Рассмотрим основные операции над отношениями, которые могут представлять интерес с точки зрения извлечения данных из реляционных таблиц. Это *объединение*, *пересечение*, *разность*, расширенное *декартово произведение* отношений, а также специальные операции над отношениями: *выборка*, *проекция* и *соединение*.

Для иллюстрации *теоретико-множественных операций* над отношениями введем абстрактные отношения (таблицы) с некоторыми атрибутами (полями).

Отношение `R`

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

```
CREATE TABLE R
(a1 CHAR(1), a2 INT, PRIMARY KEY(a1,a2))
```

Отношение `S`

S.b1	S.b2
1	h
2	g
3	h

```
CREATE TABLE S
(b1 INT PRIMARY KEY, b2 CHAR(1))
```

Операции *выборки* и *проекции* являются унарными, поскольку они работают с одним отношением.

Операция выборки

Операция ***выборки*** - построение горизонтального подмножества, т.е. подмножества кортежей, обладающих заданными свойствами.

Операция *выборки* работает с одним отношением `R` и определяет результирующее отношение, которое содержит только те кортежи (строки) отношения `R`, которые удовлетворяют заданному условию `F` (предикату).

$\sigma_F(R)$ или $\sigma_{\text{предикат}}(R)$.

Пример 5.1. Операция выборки в SQL.

Выборка $\sigma_{(a2=1)}(R) = (a, 1), (b, 1)$ записывается следующим образом:

```
SELECT a1, a2
FROM R
WHERE a2=1
```

```
5.1. Операция выборки в SQL.
```

Операция проекции

Операция ***проекции*** - построение вертикального подмножества отношения, т.е. подмножества кортежей, получаемого выбором одних и исключением других атрибутов.

Операция *проекции* работает с одним отношением R и определяет новое отношение, которое содержит вертикальное подмножество отношения R, создаваемое посредством извлечения значений указанных атрибутов и исключения из результата строк-дубликатов.

$$\Pi_{a1,a2,...an}(R)$$

Пример 5.2. *Операция проекции* в SQL.

Проекция $\Pi_{b2}(S) = (h), (g)$ записывается следующим образом:

```
SELECT DISTINCT b2
FROM S
```

5.2. Операция проекции в SQL.

К основным операциям над отношениями относится *декартово произведение*.

Декартово произведение

Декартово произведение `RxS` двух отношений (двух таблиц) определяет новое отношение - результат конкатенации (т.е. сцепления) каждого кортежа (каждой записи) из отношения `R` с каждым кортежем (каждой записью) из отношения `S` .

```
RxS={(a, 1, 1, h), (a, 2, 1, h),
     (b, 1, 1, h), ... }

SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
```

5.3. Декартово произведение отношений в SQL.

Результат *декартова произведения* двух отношений показан в таблице.

Таблица 5.1.

R x S

R.a1	R.a2	S.b1	S.b2
a	1	1	h
a	1	2	g
a	1	3	h
a	2	1	h
a	2	2	g
a	2	3	h
b	1	1	h
b	1	2	g
b	1	3	h
b	3	1	h
b	3	2	g
b	3	3	h
b	4	1	h
b	4	2	g
b	4	3	h

Если одно отношение имеет `N` записей и `K` полей, а другое `M` записей и `L` полей, то отношение с их *декартовым произведением* будет содержать `NxM` записей и `K+L` полей. Исходные отношения могут содержать поля с одинаковыми именами, тогда имена полей будут содержать названия таблиц в виде префиксов для обеспечения уникальности имен полей в отношении, полученном как результат выполнения *декартова произведения*.

Однако в таком виде (пример 5.3.) отношение содержит больше информации, чем обычно необходимо пользователю. Как правило, пользователей интересует лишь некоторая часть всех комбинаций записей в *декартовом произведении*, удовлетворяющая некоторому условию. Поэтому вместо *декартова произведения* обычно используется одна из самых важных *операций реляционной алгебры* - операция

соединения, которая является производной от операции *декартова произведения*. С точки зрения эффективности реализации в реляционных СУБД эта операция - одна из самых трудных и часто входит в число основных причин, вызывающих свойственные всем реляционным системам проблемы с производительностью.

Операция соединения по двум отношениям (таблицам)

Соединение - это процесс, когда две или более таблицы объединяются в одну. Способность объединять информацию из нескольких таблиц или запросов в виде одного логического набора данных обусловливает широкие возможности SQL.

В языке SQL для задания типа *соединения* таблиц в логический набор записей, из которого будет выбираться необходимая информация, используется операция `JOIN` в предложении `FROM`.

Формат операции:

```
FROM имя_таблицы_1 {INNER | LEFT | RIGHT}
    JOIN имя_таблицы_2
ON условие_соединения
```

Существуют различные типы операций *соединения*:

- тета-соединение $R \triangleright \triangleleft_F S$;
- *соединение по эквивалентности* $R \triangleright \triangleleft_= S$;
- естественное соединение $R \triangleright \triangleleft S$;
- *внешнее соединение* $R \supset \triangleleft S$, $R \triangleright \subset S$;
- *полусоединение* $R \triangleright_F S$.

Операция тета-соединения

Операция тета-соединения $R \triangleright \triangleleft_F S$ определяет отношение, которое содержит кортежи из *декартова произведения* отношений `R` и `S`, удовлетворяющие предикату `F`. Предикат `F` имеет вид $R.ai \Theta S.bj$, где вместо Θ может быть указан один из операторов сравнения (`>`, `>=`, `<`, `<=`, `=`, `<>`).

Если предикат `F` содержит только оператор равенства (`=`), то *соединение* называется *соединением по эквивалентности*.

Таблица 5.2.

$R \triangleright \triangleleft_F S, F = (R.a2 = S.b1)$

R.a1	R.a2	S.b1	S.b2
a	1	1	h
a	2	2	g
b	3	3	h
b	1	1	h

Операция тета-соединения в языке SQL называется `INNER JOIN` (внутреннее *соединение*) и используется, когда нужно включить все строки из обеих таблиц, удовлетворяющие условию объединения. Внутреннее *соединение* имеет место и тогда, когда в предложении `WHERE` сравниваются значения полей из разных таблиц. В этом случае строится *декартово произведение* строк первой и второй таблиц, а из полученного набора данных отбираются записи, удовлетворяющие условиям объединения.

В условиях объединения могут участвовать поля, относящиеся к одному и тому же типу данных и содержащие один и тот же вид данных, но они не обязательно должны иметь одинаковые имена.

Блоки данных из двух таблиц объединяются, как только в указанных полях будут найдены совпадающие значения.

Если в предложении `FROM` перечислено несколько таблиц и при этом не употребляется спецификация `JOIN`, а для указания соответствия полей из таблиц используется условие в предложении `WHERE`, то некоторые реляционные СУБД (например, Access) оптимизируют выполнение запроса, интерпретируя его как *соединение*.

Если перечислять ряд таблиц или запросов и не указывать условия объединения, в качестве результирующей таблицы будет выбрано *декартово (прямое) произведение* всех таблиц.

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
WHERE R.a2=S.b1

    или

SELECT R.a1, R.a2, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

5.4. Тета-соединение отношений в SQL.

Естественное соединение

Естественным *соединением* называется *соединение по эквивалентности* двух отношений `R` и `S`, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру каждого общего атрибута.

Таблица 5.3.

$$R \triangleright\triangleleft S, F = (R.a2 = S.b1)$$

R.a1	R.a2	или	S.b1	S.b2
a	1			h
a	2			g
b	3			h
b	1			h

```
SELECT R.a1, R.a2, S.b2
FROM R, S
WHERE R.a2=S.b1

    или

SELECT R.a1, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

5.5. Естественное соединение отношений в SQL.

Пример 5.6. Вывести информацию о проданных товарах.

```
SELECT *
FROM Сделка, Товар
WHERE Сделка.КодТовара=Товар.КодТовара

    Или (что эквивалентно)

SELECT *
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
```

5.6. Выборка информации о проданных товарах.

Можно создать вложенные объединения, добавив третью таблицу к результату объединения двух других таблиц.

Пример 5.7. Получить сведения о товарах, дате сделок, количестве проданного товара и покупателях.

```
SELECT Товар.Название, Сделка.Количество, Сделка.
    Дата, Клиент.Фирма
FROM Клиент INNER JOIN
    (Товар INNER JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара)
ON Клиент.КодКлиента=Сделка.КодКлиента
```

5.7. Выборка сведений о товарах, дате сделок, количестве проданного товара и покупателях.

Использование общих имен таблиц для идентификации столбцов неудобно из-за их громоздкости. Каждой таблице можно присвоить какое-нибудь краткое обозначение, псевдоним.

Пример 5.8. Получить сведения о товарах, дате сделок, количестве проданного товара и покупателях. В запросе используются псевдонимы таблиц.

```
SELECT T.Название, C.Количество,
       C.Дата, K.Фирма
FROM Клиент AS K INNER JOIN
    (Товар AS T INNER JOIN
    Сделка AS C
ON T.КодТовара=C.КодТовара)
ON K.КодКлиента=C.КодКлиента;
```

5.8. Выборка сведений о товарах, дате сделок, количестве проданного товара и покупателях с использованием псевдонима.

Внешнее соединение похоже на внутреннее, но в результирующий набор данных включаются также записи ведущей таблицы *соединения*, которые объединяются с пустым множеством записей другой таблицы.

Какая из таблиц будет ведущей, определяет вид *соединения*. `LEFT` - левое *внешнее соединение*, ведущей является таблица, расположенная слева от вида *соединения* ; `RIGHT` - правое *внешнее соединение*, ведущая таблица расположена справа от вида *соединения*.

Левое внешнее соединение

Левым *внешним соединением* называется *соединение*, при котором *кортежи отношения* `R` , не имеющие совпадающих значений в общих столбцах отношения `S` , также включаются в результирующее отношение.

Таблица 5.4.

$R \supset \triangleleft S$

R.a1	R.a2	S.b1	S.b2
a	1	1	h
a	2	2	g
b	1	1	h
b	3	3	h
b	4	null	null

```
SELECT R.a1,  R.a2, S.b1, S.b2
FROM R LEFT JOIN S ON R.a2=S.b1
```

5.9. Левое внешнее соединение отношений в SQL.

Существует и правое *внешнее соединение* $R \triangleleft \subset S$, называемое так потому, что в результирующем отношении содержатся все кортежи правого отношения. Кроме того, имеется и полное *внешнее соединение*, в его результирующее отношение помещаются все кортежи из обоих отношений, а для обозначения несовпадающих значений кортежей в нем используются определители `NULL` .

```
SELECT R.a1,  R.a2, S.b1, S.b2
FROM R RIGHT JOIN S ON  R.a2=S.b1
```

5.10. Правое внешнее соединение отношений в SQL.

Пример 5.11. Вывести информацию о всех товарах. Для проданных товаров будет указана дата сделки и количество. Для непроданных эти поля останутся пустыми.

```
SELECT Товар.*, Сделка.*
FROM Товар LEFT JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара;
```

5.11. Выборка информации о всех товарах.

Полусоединение

Операция ***полусоединения*** определяет отношение, содержащее те *кортежи отношения* `R` , которые входят в *соединение* отношений `R` и `S` .

Таблица 5.5.

$$R \triangleleft_F S, F = (R.a2 = S.b1)$$

R.a1	R.a2
a	1
a	2
b	3
b	1

```
SELECT R.a1, R.a2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1,   R.a2
FROM R INNER JOIN S ON R.a2=S.b1
```

5.12. Полусоединение отношений в SQL.

Операция объединения

Объединение (`UNION`) $R \cup S$ отношений `R` и `S` можно получить в результате их конкатенации с образованием одного отношения с исключением кортежей-дубликатов. При этом отношения `R` и `S` должны быть совместимы, т.е. иметь одинаковое количество полей с совпадающими типами данных. Иначе говоря, отношения должны быть совместимы по *объединению*.

Объединением двух таблиц `R` и `S` является таблица, содержащая все строки, которые имеются в первой таблице `R` , во второй таблице `S` или в обеих таблицах сразу.

```
SELECT R.a1, R.a2
FROM R
UNION
SELECT S.b2, S.b1
FROM S
```

5.13. Объединение отношений в SQL.

Операция пересечения

Операция ***пересечения*** (`INTERSECT`) $R \cap S = R - (R - S)$ определяет отношение, которое содержит кортежи, присутствующие как в отношении `R` , так и в отношении `S` . Отношения `R` и `S` должны быть совместимы по *объединению* .

Пересечением двух таблиц `R` и `S` является таблица, содержащая все строки, присутствующие в обеих исходных таблицах одновременно.

```
SELECT R.a1, R.a2
FROM R,S
```

```
WHERE R.a1=S.b1 AND R.a2=S.b2
```

или

```
SELECT R.a1, R.a2
FROM R
WHERE R.a1 IN
    (SELECT S.b1 FROM S
     WHERE S.b1=R.a1) AND R.a2 IN
         (SELECT S.b2
          FROM S
          WHERE S.b2=R.a2)
```

5.14. Пересечение отношений в SQL.

Операция разности

Разность (EXCEPT) R-S двух отношений R и S состоит из кортежей, которые имеются в отношении R , но отсутствуют в отношении S . Причем отношения R и S должны быть совместимы по *объединению*.

Разностью двух таблиц R и S является таблица, содержащая все строки, которые присутствуют в таблице R , но отсутствуют в таблице S .

```
SELECT R.a1, R.a2
FROM R
WHERE NOT EXISTS
    (SELECT S.b1,S.b2
     FROM S
     WHERE S.b1=R.a2 AND S.b2=R.a1)
```

5.15. Разность отношений в SQL.

Операция деления отношений

Результат операции ***деления*** R:S - набор *кортежей отношения* R , определенных на множестве атрибутов C , которые соответствуют комбинации всех *кортежей отношения* S .

$T1=\Pi_C(R);$

$T2=\Pi_C((S \times T1) - R);$

$T=T1 - T2.$

Отношение R определено на множестве атрибутов A , а отношение S - на множестве атрибутов B , причем $A \supseteq B$ и C=A - B .

Пусть A ={имя, пол, рост, возраст, вес}; B ={имя, пол, возраст}; C ={рост, вес}.

Таблица 5.6.

Отношение R

имя	пол	рост	возраст	вес
a	ж	160	20	60
b	м	180	30	70
c	ж	150	16	40

Отношение S

имя	пол	возраст
a	ж	20

$T1=\Pi_C(R)$

рост	вес
160	60
180	70
150	40

TT=(S X T1)-R

имя	пол	возраст	рост	вес
а	ж	20	180	70
а	ж	20	150	40

$T2=П_C((S X T1)-R)$

рост	вес
180	70
150	40

T=T1-T2

рост	вес
160	60

Пример 5.16. Деление отношений в SQL.

Создание отношения R

```
CREATE TABLE R
(i     int primary key,
имя    varchar(3),
пол    varchar(3),
рост   int,
возраст int,
вес    int)
```

```
5.16a. Деление отношений в SQL.
```

Создание отношения S

```
CREATE TABLE S
(i     int primary key,
имя    varchar(3),
пол    varchar(3),
возраст int)
```

```
5.16b. Деление отношений в SQL.
```

Создание отношения T1

```
CREATE VIEW T1
AS
SELECT рост,вес
FROM R
```

```
5.16c. Деление отношений в SQL.
```

Создание отношения TT

```
CREATE VIEW TT AS
SELECT S.имя, S.пол, S.возраст,
       T1.рост, T1.вес
FROM S, T1
```

```
5.16d. Деление отношений в SQL.
```

Создание отношения T2

```
CREATE VIEW T2
AS
SELECT TT.рост, TT.вес
FROM TT
```

```
WHERE NOT EXISTS
    (SELECT R.рост, R.вес
     FROM R
     WHERE TT.имя=R.имя AND TT.пол=R.пол
           AND TT.возраст=R.возраст
           AND TT.рост=R.рост
                   AND TT.вес=R.вес)
```

```
5.16e. Деление отношений в SQL.
```

Создание отношения T

```
SELECT T1.рост, T1.вес
FROM T1
WHERE NOT EXISTS
    (SELECT T2.рост,T2.вес
     FROM T2
     WHERE T1.рост=T2.рост AND T1.вес=T2.вес)
```

```
5.16f. Деление отношений в SQL.
```