Тема № 1. Базы данных специального назначения

Лекция № 3: Реляционная алгебра. Реляционное исчисление. Средства языка SQL.

Учебные цели занятия:

Сформировать представление о:

1) Положениях реляционной алгебры и ее назначении,
2) Положениях реляционного исчисления и его назначении,
3) Средствах языка SQL манипулирования данными.
4) Ограничениях целостности используемых реляционной моделью

Учебные вопросы:

5) Реляционная алгебра
6) Реляционное исчисление
7) Целостность данных

1. Реляционная алгебра
1.1 Введение в реляционную алгебру

- **Выборка** - Возвращает отношение, содержащее все кортежи заданного отношения, которые удовлетворяют указанным условиям. Операцию выборки также иногда называют операцией ограничения.

- **Проекция** - Возвращает отношение, содержащее все кортежи (подкортежи) заданного отношения, которые остались в этом отношении после исключения из него нескольких атрибутов.

- **Произведение** - Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям.

- **Объединение** - Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим.

1. Реляционная алгебра
1.1 Введение в реляционную алгебру

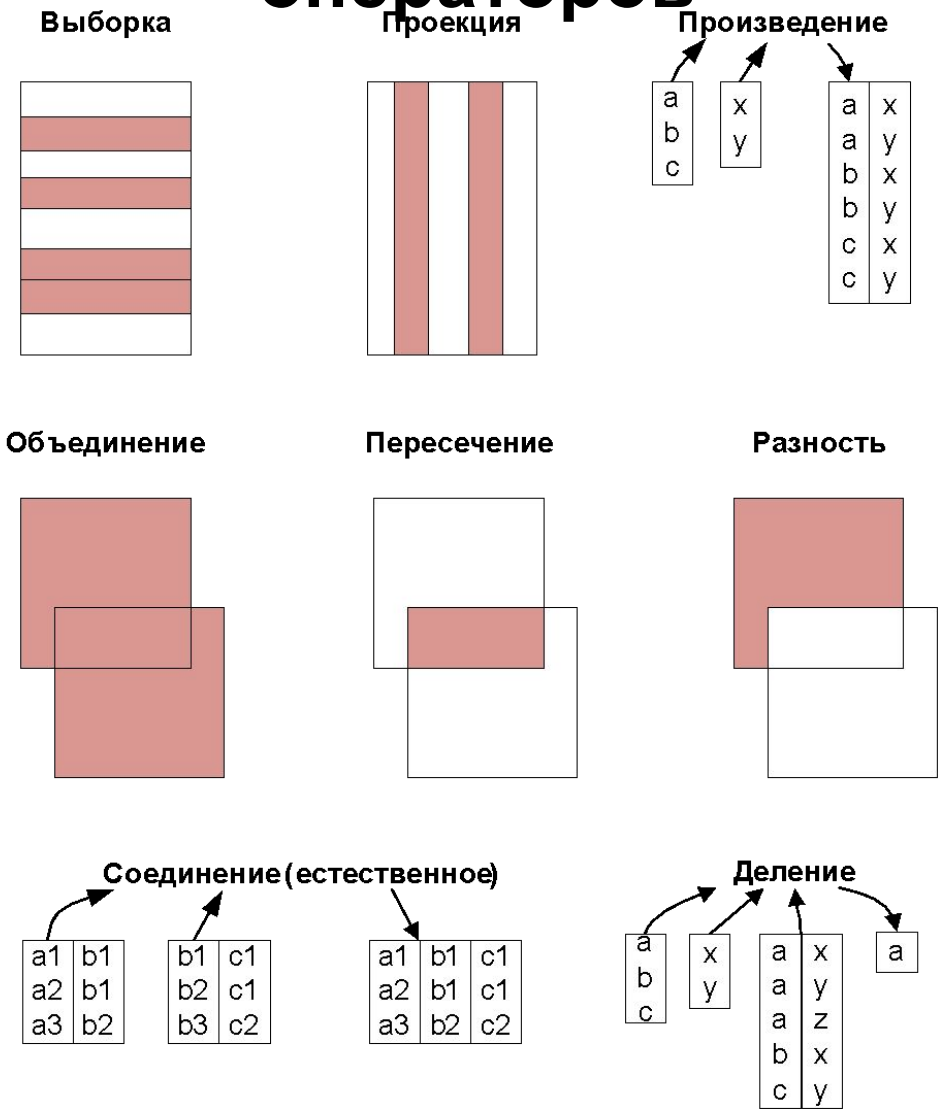
- **Пересечение** - Возвращает отношение, содержащее кортежи, которые принадлежат одновременно двум заданным отношениям.

- **Разность** - Возвращает отношение, содержащее кортежи, которые принадлежат первому отношению, но не принадлежат второму.

- **Соединение** - Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей, принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах.

- **Деление** - Для заданных двух унарных отношений и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

Графическая интерпретация восьми операторов



1.2 Реляционная замкнутость

- Результат выполнения любой операции над отношением также является отношением. Эта особенность является свойством ***реляционной замкнутости***.
- Благодаря этому свойству можно записывать ***вложенные реляционные выражения***, т.е. выражения, в которых операнды сами представлены реляционными выражениями, причем произвольной сложности.
- результат обязательно должен иметь определенный ***тип отношения***.

1.2 Реляционная замкнутость

- Необходим встроенный в реляционную алгебру набор ***правил вывода типов*** (отношений), чтобы выводить тип (отношения) на выходе произвольной реляционной операции, зная типы (отношения) на ее входе.
- Полезным в этом направлении является введение оператора переименования RENAME, который позволяет вернуть новое отношение, только указанные атрибуты которого имеют новые имена, а его значение остается прежним.
- P RENAME PNAME AS PN, WEIGTH AS WT
- Данный оператор позволяет устраниться от необходимости использования механизма уточнения имен атрибутов (P.WEIGHT, как в SQL).

1.3 Реляционная алгебра. Синтаксис (начало)

```
<реляционное выражение> ::= RELATION { <список выражений кортежей> }
| <имя переменной-отношения>
| <реляционная операция>
| ( <реляционное выражение> )

<реляционная операция> ::= <проекция> | <не проекция>
<проекция> ::= <реляционное выражение>
              { [ ALL BUT } <список имен атрибутов> }
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>.

```
<не проекция> ::= <переименование> | <объединение> | <пересечение>
| <вычитание> | <произведение> | <выборка>
| <соединение> | <деление>

<переименование> ::= <реляционное выражение>
RENAME <список переименовываемых элементов>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>.

```
<объединение>::=<реляционное выражение> UNION <реляционное выражение>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>, если только оба не объединения.

```
<пересечение> ::= <реляционное выражение> INTERSECT
<реляционное выражение>
```

Здесь <реляционное выражение> не должно иметь вид <не проекция>, если только оба не пересечения.

Реляционная алгебра. Синтаксис (конец)

<вычитание>::= <реляционное выражение> MINUS <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.

<произведение>::=<реляционное выражение> TIMES <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>,
если только оба не являются произведениями.

<выборка> ::= <реляционное выражение> WHERE <логическое выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.
Логическое выражение может иметь ссылки на атрибуты отношения,
обозначенного как реляционное выражение.

<соединение>::= <реляционное выражение> JOIN <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>,
если только хотя бы одно не является соединением.

<деление> ::= <реляционное выражение> DIVIDEBY
<реляционное выражение> PER <реляционное выражение>
Здесь <реляционное выражение> не должно иметь вид <не проекция>.

Объединение

Для заданных отношений A и B одного и того же типа ***объединением*** этих двух отношений (A UNION B) называется новое отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат или отношению A, или отношению B, или обоим отношениям одновременно.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Объединение (A UNION B)

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва
П2	Ильин	10	Тверь

Пересечение

Пересечением двух совместимых по типу отношений A и B (A INTERSECT B) называется отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат одновременно обоим исходным отношениям A и B.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Пересечение (A INTERSECT B)

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва

Вычитание

Вычитанием двух совместимых по типу отношений A и B (A MINUS B) называется отношение того же типа с телом, состоящим из множества всех кортежей t, которые принадлежат отношению A, но не принадлежат отношению B.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П4	Иванов	20	Москва

B

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь

Вычитание (A MINUS B)

S#	SNAME	STATUS	CITY
П4	Иванов	20	Москва

Вычитание (B MINUS A)

S#	SNAME	STATUS	CITY
П2	Ильин	10	Тверь

Декартово произведение

Декартовым произведением двух отношений A и B (A TIMES B), где отношения A и B не имеют общих имен атрибутов, называется новое отношение с заголовком, представляющим объединение заголовков двух исходных отношений A и B, и с телом, состоящим из множества всех кортежей t, таких, что каждый кортеж t представляет собой объединение двух кортежей, один из которых принадлежит отношению A, а другой – отношению B.

Выборка

Пусть задано отношение A с атрибутами X и Y (и, возможно, с другими атрибутами), а символ Θ обозначает любой скалярный оператор сравнения, такой, что условие XΘY корректно определено при заданных значениях этих атрибутов и дает значение *истина* или *ложь*.

Тогда **Θ-выборкой** из отношения A по атрибутам X и Y называется отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество всех кортежей t отношения A, для которых проверка условия XΘY дает значение *истина*.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

A WHERE CITY = ‘Москва’ OR STATUS > 14

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

Проекция

Пусть задано отношение A с атрибутами X, Y, …, Z (и, возможно, другими). Тогда проекцией отношения A по атрибутам X, Y, …, Z (A {X, Y, …, Z}) называется отношение, удовлетворяющее следующим требованиям:

1) Его заголовок получается из заголовка отношения A посредством удаления из него всех атрибутов, не входящих в множество {X, Y, …, Z}.

2) Его тело содержит множество всех кортежей вида {X:x, Y:y, …, Z:z}, таких для которых в отношении A значение атрибута X равно x, значение атрибута Y равно y, …, значение атрибута Z равно z.

A

S#	SNAME	STATUS	CITY
П1	Петров	20	Москва
П2	Ильин	10	Тверь
П3	Коробов	15	Смоленск
П4	Иванов	20	Москва

A {STATUS, CITY } = A { ALL BUT S#, SNAME }

STATUS	CITY
20	Москва
10	Тверь
15	Смоленск

Соединение

Пусть даны два отношения A и B имеют соответственно заголовки

{X1, X2, …, Xm, Y1, Y2,…, Yn}

и

{ Y1, Y2,…, Yn, Z1, Z2, …, Zp}.

Пусть X, Y и Z являются соответствующими составными атрибутами {X1, X2, …, Xm}, {Y1,Y2,…,Yn} и { Z1, Z2, …, Zp}. Тогда ***естественным соединением*** отношений A и B (A JOIN B) называется отношение с заголовком {X, Y, Z} и телом, содержащим множество всех кортежей вида {X:x,Y:y, Z:z}, таких, для которых в отношении A значение атрибута X равно x, а значение атрибута Y равно y, и в отношении B значение атрибута Y равно y, а значение атрибута Z равно z.

Деление

Пусть отношения A и B имеют заголовки {X1, X2, …, Xm} и {Y1,Y2,…,Yn} соответственно. Пусть также имеется отношение C с заголовком {X1, X2, …, Xm,Y1,Y2,…,Yn}. Пусть X, Y являются соответствующими составными атрибутами {X1, X2, …, Xm} и {Y1,Y2,…,Yn}.

Тогда результатом ***деления*** отношения A на отношение B по соотношению C (A DIVIDEBY B PER C) называется отношение с заголовком {X} и телом, содержащим множество всех кортежей вида {X:x}, таких, что кортеж вида {X:x, Y:y} принадлежит отношению C для всех кортежей вида {Y:y}, принадлежащих отношению B.

A

S#
П1
П2
П3
П4

B

D#
Д1
Д2
Д3
Д4

C

S#	D#
П1	Д1
П1	Д2
П1	Д3
П1	Д4
П2	Д1
П2	Д2
П3	Д2

A DIVIDEBY B PER C

S#
П1

1.5 Реляционная алгебра. Примеры

Получить имена поставщиков детали с номером ‘P2’:

```
( (SP JOIN S) WHERE P# = ‘P2’ ) {SNAME}
```

Получить имена поставщиков по крайней мере одной красной детали:

```
( ( (P WHERE COLOR = ‘Красный’) JOIN SP ){S#}
  JOIN S) {SNAME}
```

Получить имена поставщиков всех типов деталей:

```
( (S{S#} DIVIDEBY P{P#} PER SP{S#,P#})
  JOIN S) {SNAME}
```

Получить имена поставщиков, которые не поставляют деталь с номером ‘P2’:

```
( (S{S#} MINUS (SP WHERE P#=’P2’){S#} )
  JOIN S) {SNAME}
```

1.6 Назначение реляционной алгебры

- Основная цель реляционной алгебры – ***обеспечить запись реляционных выражений***.

Некоторые из возможных применений подобных выражений:

- Определение области выборки
- Определение области обновления
- Определение правил поддержки целостности данных
- Определение производных переменных-отношений
- Определение требований устойчивости, т.е. данных, которые должны быть включены в контролируемую область для некоторых операций управления параллельным доступом к информации.
- Определение ограничений защиты, т.е. данных, для которых осуществляется тот или иной тип контроля доступа.

Выражения реляционной алгебры служат для *символического высокоуровневого представления намерений пользователя*.

- Данными выражениями можно манипулировать в соответствии с различными символическими высокоуровневыми ***правилами преобразования***.
- Запрос ((SP JOIN S) WHERE P#=’P2’) {SNAME}
- может быть преобразован в более рациональное выражение вида:
- ((SP WHERE P#=’P2’) JOIN S) {SNAME}
- Таким образом, реляционная алгебра может быть хорошим основанием для выполнения ***оптимизации*** (что должно производиться оптимизатором автоматически).
- В общем случае язык называют ***реляционно полным***, если его возможности, по крайней мере, соответствуют возможностям, обеспечиваемым алгебраическими операциями, т.е. выражения этого языка позволяют определить каждое отношение, которое может быть определено с помощью алгебраических выражений

2. Реляционное исчисление

2.1 Введение в реляционное исчисление

- часть реляционной модели, которая связана с операторами манипулирования данными, основывается на использовании реляционной алгебры
- Однако можно сказать, что она построена на базе *реляционного исчисления*.
- реляционная алгебра и реляционное исчисление представляют два альтернативных подхода.
- в реляционной алгебре предоставляется в явном виде набор операторов для формирования требуемого отношения
- в реляционном исчислении имеется система обозначений для определения требуемого отношения в терминах данных отношений.

Пример

- В качестве примера рассмотрим следующий запрос: «Выбрать номера поставщиков и названия городов, в которых находятся поставщики детали с номером ‘P2’».
- Алгебраическая версия запроса выглядит следующим образом: 1) сначала выполнить соединение отношения поставщиков S и отношения поставок по атрибуту S#; 2) Выбрать из результата соединения кортежи с номером детали ‘P2’; 3)Выполнить проекцию для результата этой выборки по атрибутам S# и CITY.
- В терминах реляционного исчисления запрос формулируется следующим образом:

  Получить атрибуты S# и CITY для таких поставщиков S, для которых в отношении SP существует запись о поставке с тем же значением атрибута S# и со значением атрибута P#, равным ‘P2’.

  Т.е. указываются лишь некоторые характеристики требуемого результата, оставляя системе решать, что именно и в какой последовательности соединять, проецировать и т.д., чтобы получить необходимый результат.

Реляционное исчисление носит описательный характер, а реляционная алгебра – предписывающий, т.е. не описывается, в чем заключается проблема, а задается процедура решения этой проблемы.

- Реляционное исчисление основано на разделе математической логики, которое называется исчислением ***предикатов***.
- Основным понятием реляционного исчисления является понятие ***переменной кортежа*** – переменная, «изменяющаяся на» некотором заданном отношении, т.е. переменная, допустимыми значениями для которой являются кортежи заданного отношения.
- Другими словами, если переменная кортежа V изменяется в пределах отношения r, то в любой заданный момент времени переменная V представляет некоторый кортеж t отношения r.
- В связи с тем, что реляционное исчисление основано на переменных кортежа, его первоначальную версию называют также ***исчислением кортежей***.

2.Реляционная исчисление. 2.2 Исчисление кортежей. Синтаксис (начало)

```
<реляционное выражение> ::= RELATION { <список выражений кортежей> }
| <имя переменной-отношения>
| <реляционная операция>
| ( <реляционное выражение> )
```

Определение реляционного выражения осталось прежним, но <реляционная операция> имеет иное определение.

```
<определение переменной кортежа> ::=
   RANGEVAR <имя переменной кортежа>
   RANGES OVER <список реляционных выражений>;
```

<Имя переменной кортежа> может использоваться в следующих случаях:

- Перед точкой и последующим уточнением в параметре <ссылка на атрибут кортежа>;
- Сразу после квантора в параметре <логическое выражение с квантором>;
- Как операнд в параметре <логическое выражение>;
- Как параметр <прототип кортежа> или как подпараметр <выражение> в параметре <прототип кортежа>.

```
<ссылка на атрибут кортежа> ::=
<имя переменной кортежа>.<ссылка на атрибут> [AS <имя атрибута>]
```

2. Реляционная исчисление. Синтаксис (конец)

Параметр <ссылка на атрибут кортежа> может использоваться как параметр <выражение>, но только в определенном контексте:

- Как операнд параметра <логическое выражение>;
- Как параметр <прототип кортежа> или как подпараметр <выражение> в параметре <прототип кортежа>.

```
<логическое выражение> ::=… все обычные возможности |
<логическое выражение с квантором>

<логическое выражение с квантором> ::=
   EXISTS <имя переменной кортежа> ( <логическое выражение> )
   | FORALL <имя переменной кортежа> (< логическое выражение > )

<реляционная операция> ::= <прототип кортежа>
              [WHERE <логическое выражение>]

<прототип кортежа> ::= <выражение кортежа>
```

Переменные кортежей

- Приведем примеры определения переменных кортежей для БД поставщиков и деталей:
- RANGEVAR SX RANGES OVER S
- RANGEVAR SY RANGES OVER S
- RANGEVAR SPX RANGES OVER SP
- RANGEVAR SPY RANGES OVER SP
- RANGEVAR PX RANGES OVER P

  RANGEVAR SU RANGES OVER
  (SX WHERE SX.CITY = ‘Москва’),
  (SX WHERE EXISTS SPX (SPX.S# = SX.S# AND
  SPX.P# = ‘P1’))
- Переменная кортежа SU определенная на объединении множества поставщиков, находящихся в Москве, и множества кортежей поставщиков детали с номером ‘P1’. Конечно, отношения при их объединении должны быть совместимы по типу.
- **Замечание**. Переменные кортежей не являются переменными в обычном смысле, а скорее представляют некоторый аналог местодержателям, или параметрам, предикатов, а, следовательно, являются переменными в логическом смысле.

Свободные и связанные переменные кортежей

- Каждая ссылка на переменную кортежа является либо ***свободной***, либо ***связанной***.
- Пусть V – переменная кортежа, тогда:
- Ссылки на переменную V в логических выражениях типа NOT p свободны или связаны в пределах этого выражения в зависимости от того, свободны они или нет в формуле p. Ссылки на переменную V в логических выражениях типа (p AND q) и (p OR q) свободны или связаны в зависимости от того, свободны ли они в выражениях p и q.
- Ссылки на переменную V, которые свободны в логическом выражении p, связаны в логических выражениях типа EXISTS V(p) и FORALL V(p) в соответствии с тем, свободны ли они в формуле p

Пример

Приведем некоторые примеры свободных и связанных переменных кортежей:

- Примеры свободных переменных кортежей:

  SX.S# = ’П1’

  SX.S# = SPX.S#

  NOT (SX.CITY = ’Москва’)

  SX.S#=SPX.S# AND SPX.P# <> PX.P#

  PX.COLOR = ‘Красный’ OR PX.CITY = ’Москва’
- Примеры связанных переменных кортежей:

  EXISTS SPX (SPX.S#=SX.S# AND SPX.P#=’P2’)

  FORALL PX (PX.COLOR=’Красный’)

Кванторы

- Существует два квантора: EXISTS и FORALL.
- Квантор EXISTS является квантором существования, а FORALL – квантором всеобщности.
- Если выражение p – логическое выражение, в которой переменная V свободна, то выражения EXISTS V(p) и FORALL V(p) также являются допустимыми логическими выражениями, но переменная V в них обеих будет связанная.
- Первая формула означает: «Существует, по крайней мере, одно значение переменной V, такое, что вычисление выражения p дает для него значение *истина*». Второе выражение означает: «Для всех значений переменной V вычисление выражения p дает для него значение *истина*».

Пример

- Рассмотрим следующий квантор существования:

  EXISTS SPX (SPX.S#=SX.S# AND SPX.P#=’P2’)

- Данное выражение может быть прочитано следующим образом:

  В текущем значении переменной-отношения SP существует, по крайней мере, один кортеж (скажем, SPX), такой, для которого значение атрибута S# в этом кортеже равно значению атрибута SX.S# (какое бы оно ни было), а значение атрибута P# в кортеже SPX равно ‘P2’.

2.3 Примеры использования исчисления кортежей

1. Определить номера поставщиков из Твери со статусом, большим 20.
(SX.S#, SX.STATUS) WHERE SX.CITY = ‘Тверь’ AND SX.STATUS > 20

2. Определить имена поставщиков детали с номером ‘P2’.
SX.SNAME WHERE EXISTS SPX(SPX.S#=SX.S# AND SPX.P#=’P2’)

3. Определить имена поставщиков по крайней мере одной красной детали.
SX.SNAME WHERE EXISTS SPX (SX.S#=SPX.S# AND
EXISTS PX (PX.P# = SPX.P# AND PX.COLOR = ‘Красный’))

4. Найти имена поставщиков по крайней мере одной детали, поставляемой поставщиком с номером ‘П2’.
SX.SNAME WHERE
EXISTS SPX (EXISTS SPY(SX.S# = SPX.S# AND
SPX.P# = SPY.P# AND SPY.S# = ‘П2’))

5. Выбрать имена поставщиков всех типов деталей.
SX.SNAME WHERE FORALL PX (EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = PX.P#))

6. Определить имена поставщиков, которые не поставляют деталь с номером ‘P2’.
SX.SNAME WHERE NOT EXISTS SPX
(SPX.S# = SX.S# AND SPX.P# = ‘P2’)

2.4 Средства языка SQL (начало)

1. Указать цвета и названия городов, в которых находятся детали «не из Твери» с весом, превышающим 10 кг.

```
SELECT PX.COLOR, PX.CITY
FROM P AS PX
WHERE PX.CITY <> ‘Тверь’ AND PX.WEIGHT > 10
```

2. Для всех деталей указать номер и вес в фунтах

```
SELECT P.P#, P.WEIGHT / 0.454 AS WF
FROM P
```

3. Выбрать информацию обо всех парах поставщиков и деталей, находящихся в одном городе

```
SELECT S.*, P.P#, P.PNAME, P.COLOR, P.WEIGHT
FROM S, P
WHERE S.CITY = P.CITY
```

4. Определить общее количество поставщиков

```
SELECT COUNT(*) AS N
FROM S
```

Средства языка SQL (продолжение)

5. Для каждой поставляемой детали указать номер и общий объем поставки в штуках

```
SELECT SP.P#, SUM(SP.QTY) AS TOTQTY
FROM SP
GROUP BY SP.P#
```

6. Указать номера всех типов деталей, поставляемых более чем одним поставщиком

```
SELECT SP.P#
FROM SP
GROUP BY SP.P#
HAVING COUNT(SP.S#) > 1;
```

7. Определить имена поставщиков детали с номером ‘P2’

```
SELECT DISTINCT S.SNAME
FROM S
WHERE S.S# IN
      (SELECT SP.S#
      FROM SP
      WHERE SP.P# = ‘P2’);
```

Средства языка SQL (продолжение)

8. Определить имена поставщиков, по крайней мере, одной красной детали

```
SELECT DISTINCT S.SNAME
FROM S
WHERE S.S# IN     (SELECT SP.S#
            FROM SP
            WHERE SP.P# IN    (SELECT P.P#
                        FROM P
                        WHERE P.COLOR=’Красный’));
```

9. Указать имена поставщиков, статус которых меньше текущего максимального статуса в таблице S

```
SELECT S.S#
FROM S
WHERE S.STATUS < (SELECT MAX(S.STATUS) FROM S)
```

10. Выбрать имена поставщиков, которые не поставляют деталь с номером ‘P2’

```
SELECT DISTINCT S.SNAME
FROM S
WHERE NOT EXISTS
      (SELECT *
      FROM SP
      WHERE SP.S#=S.S# AND SP.P#=’P2’)
```

Средства языка SQL (конец)

11. Определить имена поставщиков все типов деталей

```
SELECT DISTINCT S.SNAME
FROM S
WHERE NOT EXISTS
        (SELECT *
        FROM P
        WHERE NOT EXISTS
                (SELECT *
                FROM SP
                WHERE SP.S#=S.S# AND SP.P#=P.P#)
```

Типы (категории) ограничений целостности данных

Ограничения целостности можно классифицировать по четырем основным категориям:

- *Ограничения целостности типа*, в которых задаются допустимые значения для данного типа.
- *Ограничения целостности атрибута*, в которых задаются допустимые значения для данного атрибута.
- *Ограничения целостности переменной-отношения*, в которых задаются допустимые значения для переменной-отношения.
- *Ограничения целостности БД*, в которых задаются допустимые значения для БД.

Ограничения переменной-отношения и БД. Примеры

Примеры ограничений переменной-отношения:

«Поставщики в Твери должны обладать статусом, равным 20»:

```
CONSTRAINT SC5
    IS_EMPTY ( S WHERE CITY = 'Тверь' AND STATUS <> 20 ).
```

«Номера поставщиков должны быть уникальны» или «Ключ {S#} – это потенциальный ключ отношения поставщиков»:

```
CONSTRAINT SCK
    COUNT ( S ) = COUNT ( S { S#} )
```

«Если детали вообще имеются, то одна из них должна быть красной»:

```
CONSTRAINT PC1
    IF NOT ( IS_EMPTY( P ) ) THEN
        COUNT ( P WHERE COLOR = 'Красный') > 0
    END IF
```

Примеры ограничений БД:

«Поставщики со статусом, меньшим 20, не могут поставлять детали в количестве свыше 500 штук»:

```
CONSTRAINT DBC1
    IS_EMPTY( (S JOIN SP)
            WHERE STATUS < 20 AND QTY > 500)
```

«Каждая деталь должна быть поставлена хотя бы один раз»:

```
CONSTRAINT DBC2 SP {P#} = P{P#}
```

«Золотое правило»

Вариант 1:
Ни одна из операций изменения не имеет права переводить переменную-отношение в состояние, нарушающее ее собственный предикат.

Вариант 2 (уточненный):
Ни одна из операций изменения не имеет права переводить переменную-отношение в состояние, нарушающее ее собственный предикат. Аналогично ни одна из транзакций изменения не имеет права переводить БД в состояние, нарушающее ее собственный предикат.

Потенциальные ключи

Пусть K – множество атрибутов переменной-отношения R. В этом случае множество K будет ***потенциальным ключом*** переменной-отношения R тогда и только тогда, когда оно обладает следующими свойствами:

а) ***Уникальность***. Никакие допустимые значения переменной-отношения R не содержат двух различных кортежей с одинаковыми значениями атрибутов множества K.

б) ***Неизбыточность***. Никакое из собственных подмножеств множества K не обладает свойством уникальности.

Суперключом называется некоторое надмножество потенциального ключа. Суперключ обладает свойством уникальности, но не обладает свойством неизбыточности.

Пример:

```
VAR MARRIAGE BASE RELATION {
    HUSBAND      /* Муж */    NAME,
    WIFE     /* Жена */   NAME,
    DATE     /* Дата бракосочетания */  DATE }
/* Подразумевается, что муж может иметь одну жену, а жена одного мужа,
причем не допускается повторного брака между одними и теми людьми */
        KEY { HUSBAND, DATE }
        KEY { DATE, WIFE }
        KEY { WIFE, HUSBAND }
```

Внешние ключи

Пусть R2 – некоторая переменная-отношение. Тогда ***внешний ключ*** (скажем, FK) в переменной-отношении R2 представляет собой множество атрибутов этой переменной-отношения, такое, что:

а) существует переменная-отношение R1 (причем переменные-отношения R1 и R2 необязательно различны) с потенциальным ключом CK;

б) каждое значение внешнего ключа FK в текущем значении переменной-отношения R2 обязательно совпадает со значением ключа CK некоторого кортежа в текущем значении переменной-отношения R1.

Ссылочная целостность – ограничение целостности на то, что БД не должна содержать внешних ключей, не имеющих соответствия.

Ограничения целостности в SQL (начало)

Ограничения домена

CREATE DOMAIN COLOR CHAR(6) DEFAULT '???'
    CONSTRAINT VALID_COLORS CHECK (VALUE IN
        ('Красный', 'Желтый', 'Синий','Зеленый','???')

Ограничения базовой таблицы

Потенциальные ключи

UNIQUE (<список имен столбцов>) или для первичного ключа:
PRIMARY KEY (<список имен столбцов>)

Внешние ключи

FOREIGN KEY (<список имен столбцов>)
    REFERENCES <имя базовой таблицы> [<список имен столбцов>]
    [ON DELETE <ссылочная операция>]
    [ON UPDATE <ссылочная операция>]

Проверочные условия

CHECK (<условное выражение>)

Пример.
```
CREATE TABLE SP (   S# S# NOT NULL, P# P# NOT NULL, QTY QTY NOT NULL,
        PRIMARY KEY (S#, P#),
        FOREIGN KEY (S#) REFERENCES (S)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (P#) REFERENCES (P)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CHECK (QTY > 0 AND QTY < 5001) );
```

Ограничения целостности в SQL (конец)

Утверждения

CREATE ASSERTION <имя ограничения>
CHECK (<условное выражение>);

Для отмены общего ограничения используется оператор DROP ASSERTION:
DROP ASSERTION <имя ограничения>;

Примеры:

1. Каждый поставщик должен иметь статус не менее 5.

```
CREATE ASSERTION AS1 CHECK
    ( (SELECT MIN (S.STATUS) FROM S) > 4 );
```

2. Значение веса любой детали должно быть положительным:

```
CREATE ASSERTION AS2 CHECK
    ( NOT EXISTS ( SELECT * FROM P
            WHERE NOT (P.WEIGHT > 0) ) );
```

3. Поставщики со статусом меньшим 20, не имеют права поставлять любую деталь в количестве более 500 штук:

```
CREATE ASSERTION AS3 CHECK
    ( NOT EXISTS ( SELECT * FROM S, SP
            WHERE S.STATUS < 20 AND S.S# = SP.S#
            AND SP.QTY > 500) );
```

Вопросы на самоподготовку:

1. Реляционная алгебра. Операторы. Реляционная замкнутость. Примеры.
2. Реляционная алгебра. Семантика операторов. Назначение реляционной алгебры. Примеры.
3. Реляционное исчисление. Исчисление кортежей. Переменные кортежей. Свободные и связанные переменные. Кванторы. Примеры.
4. Средства языка SQL манипулирования данными: Запросы SQL. Структура запроса. Вложенные подзапросы. Обобщающие функции. Примеры.
5. Средства языка SQL манипулирования данными: Запросы SQL. Структура запроса. IN-условия. Кванторы. Примеры.
6. Ограничения целостности данных. Типы ограничений целостности. Ограничения целостности типа и атрибута. «Золотое правило». Триггеры.
7. Ограничения целостности данных. Типы ограничений целостности. Ограничения целостности переменной-отношения и БД. Ключи.
8. Средства языка SQL поддержания ограничений целостности данных: Ограничения домена, базовой таблицы и утверждения. Операторы языка SQL. Примеры.