

DB LEC 03

isagila

Собрано 15.09.2023 в 19:11



Содержание

1. Лекции	3
1.1. Лекция 23.09.01.	3
1.2. Лекция 23.09.08.	3
1.3. Лекция 23.09.15.	4

1. Лекции

1.1. Лекция 23.09.01.

TODO: Без понятия, буду ли писать лекции по БД, но пусть пока все остается как есть.

Информация бывает трех видов

1. Сигнал.

Например, для человека, который не знает какой-либо язык, текст на этом языке будет сигналом: т.е. можно понять, что какая-то информация была передана, но принять (понять) эту информацию нельзя.

2. Знание.

Студент, слушающий лектора и пишущий конспект, получает знания.

3. Данные.

Данные это формализованные знания. В отличие от знаний данные не искажаются в процессе передачи. Если лектор передаст студентам конспект лекции, который он написал сам, то он передаст именно данные.

Def 1.1.1. Данные это поддающиеся многократной интерпретации представления информации в формализованном виде пригодном для передачи, интерпретации и обработки.

Все данные проходят путь вида

Сбор → Обработка → Передача → Хранение → Представление

TODO: Да... вводная лекция она такая... не очень большая — не знаю, что еще можно добавить.

1.2. Лекция 23.09.08.

Уровни архитектуры данных

1. Внешний.

Решаем проблему представления данных пользователю (как агрегировать и детализировать данные?).

2. Концептуальный.

Занимаемся выделением сущностей, решаем проблемы безопасности (разрешения + ограничения). Определяем семантику (например договариваемся, что кредитный рейтинг будет храниться в виде числа от 1 до 5 и сопоставляем каждому числу некоторое словесное описание, которое записано в документации, но не в самой базе данных).

3. Внутренний.

Решаем проблемы хранения данных на физическом уровне.

Для простоты выделяют три модели данных, которые связаны с некоторыми описанными выше уровнями архитектуры данных.

1. Модель Сущность–Связь (внешний + концептуальный уровни).

2. Логическая (даталогическая модель) модель (все уровни).

3. Физическая модель (концептуальный + внутренний уровни).

Модель Сущность–Связь

Def 1.2.1. Сущность это множество экземпляров (реальных или абстрактных) однотипных объектов предметной области.

Def 1.2.2. Сущность называется сильной, если ее экземпляры могут существовать независимо. Слабые сущности могут существовать только при наличии одного или нескольких экземпляров сильной сущности.

Пример 1.2.3. Пусть у нас есть две сущности: Студент и Группа. Студент будет сильной сущностью, т.к. он может существовать и без группы, а Группа будет слабой сущностью, т.к. она не может существовать без студентов. Этот пример очень условный: деление сущностей на слабые и сильные зависит от обстоятельств. В некоторых случаях Группа вполне может быть сильной сущностью.

Каждая сущность обладает одним или несколькими атрибутами. Атрибуты делятся на

1. Простые (например дата рождения)

2. Составные (например адрес, если хранить город, улицу, дом и т.п. отдельно)

Замечание 1.2.4. Один и тот же атрибут (например, ФИО) может в разных случаях быть как простым (если его хранить как одну строку), так и составным (если отдельно хранить фамилию, отдельно имя и отдельно отчество).

Также атрибуты можно делить по другому принципу.

1. Обязательные (например email в некоторых случаях может быть обязательным атрибутом).

2. Необязательные (например отчество может быть необязательным атрибутом).

Помимо этого, атрибуты делятся на

1. Однозначные (например дата рождения, она у каждого ровно одна).

2. Многозначные (например номер телефона, у кого-то может быть несколько номеров телефона).

Для каждого атрибута мы определяем домен, т.е. множество допустимых значений. Доменом может быть перечисление, множество всех натуральных чисел, регулярное выражение и т.д.

Чтобы показать отношения между сущностями, используются связи. На уровне Сущность–Связь они обычно несут семантическую нагрузку, например Студент **принадлежит** Группе. Существует три вида связи

1. Один-к-одному (1:1).

Обычно все связи этого вида это искусственно выделенные атрибуты. Зачем же тогда нужна эта связь? Рассмотрим на примере. Допустим, у нас есть две сущности: Студент и Паспорт. Логично, что у каждого студента один паспорт и у каждого паспорта один студент, т.е. связь вида 1:1. Почему же нельзя добавить паспорт в атрибуты сущности Студент? Так можно сделать, но связь вида 1:1 может быть удобна в целях безопасности: в таком случае все паспорта можно будет хранить отдельно от студентов и (например) дополнительно шифровать (шифровка же всей сущности Студент будет мало того, что излишне затратной, так еще и не очень нужной).

2. Один-ко-многим (1:N).

3. Многие-ко-многим (N:N).

1.3. Лекция 23.09.15.

Def 1.3.1 (по Коннолли и Бегг). База данных это совместно используемый набор логически связанных данных и описание этих данных, предназначенный для удовлетворения информационных потребностей предприятия.

Def 1.3.2 (по Дейту). База данных это набор постоянно хранимых данных, используемых прикладными системами предприятия.

Def 1.3.3 (по Хомоненко). База данных это совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Рассмотрим некоторые модели данных.

Иерархическая модель данных

Эта модель удобная для восприятия человека, т.к. он часто сталкивается с разного рода иерархией в реальном мире. У нас есть некоторые поля данных, которые являются неделимыми атомами, а некоторые объединение этих полей называется сегментом данных. Каждая запись в БД это экземпляр сегмента данных.

К минусам этой модели можно отнести дублирование данных, вследствие чего возникает некоторая сложность поддержки их целостности.

Сетевая модель данных

Если иерархическая модель данных по сути являлась деревом, то сетевая модель представляет собой граф, в котором вершины представляют собой некоторые экземпляры объектов.

Реляционная модель данных

В отличие от иерархической и сетевой модели элементы модели сразу двумерные (т.е. таблицы) и при этом мы не храним связь явно — мы выделяем некоторый атрибут, который наделяем семантикой связи. Это может привести к некоторым проблемам — семантика может быть неправильна трактована, в результате чего запрос к БД либо не выполнится, либо выполнится некорректно. Проблему поддержки целостности данных будем решать нормализацией (об этом будет рассказано в последующих лекциях). Одним из плюсов этой модели является то, что мы можем оценить время работы запроса к БД.

Постреляционная модель данных

Берем реляционную модель и снимаем запрет на неделимость поля, т.е. в качестве атрибута теперь можно хранить массив строк через запятую или сразу JSON с какими-либо данными.

Эта модель была сделана для того, чтобы решить проблему очень большой персонификации данных. Т.е. допустим есть таблица с деталями и у некоторых деталей есть какие-то атрибуты, которых в принципе не может быть у других. Если добавить все возможные атрибуты каждой детали, то получим очень разряженную таблицу почти полностью состоящую из null-значений \Rightarrow проблемы с памятью.

С другой стороны мы усложняем себе поиск по такой базе (сложно проверять детали, которые имеют разный набор атрибутов) и поддержку целостности — поле JSON-данными при желании можно записать некорректное значение (либо нужно жестко валидировать все записываемое в каждую такую ячейку \Rightarrow проблемы со скоростью работы).

Многомерная модель данных

Пусть есть таблица со столбцами «Продавец», «Товар», «Регион», «Квартал» и «Количество». Мы хотим уметь быстро строить сводки по продавцу, товару, региону, кварталу (или по комбинации параметром), находить наибольшее и наименьшее значения и т.п.

Можно представить наши данные в 4ех мерном пространстве, где первые четыре колонки будут задавать координаты, а пятая (количество товара) — значение. Теперь, чтобы получать различные метрики, мы будем рассекать полученный гиперкуб гиперплоскостями, искать в них минимум/максимум и т.п.

Данная модель имеет большие затраты по памяти, но зато с ее помощью можно быстро выполнять запросы. Ее можно использовать в следующем ключе: храним данные в реляционной модели. Ночью строим по ней многомерную модель, а днем пользуемся ей не внося никаких изменений (будем считать, что продажи товаров за день не сильно влияют на глобальную ситуацию \Rightarrow ими можно пренебречь). Следующей ночью заново строим многомерную модель по уже новым данным и так далее. Таким образом в течение дня мы получаем возможность быстро делать разные запросы и узнавать разные метрики.

Объектно-ориентированная модель данных

Похожа на постреляционную модель: мы сериализуем все объекты в системе и храним пары вида `id: [serialized]`. Это усложняет поиск, т.к. нужно десериализовывать данные.