**72. СУБД. Функции СУБД. Типовая организация СУБД**

СУБД – комплекс программ, предназначенная для управления БД (организации информации, помещения ее в таблицы, манипулирования ею). Основные функции СУБД:

* непосредственное управление во внешней памяти, обеспечение необходимых структур внешней памяти как для хранения данных, вводящихся в БД, так и для служебных целей (индексы). В развитых СУБД пользователь не знает особенностей организации внешней памяти.
* управление буферами оперативной памяти
* управление транзакциями (Транзакция – последовательность операций, рассматриваемых СУБД как единое целое, переводящее БД из одного целостного состояния в другое.
* журнализация СУБД должна быть в состоянии восстановить последнее состояние БД после аппаратного или программного сбоя. Для восстановления БД необходимо иметь информацию. Существует журнал – особая часть БД, недоступная пользователю СУБД, в который поступают записи обо всех изменениях основной части БД
* поддержка языков БД. В ранних – языки манипулирования, организация схемы СУБД, в современных – SQL

Типовая структура современной СУБД.

Организация соответствует набору функций СУБД. Логически в современной реляционной СУБД можно выделить наиболее внутреннюю часть - ядро СУБД (часто его называют Data Base Engine), компилятор языка БД (обычно SQL), подсистему поддержки времени выполнения, набор утилит. В некоторых системах эти части выделяются явно, в других - нет, но логически такое разделение можно провести во всех СУБД.

Ядро СУБД отвечает за управление данными во внешней памяти, управление буферами оперативной памяти, управление транзакциями и журнализацию. Соответственно, можно выделить такие компоненты ядра (по крайней мере, логически, хотя в некоторых системах эти компоненты выделяются явно), как менеджер данных, менеджер буферов, менеджер транзакций и менеджер журнала.

Функции этих компонентов взаимосвязаны, и для обеспечения корректной работы СУБД все эти компоненты должны взаимодействовать по тщательно продуманным и проверенным протоколам. Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую и используемым в программах, производимых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах БД. Ядро СУБД является основной резидентной частью СУБД. При использовании архитектуры "клиент-сервер" ядро является основной составляющей серверной части системы.

Основной функцией компилятора языка БД является компиляция операторов языка БД в некоторую выполняемую программу. Основной проблемой реляционных СУБД является то, что языки этих систем (а это, как правило, SQL) являются непроцедурными, т.е. в операторе такого языка специфицируется некоторое действие над БД, но эта спецификация не является процедурой, а лишь описывает в некоторой форме условия совершения желаемого действия. Поэтому компилятор должен решить, каким образом выполнять оператор языка прежде, чем произвести программу. Результатом компиляции является выполняемая программа, представляемая в некоторых системах в машинных кодах, но более часто в выполняемом внутреннем машинно-независимом коде. В последнем случае реальное выполнение оператора производится с привлечением подсистемы поддержки времени выполнения, представляющей собой, по сути дела, интерпретатор этого внутреннего языка.

Наконец, в отдельные утилиты БД обычно выделяют такие процедуры, которые слишком накладно выполнять с использованием языка БД, например, загрузка и выгрузка БД, сбор статистики, глобальная проверка целостности БД и т.д. Утилиты программируются с использованием интерфейса ядра СУБД, а иногда даже с проникновением внутрь ядра.

**73. Этапы разработки базы данных.**

БД - это совокупность сведений о каких-либо объектах реального мира, предназначенных для совместного использования.

На стадии проектирования информационной системы проектировщик должен сделать:

1. обследовать предметную область
2. определить объекты и их атрибуты
3. установить все структуры и запросы, установить структурные и иерархические связи между объектами, запросные связи, начертить схему проекта
4. выработать технологию обслуживания информационной системы
5. выбрать технические и инструментальные средства для реализации проекта, реализовать проект
6. протестировать проект

На этапе 1 необходимо внимательно выслушать заказчика, необходимо прояснить следующие вопросы:

* Каковы границы предметной области?
* Будет ли предметная область изменяться?
* Каков перечень фрагментов предметной области?
* Какая информация, с какой степенью детальности нужна пользователям каждого фрагмента?
* Определить перечень пользователей и их информационные потребности
* Какие процессы передачи, обработки данных происходят в каждом фрагменте, с какой интенсивностью
* Какие существуют технологии накопления и обработки информации?
* Какие технические средства для реализации системы будут использоваться?
* Определить требования технологического функционирования системы, ознакомиться со всеми входными и выходными документами.

Основные этапы разработки структуры БД.

1. Инфологическая (концептуальная модель). Это описание предметной области, выполненное без ориентации на используемые в дальнейшем программы и технические средства. Транслируется в даталогическую модель. Требования к этой модели:

* адекватность отражения предметной области
* непротиворечивость
* недопустимость неосознанной трактовки модели
* легкая расширяемость модели (изменение)

Компоненты модели:

* описание объектов предметной области и связей между ними (знаковая система или ER-модель)
* алгоритмические связи показателей
* описание информационных потребностей пользователей
* ограничения целостности

1. Даталогическая модель – трансляция концептуальной модели в логическую модель данных на языке описания данных
2. Даталогическая модель (логическая) - описание логической связи между элементами данных без отношения к среде их содержания. Строиться даталогическая модель в терминах и информационных единиц допустимых конкретной СУБД. Описание логической структуры данных называют схемой. Даталогическая модель отображается затем в физической памяти.

Критерии качества логической модели данных.

* адекватность БД предметной области
* легкость разработки и сопровождения БД
* скорость выполнения операции обновления данных
* скорость выполнения операции выборки данных

1. Физическая модель. Определяет используемые запоминающие устройства. Способы физической организации данных, выбор метода доступа к данным, управление свободной памятью, описание физической структуры данных называется **схемой хранения**.

**74. ER - модель. Основные понятия.**

“Сущность – связь” (Entity - Relationship). Отображает инфологическую модель.

Сущность – класс однотипных объектов, информация о которых должна быть учтена в модели.

Каждая сущность в модели показывается:

* Экземпляр сущности – конкретный представитель данной сущности. Экземпляры сущности различны, они должны иметь различные свойства. Пример: студент Иванов, 2 курс, 2 группа
* Атрибут – поименованная характеристика, являющаяся некоторым свойством сущности или информационное отображение свойств объекта. Выражается существительным в ед. ч. Пример: ФИО.
* Ключ сущности – неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточный значит, что удаление любого атрибута нарушит уникальность ключа.
* Связь – некоторая ассоциация между 2 сущностями. Позволяет по одной сущности находить другую сущность, связанную с ней.

3 типа связей:

* 1:1 (один к одному). Означает, что один экземпляр первой сущности связан с одним экземпляром второй сущности.
* 1:M (один ко многим). Один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Сущность со стороны 1 – родитель, со стороны M – потомок.
* M:M (многие ко многим). Каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.
* Модальность - Обязательность или необязательность связи. Пример: при поступлении в вуз абитуриент обязан знать иностранный язык, но никто не обязан знать более 4х языков.
* Класс объектов – совокупность объектов, обладающих одинаковым набором свойств.

**75. Общая характеристика реляционной модели данных. Типы данных. Простые типы данных. Структурированные типы данных. Ссылочные типы данных. Типы данных, используемые в реляционной модели. Домены. Отношения. Атрибуты. Кортежи.**

Кодд предложил реляционную модель в 1970. Дейт дает наиболее распространенную трактовку этой модели. Реляционная модель (РМ) данных состоит из 3 частей:

1. структурная. Описывает, как объекты рассматриваются реляционной моделью. Единственной структурной единицей являются формализованные отношения.
2. целостная. Описывает ограничения, которые должны выполняться для отношений
3. манипуляционная. Описывает способы манипулирования реляционными данными (реляционная алгебра и реляционные исчисления)

Типы данных:

* + простые (логические, строковые, численные)
  + структурированные (массивы, записи)
  + ссылочные (указатели)

Для реляционной модели тип данных не важен, важно, чтобы тип был простым.

**Домен** – набор значений элементов, данных одного типа, отвечающих поставленным условиям. D={n∈N: n>=18 and n<=60}Свойства домена:

* уникальное имя
* определен на простом типе данных или на другом домене
* может нести логическое условие для описания подмножества данных, допустимых для данных домена.
* Домен имеет определенную смысловую нагрузку. Этим он отличается от понятия подмножества.

**Атрибут отношения** – пара вида <Имя атрибута : Имя домена>. Имя атрибута должно быть уникальным.

**Отношение** *R* определенное на множестве доменов D1, D2, …Dn содержит 2 части: заголовок отношения - фиксированное количество атрибутов отношения <A1:D1><A2:D2>…<An:Dn>. Тело отношения содержит множество кортежей отношения. Отношение “Сотрудники”: Сотрудники (Номер сотр, Фамилия, Зарплата, Номер отдела).

Каждый**кортеж отношения**представляет собой множество пар вида: <имя атрибута: значение атрибута> Отношение записывается в виде R(<A1:D1><A2:D2>…<An:Dn>); R(A1, A2,…,An) Пример:

(1, Иванов, 1000)

(2, Петров, 2000)

Количество атрибутов в отношении называется *степенью отношения* (n – арное отношение). Мощность множества кортежей называется *мощностью отношения*.

Отношения представляются в виде таблицы, но не являются таблицами.

Схема реляционной БД является набор заголовков отношений.

**76. Свойства отношений. Потенциальные ключи. Целостность сущностей. Внешние ключи. Целостность внешних ключей. Стратегия поддержания ссылочной целостности.**

Свойства отношений:

* в отношении нет одинаковых кортежей
* кортежи не упорядочены
* атрибуты не упорядочены
* все значения атрибутов атомарны

Реляционная БД – набор взаимосвязанных отношений.

**Потенциальные ключи**– пусть дано отношение R, подмножество атрибутов K отношения R называют потенциальным ключом. Если K обладает свойствами:

* свойство уникальности
* свойство неизбыточности - никакое подмножество из K не обладает свойством уникальности.

Потенциальный ключ может быть простым и составным (сложным). Отношение может иметь несколько потенциальных ключей, один из которых первичный, все остальные альтернативные. Не понимая смысла данных, невозможно судить о том, какая совокупность атрибутов будет являться потенциальным ключом. Потенциальный ключ является средством идентификации объектов предметной области. Потенциальный ключ является единственным средством адресации на уровне кортежей, поэтому значения этих идентификаторов не могут содержать неизвестные значения.

|  |  |
| --- | --- |
| ID\_o | Tel |
| 1  2 | 222  333 |

Правило целостности сущностей:

Атрибут, входящий в состав потенциального ключа не может принимать NULL значения. Пусть дано отношение R. Подмножество атрибутов FK отношения R называют **внешним ключом**, если существует отношение S с потенциальным ключом. K, и каждое значение FK в отношении R всегда совпадает со значением K для некоторого кортежа из S, либо является NULL значением. Отношение S- родительское, отношение R- дочернее. Внешние ключи не обладают свойством уникальности.

|  |  |  |
| --- | --- | --- |
| ID | Name | ID\_otd |
| 1  2  3 | Ваня  Петя  Вася | 1  2  1 |

ID\_otd – внешний ключ по отношению к ID\_o.

Правила целостности внешних ключей:

Внешние ключи должны быть согласованными, для каждого значения внешнего ключа должно существовать значение потенциального ключа в родительском отношении.

Нарушение ссылочной целостности:

* Обновление кортежа в родительском отношении
* Удаление кортежа в родительском отношении
* Вставка кортежа в дочернее отношение
* Обновление кортежа в дочернем отношении

Стратегии поддержания ссылочной целостности:

* 1. Основные
     + RESTRICT – запрет выполнения операций, приводящих к нарушению целостности
     + CASCADE – разрешать выполнение операций, внося поправки в другое отношение так, чтобы не нарушить ссылочную целостность и сохранить все имеющиеся связи
  2. дополнительные
     + SET NULL – разрешить выполнение операции, некоторые атрибуты изменить на NULL
     + SET DEFAULT – разрешает выполнение требуемой операции но все возникающие некорректные значения внешних ключей заменяются значениями по умолчанию
     + IGNORE – не соблюдать ограничения ссылочной целостности

**77. Реляционные операторы: выборка, проекция, соединение, деление. Реализация на SQL.**

**Реляционная алгебра** – набор операторов, использующих отношения в качестве переменных и возвращающих отношение в качестве результата. R=f(R1,R2,…,Rn)

Выборка (A Where C):

Выборка на отношении A с условием C называется отношение с тем же заголовком, что и у A и телом, состоящим из кортежей, значения атрибутов которых при подстановке в условие C дают значение истина. С представляет собой логическое выражение в которое могут входить атрибуты А и скалярные выражения

Проекция (A[x,y,z]):

Проекция отношения A по атрибутам X,Y,Z, где каждый из этих атрибутов принадлежит A называется отношение с заголовком (X,Y,Z) и телом (x,y,z), состоящим из множества кортежей, таких, для которых в отношении A найдутся кортежи со значением атрибута X=x, Y=y, Z=z

Соединение ((A TIMES B) Where C):

* (A TIMES B) Where С - соединение отношений A и B по условию C
* (A TIMES B) Where X θ Y, θ - оператор сравнения, A[X θ Y]B
* A[X = Y]B – эквисоединие
* A JOIN B естественное соединение - В синтаксисе естественного соединения не указывается по каким атрибутам производится соединение. Оно производится по всем одинаковым атрибутам.

Деление (A DEVID BY B): A – делимое, B – делитель

Пусть данные отношения A(X1,X2,…Xn,Y1,Y2,…Ym) и B(Y1, Y2,…Ym) атрибуты Y являются общими для 2 отношений. Делением отношения A на B называют отношение с заголовком X1 X2 Xn и телом из кортежей (x1,x2,…,xn), таких, что для кортежей (y1,y2,…,ym)∈B в отношении A всегда найдется кортеж (x1,x2,…,xn,y1,y2,…ym)

**78. Реляционные операторы: объединение, пересечение, вычитание, декартово произведение множеств. Реализация на SQL.**

**Реляционная алгебра** – набор операторов, использующих отношения в качестве переменных и возвращающих отношение в качестве результата. R=f(R1,R2,…,Rn)

Объединение:

Объединением двух совместных по типу отношений A и B (A UNION B) называют отношение с тем же заголовком, что и отношение A и B и телом, состоящим из кортежей, принадлежащих или A или B или обоим отношениям. В результаты объединения входят все кортежи А и недостающие кортежи В.

Пересечение:

Пересечением двух совместных по типу отношений A и B (A INTERSECT B) называют отношение с тем же заголовком, что и у отношений A и B и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям.

Вычитание:

Вычитанием двух совместных по типу отношений A и B (A MINUS B) называется отношение с тем же заголовком, что и у отношений A и B и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B

Декартово произведение (A TIMES B):

Декартовым произведением 2 отношений A(A1,A2,…,An) B(B1,B2,…,Bn) называют отношение, заголовок которого является сцеплением заголовков отношений A и B (A1,A2,…,An,B1,B2,…,Bn), а тело состоит из кортежей (a1,a2,…,an,b1,b2,…,bn) отношений A и B, таких что (a1,a2,…,an)∈A и (b1,b2,…,bn)∈B. Мощность произведения равна произведению мощностей. Если есть одинаковые атрибуты, то их необходимо переименовать.

**79. Концептуальная схема выполнения оператора SELECT. Синтаксис оператора выборки данных. Синтаксис соединенных таблиц. Синтаксис условных выражений.**

Концептуальная схема выполнения оператора SELECT.

1. вычисляется прямое декартово произведение всех таблиц, указанных в обязательном разделе FROM, получаем некоторую таблицу A.
2. если присутствует WHERE, то сканируется таблица A, для каждой строки таблицы A вычисляется условие из этого раздела; в результат включаются строки, для которых условие возвращает значение TRUE, результат: таблица B, если WHERE нет, то шаг 3
3. если присутствует GROUP BY, то строки таблицы B группируются в соответствии со списком группировки из раздела, получаем таблицу С. Если нет, то шаг 4.
4. если присутствует HAVING, то группы, не удовлетворяющие условию из этого раздела исключаются, в результате таблица D
5. каждая группа, полученная на шаге 4, генерирует одну строку результата следующим образом: вычисляются все скалярные выражения, указанные в разделе SELECT для каждой группы, вычисляются агрегатные функции, приведенные в разделе SELECT. Если GROUP BY нет, а функции есть, то считается, что имеется одна группа. Если нет агрегатных функций, ни раздела, то считается, что есть столько групп, сколько строк. В результате таблица E содержит столько колонок, сколько элементов приведено в разделе SELECT и столько строк, сколько отобрано групп. Если есть ORDER BY, то строки таблицы упорядочиваются в соответствии со списком упорядочивания.

Синтаксис оператора выборки данных:

Табличное выражение:

[ORDER BY {{имя столбца результата [ASC|DESC|}

|{положит целое [ASC|DESC}]

табл выражение := select выражение

[UNION | EXCEPT | INTERSECT | ALL }

{SELECT выражение | TABLE имя табл | конструктор значений таблицы}

SELECT выр-е:= SELECT [ALL|DISTINCT]

{{скалярное выражение | функция агрегирования |

SELECT выр-е} [AS имя столбца]}…}

|{имя табл | имя корреляции}…}

FROM {имя табл [AS] [имя корреляции]

[(имя столбца,…)]}

|SELECT выр-я [AS] имя корреляции

[(имя столбца,…)]}

| соединенная таблица}…

[WHERE условное выражение]

[GROUP BY [имя табл|имя корреляции}]

имя столбца]…}

[HAVING условное выражение];

Условное выражение вычисляется для каждой строки, являющейся кандидатом в результирующее множество строк. Можно использовать подзапросы. Раздел HAVING содержит условное выражение, вычисляющееся для каждой группы в соответствии со списком группировки в разделе GROUP BY. Если в разделе SELECT присутствует агрегатная функция и присутствует раздел GROUP BY, то агрегатная функция вычисляется для каждой группы отдельно. Если раздел отсутствует, то агрегатная функция вычисляется по всем строкам, удовлетворяющим условному выражению в разделе WHERE. Скалярное выражение – имя столбцов таблицы или функции.

Синтаксис соединенных таблиц:

Соединенная таблица :=

* перекрестное соединение - A CROSS JOIN B - декартово произведение таблицы
* естественное соединение - A NATURAL [тип соединения] JOIN B - произведение по всем столбцам таблиц, имеющим одинаковые имена.
* соединение посредством предиката - A [тип соединения] JOIN B ON предикат - соединение по общим столбцам - BETWEEN | IN | LIKE | IS NULL | EXIST | UNIQUE | MATCH | OVERLAPS
* соединение посредством имен столбцов - A [тип соед-я] JOIN B USING {имя столбца}
* соединение объединения - A UNION JOIN B – обратное к внутреннему соединению, обратно FULL

Тип соединения

INNER - соединяются те строки, для которых найдены совпадения

| LEFT [OUTER] – результат включает все строки из левой таблицы и те из правой таблицы, для которых найдены совпадения. Для строк из A, для которых не найдены совпадения в B, заносятся со значениями NULL.

| RIGHT [OUTER]

| FULL [OUTER] – все строки таблиц, для совпадающих – реальные значения, для несовпадающих – NULL.

Синтаксис условных выражений раздела WHERE:

Условное выражение:= [( ] [NOT]

{Предикат сравнения | BETWEEN | IN | LIKE | NULL | предикат количественного сравнения | EXIST | UNIQUE | MATСH | OVERLAPS | } [ { AND | OR } условное выражение] [ ) ] [IS [NOT] {TRUE | FALSE | UNKNOWN}

BETWEEN – диапазоны значения

IN – проверяет вхождение во множество

EXIST – проверяет существует ли предикат подзапроса

MATСH – проверяет будет ли значение в любой строке совпадать с результатом подзапроса.

**80. Определение функциональной зависимости. Вторая нормальная форма. Третья нормальная форма. Алгоритм приведения к 3НФ.**

Функциональная зависимость – пусть R является отношением множества атрибутов Y функционально зависит от множества атрибутов Х (Х->Y). Когда для любого состояния R во всех кортежах, имеющих одинаковое значение атрибутов Х, значение атрибутов Y так же совпадают.

r1,r2∈R

r1X=r2X следует, что r1Y=r2Y

Y- зависимая часть, Х- детерминант функциональной зависимости.

Пусть дано соотношение R с атрибутами Х и Y. Если каждому значению атрибута Х соответствует ровно одно значение Y, то Y функционально зависит от Х.

Н\_отделения -> телефон

Н\_сотрудника -> фамилия

Н\_проекта-> проект

{Н\_сотрудник,Н\_проект}->Н\_задание

Вторая нормальная форма.

Отношение находится во 2ой норм форме, если оно находиться в 1ой норм форме и отсутствует зависимость неключевых атрибутов от части сложного ключа. Чтобы устранить зависимость неключевых атрибутов от части сложного ключа, компоненты вносятся в отдельное отношение.

Декомпозиция: проекты (H\_пр, проект)

Задание (Н\_сотр, Н\_пр, Н\_задание)

Сотрудники\_отделы(Н\_сотр, фамилия, Н\_отд, телефон)

Третья нормальная форма

Отношение R находиться в 3ей нормальной форме если отношение находиться во 2ой норм форме и все неключевые атрибуты взаимно независимы.

Н\_\_отдел-> телефон

Чтобы устранить взаимозависимость неключевых атрибутов проводиться декомпозиция, при этом независимые атрибуты вносятся в отдельное отношение. Детерминант функциональной зависимости становиться ключом.

Сотрудники (Н\_сотр,фамилия, Н\_отд)

Отделы (Н\_отд, телефон)

Алгоритм привидения отношения к третьей нормальной форме

* + 1. Задается одно или несколько отношений, образующих понятие предметной области.
    2. Если в отношении обнаружена зависимость от части сложного ключа, то приводиться декомпозиция этих отношений на несколько, причем атрибуты которые зависят от части сложного ключа вносятся в отдельное отношение вместе с частью этого ключа. В исходном отношении остаются все ключевые атрибуты.

R(K1,K2, A1,A2,…,An,B1,B2,…,Bm)

{K1,K2}->{ A1,A2,…,An,B1,B2,…,Bm }

{K2}->{ B1,B2,…,Bm }

R1(K1,K2, A1,A2,…,An)

R2(K2, B1,B2,…,Bm)

* + 1. Если в некоторых отношения обнаружена зависимость некоторых неключевых атрибутов от др неключевых атрибутов, то проводиться декомпозиция этих отношений, т.е. не ключевые атрибуты, которые зависят от др не ключевых атрибутов, выносятся в отдельное отношение. В новом отношении ключом становиться детерминант функц зависимости.

R(K,A1,A2,…,An,B1,B2,…,Bm)

K->{ A1,A2,…,An,B1,B2,…,Bm }

{A1…An}->{B1…Bm}

R1(K,A1,A2,…,An)

R2(A1,A2,…,An,B1,B2,…,Bm)

**81. Нормальная форма Бойса-Кодда. Четвертая нормальная форма.**

Поставки

|  |  |  |  |
| --- | --- | --- | --- |
| PNUM | PNAME | DNUM | VOL |
| 1 | Фирма 1 | 1 | 100 |
| 1 | Фирма 1 | 2 | 200 |
| 1 | Фирма 1 | 3 | 300 |
| 2 | Фирма 2 | 1 | 150 |
| 2 | Фирма 2 | 2 | 250 |
| 3 | Фирма 3 | 1 | 1000 |

Считаем что наименование поставщиков является уникальным, каждый поставщик имеет свой уникальный номер.

2 потенциальных ключа {PNUM, DNUM}, {PNAME,DNUM}. Данные хранятся с избыточностью.

PNUM->PNAME

PNAME->PNUM

{PNUM, DNUM}->VOL

{PNUM, DNUM}->PNAME

{PNAME,DNUM}->VOL

{PNAME,DNUM}->PNUM

Отношение относиться в 3ей норм форме поскольку единственный неключевой атрибут зависит от всего ключа сразу. Очевиден его способ декомпозиции: P(PNUM,PNAME) – поставщики, PD(PNUM,DNUM,VOL)- поставки.

Отношение находиться в норм форме Бойса-Кодда, если детерминант всех функций зависимостей являются потенциальными ключами.

Правило: для того, чтобы устранить зависимость от детерминантов, не являющихся потенциальными ключами, проводиться декомпозиция, при этом детерминант и зависимые от них части вносятся в отдельное отношение.

Четвертая нормальная форма

4НФ Пусть необходимо хранить данные об абитуриентах, поступающих в ВУЗ. Каждый абитуриент имеет право сдавать экзамены на несколько факультетов одновременно. Каждый факультет имеет список сдаваемых предметов. Один и тот же предмет может сдаваться на нескольких факультетах. Абитуриент обязан сдавать предмет на факультете, независимо от сдачи этого предмета на другом факультете.

Отношения – Абитуриент.Факультет.Предмет(Абитуриент, Факультет, Предмет)

Декомпозируем отношение. В результате получим:

Абитуриенты (Номер, Фамилия); Факультеты (Номер, Факультет); Предмет (Номер, Предмет); Абитуриент.Факультет.Предмет(Номер\_А, Номер\_Ф, Номер\_П)

Декомпозиция отношения А.Ф.П не может быть выполнена на основе функциональной зависимости. В отношении нет функциональных зависимостей.

Многозначная зависимость.

Пусть дано отношение R(X,Y,Z). Атрибуты Y и Z многозначно зависят от Х, если из того, что в отношении R соединяются кортежи r1=(x,y,z1) r2=(x,y1,z). Следует, что в отношении r содержится и кортеж r3=(x,y,z)

x->>y|z

Факультет(х)->>Абитуриент(y)|Предмет(z)

Для каждого факультета х каждый поступающий абитуриент у сдает один и тот же список предметов z или для каждого факультет каждый сдаваемый на факультете экзамен сдается одним и тем же списком абитуриентов.

Замечание: если в отношении R имеется не менее 3х атрибутов и х->у, то имеется многозначная зависимость x->>y|z

Многозначная зависимость атрибутов у и z от х называется нетривиальной многозначной зависимостью, если не существует х->у и х->z

Теорема Фейджина-Пусть x,y,z – непересекающееся множество атрибутов отношения R. Декомпозиция отношения R на проекции R1[x,y] и R2[x,z] является декомпозицией без потерь, когда имеется многозначная зависимость атрибутов y,z от x.

Отношение R находиться в 4ой норм форме тогда и только тогда, когда отношения находиться в нормальной форме Бойса-Кодда и не содержит нетривиальных многозначных зависимостей.

**82. Корректность процедуры нормализации - декомпозиция без потерь. Теорема Хеза.**

При декомпозиции из одного отношения получают одно или более отношений, каждое из которых содержит часть исходного отношения. В полученных новых отношениях удаляются дубликаты строк. Это означает, что декомпозиция отношений является проекциями сходного отношения, при чем эти проекции в совокупности содержат все атрибуты исходного отношения. При декомпозиции не должны теряться данные. Данные считаются не потерянными в случае, если возможна обратная операция – восстановления исходного отношения в точности в прежнем виде по декомпозированным отношениям. Операция, обратная операции проекции – это операция соединения (естественное соединение).

Проекция R [x] на множество атрибутов х является собственной, если множество атрибутов х является собственным подмножеством множества атрибутов отношения R.

Собственные проекции R1 и R2 называются декомпозицией без потерь если отношение R точно восстанавливается из них при помощи естественного соединения R1 JOIN R2 любого состояния отношения R.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R Номер | Фамилия | | | Зарплата | | |
| 1 | Иванов | | | 1000 | | |
| 2 | Петров | | | 1000 | | |
| R1  Номер | | Зарплата | | |
| 1 | | 1000 | | |
| 2 | | 1000 | | |
| R2 Фамилия | | | Зарплата | | |
| Петров | | | 1000 | | |
| Иванов | | | 1000 | | |

|  |  |
| --- | --- |
| R2  Номер | Фамилия |
| 1 | Иванов |
| 2 | Петров |

|  |  |  |
| --- | --- | --- |
| R1 JOIN R2  Номер | Фамилия | Зарплата |
| 1 | Иванов | 1000 |
| 1 | Петров | 1000 |
| 2 | Иванов | 1000 |
| 2 | Петров | 1000 |

Теорема Хеза:

Пусть R(A,B,C) является отношением, а А,В,С- множество атрибутов этого отношения. Если имеется функциональная зависимость А->В, то проекции R1=R[A,B] и R2=R[A,C] образуют декомпозицию без потерь.

**83. Понятие транзакции. Свойства транзакций. Работа транзакций. Проблемы при работе транзакций.**

**Транзакция** – последовательность операторов манипулирования данными, выполняющаяся как единое целое и переводящая БД из одного целостного состояния в другое. Это логическая единица работы, единицы восстановления данных после сбоя.

4 свойства транзакций:

* атомарность –транзакция выполняется как атомарная, либо выполняется целиком, либо не выполняется
* согласованность – транзакция переводит БД из одного согласованного состояния в другое
* изоляция - транзакции разных пользователей не должны мешать друг другу
* долговечность - если транзакция выполнена, то результат ее работы должен сохраниться в БД, даже если в следующий момент произойдет сбой системы

Продолжается транзакция до наступления одного из 4 событий.

* Commit– завершить транзакцию
* Rollback – откатить
* Произошло отсоединение пользователя от СУБД
* Сбой системы

При последовательном запуске системы происходит анализ транзакций, которые можно восстановить.

Проблемы при параллельной работе транзакции.

1. Проблема потери результатов обновления. Две транзакции по очереди делают запись в одну и туже строку и фиксируют изменения. После окончания транзакции P имеет значение Р2. Транзакция А ничего не знает о транзакции В и, следовательно, транзакция А потеряла результаты своей работы.
2. Проблема незафиксированной зависимости (чтения грязных данных). Транзакция В изменяет данные в строке, затем транзакция А читает измененные данные и работает с ними. Транзакция В откатывается и восстанавливает измененные данные.
3. Проблема несовместимого анализа.
   1. Неповторяемое считывание. Транзакция А дважды читает одну и туже строку, между считываниями вклинивается транзакция В, изменяющая значения в строке. Транзакция А работает с данными, которые с точки зрения транзакций самопроизвольно изменяются
   2. Фантомы. Транзакция А выполняет выборку строк, удовлетворяющих некоторому условию
   3. Несовместный анализ. Длинная транзакция выполняет анализ по таблице, подсчитывает сумму денег на счетах клиентов банка. Пусть на всех счетах сумма 100 денежных единиц. Короткая транзакция выполняет перевод 50 денежных единиц со счета на счет так, чтобы сумма на счетах не уменьшилась

Транзакции называются конкурирующими если они пересекаются во времени и обращаются к одним и тем же данным. В результате конкуренции транзакции возникают конфликты:

1. WRITE- WRITE. Первая транзакция изменила объект и не закончилась. Вторая транзакция пытается изменить этот объект. В результате происходит конфликт обновления.
2. READ-WRITE. Первая транзакция прочитала объект и не закончилась. Вторая транзакция пытается изменить объект.
3. WRITE-READ. Первая транзакция изменила объект и не закончилась. Вторая транзакция пытается прочитать объект. В результате чтение грязных данных

Имеется 2 способа разрешить конкуренцию между поступающими в произвольные моменты времени транзакциями:

1. Обеспечить чтобы конкурирующие транзакции выполнялись в разное время. Реализуется с помощью временных меток.
2. Обеспечить чтобы конкурирующие транзакции работали с разными версиями данных. Реализуется с помощью использования журнала транзакции

**84. Ограничения целостности. Ограничения атрибута, кортежа, отношения, базы данных.**

Ограничение целостности – это некоторое утверждение которое может быть либо истинным, либо ложным в зависимости от БД. БД находиться в согласованном целостном состоянии, если выполнены все ограничения целостности, определенные для БД.

Реакции на попытку нарушения целостности:

1. отказ выполнить незаконную операцию
2. выполнение компенсирующих действий

Классификация ограничения целостности

1. По способам реализации:
2. декларативная поддержка ограничения целостности. Заключается в определении ограничений средствами языка DDL.

CREATE TABLE PERSON

(P\_ID integer primary key,

P\_NAME char(30) not null,

P\_id REFERENCES DEPART (P\_id)

ON DELETE CASCADE ON UPDATECASCADE);

1. Процедурная поддержка ограничения целостности заключается в использовании триггеров и хранимых процедур
2. По времени проверки:
3. немедленно проверяемые ограничения (пример - проверка уникальности ключа)
4. ограничение с отложенной проверкой (пример - в момент фиксации транзакции)
5. По области действия:
6. ограничение целостности домен
7. ограничение целостности атрибута - представляет собой ограничение, накладываемое на значение атрибута. Совпадает с ограничениями домена
8. ограничение целостности кортежа - представляет собой ограничение, накладываемое на значение конкретного кортежа и не является ограничением атрибута. Является немедленно проверяемым ограничением
9. ограничение целостности отношения представляет собой ограничения накладываемые на отдельные значения отношения и не являются ограничением целостности кортежа
10. ограничение целостности функциональной зависимости
11. ограничение целостности БД представляет собой ограничения, накладываемые на значения 2х или более связанных между собой отношений (пример - ограничение целостности ссылок)

**85. Блокировки. Совместимость блокировок. Протокол доступа к данным. Решение проблем параллелизма при помощи блокировок.**

2 типа блокировок: S и Х блокировки. S – Разделяемая блокировка (блокировка чтения), Х – монопольная блокировка (блокировка без взаимного доступа, блокировка записи).

|  |  |  |
| --- | --- | --- |
| Блокировки тр. А. | Блокировки тр. В. | |
| S | X |
| S | Да | Нет R-W |
| X | Нет W-R | Нет W-W |

Доступ к объектам БД осуществляется с помощью протокола БД:

1. прежде чем прочитать объект транзакция должна наложить на объект S блокировку. Прежде чем обвить объект транзакция должна наложить Х блокировку
2. если объект заблокирован S, то перед обновлением объекта S блокировка должна быть заменена на Х блокировку
3. если блокировка объекта транзакции отвергается от того, что объект заблокирован конкурирующей транзакцией, то транзакция, пытающаяся захватить объект, переходит в состояние ожидания и находиться в этом состоянии ожидания до тех пор, пока захватившая объект транзакция не освободит его
4. Х блокировки, наложенные транзакцией А, сохраняются до конца транзакции А

Проблема потери результатов обновления

|  |  |  |
| --- | --- | --- |
| S-бл. P успешно | t1 | - |
| Чтение P->P0 | t2 | - |
| - | t3 | S-бл. P успешно |
| - | t4 | Чтение P->P0 |
| x-бл отвергается | t5 | - |
| режим ожидания | t6 | x-бл отвергается, ожидание |

Обе транзакции накладывают S-блокировки и читают объект P. Транзакция А пытается наложить x-блокировку. Она отвергается, так как объект s-блокирован транзакцией В. Транзакция А переходит в режим ожидания до тех пор, пока транзакция В не освободит объект. В аналогичной ситуации находится транзакция В. Обе транзакции ожидают друг друга и не могут продолжаться. Возникла ситуация ТУПИКА.

Проблема чтения грязных данных

|  |  |  |
| --- | --- | --- |
| - | t1 | S-блкировка строки Р, успешно |
| - | t2 | Чтение P->P0 |
| - | t3 | x-блокировка, успешно |
| - | t4 | Запись P1->P |
| S-блокировка отвергается | t5 | - |
| ожидание | t6 | Откат P0->P блокировки снимаются |
| S блокировка Р успешна | t7 | - |
| Чтение P=P0 | t8 | - |
| Фиксация | t9 | - |

Работа тр. А. притормаживается до окончания работы тр. В. После этого тр. А. продолжает работу с правильными данными. “- “увеличение времени работы тр. А.

Проблема неповторяемого считывания

|  |  |  |
| --- | --- | --- |
| S-бл Р успешно | t1 | - |
| P=P0 | t2 | - |
| - | t3 | x-бл Р отвергается |
| - | t4 | ожидание |
| Чтение P=P0 | t5 |  |
| Фиксация, снимаются блокировки | t6 |  |
| - | t7 | x-бл успешно |
| - | t8 | Запись P1->P |
| - | t9 | фиксация |

Тр. В. притормаживается до окончания работы тр. А. В результате тр. А. дважды читает одни и те же данные правильно. После окончания тр. А. тр.В продолжает работу

Проблема фантомов

|  |  |  |
| --- | --- | --- |
| S-бл строк, удовл усл. Заблок N строк | t1 | - |
| Выборка по усл α, отобр N строк | t2 | - |
| - | t3 | Вставл строки удовл усл α |
| - | t4 | фиксация |
| Повторная выборка, соотв условию α (N+1) | t5 | - |
| Фиксация | t6 | - |

Блокировка на уровне строк не решила проблему фантомов.

Несовместимый анализ

|  |  |  |
| --- | --- | --- |
| S-бл счета P1 успешно | t1 | - |
| Чтение P1 = 100, sum=100 | t2 | - |
| - | t3 | x-бл счета Р3 успешно |
| - | t4 | Снимает деньги со счета Р3, Р3: 100->50 |
| - | t5 | x-бл счета Р1 отвергается |
| S-бл счета P2 успешно | t6 | ожидание |
| Чтение P2 =100, sum=200 | t7 | ожидание |
| S-бл P3 отвергается | t8 | ожидание |
| ожидание | t9 | ожидание |

При использовании протокола доступа к данным проблема незафиксированной зависимости разрешилась, неповторяемого чтения разрешилась, проблема фантомов не разрешилась, проблема потери результатов обновления и несовместного анализа привели к новой проблеме – ТУПИК.

2 подхода решения тупика:

* 1 подход: СУБД не следит за выбором транзакции жертвы
* 2 подход: за возникновение тупиковой ситуации следит СУБД. Строиться граф ожидания транзакции.

**86. Преднамеренные блокировки. Метод временных меток. Механизм выделения версий данных.**

При блокировании объектов разной величины возникает проблема обнаружения ранее наложенных блокировок. Протокол преднамеренных блокировок: перед тем как наложить блокировку на объект необходимо наложить специальную преднамеренную блокировку на объект, в состав которого входит блокируемый объект. Наличие блокировки таблицы будет свидетельствовать о наличии блокировки строк таблицы для другой транзакции, пытающейся блокировать целую таблицу.

Виды блокировок:

* IS –преднамеренная блокировка с возможностью взаимного доступа. Накладывается на некий составной объект Т и означает намерение блокировать входящий в Т- объект в режиме S блокировки.
* IХ – преднамеренная блокировка без взаимного доступа. Накладывается на некий составной объект Т и означает намерение блокировать входной в него объект в режиме Х блокировки.
* SIХ – преднамеренная блокировка как с возможностью взаимного доступа, так и без него. Накладывается на некий составной объект Т и означает разделяемую блокировку всего этого объекта с намерением в последствии блокировать входящие в него объекты в режиме Х-блокировки

Матрица совместимости блокировок

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IS | S | SIX | IX | X |
| IS | + | + | + | + | - |
| S | + | + | - | - | - |
| SIX | + | - | + | - | - |
| IX | + | - | - | - | - |
| X | - | - | - | - | - |

Сила блокировок: Х самая сильная блокировка, а IS самая слабая блокировка.

Протокол преднамеренной блокировки для доступа к данным:

* 1. При задании Х-блокировок для сложных объектов неявно задаются Х-блокировки для дочерних объектов этого объекта.
  2. При задании S-блокировок или SIX для сложных объектов неявно задаются X-блокировки для дочерних объектов этого об-та.
  3. Перед наложением требуемой S или IS на объект, задаются IS-блокировка или более сильная хотя бы для одного родительского объекта.
  4. Перед наложением требуемой X или IS, SIX на объект, задаются IX-блокировка или более сильная для всех родительских объектов.
  5. Перед отменой блокировки объекта для транзакции, должны быть отменены все блокировки для дочерних объектов этого объекта.

Проблема фантомов решается, если транзакция А использует преднамеренную блокировку этой таблицы S или IX

Метод временных меток:

Если транзакция T1 началась раньше транзакции T2, то система обеспечивает такой режим выполнения, как если бы T1 была целиком выполнена до начала T2.

Для этого каждой транзакции T предписывается временная метка t, соответствующая времени начала транзакции. При выполнении операции над объектом r транзакция T помечает его своей временной меткой и типом операции (чтение или изменение).

Перед выполнением операции над объектом r транзакция T1 проверяет, не закончилась ли транзакция T, пометившая этот объект. 1 - Если T закончилась, T1 помечает объект r и выполняет свою операцию. 2 - Если транзакция T не завершилась, то T1 проверяет конфликтность операций. Если операции неконфликтны, при объекте r остается или проставляется временная метка с меньшим значением, и транзакция T1 выполняет свою операцию. 3 - Если операции T1 и T конфликтуют, то если t(T)> t(T1) (т.е. транзакция T является более "молодой", чем T), производится откат T и T1 продолжает работу. 4 - Если же t(T) <t(T1) (T "старше" T1), то T1 получает новую временную метку и начинается заново.

Механизм выделения версий данных:

1. Для каждой транзакции записывается текущий системный номер (SCN); чем позже начата транзакция, тем он больше.
2. При записи страниц данных фиксируется SCN транзакции, производившей запись. Этот номер ставится текущим в системе данных.
3. Если транзакция А читает страницу данных, то SCNа сравнивается с SCNp, читающей со страницы данных.
4. Если SCNp <= SCNa то транзакция читает эту страницу
5. Если SCNa<SCNp это означает, что некоторая транзакция В, начавшаяся позже транзакции А успела изменить и сейчас изменяет данные страницы. В этом случае транзакция А просматривает журнал транзакции назад в поиске первой записи об изменении нужной страницы данных с SCNa<SCNp. Найдя такую запись транзакция А использует старый вариант данных страницы.

|  |  |  |
| --- | --- | --- |
| A | t | B |
| Проверка SCNа>SCNp1. Чтение p1 = 100 sum = 100 | t1 | - |
|  | t2 | Х-бл. р3 успешно |
| - | t3 | Снятие денег со счета р3 : 100 -> 50 |
| - | t4 | Х-бл. р1 успешно |
| - | t5 | Помещение денег на счет р1 : 100 -> 150 |
| - | t6 | Фиксация В |
| Проверка SCN(A) > SCN(р2). Чтение р2 = 100 sum = 200 | t7 |  |
| Проверка SCN(A) < SCN(р3). Чтение старого варианта счета р3 = 100 sum = 300 | t8 | - |
| Фиксация А | t9 | - |

**87. Виды восстановления данных. Индивидуальный откат транзакции. Восстановление после жесткого сбоя. Восстановление после мягкого сбоя.**

Восстановление БД должно проводиться в следующих видах:

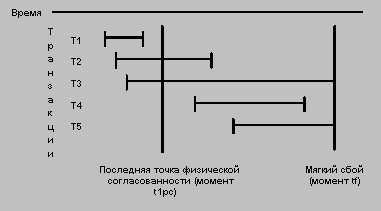
* 1. Индивидуальный откат: откат транзакции может быть инициирован самой транзакцией путем подачи команды Rollback или системой. СУБД может инициировать откат транзакции в случае ошибки или при разрешении тупиковой ситуации
  2. Мягкий сбой системы (аварийный отказ ПО). Характеризуется утратой оперативной памяти, при этом поражаются выполняющиеся на момент сбоя транзакции, теряется содержимое буферов БД, но данные, хранящиеся на диске, остаются неповрежденными. Мягкий сбой может произойти по факту отключения питания.
  3. Жесткий сбой системы (аварийный отказ аппаратуры). Характеризуется повреждением внешних носителей памяти.

Индивидуальный откат транзакции:

Все записи в журнале от данной транзакции связываются в обратный список. Началом списка для не закончившихся транзакций является запись о последнем изменении БД, произведенном данной транзакцией. Для закончившихся транзакций (индивидуальные откаты которых уже невозможны) началом списка является запись о конце транзакции, которая обязательно вытолкнута во внешнюю память журнала. Концом списка всегда служит 1ая запись об изменении БД, произведенном данной транзакцией. Каждой записи проставляется уникальный ID транзакции, чтобы можно было восстановить прямой список записей об изменениях БД данной транзакцией.

1. Просматривается список записей, сделанных данной транзакцией в журнале транзакций.
2. Выбирается очередная запись из списка данной транзакции.
3. Выполняется противоположная по смыслу операция: вместо INSERT - DELETE.
4. Любая из операций журнализуются.
5. При успешном завершении отката в журнал заносится запись о конце транзакции.

Мягкий сбой: Существуют 5 вариантов состояния транзакции к последней контрольной точке.



T1 - закончилась до принятия контр точки. Все ее результаты отражены во внешней памяти базы данных. Никаких операций по восстановлению не требуется.

T2 – начата до контр. точки и успешно завершилась до момента мягкого сбоя. Записи журнала вытолкнуты во внешнюю память, страницы данных – частично. Нужно повторно выполнить оставшуюся часть операций.

T3 - начата до контр. точки и не завершилась к моменту сбоя. Часть страниц данных – во внеш. памяти (до контр. точки), после контр точки следов изменений нет. Записи журнала до контр точки – вытолкнуты, записи журнала после контр точки – нет. Транзакцию нужно откатить.

T4 – начата после контр точки и успела закончиться до сбоя. Записи журнала – вытолкнуты, страницы – отсутствуют. Нужно повторить транзакцию целиком.

T5 - начата после контр точки и не завершена в результате сбоя. Следов т. нет во внешней памяти, значит действий не требуется.

Восстановление системы после мягкого сбоя осуществляется как часть процедуры перезагрузки системы.

1. Создаются 2 списка транзакций Undo и Redo. В Undo заносятся транзакции из последней записи контрольной точки. Undo = {T2, T3}. Redo остаются пустым
2. Начиная с записи контрольной точки, просматривается вперед журнал транзакции и если в журнале обновится запись о начале транзакции, то она добавится в список Undo. Undo = {T2, T3, T4}
3. Если в журнале обновится запись Commit об окончании транзакции, то она добавится в список Redo. Redo = {T2, T4}.
4. При достижении конца журнала транзакции списки анализируются. Из списка Undo удаляются транзакции, попавшие в список Redo.
5. Система просматривает журнал транзакции назад с момента контрольной точки, открывает транзакции из списка Undo. Система просматривает журнал транзакции вперед с контрольной точки и повторно выполняет все операции транзакции из списка Redo.

Жесткий сбой

БД на диске нарушается физически. Основой восстановления являются журнал и архивная копия БД. Восстановление начинается с обратного копирования базы данных из архивной копии. Затем просматривается журнал для выполнения в прямом направлении всех транзакций, которые успешно завершаются до сбоя. Нет необходимости отката транзакций, прерванных в результате сбоя, т.к. их нет в резервной копии.