«... И ясен будет язык гугнивых».
Книга пророка Исаии 35,6
(церковнославянский перевод)

Лекция 1. Диалект SQL фирмы Oracle

Рассматриваются понятия, которые определяют диалект SQL, предлагаемый фирмой Oracle, в его нынешнем состоянии и формируют контекст употребления этого диалекта. В основном это реляционная модель данных и реляционное проектирование, а также стандартный SQL.

Происхождение и объем диалекта SQL фирмы Oracle

Для успешного программирования в Oracle на SQL недостаточно знать сугубо формальное описание языка. Необходимо владеть более широким набором имеющих отношение к делу знаний. С этой целью ниже рассматривается цепочка понятий, подводящая к «диалекту SQL, предлагаемому фирмой Oracle». Она устроена следующим образом: база данных → модель данных, СУБД; реляционная модель; язык запросов к данным и изменения данных → SQL → диалект SQL в Oracle.

База данных и модель данных

База данных

В буквальном переводе на русский язык «база данных» (БД) означает специально подготовленную на компьютере «основу» (base) для работы потребителей с «данными» (data). Непосредственным потребителем является, конечно, программа.

Общепринятого понятия базы данных, несмотря на широкое распространение самого явления, не существует. Вот некоторые примеры разнохарактерных определений.

(1) «Обычно большое собрание данных, организованных для особо быстрого и удобного способа поиска и извлечения (например, из ЭВМ)» (*Merriam-Webster's Collegiate Dictionary,* www.merriam-webster.com/dictionary/database, датировано 1962 годом).

(2) «Собрание структуризованных данных в ЭВМ, поддерживаемое СУБД, которая обеспечивает различным приложениям различный вид данных» (*F. Pascal, Understanding Relational Databases with examples in SQL-92, New York, NY: John Whiley & Sons, 1993*).

(3) «База данных – набор аксиом. Результат на запрос к базе есть теорема. Процесс вывода теоремы из аксиом есть доказательство. Доказательство осуществляется манипулированием символов по условленным математическим правилам. Доказательство [то есть результат запроса к базе] настолько же здраво и логично (consistent), насколько здравы и логичны правила» (*H. Darwen. The Duplicity of Duplicate Rows. Relational Database Writings 1989-1991, Reading, MA: Addison-Wesley, 1992*).

Возможное обобщение этих и других определений:

(0) «Совокупность всех данных некоторой прикладной области» [для использования в программных системах] (*Филиппов В. И. Общее описание системы КОМПАС. // Автоматизация программирования. Москва: ВЦ АН СССР, 1989*).

Существенные элементы в определениях БД:

- **Модель.** Всякая БД, независимо от того, сознает это ее разработчик или нет, воплощает собой некоторую «модель данных» «предметной области»: понятийную (говоря по-иному, «концептуальную», «бизнес-модель»), логическую (данных) и физическую (организации данных)*.
- **Собственно БД.** Организованные для долговременного хранения, обычно на внешнем носителе, данные общего пользования.
- **СУБД** (система управления базой данных). Компьютерная программа для управления данными и доступа к ним. Во всех промышленных системах прикладные программы для работы с данными не имеют возможности обратиться к данным БД иначе как через СУБД.

Моделированием (например, составлением карт местности) человечество занимается не одну тысячу лет, и современные базы данных лишь переводят эту деятельность в компьютерную область. Но современное моделирование средствами БД наследует из далекого прошлого и несколько общих проблем. Например:

- ни одна модель по определению не в состоянии учесть все обстоятельства предметной области и обязательно чего-то не будет учитывать (на деле, наоборот, «учитывать всего лишь кое-что»);
- существует опасность «неадекватного» моделирования, когда модель формально построена корректно (например, СУБД не видит ошибок в запросах и выдает ответы), но некорректно отражает предметную область, причем формального аппарата для обнаружения подобных расхождений не существует;

* Иногда тройку «концептуальная», логическая и физическая модель выстраивают по-другому; здесь она соответствует определениям, используемым в промышленных системах проектирования БД.

- чем дольше используется модель, тем чаще возникает необходимость ее подправить и привести в (лучшее) соответствие предметной области.

Последнее обстоятельство (необходимость внести в модель данных изменения) может быть вызвано как субъективными причинами, из-за небрежного начального проектирования модели, так и объективными – из-за изменения самой предметной области. Например, в базах личностных данных до 2000-х годов сведения о браке достаточно моделировались парой «муж–жена», тогда как с наступлением XXI века это все чаще становится неприемлемым, и старые базы требуется подправлять.

Перечисленные проблемы моделирования вообще автоматически сопровождают и моделирование с помощью БД, а проблема внесения изменений в модель, используемую в БД, часто к тому же усугубляется технологическими сложностями, вынуждающими в жизни откладывать перестройку БД «до последнего момента».

Что касается СУБД, то в БД именно эта программа обеспечивает «особо быстрый и удобный способ поиска и извлечения», притом берет на себя решение этих задач монопольно, запрещая прикладным программам обращаться к данным в обход себя.

Относительно терминологии нередко бытуют вольности словоупотребления. Например, в материалах фирмы Oracle часто говорят о «системе базы данных» (database system), вероятно, подразумевая под этим сосуществующую пару «СУБД – БД». В склонном к упрощениям американском английском слово «система» в полном термине порою выпадает, в результате чего пару СУБД – БД часто именуют просто словом database. По той же причине вместо «тип СУБД» часто говорят просто «СУБД», и тогда возникает двусмыслица: СУБД как конкретная работающая программа и СУБД как конкретный набор программного обеспечения для обслуживания доступа приложений к данным. Это не исключение: подобная многосмыслица имеется и для понятия «модель данных», о чем будет сказано ниже, а также многих других понятий в базах данных в частности и в информационных технологиях вообще. Иногда в этом ничего страшного нет и можно сориентироваться по контексту употребления, но иногда такие вольности приводят к туману в понимании и в выражении мыслей.

СУБД

СУБД обеспечивает работу приложений, а в конечном счете пользователей, с информационной моделью. Основное назначение СУБД состоит в делегировании управления данными от прикладной программы одной специальной программной системе, которая вне зависимости от

того, какая прикладная программа или же какой пользователь работает с данными, единым во всех случаях образом:

– защищает данные от рассогласованности,
– оптимизирует выполнение операций над данными,
– оптимизирует обращение к данным,
– выполняет прочие необходимые действия.

В число функций, которые обеспечивают современные СУБД, входят следующие:

- Поддержка логической модели данных (определение данных, оперирование данными).
- Восстановление данных (транзакции, журнализация, контрольные точки).
- Управление одновременным доступом к данным в БД.
- Безопасность данных (права доступа и прочее).
- Самостоятельная оптимизация выполнения операций.
- Прочие, в том числе вытекающие из перечисленных (администрирование, статистика, распределение данных и т. д.).

Реляционный подход к моделированию данных

Наиболее существенное влияние на современные промышленные виды СУБД, включая Oracle, оказал реляционный подход к моделированию данных. Он основывается на использовании «реляционной теории», частью которой является «реляционная модель». Последняя берет свое начало со статьи своего основателя, Э. Кодда (E. F. Codd), «Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks», опубликованной в IBM Research Report RJ599 в 1969 году. Впоследствии реляционная модель пережила всплеск интенсивного изучения и уточнения широким кругом специалистов, а в настоящее время она развивается главным образом усилиями К. Дейта (C. J. Date), сподвижника и коллеги Кодда во времена создания модели.

Реляционная модель данных

Все, что я съел, и все, что я выпил, осталось со мною;
Все остальное, что есть, право, не стоит щелчка.
Надпись на могильной плите
Сарданапала

Выражение «модель данных» часто понимается в двух разных смыслах: как формальное описание некоторой конкретной предметной области и как инструмент для составления подобных описаний. Нужный смысл обычно приходится определять по контексту. Здесь выражение

«реляционная модель данных» понимается как инструмент составления в БД конкретных описаний конкретных предметных областей.

Реляционная модель при необходимости может быть описана математическим языком, то есть наиболее точным из изобретенных человеком. Ниже приводятся нестрогие определения некоторых понятий реляционной модели.

- «Тип данных» (type) – множество допустимых величин («область определения») и операций. Для всех типов существуют операции сравнения и присвоения. Величинам не запрещено иметь структуру, например, объекта.
- «Отношение» (relation) – множество атрибутов: уникальных имен с уточнением типа данных; плюс множество «наборов величин» («рядов»), соответствующих атрибутам. Величины в наборах могут быть представлены только единичными значениями соответствующих атрибутам типов, то есть быть скалярами («1-я нормальная форма»).
- «Переменная отношения» (relation variable) – *переменная* типа отношения конкретного вида, необходимое понятие для определения в базе данных действий по «обновлению отношений», «внесения изменений в данные». В нарушение точности и в силу поверхностно-ознакомительного характера настоящего материала далее вместо названия «переменная отношения» будет употребляться просто «отношение» (подобно вольному употреблению слова «целое» вместо «переменная целого типа»).
- «Ключ» (key) – группа атрибутов, величины для которых во всех наборах в отношении различны, но ни одна подгруппа этих атрибутов таким свойством уже не обладает (свойство «минимальности» ключа). В частности, группа может состоять из единственного атрибута. Ключ в отношении обязан иметься всегда, а если их несколько, один из них обязан быть назначен «первичным» (primary).
- «Внешний ключ» (foreign key) – группа атрибутов, величины для которых в каждом наборе величин отношения обязаны совпадать с набором величин какого-нибудь ключа. Внешние ключи в отношении не обязательны и провозглашаются по потребностям моделирования.
- «Операции» (operation) – множество общих действий над отношениями, дающих в результате опять-таки отношения («замкнутость операций»). Используются для получения новых отношений в нуждах последующего моделирования или при извлечении из базы нужных данных. Перечень операций можно определять по-разному; в первых предложениях модели приводилось восемь операций (проекции, соединения, отбора и пр.), уже не минимальный набор, как компромисс между отсутствием избыточности и удобством употребления.

- «**Реляционная база данных**» (relational database) – набор отношений.

«Тип данных» иногда называют «доменом» (domain), но иногда под «доменом» разумеют только «область определения» величин. «Набор величин» (tuple) по-русски иначе называют «кортежем» или «*n*-кой».

Для удобства отношения часто изображают в виде таблиц, хотя такое представление неправомерно (в отношении не определен ни порядок атрибутов, ни порядок наборов величин, в отличие от таблицы). В SQL, на основе которого построена в том числе СУБД Oracle, понятие «отношения» (а точнее, понятие «переменной отношения») как инструмента моделирования заменено как раз на «таблицу». Другим представлением данных отношения может быть гиперкуб, и к нему тоже иногда удобно прибегать в рассуждениях об имеющейся БД.

Если отказаться от определительного слова-кальки «реляционный», то термин «реляционная БД» можно перевести как «БД отношений» (точнее, «БД построенная *посредством* отношений»; отношений как инструмента, а не объекта моделирования: иначе исходный термин был бы relation database). Точно так же термин «реляционная модель» можно перевести как «модель отношений», то есть «система понятий для построения модели предметной области в виде набора отношений». По ряду причин, в том числе исторического и языкового характеров, этого не было в свое время сделано.

Все взаимоотношения данных описываются *явно* и *только* величинами в наборах (в других подходах к моделированию может быть иначе). Никаких «подразумеваемых» зависимостей (в том числе на уровне программной логики), кроме сформулированых переменными отношений, нет. Реляционный подход разграничивает описание данных и сопутствующую приложению программную логику (в противовес, например, объектному подходу).

Приведенный взгляд на реляционную БД (набор отношений и операции) характерен для *реляционной алгебры*. Это не единственная точка зрения. Каждый набор величин в переменной отношения можно понимать как истинное высказывание («предикат»): имеется такой-то сотрудник с такими-то свойствами; такой-то отдел и так далее. Тем самым реляционная база данных в каждый момент времени представляет собой набор истинных высказываний о предметной области, сформулированный через отношения. По сути, набор высказываний в переменных отношений и образует модель предметной области, представленную базой данных. Такой взгляд на реляционную БД характерен для *реляционного исчисления*. Оба взгляда на реляционную модель хорошо изучены и доказана их выразительная равносильность.

Проектирование реляционной базы данных

Задачей проектирования реляционной базы данных можно считать достижение такой системы отношений, которая воспроизводила бы в БД необходимые утверждения о предметной области, и при этом все сведения о предметной области были бы представлены в БД однократно, не повторялись. Такое проектирование, в отличие от реляционной модели, не поддается математическому описанию и в значительной мере определяется здравым смыслом и опытом разработчика.

Существующая теория проектирования реляционной БД не учит, как нужно строить БД. Вместо этого она учит тому, какими неприятностями чревато «неправильное» проектирование: ее иногда называют «хорошим источником плохих примеров».

Устранение избыточности

Значительная часть усилий по проектированию реляционной БД связана с устранением избыточности. Избыточность провоцирует «аномальности обновлений» данных, в результате которых формально правильно составленные вопросы к БД смогут выдавать неверные данные. К сказанному есть два важных замечания. Во-первых, избыточность тут подразумевается применительно к логическому описанию данных, в то время как избыточность физического хранения может быть оправданна и разумна. Во-вторых, устранение избыточности, будучи необходимым для «правильного» построения БД, само по себе не гарантирует правильности моделирования предметной области.

Простой пример устранения избыточности показан на рис. 1.1.

Сотрудники

…	Зарплата	Комиссионные	Доход
…	1600	300	1900
…	1250	500	1750

Сотрудники

…	Зарплата	Комиссионные
…	1600	300
…	1250	500

×

Рис. 1.1.

Пусть в отношении, представляющем сведения о сотрудниках, есть атрибуты «зарплата», «комиссионные» и «доход». Если по правилам моделируемой предметной области доход сотрудника складывается исключительно из его зарплаты и комиссионных, один из перечисленных атрибутов следует из определения отношения убрать – скорее всего, это будет «доход».

В других случаях устранение избыточности может выглядеть отнюдь не столь очевидно. Представьте, что в БД требуется хранить почтовый адрес, включая индекс. Часто для этого используют отдельные атрибуты для индекса и прочих частей адреса, таких как город, улица и так далее. Однако подобное хранение, строго говоря, избыточно, так как почтовый индекс однозначно определяется «словесной частью» адреса. Двойное хранение сведений будет держать открытой лазейку для появления содержательно неверных данных в БД, но чтобы воспрепятствовать этому, потребуется заметно усложнить схему и утяжелить БД (что же, идти на хранение ненужного для иной цели соответствия «словесной части» адреса почтовому индексу?) и, как следствие, – усложнить и замедлить, казалось бы, простые обращения к БД. В силу таких издержек разработчик в жизни нередко предпочитает простоту организации данных рискам, связанным с избыточностью (или, если удается, снимает проблему рассогласования данных с помощью ограничений целостности либо триггерных процедур).

Хотя в общем борьба с избыточностью данных в БД – процесс неформализованный, известно две техники, позволяющие устранять избыточность и доведенные до математического уровня описания: нормализация и ортогонализация. Ниже приводится их нестрогое ознакомительное изложение.

Нормализация реляционной базы данных

Нормализация есть приведение отношения к «нормальному виду» путем дробления его на несколько других, связанных внешним ключом. Дробление осуществляется построением проекций исходного отношения, однако задача состоит в том, чтобы обратное соединение полученных в результате дробления отношений возвращало бы нас к исходному отношению, то есть чтобы такое дробление происходило без потерь (искажения) информации.

Наиболее популярная схема подобного «правильного» дробления состоит в устранении из отношения лишних функциональных зависимостей. Данные в отношении функционально зависят друг от друга, если одни из них могут быть определены через другие.

Пусть в отношении с данными о сотрудниках присутствуют и номер отдела сотрудника, и название отдела. Чтобы устранить зависимость

между номером и названием отдела в отношении «сотрудники», достаточно оставить в нем только один из двух атрибутов (скорее всего, это будет «номер отдела»), создать отношение «отделы» с атрибутами «номер отдела» и «название», объявить в нем «номер отдела» ключом. «Номер отдела» в «сотрудниках» следует объявить внешним ключом, связав его с «номером отдела» из «отделов» (см. рис. 1.2).

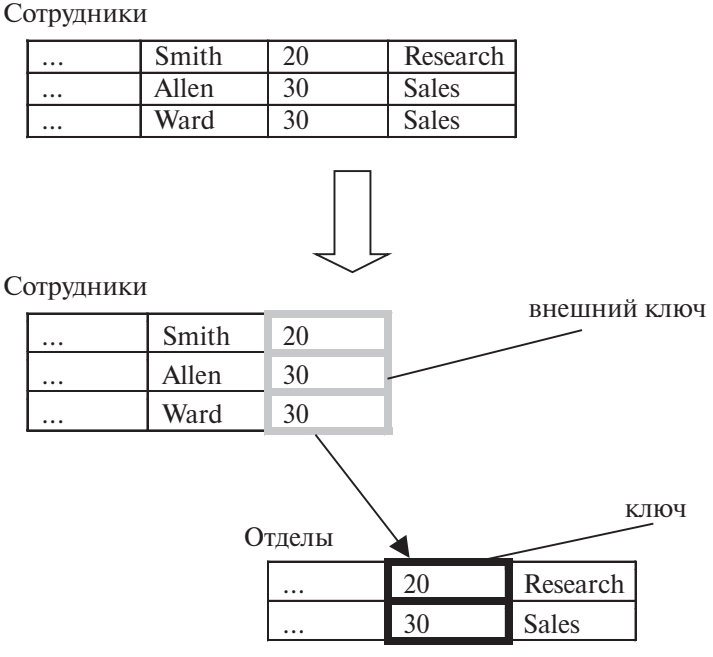


Рис. 1.2.

Избавляться от подобного рода функциональных зависимостей в отношениях можно, пока они не окажутся приведенными к «нормальной форме BCNF», Бойса-Кодда (Boyce-Codd); по праву первенства ее следовало бы назвать «нормальной формой Хита» (Heath). В таких отношениях, образно выражаясь, «каждое сведение относится к ключу, всему ключу (со всеми его атрибутами) и только к ключу» (в оригинальной фразе на английском вместо «сведения» дословно сказано «факт»), с тем добавлением, что ключей может быть несколько. Иными словами, в отношении, удовлетворяющем BCNF, произвол величин в наборах ограничен только правилами ключа, возможно внешнего ключа и типов атрибутов. Несколько менее строгий вариант BCNF именуется «3-й нормальной формой».

Если ключи в отношениях состоят из групп атрибутов (не являются «простыми»), то для избавления от лишних функциональных зависимостей данных дополнительно потребуется привести отношения к «пятой нормальной форме». Если же для дробления рассмотреть некоторые обобщенные версии операций проекции и соединения, то вдобавок потребуется привести отношения к «шестой нормальной форме». Таким образом, на практике получить выгоды от нормализации проще всего, имея дело с простыми (одноатрибутными) ключами в отношениях.

Все предложенные нормальные формы линейно упорядочены, так что достижение в отношении какой-нибудь из них по определению означает выполнение ряда «предыдущих».

Нормализация отношений способствует:

- устранению избыточности данных в БД;
- как следствие, избавлению от некоторых аномалий обновления данных;
- упрощению изменения, а иногда и запросов к данным;
- упрощению формулирования ограничений целостности;
- представлению данных предметной области в БД в более естественном виде.

Часто процесс устранения избыточности автоматически влечет за собой прочие перечисленные выгоды. Приведение отношений к нормальным формам неспособно устранить все виды избыточности, но считается действенным средством для достижения этой цели. Иногда нормализацию данных называют формализованным проявлением здравого смысла. В то же время:

- механическое сведение к нормальной форме BCNF в случае составных ключей может оказаться неоправданным и вступить в противоречие с необходимостью сохранить в модели определенную долю избыточности;
- сохранение избыточности данных, в том числе вследствие недонормализованности отношений, способно приводить к аномалиям обновлений;
- нормализованные данные может оказаться сложнее изменять;
- нормализация неоднозначна и часто допускает разные способы дробления таблиц;
- нормализация не решает всех проблем моделирования, заставляя разработчика БД прибегать к дополнительным правилам ограничения целостности данных.

В жизни в SQL-системах нормализация (применительно к таблицам) часто не соблюдается в угоду ожиданию скорости доступа, простоты изменения данных и их представления. Для обозначения такой намеренной практики даже используется особый термин: «денормализация».

Иногда подобные ожидания оправдываются, однако в качестве оборотной стороны это порождает риски расхождения модели в БД с предметной областью, а то и некорректности используемой модели. Некоторые эксперты полагают, что выигрыш, который в определенных обстоятельствах способна дать денормализация, – мифический.

Ортогонализация отношений

В то время как нормализация отношений ставит своей целью избавление от избыточности в пределах отдельных отношений, ортогонализация данных пытается избавиться от повторений общих данных в разных отношениях. Тем самым она дополняет нормализацию.

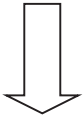
Упрощенный вариант принципа ортогонализации данных утверждает, что один и тот же набор величин не имеет право повторяться в разных отношениях. Например, с точки зрения этого принципа желательно преобразование отношений в БД, показанное на рис. 1.3.

Сотрудники_20

Номер	Имя	Зарплата	...
7369	Smith	900	...

Сотрудники_30

Номер	Имя	Зарплата	...
7499	Allen	1600	...
7521	Ward	1250	...



Сотрудники

Номер	Имя	Зарплата	...	Отдел
7499	Allen	1600	...	30
7521	Ward	1250	...	30
7369	Smith	900	...	20

Рис. 1.3.

Существует и более сложная формулировка принципа ортогональности для проектируемых отношений, требующая для иллюстрации менее очевидных примеров.

[. . .]