому ее следует переименовать согласно бизнес-логике в «УЧЕБНОЕ ЗАНЯТИЕ».Один и тот же предмет может много раз читаться преподавателем, например в разное время, поэтому для того, чтобы идентифицировать учебное занятие, необходимо в состав первичного ключа сущности «УЧЕБНОЕ ЗАНЯТИЕ» добавить дополнительный атрибут, например дату проведения занятия.



Распространенные ошибки при моделировании сущностей и выборе ключей

Этот раздел, посвященный распространенным ошибкам при моделировании, не претендует на полноту. Его цель – указать на наиболее распространенные ошибки, которые возникают у разработчиков моделей.

Моделирование ролей

Что подразумевается под моделированием ролей? Во время рабочих сессий пользователи могут сказать вам, что им необходимо хранить информацию о сотрудниках. Возникает искушение создать сущность СОТРУДНИК. Более тщательный анализ информации, представляющей интерес для корпорации, например, такой как имя, адрес и номер социального страхования показывает, что эти значения не зависят от сущности СОТРУДНИК. Для конкретного СОТРУДНИКА значение атрибута ИМЯ не зависит от сущности СОТРУДНИК. Это легко понять, если задуматься о том, что ваше имя остается вашим именем вне зависимости от того, являетесь ли вы СОТРУДНИКОМ или нет.

Перегрузка сущностей

Перегруженными являются сущности, содержащие информацию более чем об одном концептуальном объекте. Если некоторые атрибуты сущности описывают одну и ту же концепцию, такие сущности следует проверить. Перегруженные сущности имеют значения не для каждого из атрибутов.

Иногда эксперты из разных предметных областей в корпорации используют имя сущности, которое звучит и пишется одинаково, но имеет разный смысл для разных экспертов. Единственный способ убедится, что одинаковые имена описывают одинаковые объекты, это проверка описаний. Убедитесь, что сущность содержит данные, описывающие единственную концепцию.

Например, сущность ОБОРУДОВАНИЕ может иметь совершенно разное значение для подразделений информационных технологий и для отдела средств массовой информации и коммуникаций.

Избыточные сущности

Избыточными являются сущности, имеющие различные имена, но содержащие информацию о сходных концепциях. Английский язык включает много слов для представления одних и тех же вещей. Один из способов обнаружить такие сущности – это поиск сущностей, содержащих сходные атрибуты. Сравните описания каждой из таких сущностей, чтобы определить, не представляют ли они сходные концепции. Избыточные сущности часто появляются в результате тенденции к моделированию ролей в качестве сущностей.

Например, сущности УПРАВЛЕНЕЦ и СОТРУДНИК могут содержать сходную информацию, поскольку обе являются ролями, которые может играть экземпляр сущности ПЕРСОНА.

Выбор неправильного первичного ключа

Выбор неправильного первичного ключа означает, что вы выбрали первичный ключ, не выдерживающий тестирования. Распространенными ошибками, связанными с первичным ключом, являются:

- Не уникальность: первичный ключ не является уникальным для каждого из экземпляров. Например, разработчик модели может считать, что номер социального страхования является уникальным для каждой ПЕРСОНЫ. Однако номер социального страхования может

повторно использоваться в том случае, если первоначальный его владелец скончался.

- Требуемое значение/неопределенность: первичный ключ не имеет значения для некоторых из экземпляров. Например, не каждый экземпляр сущности ПЕРСОНА будет иметь номер социального страхования. Иностранцы и маленькие дети – вот две категории людей, у которых он будет отсутствовать.

Использование неудачных имен сущностей

Непонятные, неоднозначные или неточные имена затрудняют для новых пользователей и команд разработчиков повторное использование или расширение существующей модели.

Не используйте аббревиатуры или акронимы в качестве части имени. Аббревиатуры и акронимы открыты для неправильной интерпретации и даже могут иметь разное значение в разных предметных областях.

Не включайте месторасположение в качестве части имени. Как правило, вам неизбежно потребуется и другое месторасположение. Имя с указанием расположения является признаком того, что вы моделируете конкретный экземпляр вместо класса сущностей.

Использование неудачных описаний сущностей

Не используйте описаний, заимствованных только из словаря. Описания из словаря не будут включать значимую для бизнеса информацию.

Не пытайтесь перефразировать имя сущности. Не используйте имя сущности в ее описании.

Неясные, расплывчатые или, что еще хуже, неполные описания затрудняют повторное использование и расширение существующей модели. Пользователь не сможет проверить, содержит ли сущность всю необходимую информацию.

При этом значительно повышается риск возникновения перегруженных сущностей и использования их для хранения информации о разных объектах.

Концепции, которые кажутся очевидными для всех участников рабочих сессий, могут перестать быть столь очевидными с течением времени, когда перед новой командой разработчиков будет поставлена задача расширения существующей модели.

Полная атрибутивная модель

Полная атрибутивная модель — это логическая модель, наиболее детально представляющая структуру данных, представленных в третьей нормальной форме.

Нормализация

Нормализация это процесс организации данных в базе данных. Это включает создание таблиц и установку отношений между этими таблицами в соответствии с правилами, предназначенными для защиты данных и обеспечения большей гибкости базы данных за счет исключения избыточности и несогласованности зависимости.

Избыточные данные отнимают место на диске и создают проблемы с обслуживанием. Если необходимо изменить данные, которые находятся в нескольких местах, их необходимо изменить точно так же, как во всех местах. Изменение адресов клиентов значительно упрощается, если эти данные хранятся только в таблице Customers, а не в базе данных.

Что такое "несогласованная зависимость"? В отличие от того, что пользователь может искать в таблице Customers по адресу определенного клиента, может не иметь смысла искать зарплаты сотрудников, обращающихся к этому клиенту. Зарплата сотрудника связана с сотрудником или зависит от него, поэтому его следует переместить в таблицу Employees (сотрудники). Несогласованные зависимости могут усложнить доступ к данным, так как путь для поиска данных может отсутствовать или быть нарушен.

Существует несколько правил нормализации баз данных. Каждое правило называется "обычной формой". Если выполняется первое правило, база данных называется "Первая нормальная форма". Если выполняются первые три правила, база данных считается "третьей нормальной форме". Хотя возможны и другие уровни нормализации, Третья нормальная форма считается самым высоким уровнем, необходимым для большинства приложений.

Перед тем как переходить к процессу нормализации и приведению базы данных к определённой нормальной форме, необходимо привести базу данных к табличному виду, но так, чтобы он отвечал базовым принципам реляционной теории, так как мы говорим о реляционных базах данных, и только после этого задумываться о процессе нормализации.

По реляционной теории строки в таблицах не должны быть пронумерованы, т.е. порядок строк не имеет значения, так же как не имеет значения порядок столбцов. Т.е. например, если мы поменяем порядок столбцов, или порядок строк, ничего измениться не должно, это не должно ни на что повлиять. Таким образом по реляционной теории мы не можем обратиться к определённой строке или столбцу по ее номеру.

И если Ваши таблицы соблюдают эти принципы, то можно переходить к нормализации базы данных.

Требования первой нормальной формы (1NF)

Требование первой нормальной формы (1NF) очень простое и оно заключается в том, чтобы таблицы соответствовали реляционной модели данных и соблюдали определённые реляционные принципы.

Таким образом, чтобы база данных находилась в 1 нормальной форме, необходимо чтобы ее таблицы соблюдали следующие реляционные принципы:

- В таблице не должно быть дублирующих строк
- В каждой ячейке таблицы хранится атомарное значение (одно не составное значение)

– В столбце хранятся данные одного типа
– Отсутствуют массивы и списки в любом виде

Требования второй нормальной формы (2NF)

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

– Таблица должна находиться в первой нормальной форме
– Таблица должна иметь ключ
– Все неключевые столбцы таблицы должны зависеть от полного ключа (в случае если он составной)

Ключ – это столбец или набор столбцов, по которым гарантировано можно отличить строки друг от друга, т.е. ключ идентифицирует каждую строку таблицы. По ключу мы можем обратиться к конкретной строке данных в таблице.

Если ключ составной, т.е. состоит из нескольких столбцов, то все остальные неключевые столбцы должны зависеть от всего ключа, т.е. от всех столбцов в этом ключе. Если какой-то атрибут (столбец) зависит только от одного столбца в ключе, значит, база данных не находится во второй нормальной форме.

Иными словами, в таблице не должно быть данных, которые можно получить, зная только половину ключа, т.е. только один столбец из составного ключа.

Главное правило второй нормальной формы (2NF) звучит следующим образом. Таблица должна иметь правильный ключ, по которому можно идентифицировать каждую строку.

После того как таблицы базы данных находятся во второй нормальной форме, мы можем начинать приводить базу данных к третьей нормальной форме и рассматривать соответствующие требования.

Требования третьей нормальной формы (3NF)

Требование третьей нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость.

Транзитивная зависимость – это когда неключевые столбцы зависят от значений других неключевых столбцов.

Если в первой нормальной форме наше внимание было нацелено на соблюдение реляционных принципов, во второй нормальной форме в центре нашего внимания был первичный ключ, то в третьей нормальной форме все наше внимание уделено столбцам, которые не являются первичным ключом, т.е. неключевым столбцам.

Чтобы нормализовать базу данных до третьей нормальной формы, необходимо сделать так, чтобы в таблицах отсутствовали неключевые столбцы, которые зависят от других неключевых столбцов.

Иными словами, неключевые столбцы не должны пытаться играть роль ключа в таблице, т.е. они действительно должны быть неключевыми

столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Главное правило третьей нормальной форме (3NF) звучит следующим образом: Таблица должна содержать правильные неключевые столбцы

Как и в случае с множеством формальных правил и спецификаций, реальные сценарии не всегда позволяют обеспечить оптимальное соответствие требованиям. Как правило, нормализация требует дополнительных таблиц, и некоторые клиенты находят эту громоздкий. Если вы решите, что вы нарушаете одно из первых трех правил нормализации, убедитесь, что приложение предвидит все возможные проблемы, такие как избыточные данные и несогласованные зависимости.

При использовании универсального отношения возникают две **проблемы**:

- избыточность данных;
- потенциальная противоречивость (аномалии).

Под *избыточностью* понимают повторение данных в разных строках одной таблицы или в разных таблицах БД.

Аномалии– это проблемы, возникающие в данных из-за дефектов проектирования БД. Существуют три вида аномалий: вставки, удаления и модификации.

Аномалии вставки проявляются при вводе данных в дефектную таблицу. Добавляя информацию о новом сотруднике, мы должны добавить номер и название отдела. Если ввести данные, не соответствующие имеющимся в таблице (например, 42, отдел проектирования), будет не ясно, какая из строк БД содержит правильную информацию.

Аномалии удаления возникают при удалении данных из дефектной схемы. Предположим, что все сотрудники отдела 128 уволились в один и тот же день. После удаления записей этих сотрудников в БД больше не будет ни одной записи, содержащей информацию об отделе 128.

Аномалии модификации возникают при изменении данных дефектной схемы. Предположим, что отдел 128 решили переименовать в отдел передовых технологий. Необходимо изменить соответствующие данные о каждом сотруднике отдела. Если мы пропустим хотя бы одну запись, возникнет аномалия модификации.

Первая нормальная форма

- Исключите повторяющиеся группы в отдельных таблицах.
- Создайте отдельную таблицу для каждого набора связанных данных.
- Идентифицируйте каждый набор связанных данных с помощью первичного ключа.

Не используйте несколько полей в одной таблице для хранения похожих данных. Например, для отслеживания складской позиции, которая может поступать из двух возможных источников, в записи инвентаризации могут содержаться поля для кода поставщика 1 и поставщика с кодом 2.

Что происходит при добавлении третьего поставщика? Добавление поля не является ответом; для этого требуется изменение программ и таблиц, а также нестабильное динамическое количество поставщиков. Вместо этого разместите все сведения о поставщиках в отдельной таблице "поставщики", а затем свяжите запасы с поставщиками с ключом номера номенклатуры или поставщиками для запасов с ключом кода поставщика.

Вторая нормальная форма

- Создайте отдельные таблицы для наборов значений, которые применяются к нескольким записям.
- Свяжите эти таблицы с помощью внешнего ключа.

При необходимости записи не должны зависеть от первичного ключа таблицы (составного ключа). Например, рассмотрим адрес клиента в системе учета. Этот адрес необходим для таблицы Customers, но также в таблицах заказы, отгрузка, счета, расчеты с клиентами и сборы. Вместо того чтобы хранить адрес клиента в отдельной записи в каждой из этих таблиц, сохраните его в одном месте, в таблице Customers или в отдельной таблице адресов.

Третья нормальная форма

- Исключите поля, которые не зависят от ключа.

Значения в записи, не являющиеся частью ключа записи, не принадлежат таблице. В общем случае, когда содержимое группы полей может быть применено к более чем одной записи в таблице, рекомендуется размещать эти поля в отдельной таблице.

Например, в таблице сотрудников сотрудников можно включить название и адрес университета. Но вам нужен полный список университетов для групповых сообщений. Если сведения об университетах хранятся в таблице кандидатов, не существует способа перечисление университетов без текущих кандидатов. Создайте отдельную таблицу университетов и свяжите ее с таблицей кандидатов с помощью ключа кода университета.

ИСКЛЮЧЕНИЕ: следование третьей нормальной форме, хотя теоретически желательно, не всегда практично. Если у вас есть таблица Customers и вы хотите устранить все возможные зависимости между полями, необходимо создать отдельные таблицы для городов, почтовых индексов, торговых представителей, классов клиентов и других факторов, которые могут дублироваться в нескольких записях. Теоретически, нормализация стоит пурсинг. Тем не менее, многие небольшие таблицы могут понизить производительность или превышать количество файлов и емкости памяти.

Более часто можно применить третью обычную форму только к данным, которые часто изменяются. Если остались зависимые поля, разработайте приложение, чтобы требовать от пользователя проверки всех связанных полей при изменении любого из них.

Другие формы нормализации

Четвертая обычная форма, также называемая обычной формой бойце Кодд (БКНФ) и пятая обычная форма, существует, но редко рассматривается в практическом проекте. Отсутствие учета этих правил может привести к меньшему созданию структуры базы данных, но не влияет на функциональность.

На практике обычно ограничиваются приведением данных к третьей нормальной форме. Нормальные формы основаны на понятии функциональной зависимости. Приведем формальное определение для функциональной зависимости.