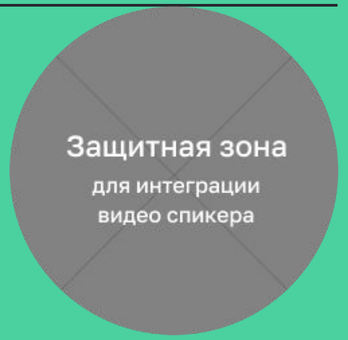


Продвинутый SQL

Лекция 4.
Зависимости.

нетология

Николай Хащанов

Full–stack developer

Защитная зона
для интеграции
видео спикера

О спикере:

- Разрабатываю и поддерживаю crm/erp системы
- Преподаю в Нетологии
- Окончил РГТЭУ по специальности Менеджмент
- Оптимизация и автоматизация бизнес-процессов

Я в Слаке:

@Николай Хащанов

Содержание

1 Зависимости

2 Денормализация

3 OLAP

Зависимости

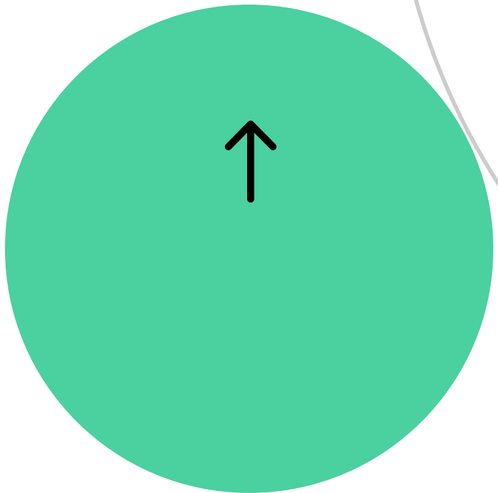
1

Николай Хащанов

Продвинутый SQL

нетология

Функциональные зависимости — это основа нормализации баз данных



Защитная зона
для интеграции
видео спикера

Функциональные зависимости.

Под функциональной зависимостью подразумевается зависимость значения одного атрибута от другого.

Если даны два атрибута А и Б некоторого отношения, то говорят, что Б функционально зависит от А, если в любой момент времени каждому значению А соответствует ровно одно значение Б.

aircraft_code	model	range
773	Boeing 777-300	11 100
763	Boeing 767-300	7 900
SU9	Sukhoi SuperJet-100	3 000
320	Airbus A320-200	5 700
321	Airbus A321-200	5 600
319	Airbus A319-100	6 700
733	Boeing 737-300	4 200
CN1	Cessna 208 Caravan	1 200
CR2	Bombardier CRJ-200	2 700

Возьмем таблицу aircrafts.
Атрибуты model и range функционально зависят от aircraft_code.

Избыточные ФЗ.

Избыточной зависимостью называют ситуацию, при которой зависимость заключает информацию, которая может быть получена из других зависимостей.

Правила вывода зависимостей применяются к списку функциональных зависимостей с целью избавиться от избыточных зависимостей.

Набор функциональных зависимостей, получаемый из исходного набора функциональных зависимостей удалением всех избыточных функциональных зависимостей с помощью правила вывода называется минимальным покрытием.

Избыточные функциональные зависимости следует удалять из набора по одной, каждый раз заново анализируя полученный набор функциональных зависимостей на присутствие в нем избыточных зависимостей.

Аксиомы Армстронга (правила вывода).

1. Рефлексивность
2. Пополнение
3. Транзитивность

Рефлексивность.

Если множество Б является подмножеством множества А, то Б функционально зависит от А. Такие зависимости называются тривиальными, правая часть таких зависимостей содержится в левой. То есть А->Б.

123 flight_id	ABC flight_no
3 395	PG0227
3 396	PG0228
3 397	PG0229
3 398	PG0227
3 399	PG0228
3 400	PG0229
3 401	PG0671
3 402	PG0671
3 403	PG0671
3 404	PG0671
3 405	PG0671
3 406	PG0671
3 407	PG0671

Пополнение.

Если Б функционально зависит от А, то БВ функционально зависит от АВ. То есть можно дополнить левую и правую части функциональной зависимости одинаковыми атрибутами.

flight_id	departure_airport
3 212	DME
3 213	DME
3 214	DME
3 215	DME
3 216	DME
3 217	DME
3 218	VKO
3 219	VKO
3 220	VKO

flight_id	departure_airport_name	departure_airport
3 212	Домодедово	DME
3 213	Домодедово	DME
3 214	Домодедово	DME
3 215	Домодедово	DME
3 216	Домодедово	DME
3 217	Домодедово	DME
3 218	Внуково	VKO
3 219	Внуково	VKO
3 220	Внуково	VKO

flight_id	departure_airport_name	departure_airport
3 215	Домодедово	DME
3 216	Домодедово	DME
3 217	Домодедово	DME
3 218	Внуково	VKO
3 219	Внуково	VKO
3 219	Внуково	VKO
3 219	Внуково	VKO
3 222	Внуково	VKO

Раз есть А->Б, то есть и АВ->Б. Как бы не меняли значения В, в таблице каждой паре значений АВ всегда соответствует одно значение то же что и для А.
Для того чтобы не было АВ->Б, нужно найти таких два кортежа, чтобы значения пары АВ в них были одинаковые, а значения Б разные: <ai, bi, бi>, <ai, bi, бj>. В таблице одинаковой пары нет. Выберем другое значение В что бы были. Но в любом случае, что для АВ, что для А будет соответствовать одно значение Б.

Транзитивность.

Транзитивная зависимость является избыточной.

Если А -> Б, Б -> В и А -> В. Любой не ключевой атрибут не должен быть зависим от атрибута, который не является первичным ключом отношения.

flight_id	scheduled_departure	scheduled_arrival	departure_airport	departure_airport_name	departure_city
1	2016-09-13 08:35:00	2016-09-13 09:30:00	DME	Домодедово	Москва
2	2016-10-03 18:05:00	2016-10-03 19:00:00	DME	Домодедово	Москва
3	2016-10-03 08:35:00	2016-10-03 09:30:00	DME	Домодедово	Москва
4	2016-11-07 11:25:00	2016-11-07 12:20:00	DME	Домодедово	Москва
5	2016-10-14 08:35:00	2016-10-14 09:30:00	DME	Домодедово	Москва
6	2016-10-14 18:05:00	2016-10-14 19:00:00	DME	Домодедово	Москва

Здесь мы видим, что departure_airport_name и departure_city функционально зависят от departure_airport и эта зависимость не по первичному ключу, которым является атрибут flight_id.

Практика 1.

Устраните транзитивную зависимость в предыдущей таблице.

Какой нормальной форме будет соответствовать результат и почему?

Аксиомы Армстронга 4 - 8.

Следующие аксиомы необходимо изучить самостоятельно:

4. Самодетерминированность
5. Декомпозиция
6. Объединение
7. Композиция
8. Накопление

Итог:

С помощью функциональных зависимостей можно накладывать на реляционную базу дополнительные ограничения. Основной идеей является то, что значением одного атрибута в кортеже однозначно определяется значение другого атрибута.

Чем выше форма нормализации, тем меньше функциональных зависимостей может существовать.

Денормализация

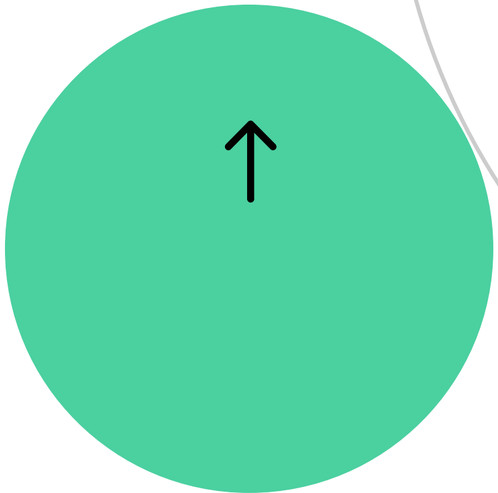
2

Николай Хащанов

Продвинутый SQL

нетология

Денормализация - это процесс ухода от правил нормализации там, где это необходимо



Отсутствие алгоритма.

Для процесса денормализации не существует стандартного алгоритма. Процесс денормализации индивидуален и требует четкого понимания для чего он необходим в связи с появлением избыточности.

К денормализации прибегают для сокращения времени обработки запросов и уменьшения затрат ресурсов. В нормализованных базах часто приходится соединять большое количество таблиц или добавлять агрегацию.

Таким образом денормализацию можно выполнить сократив количество таблиц или добавив новые столбцы в существующую таблицу. При этом учитывая избыточность данных необходимо следить за целостностью данных.

bookings.flights_v:

За основу возьмем знакомое представление flights_v, которое уже денормализовано по отношению к таблице flights. Не все РСУБД поддерживают материализованные представления, которые позволяют применять денормализацию не затрагивая базу на физическом уровне.

Надо понимать, что лучше объединять несколько таблиц в одну, которые имеют небольшой размер и содержат редко изменяемую информацию, тесно связанную между собой.

bookings.flights_v:


```
explain analyze
select f.flight_id,
    f.flight_no,
    f.scheduled_departure,
    timezone(dep.timezone, f.scheduled_departure) AS scheduled_departure_local,
    f.scheduled_arrival,
    timezone(arr.timezone, f.scheduled_arrival) AS scheduled_arrival_local,
    f.scheduled_arrival - f.scheduled_departure AS scheduled_duration,
    f.departure_airport,
    dep.airport_name AS departure_airport_name,
    dep.city AS departure_city,
    f.arrival_airport,
    arr.airport_name AS arrival_airport_name,
    arr.city AS arrival_city,
    f.status,
    f.aircraft_code,
    f.actual_departure,
    timezone(dep.timezone, f.actual_departure) AS actual_departure_local,
    f.actual_arrival,
    timezone(arr.timezone, f.actual_arrival) AS actual_arrival_local,
    f.actual_arrival - f.actual_departure AS actual_duration
from bookings.flights f
left join bookings.airports dep on f.departure_airport = dep.airport_code
left join bookings.airports arr on f.arrival_airport = arr.airport_code
```

```
1   Hash Left Join  (cost=8.68..1409.67 rows=33121 width=195)

Planning Time: 0.346 ms
Execution Time: 63.106 ms
```

bookings.flights_v + стоимость:


```sql
explain analyze
select f.flight_id,
    f.flight_no,
    f.scheduled_departure,
    timezone(dep.timezone, f.scheduled_departure) AS scheduled_departure_local,
    f.scheduled_arrival,
    timezone(arr.timezone, f.scheduled_arrival) AS scheduled_arrival_local,
    f.scheduled_arrival - f.scheduled_departure AS scheduled_duration,
    f.departure_airport,
    dep.airport_name AS departure_airport_name,
    dep.city AS departure_city,
    f.arrival_airport,
    arr.airport_name AS arrival_airport_name,
    arr.city AS arrival_city,
    f.status,
    f.aircraft_code,
    f.actual_departure,
    timezone(dep.timezone, f.actual_departure) AS actual_departure_local,
    f.actual_arrival,
    timezone(arr.timezone, f.actual_arrival) AS actual_arrival_local,
    f.actual_arrival - f.actual_departure AS actual_duration,
    amount.avg
from bookings.flights f
left join bookings.airports dep on f.departure_airport = dep.airport_code
left join bookings.airports arr on f.arrival_airport = arr.airport_code
left join (select avg(tf.amount), tf.flight_id from ticket_flights tf group by tf.flight_id) as amount
    on amount.flight_id = f.flight_id
```

```
1   Hash Left Join  (cost=20222.95..21710.89 rows=33121 width=227)

Planning Time: 0.512 ms
Execution Time: 267.380 ms
```

flights_v + стоимость + количество:


```
explain analyze
select f.flight_id,
    f.flight_no,
    f.scheduled_departure,
    timezone(dep.timezone, f.scheduled_departure) AS scheduled_departure_local,
    f.scheduled_arrival,
    timezone(arr.timezone, f.scheduled_arrival) AS scheduled_arrival_local,
    f.scheduled_arrival - f.scheduled_departure AS scheduled_duration,
    f.departure_airport,
    dep.airport_name AS departure_airport_name,
    dep.city AS departure_city,
    f.arrival_airport,
    arr.airport_name AS arrival_airport_name,
    arr.city AS arrival_city,
    f.status,
    f.aircraft_code,
    f.actual_departure,
    timezone(dep.timezone, f.actual_departure) AS actual_departure_local,
    f.actual_arrival,
    timezone(arr.timezone, f.actual_arrival) AS actual_arrival_local,
    f.actual_arrival - f.actual_departure AS actual_duration,
    amount.avg,
    seats.count
from bookings.flights f
left join bookings.airports dep on f.departure_airport = dep.airport_code
left join bookings.airports arr on f.arrival_airport = arr.airport_code
left join (select avg(tf.amount), tf.flight_id from ticket_flights tf group by tf.flight_id) as amount
    on amount.flight_id = f.flight_id
left join (select count(bp.seat_no), bp.flight_id from boarding_passes bp group by bp.flight_id) as seats
    on seats.flight_id = f.flight_id
```

```
1   Hash Left Join  (cost=31384.02..32958.91 rows=33121 width=235)

Planning Time: 0.672 ms
Execution Time: 352.528 ms
```

flights_v + N1 + N2 + N3 + PostgreSQL:


```
create materialized view temp_flights as


refresh materialized view temp_flights


explain analyze
select * from temp_flights
```

1	Seq Scan on temp_flights (cost=0.00..1271.21 rows=33121 width=215)
2	Planning Time: 0.042 ms
3	Execution Time: 3.793 ms

Как же лучше?

Как уже говорили, нет определенного алгоритма, есть конкретная задача.

При появлении избыточности или дублировании атрибутов необходимо контролировать целостность при внесении и модификации данных.

Можно использовать материализованные представления.

Можно нормализованные таблицы хранить на одном сервере, а денормализованные на другом и через триггеры контролировать целостность данных.

OLAP

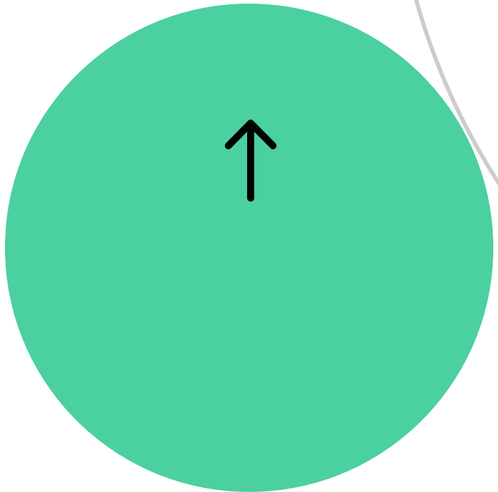
3

Николай Хащанов

Продвинутый SQL

нетология

OLAP - Online Analytical Processing - оперативный анализ данных.



OLAP.

В 1993 г. Е. Ф. Кодд сформулировал 12 определяющих принципов OLAP. Позже его определение было переработано в тест FASMI, требующий, чтобы OLAP-приложение предоставляло возможности быстрого анализа разделяемой многомерной информации.

Пользователь получает интуитивно понятную модель данных, организуя их в виде многомерных кубов (Cubes). Осями многомерной системы координат служат основные атрибуты анализируемого бизнес-процесса.

Для визуализации данных, хранящихся в кубе, применяются, как правило, привычные двумерные представления в виде таблиц, имеющие сложные иерархические заголовки строк и столбцов.

FASMI.

Fast (Быстрый) – анализ должен производиться одинаково быстро по всем аспектам информации. Приемлемое время отклика – 5 с или менее.

Analysis (Анализ) – должна быть возможность осуществлять основные типы числового и статистического анализа, предопределенного разработчиком приложения или произвольно определяемого пользователем.

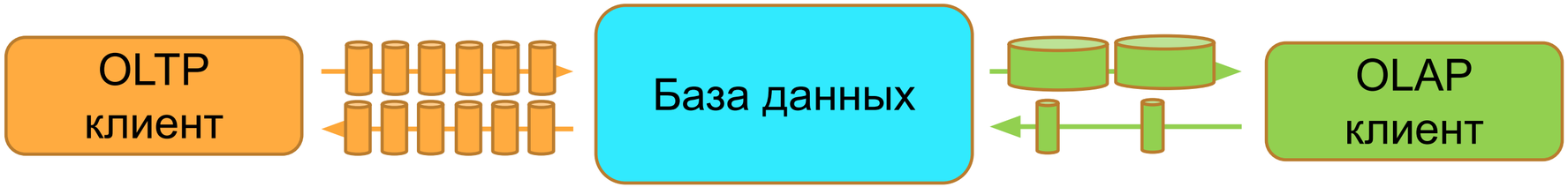
Shared (Разделяемой) – множество пользователей должно иметь доступ к данным, при этом необходимо контролировать доступ к конфиденциальной информации.

Multidimensional (Многомерной) – это основная, наиболее существенная характеристика OLAP.

Information (Информации) – приложение должно иметь возможность обращаться к любой нужной информации, независимо от ее объема и места хранения.

OLTP + OLAP.

Защитная зона
для интеграции
видео спикера



- Большое количество транзакций
- Повышенные требования к скорости
- Низкая нагрузка

- Небольшое количество транзакций
- Повышенные требования к пропускной способности
- Высокая нагрузка

OLTP VS OLAP.

Параметры	OLAP	OLTP
Резервное копирование и восстановление	Требуется редко.	Постоянно требуется полное или инкрементное резервное копирование.
Данные	Используется концепция хранилища данных и представляет собой онлайн-систему управления запросами к базе данных.	Опирается на традиционные системы управления базами данных.
Целостность данных	Из-за редких транзакций не обязательно требовать соблюдения целостности данных.	Из-за частых транзакций требуется контролировать целостность данных
Основное намерение	Система OLAP позволяет пользователю извлекать многомерные данные, которые можно анализировать и использовать для принятия решений.	Система OLTP фокусируется на добавлении, модификации и удалении информации из базы данных.
Требования к пространству для хранения	Требуют значительного пространства для хранения из-за наличия структур агрегирования и исторических данных.	Требует меньшего пространства для хранения и еще меньше места для систем с архивными историческими данными.

OLTP VS OLAP.

Параметры	OLAP	OLTP
Тип системы	OLAP - это онлайн-система поиска и анализа данных.	OLTP - это система онлайн-транзакций.
Транзакции	Отрабатывают долго, но запускаются редко.	Отрабатывают быстро, но частота транзакций высокая.
Сложность запроса	Сложные запросы	Простые запросы
Нормализация	Таблицы не нормализованы	Таблицы нормализованы
Время отклика	От секунд до минут	В миллисекундах

«Разделяй и властвуй» - не верно!

Разделять OLTP и OLAP системы и выбирать какую-то одну конкретную не верно. Каждая система рассчитана под свои задачи и требуется для получения своего результата.

При этом надо понимать, что эти две системы взаимосвязаны, OLTP система позволяет собирать данные, а OLAP система позволяет анализировать полученные данные и на анализе этих данных принимать верные решения.

Как организовать структуру уже говорили ранее.

Дальше все в ваших руках и конкретных ТЗ.

Итоги

4

Николай Хащанов

Подведем итог:

- Поговорили о зависимостях
- Разобрались для чего нужна денормализация
- Сравнили OLTP и OLAP системы

Домашнее задание

5

Николай Хащанов

Продвинутый SQL

нетология

Задание 1. База данных dvd-rental.

Необходимо денормализовать таблицу с платежами payment.

Сейчас на основе этой таблицы сложно сформировать отчет, в котором должны быть:

- идентификатор платежа
- почтовая информация о клиенте
- информация по фильмам, которые были оплачены
- сумма платежей за месяц (если платеж был в январе, то сумма всех платежей за январь и т.д.)
- сумма платежей за неделю (если платеж был 22.09.20, то сумма всех платежей с 21.09.20 по 27.09.20 включительно и т.д.)
- информация по магазину, в котором была произведена продажа
- фамилия и имя продавца

Дополнительное задание.

Выполните основное задание, создав физическую таблицу и написав все необходимые функции и триггеры для автоматического внесения данных в новую денормализованную таблицу.

Полезные ссылки

6

Николай Хащанов

Аксиомы Армстронга:

https://en.wikipedia.org/wiki/Armstrong%27s_axioms

12 правил Кодда:

https://ru.wikipedia.org/wiki/12_%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB_%D0%9A%D0%BE%D0%B4%D0%B4%D0%B0

Введение в функциональные зависимости:

https://habr.com/ru/company/JetBrains-education/blog/473882/

Системы OLAP - это что такое?

https://yandex.ru/turbo/businessman.ru/s/sistemyi-olap---eto-chto-takoe.html

Спасибо за внимание!

Николай Хащанов

нетология