13. Внутренняя организация реляционных СУБД

Особенности данных СУБД:

- 2 уровня представления - уровень, на котором производится непосредственное управление данными во внешней памяти (единицами являются физические сектора, кластеры, буферы или цепочки буферов ОЗУ) и языковой уровень - язык SQL (единица отношения, кортежи, атрибуты);

- наличие специальных отношений - каталогов, которые содержат информацию об именовании объектов, пути доступа к объекту и конкретные средства этих объектов (владелец, ключ индекса и т.д.);

- регулярность структур данных;

- необходимо поддерживать избыточность данных: резервные копии для поддержания безопасности работы (журнальные структуры) и восстановлении при сбоях.

В СУБД можно выделить несколько структур (объектов во внешней памяти):

- стоки отношений - основная часть БД, видимая пользователю;

- управляющие структуры - индексы:

1) по инициативе пользователя;

2) по инициативе верхнего уровня для быстрого доступа;

3) по инициативе системы нижнего уровня для размещения данных и автоматической поддержки индексов при изменении данных(!);

- журнальная информация, фиксирующая все изменения в БД;

- служебная информация для обеспечения работы нижнего уровня (о свободном дисковом пространстве).

**Хранение отношений**

существует 2 подхода к физической организации хранения:

1. кортежное хранение(!) - более распространённое;

2. поатрибутное хранение.

При 1 обеспечивается быстрый доступ к целому кортежу, но т.к. в разных кортежах значения атрибутов могут дублироваться, то такая БД может создать лишнюю дублируемую информацию, и если требуется часть кортежа, то может потребоваться лишнее обращение к памяти. При 2 столбец хранится с исключёнными дубликатами  требуется меньше внешней памяти; при операциях соединения есть преимущества; но требуются дополнительные операции, когда требуется целый кортеж.

**Индексы**  
главное назначение - обеспечить эффективный прямой доступ к кортежу по ключу. Обычно определяется для одного отношения и строится по значению какого-либо атрибута. Если этот атрибут является ключом отношения, то индекс должен быть уникальным, т.е. не содержать дублированных значений ключа.  
Помимо таких индексов рассматриваются индексы, связанные с несколькими отношениями (мультииндексы). В них ключ индекса является составным, по отдельным составляющим его составляются кортежи из разных отношений. Такие мультииндексы используются для сложных операций (эквисоединение).  
Общая идея организации - хранение упорядоченного списка ключей с привязкой к каждому ключу идентификатора кортежа или списка идентификаторов кортежа, если ключи индекса не уникальны.

**Журнальная информация**

**В журнале должны хранится все** изменения, происходящие в БД. С технической точки зрения журнал должен представлять файл последовательного доступа с записями переменного размера. Конкретная структура записей является частным делом каждой реализации. Журнал недоступен пользователю, а доступен самой СУБД. Из соображения надёжности существует несколько копий журнала на разных носителях.

5 Служебная информация

Служебная информация необходима для корректной работы с внешней памятью. В число файлов служебной информации входят:

- внутренние каталоги, которые описывают свойства объектов БД;

- описатели свободной и занятой памяти в терминах страницы;

- информация, которая связывает страницы одного отношения (если в одном файле внешней памяти располагается несколько отношений, то страницы каждого отношения должны быть связаны между собой).

14. Индексы БД

Индексы - это специальные структуры в базах данных, которые позволяют ускорить поиск и сортировку по определенному полю или набору полей в таблице, а также используются для обеспечения уникальности данных. Проще всего индексы сравнить с указателями в книгах. Если нет указателя, то нам придется просмотреть всю книгу, чтобы найти нужное место, а с указателем то же действие можно выполнить намного быстрее.

Обычно чем больше индексов, тем больше производительность запросов к базе данных. Однако при излишнем увеличении количества индексов падает производительность операций изменения данных (вставка/изменение/удаление), увеличивается размер БД, поэтому к добавлению индексов следует относиться осторожно.

Некоторые общие принципы, связанные с созданием индексов:

* индексы необходимо создавать для столбцов, которые используются в джойнах, по которым часто производится поиск и операции сортировки. При этом необходимо учесть, что индексы всегда автоматически создаются для столбцов, на которые накладывается ограничение primary key. Чаще всего они создаются и для столбцов с foreign key (в Access - автоматически);
* индекс обязательно в автоматическом режиме создается для столбцов, на которые наложено ограничение уникальности;
* лучше всего индексы создавать для тех полей, в которых - минимальное число повторяющихся значений и данные распределены равномерно. В Oracle есть специальные битовые индексы для столбцов с большим количеством повторяющихся значений, в SQL Server и Access такой разновидности индексов не предусмотрено;
* если поиск постоянно производится по определенному набору столбцов (одновременно), то в этом случае, возможно, есть смысл создать композитный индекс (только в SQL Server) - один индекс для группы столбцов;
* при внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу. В результате индекс может быть сильно фрагментирован, что сказывается на производительности. Периодически следует проверять степень фрагментации индексов и дефрагментировать их. При загрузке большого количества данных иногда есть смысл вначале удалить все индексы, а после завершения операции создать их заново;
* индексы можно создавать не только для таблиц, но и для представлений (только в SQL Server). Преимущества - возможность вычислять поля не в момент запроса, а в момент появления новых значений в таблицах.

В SQL Server предусмотрено два типа индексов: **кластерные** и **некластерные**.

**Кластерный индекс** в таблице может быть только один. Проще всего сравнить таблицу, на которую наложен такой индекс, с телефонным справочником: все записи в данной таблице упорядочены по кластерному индексу. Относиться к выбору поля для кластерного индекса следует очень осторожно - например, если в эту таблицу часто производится вставка данных, а кластерный индекс наложен не на поле с автоприращением, то вполне может получиться так, что нам часто придется вставлять новые записи в середину таблицы. Результат - большое количество операций page split, фрагментация таблицы и, как следствие, серьезное падение производительности (за счет фрагментации и за счет того, что само по себе page split - достаточно ресурсоемкая операция. По умолчанию кластерный индекс создается для поля первичного ключа, и, учитывая это, лучше делать первичный ключ числовым полем с автоприращением.

**Некластерный индекс** больше всего похож на указатель в конце книги. Для таблицы можно создавать таких индексов очень много (можно даже по нескольку для каждого столбца, но большой пользы это не приносит).

15. Связи между отношениями БД

Между отношениями БД могут быть установлены *связи* или ассоциации, показывающие, каким образом отношения соотносятся или взаимодействуют между собой.

*Степень связи* – количество типов отношений, которые охвачены данной связью.

Между отношениями предметной области могут иметь место:

−    бинарные связи (между двумя отношениями или между отношением  и ем же самим - рекурсивная связь);

−    тренарные связи (между тремя отношениями);

−    и в общем случае - n-арные связи.

На практике чаще всего встречаются связи со степенью два, то есть бинарные связи.

*Рекурсивная связ*ь – связь, в которой одни и те же отношения участвуют несколько раз в разных ролях.

         Каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;

         Каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ ("ЧЕЛОВЕКОВ").

Cреди бинарных связей существуют три фундаментальных вида связи:

−    один к одному (1:1);

−    один ко многим (1:M);

−    многие ко многим (M:N).

*Связь один к одному* (1:1) существует, когда один экземпляр одного отношения связан с единственным экземпляром другого отношения (студент может не "получать" стипендию, а может получать).

*Связь один ко многим* (1:M) существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан только с одним экземпляром первого отношения (квартира может пустовать, в ней может жить один или несколько жильцов).

Связь многие ко многим (М:N) существует, когда один экземпляр одного отношения связан с одним или более экземпляром другого отношения и каждый экземпляр второго отношения связан с одним или более экземпляром первого отношения.

В связи между данными, хранимыми в разных отношениях, в реляционной БД устанавливаются с помощью ключей.

*Первичный ключ* (primary key) - это столбец или группа столбцов, однозначно определяющие кортеж (запись). *Ключ* или *возможный ключ* – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр отношения. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать кортеж по оставшимся.

***Внешний ключ (***foreign key) - это столбец или группа столбцов, ссылающиеся на столбец или группу столбцов другой таблицы. Таблица, в которой находится внешний ключ называется *дочерней*. Таблица, на которую ссылается внешний ключ, называется *родительской таблицей*, а столбцы, на которые ссылается внешний ключ - родительским ключом. Родительский ключ должен быть первичным или уникальным ключом, значения же внешнего ключа могут повторяться хоть сколько раз. То есть с помощью внешних ключей поддерживаются связи "один ко многим". Типы данных (а в некоторых СУБД и размерности) соответствующих столбцов внешнего и родительского ключа должны совпадать.

И самое главное. Все значения внешнего ключа должны совпадать с каким-либо из значений родительского ключа.

16. Классификация ограничений целостности

Ограничения целостности можно классифицировать несколькими способами:

* По способам реализации.
* По времени проверки.
* По области действия.

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

Декларативная поддержка ограничений целостности.

Процедурная поддержка ограничений целостности.

**Декларативная поддержка ограничений целостности** заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

***Процедурная поддержка ограничений целостности*** заключается в использовании триггеров и хранимых процедур.

**Классификация ограничений целостности по времени проверки**

По времени проверки ограничения делятся на:

* Немедленно проверяемые ограничения.
* Ограничения с отложенной проверкой.

*Определение 6*. ***Немедленно проверяемые ограничения*** проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Например, проверка уникальности потенциального ключа проверяется в момент вставки записи в таблицу. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности, обычно откатывается.

***Ограничения с отложенной проверкой*** проверяется в момент фиксации транзакции оператором COMMIT WORK. Внутри транзакции ограничение может не выполняться. Если в момент фиксации транзакции обнаруживается нарушение ограничения с отложенной проверкой, то транзакция откатывается. Примером ограничения, которое не может быть проверено немедленно является ограничение из примера 1. Это происходит оттого, что транзакция, заключающаяся во вставке нового сотрудника в таблицу PERSON, состоит не менее чем из двух операций - вставки строки в таблицу PERSON и обновления строки в таблице DEPART. Ограничение, безусловно, неверно после первой операции и становится верным после второй операции.

#### Классификация ограничений целостности по области действия

По области действия ограничения делятся на:

* Ограничения домена
* Ограничения атрибута
* Ограничения кортежа
* Ограничения отношения
* Ограничения базы данных

***Ограничения целостности домена*** представляют собой ограничения, накладываемые только на допустимые значения домена. Фактически, ограничения домена обязаны являться частью определения домена

***Ограничение целостности атрибута*** представляют собой ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене. Ограничение атрибута в точности совпадают с ограничениями соответствующего домена. Отличие ограничений атрибута от ограничений домена в том, что ограничения атрибута *проверяются*.

***Ограничения целостности кортежа*** представляют собой ограничения, накладываемые на допустимые значения *отдельного* кортежа отношения, и *не являющиеся* ограничением целостности атрибута. Требование, что ограничение относится к *отдельному* кортежу отношения, означает, что для его проверки *не требуется* никакой информации о других кортежах отношения.

***Ограничения целостности отношения*** представляют ограничения, накладываемые только на допустимые значения *отдельного* отношения, и *не являющиеся* ограничением целостности кортежа. Требование, что ограничение относится к отдельному отношению, означает, что для его проверки не требуется информации о других отношениях (в том числе не требуется ссылок *по внешнему ключу* на кортежи *этого же* отношения).

***Ограничения целостности базы данных*** представляют ограничения, накладываемые на значения двух или более связанных между собой отношений (в том числе отношение может быть связано само с собой).

17. Общая интерпретация реляционных операций

18. Транзакции в реляционных БД. Модели транзакций

19. Проблемы параллелизма транзакций

20. Способы и механизмы решений проблем параллелизма транзакций

21. Язык реляционных баз данных SQL. Основные возможности и функции

22. Реляционная алгебра. Операции объединения и пересечения отношений

23. Реляционная алгебра. Операция вычитания отношений. Декартово произведение

24. Реляционная алгебра. Проекция отношения. Операция выборки

25. Реляционная алгебра. Соединение отношений. Тэта-соединение, эквисоединение