Ота Никита Терентьевич

# 1. Теоретический вопрос

## Секционирование

## Ответ

Секционирование – разбиение таблицы на фрагменты, хранящиеся

в разных местах.

Называется шардингом, если разбиение осуществляется на разные компьютеры.

Есть два вида секционирования:

* Вертикальное

Таблица разбивается по столбцам. Реализуется посредством проекции и соединения.

Например:

Students ( SId , GId , FirstName , LastName , PassSeries ,

PassNo , PassIssued , Photo ) – исходная таблица

Разобьём на:

StudentData ( SId , GId , FirstName , LastName )

StudentPasses ( SId , PassSeries , PassNo , PassIssued )

Student Photos ( SId , Photo )

Обращение к фото (StudentPhotos) происходит значительно реже, чем к основ-

ным данным студента (StudentData). Также таблица с персональными данными

(StudentPasses) требует повышенных прав доступа.

Создадим также представление для работы с исходной таблицей.

create view Students as StudentData

natural join StudentPasses

natural join StudentPhotos;

**Преимущества**

Отделение данных, к которым часто обращаются, от тех, к которым обраща-

ются редко.

Защита информации.

Поддерживается во многих СУБД для CLOB и BLOB.

**Недостатки**

Нет специальной поддержки в СУБД. Считается, что проекции и соединения

для указанных целей достаточно.

Зависимость от представления (соединенных данных). Некоторые СУБД накладывают ограничения на представления, например, запрещают создавать

внешние ключи на них.

Необходимость обновляемых представлений также не гарантирована.

* Горизонтальное

Таблица разделяется по строкам. Корректно, когда каждая строка попадает ровно в одну секцию. Реализуется посредством фильтрации и объединения.

Например:

Points ( SId , CId , Points , Date ) – исходная таблица

Разобьём по Date

Points2021−1 – оценки за весенний семестр 2021,

Points2021−2 – оценки за осенний семестр 2021,

Points2020−1 – оценки за весенний семестр 2020

**Преимущества**

Отделение данных, к которым часто обращаются, от тех, к которым обраща-

ются редко. Например, чаще всего старые данные нужны реже новых.

При уменьшении размера секции уменьшается размер индекса.

Требуется встроенная поддержка.

Прозрачно для пользователя.

**Недостатки**

В некоторых случаях может приводить к замедлению работы.

**Методы секционирования**

* Простые.

– По диапазонам значений,

– По значениям,

– По хешу.

* По выражению. Поддерживаются реже.
* Составные.

– По диапазонам и хешу

Например, секционирования по диапазонам

create table Points ( . . . )

partition by range ( Date ) (

partition pHist values less than ’ 2021 −02 −01 ’ ,

partition p2021s1 values less than ’ 2021 −07−01 ’ ,

partition p2021s2 values less than ’ 2022 −02−01 ’ ,

partition pFuture values less than maxvalue

);

Секционирование по значениям.

create table Points ( . . . )

partition by list ( Term ) (

partition p Hist values in (

’ t2020 −1 ’ , ’ t2020 − 2 ’ , …),

partition p2021s1 values in

( ’ t2021 −1 ’ ) ,

partition p2021s2 values in

( ’ t2021 −2 ’ ) ,

partition pFuture values in

( ’ t2022 −1 ’ )

);

При таком подходе секция может быть не определена при записи.

Insert into Points ( Term ) values ( ’ t2001 −1 ’ ) ;

При чтении из несуществующей секции будет получен пустой результат.

Секционирование по хешу.

Хешируется по набору столбцов. Работает эффективно при хорошей и быстрой

хеш-функции.

create table Points ( . . . )

partition by hash ( Term )

partitions 4;

Секционирование по выражению.

Зависит от определенной в БД функции.

create table Points ( . . . )

partition by year ( Date ) (

partitionp Hist values less than 2021 ,

partition pCurrent values less than 2022 ,

partition pFuture values less than maxvalue

);

Составное секционирование.

Секции разбиваются на подсекции, возможно, по разным атрибутам.

create table Points ( . . . )

partition by year ( Date )

subpartition by hash ( Term ) (

partition History values

less than 2021 (

subpartition History1

, subpartition History2

),

partition Current values

less than 2022 (

subpartition Current1

, subpartition Current2

)

);

**Управление секциями**

Синтаксис может отличаться в зависимости от СУБД.

Удаление секции

alter table <table> drop partition <section>;

Разбиение секции

alter table <table> reorganise <section> into ( . . . ) ;

Перехеширование

−− Add count of partitions

alter table <table> add partition <count>;

−− Delete count of partitions

Alter table <table> coalesce partition <count>;

Оптимизатор владеет информацией о секциях.

В частности, где какие данные находятся.

select \* from Points where Date = ’2021−12−06’

При таком запросе выборка будет производиться только из секции 2021 года.

**Секционирование и индексы**

Можно определить следующие индексы:

* Локальный – один на секцию. Ускорение при выборе секций.
* Глобальный – один на таблицу. Также ускорение при выборе секций.
* Секционированный – разбит на секции. Обеспечивает согласованное секционирование (могут помочь при склеивании секций).

**Секционирование и ключи**

Лучше всего, если множество столбцов, по которым идет секционирование, образует подключ.

Еще лучше – подключ всех ключей (например, если внешние ключи на таблицу ссылаются на разные ее ключи).

# 2. Теоретическое задание

## **Построение запросов в терминах исчисления кортежей**

Даны отношения: Поставщик (НомерПоставщика, НазваниеПоставщика, ГородПоставщика) , Деталь (НомерДетали, Цвет, Вес, ГородДетали), Цены (НомерПоставщика, НомерДетали, Цена). Требуется записать в терминах исчисления кортежей следующие запросы:

1. Детали, поставляемые поставщиком, находящимся в том же городе, в котором нужна деталь.

2. Детали, которые поставляют все поставщики в заданном городе.

## Решение

1

Деталь where ∀Поставщик (

∃Цены (

Деталь.ГородДетали = Поставщик.ГородПоставщика ∧

Цены.НомерПоставщика = Поставщик.НомерПоставщика ∧

Цены.НомерДетали = Деталь.НомерДетали

)

)

2

Деталь where ∀Цены (

∀Поставщик (

Поставщик.ГородПоставщика = ЗаданныйГород ∧

Цены.НомерПоставщика = Поставщик.НомерПоставщика ∧

Цены.НомерДетали = Деталь.НомерДетали

)

)

# 3. Практическое задание

Задание ограничений целостности  
Задайте ограничения целостности так, чтобы в парах мужчина не мог быть female figure skater и наоборот.

## Решение

В отношении People отсутствует маркировка гендерного признака человека.

Из-за этого получается, что нужно либо создавать новые функции для определения гендера человека (что представляет крайнюю сложность, например, в китайских именах, где имя может быть как мужское, так и женское), либо добавить новый атрибут гендера для создания нового ограничения целостности.

Давайте добавим новый атрибут у отношения People.

## Отношение People

Атрибуты:

* PersonId — уникальный идентификатор человека.
* PersonName — имя человека.
* BirthDate — дата рождения человека.
* Country — страна.
* Gender — пол человека.

Функциональные зависимости:

* PersonId → PersonName
* PersonId → BirthDate
* PersonId → Country
* PersonId → Gender

Ключи:

* {PersonId}

create type GenderType as enum (

'Male',

'Female'

);

create table People (

PersonId int not null generated always as identity,

PersonName varchar(128) not null,

BirthDate date not null,

Country varchar(32) not null,

Gender GenderType not null,

constraint People\_PK primary key (PersonId)

);

Создадим новое ограничение целостности для отношения Pairs

create assertion assert

check

(

1 = not exists (

select Gender

from Pairs

inner join People P1 on Pairs.PersonMId = P1.PersonId

inner join People P2 on Pairs.PersonFId = P2.PersonId

where P1.Gender != 'Male'

or P2.Gender != 'Female'

)

);