База Данных

Лекция 8

Реляционная алгебра

Прошлая лекция

- выборка (selection)
- проекция (projection)
- декартово произведение (cartesian product)
- объединение (union)
- разность (set difference)
- пересечение (intersection)

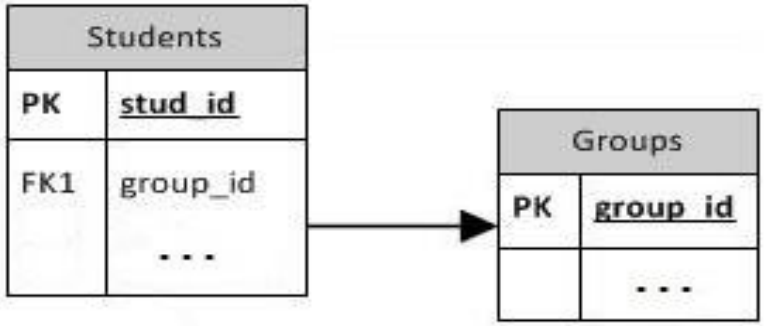
Соединение (Join)

- Операция **соединение (Join)** позволяет запрашивать информацию из двух или более связанных таблиц.
- Это аналогично условию выбора, за исключением того, что сравниваются значения в двух разных таблицах.
- Каждая таблица должна содержать хотя бы одно поле, служащее первичным ключом. Первичный ключ одной таблицы, как правило, является внешним ключом для другой таблицы.

Join: example 1

```
CREATE TABLE Groups (
group_id int PRIMARY KEY,
group_name varchar(15));

CREATE TABLE Students (
stud_id int PRIMARY KEY,
first_name varchar(20),
last_name varchar(20),
bdate date,
group_id int REFERENCES Groups(group_id));
```



Join: example 1

stud_id	*last_name*	*group_name*
…	*…*	*…*
…	*…*	*…*

```
SELECT stud_id, last_name, group_name FROM
Students, Groups
WHERE
    Students.group_id = Groups.group_id;
```

Псевдонимы соединяемых таблиц. table.column формат

- В приведенном выше примере в разделе условия выбора приведен формат **table.column**.
- **table.column** формат используется для разрешения конфликтов имен, если поля в таблицах имеют одинаковые имена.
- Псевдонимы для соединений должны быть краткими и удобочитаемыми. Они, как правило, состоят из одной буквы, причем используется первая буква соответствующей таблицы (tablename.FieldName будет записано как t.FieldName).
- Этот синтаксис может использоваться в разделах SELECT или WHERE.

Псевдонимы соединяемых таблиц. table.column формат

```
SELECT *
FROM Students, Groups
WHERE
        Students.group_id = Groups.group_id;

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
SELECT *
FROM Students AS s, Groups AS g
WHERE
        s.group_id = g.group_id;
```

Join: example 2

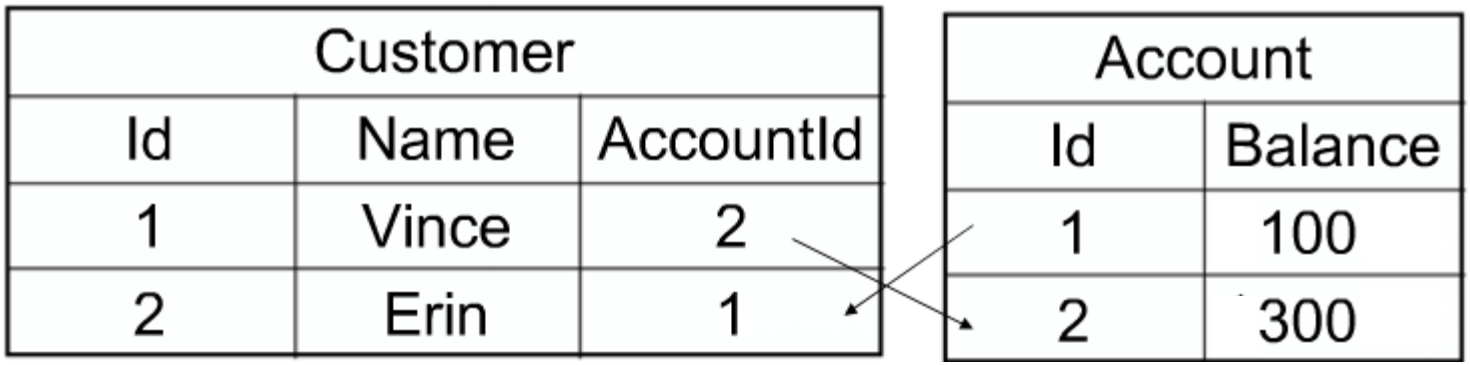
```
CREATE TABLE Account (
   id int PRIMARY KEY,
   balance int);
CREATE TABLE Customer (
   id int PRIMARY KEY,
   name varchar (20),
   accountid int REFERENCES Account (id));
```

Customer		
Id	Name	AccountId
1	Vince	2
2	Erin	1

Account	
Id	Balance
1	100
2	300

Join: example 2

- Предположим, мы хотим запросить имя Клиента, баланс которого равен 100.
- Мы можем сделать это, объединив таблицы Account и Customer, где FK Клиента (AccountId) равен PK учетной записи (Id).



Customer		
Id	Name	AccountId
1	Vince	2
2	Erin	1

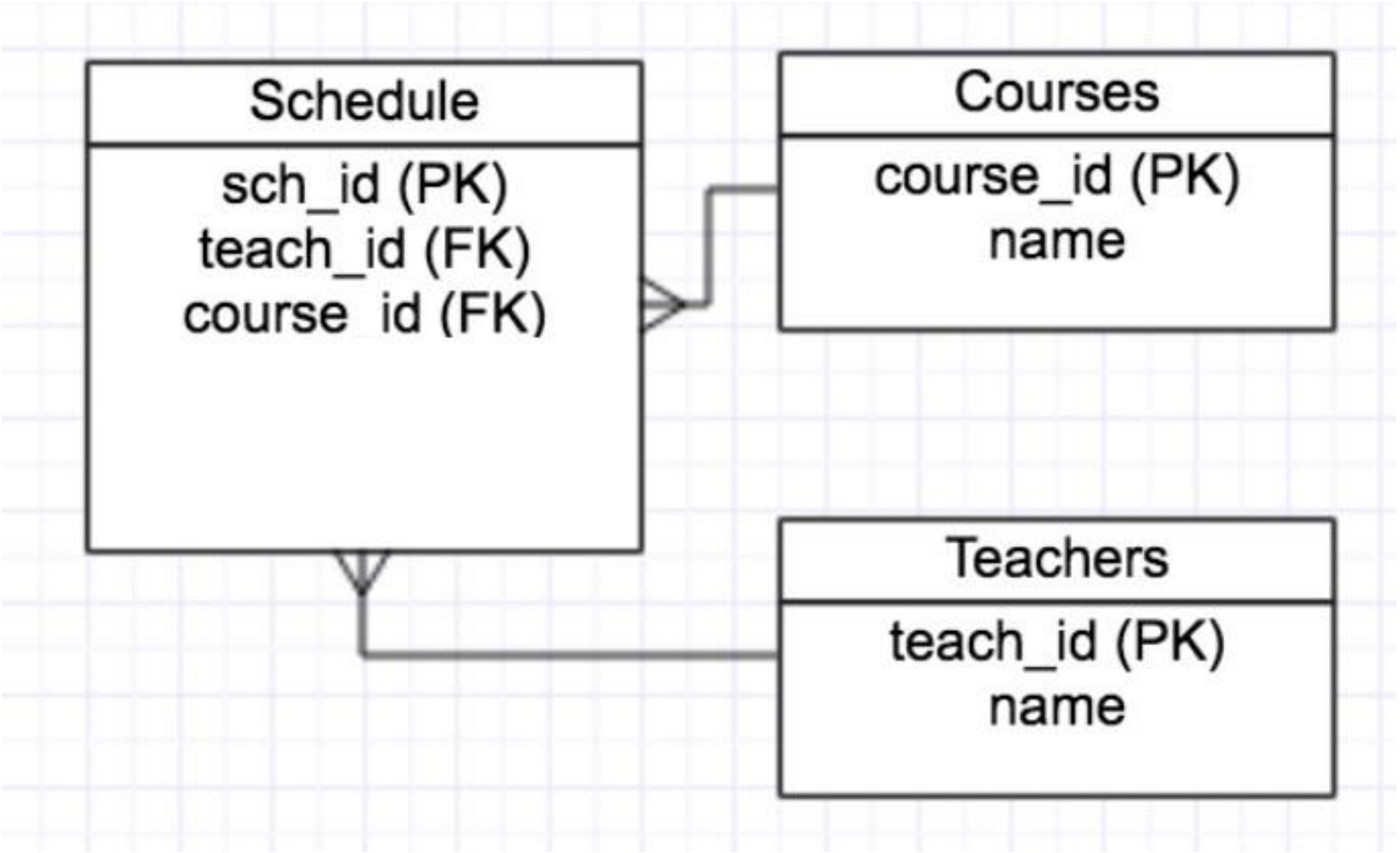
Account	
Id	Balance
1	100
2	300

Join: example 2

- SQL-запрос:

```
SELECT name
FROM Customer, Account
WHERE
Customer.accountid= Account.id
AND Account.Balance=100;
```

Join: пример 3



Join: пример 3

```
CREATE TABLE Courses (
    course_id int PRIMARY KEY,
    name varchar(30));
CREATE TABLE Teachers (
    teach_id int PRIMARY KEY,
    name varchar (30));
 CREATE TABLE Schedule (
    sch_id int PRIMARY KEY,
    course_id int REFERENCES Courses (course_id),
    teach_id int REFERENCES Teachers (teach_id));
```

Join: пример 3

course_name	*teach_name*
…	*…*
…	*…*

```
SELECT Courses.name, Teachers.name
FROM Courses, Teachers, Schedule
WHERE
Courses.course_id = Schedule.course_id
AND
Teachers.teach_id = Schedule.teach_id;
```

JOIN ключевое слово

Предложение SQL JOIN используется для объединения строк из двух или более таблиц.

Типы:

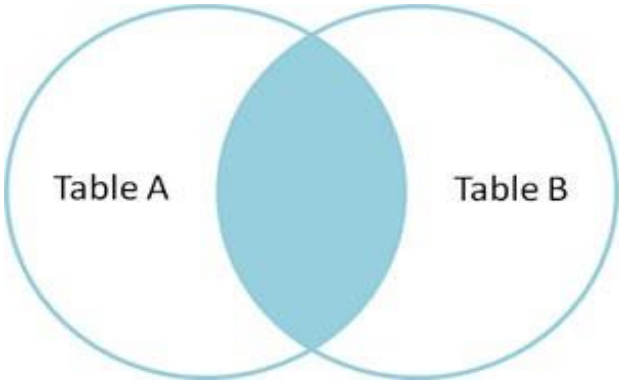
- INNER JOIN (ВНУТРЕННЕЕ СОЕДИНЕНИЕ)
- OUTER JOIN (ВНЕШНЕЕ СОЕДИНЕНИЕ)
  - LEFT JOIN (ЛЕВОЕ СОЕДИНЕНИЕ)
  - RIGHT JOIN (ПРАВОЕ СОЕДИНЕНИЕ)
  - FULL JOIN (ПОЛНОЕ СОЕДИНЕНИЕ)
- CROSS JOIN (ПЕРЕКРЕСТНОЕ СОЕДИНЕНИЕ)

INNER JOIN (ВНУТРЕННЕЕ СОЕДИНЕНИЕ)

- Наиболее распространенным типом соединения является INNER JOIN (ВНУТРЕННЕЕ СОЕДИНЕНИЕ) или просто соединение (JOIN).
- Внутреннее соединение SQL возвращает все строки из нескольких таблиц, в которых выполняется условие соединения.
- При использовании внутреннего соединения возвращаются только совпадающие записи. Любые несовпадающие данные из любой таблицы игнорируются.

Синтаксис:

```
SELECT column_name(s)  FROM tableA
INNER JOIN tableB
ON tableA.column_name =
tableB.column_name;
```



INNER JOIN тоже самое что и **JOIN**.

ВНУТРЕННЕЕ СОЕДИНЕНИЕ: пример

```
SELECT Students.stud_id, Students.fname,
    Groups.group_name
FROM Students
INNER JOIN Groups
ON Students.group_id = Groups.group_id;
```

Следующий пример эквивалентен предыдущему:

```
SELECT Students.stud_id, Students.fname,
    Groups.group_name
FROM Students, Groups
WHERE Students.group_id = Groups.group_id;
```

ВНУТРЕННЕЕ СОЕДИНЕНИЕ: пример

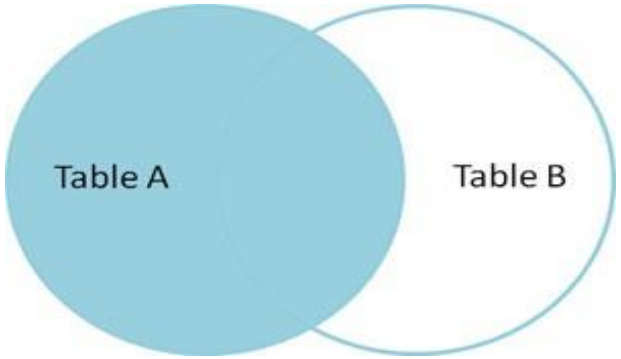
Students		
stud_id	fname	group_id
1	student1	2
2	student2	2
3	student3	

Groups	
group_id	group_name
1	CSSE-1
2	CSSE-2

Result table for INNER JOIN		
stud_id	fname	group_name
1	student1	CSSE-2
2	student2	CSSE-2

LEFT JOIN (ЛЕВОЕ СОЕДИНЕНИЕ)

Ключевое слово LEFT JOIN возвращает все строки из левой таблицы (TableA) с соответствующими строками в правой таблице (TableB). Результат равен NULL в правой части, если нет совпадения.



Синтаксис:

```
SELECT column_name(s)
FROM tableA
LEFT JOIN tableB
ON tableA.column_name = tableB.column_name;
```

В некоторых базах данных LEFT JOIN (ЛЕВОЕ СОЕДИНЕНИЕ) используется только как LEFT OUTER JOIN (ЛЕВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ).

ЛЕВОЕ СОЕДИНЕНИЕ пример

Следующая инструкция SQL вернет всех учащихся и группы, которые у них могут быть:

```
SELECT Students.stud_id, Students.fname,
Groups.group_name
    FROM Students
    LEFT JOIN Groups
    ON Students.group_id = Groups.group_id;
```

Ключевое слово LEFT JOIN возвращает все строки из левой таблицы (учащиеся), даже если в правой таблице (группы) нет совпадений:

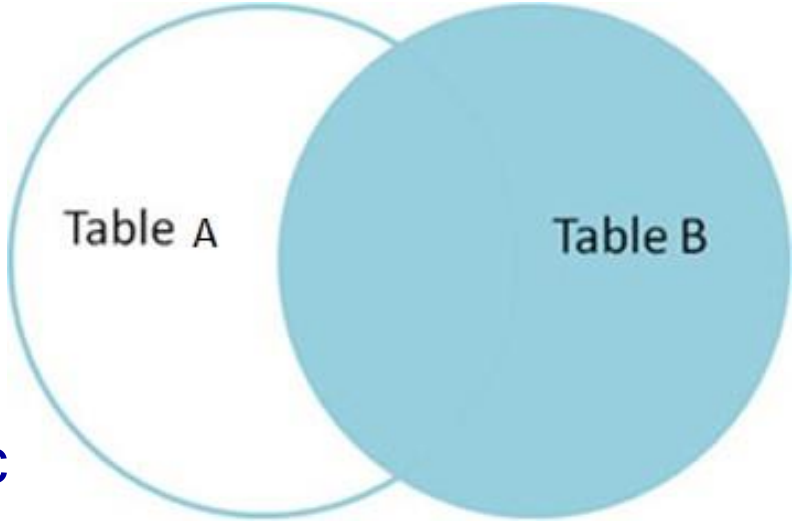
Result table for LEFT JOIN		
stud_id	fname	group_name
1	student1	CSSE-2
2	student2	CSSE-2
3	student3	

RIGHT JOIN (ПРАВОЕ СОЕДИНЕНИЕ)

Ключевое слово RIGHT JOIN возвращает все строки из правой таблицы (TableB) с соответствующими строками в левой таблице (TableA). Результат равен NULL в левой части, если нет совпадения.

Синтаксис:

```
SELECT column_name(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.column_name=tableB.c
```



некоторых базах данных RIGHT JOIN (ПРАВОЕ СОЕДИНЕНИЕ) используется только как RIGHT OUTER JOIN (ПРАВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ).

Правое СОЕДИНЕНИЕ пример

Следующая инструкция SQL вернет все группы и учащихся, которые у них могут быть:

```
SELECT Students.stud_id, Students.fname, Groups.group_name
    FROM Students
    RIGHT JOIN Groups
    ON Students.group_id = Groups.group_id;
```

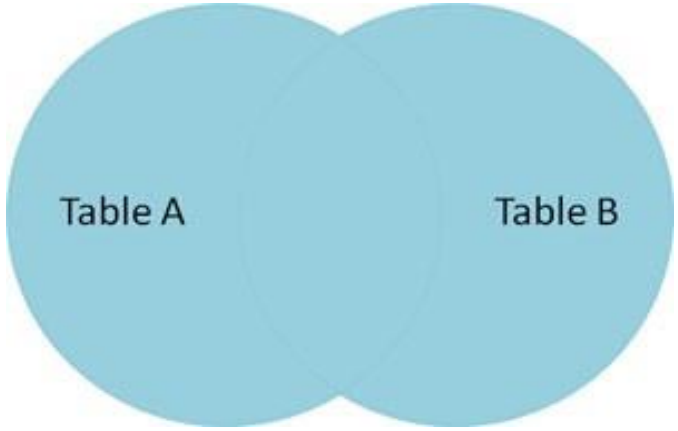
Ключевое слово RIGHT JOIN возвращает все строки из правой таблицы (группы), даже если в левой таблице (учащиеся) нет совпадений:

Result table for RIGHT JOIN		
stud_id	fname	group_name
1	student1	CSSE-2
2	student2	CSSE-2
		CSSE-1

FULL OUTER JOIN (ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ)

Ключевое слово FULL OUTER JOIN возвращает все строки из левой таблицы (TableA) и из правой таблицы (TableB).

Ключевое слово FULL OUTER JOIN объединяет результат как левого, так и ПРАВОГО соединений.



Синтаксис:

```
SELECT column_name(s)
FROM tableA
FULL OUTER JOIN tableB
ON tableA.column_name=tableB.column_name;
```

FULL OUTER JOIN (ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ) пример

Следующая инструкция SQL выбирает всех учащихся и все группы:

```
SELECT Students.stud_id, Students.fname,
   Groups.group_name FROM Students
   FULL OUTER JOIN Groups
   ON Students.group_id = Groups.group_id;
```

Ключевое слово FULL OUTER JOIN возвращает все строки из левой таблицы (учащиеся) и все строки из правой таблицы (группы).
Если в "Студентах" есть строки, у которых нет совпадений в “Группах", или если в “Группах" есть строки, у которых нет совпадений в “Студентах", эти строки также будут перечислены:

Result table for FULL OUTER JOIN

stud_id	fname	group_name
1	student1	CSSE-2
2	student2	CSSE-2
3	student3	
		CSSE-1

CROSS JOIN (ПЕРЕКРЕСТНОЕ СОЕДИНЕНИЕ)

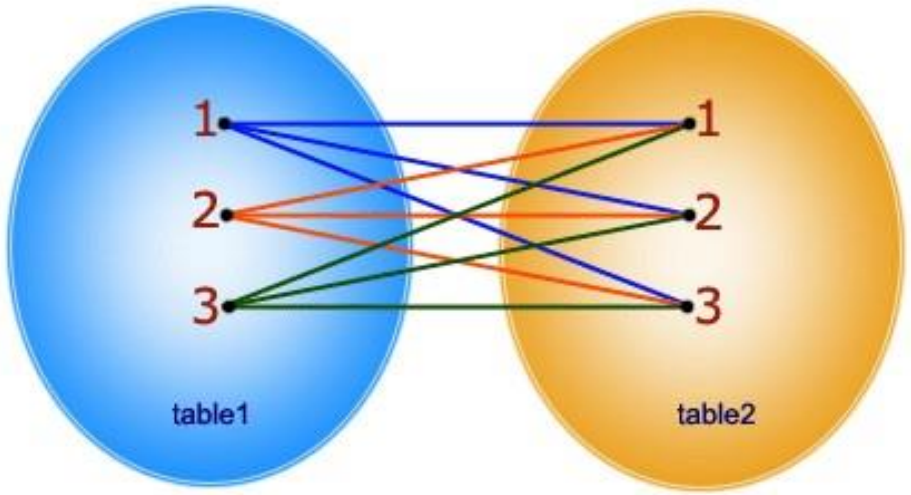
CROSS JOIN (ПЕРЕКРЕСТНОЕ СОЕДИНЕНИЕ) SQL выдает результирующий набор, который представляет собой количество строк в первой таблице, умноженное на количество строк во второй таблице (“Все ко всем”). Условие WHERE не используется вместе с ПЕРЕКРЕСТНЫМ соединением. Такой результат называется декартовым произведением.

```
SELECT *
FROM  tableA
CROSS JOIN tableB;
```

или

```
SELECT *
FROM tableA, tableB
```

SELECT * FROM table1 CROSS JOIN table2;



In CROSS JOIN, each row from 1st table joins with all the rows of another table. If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

CROSS JOIN (ПЕРЕКРЕСТНОЕ СОЕДИНЕНИЕ). Пример

```
SELECT *
FROM  Students
CROSS JOIN Groups;
```

или

```
SELECT *
FROM Students, Groups;
```

ПЕРЕКРЕСТНОЕ СОЕДИНЕНИЕ: пример

Result table for CROSS JOIN				
stud_id	fname	group_id	group_id	Group_name
1	student1	2	1	CSSE-1
2	student2	2	1	CSSE-1
3	student3		1	CSSE-1
1	student1	2	2	CSSE-2
2	student2	2	2	CSSE-2
3	student3		2	CSSE-2

Полный синтаксис JOIN (СОЕДИНЕНИЯ)

```
SELECT Attribute(s)
FROM TableA
{INNER | {LEFT | RIGHT | FULL} OUTER | CROSS } JOIN TableB
ON <condition>
```

JOIN with USING

Предложение USING - это сокращение, которое позволяет воспользоваться преимуществами конкретной ситуации, когда обе стороны соединения используют одно и то же имя для соединяемых столбцов. Он берет разделенный запятыми список имен общих столбцов и формирует условие соединения, которое включает сравнение на равенство для каждого из них.

```
SELECT      Attribute(s)
FROM TableA
{INNER | {LEFT | RIGHT | FULL} OUTER } JOIN
TableB
USING (join column list)
```

JOIN with USING: example

Когда мы используем JOIN с USING в SQL, мы объединяем таблицы по столбцу, который присутствует в обеих таблицах с одинаковым именем. Конструкция USING упрощает синтаксис, так как нам не нужно указывать имя таблицы для столбца, по которому происходит объединение.

```
SELECT *
FROM Students  INNER
JOIN Groups USING
(group_id);
```

Выходные данные JOIN USING подавляют избыточные столбцы: нет необходимости печатать оба совпадающих столбца, поскольку они должны иметь одинаковые значения.

NATURAL JOIN (ЕСТЕСТВЕННОЕ СОЕДИНЕНИЕ)

NATURAL - это сокращенная форма USING: она

формирует список USING, состоящий из всех имен столбцов, которые отображаются в обеих входных таблицах. Как и в случае USING, эти столбцы отображаются только один раз в выходной таблице.

Natural join (естественное объединение) — это тип операции объединения в SQL, при котором объединение таблиц происходит по всем столбцам с одинаковыми именами в обеих таблицах. Результатом является набор строк, содержащих все столбцы из обеих таблиц, за исключением дублирующихся столбцов, где значения совпадают.

```
SELECT Attribute(s)
FROM TableA
NATURAL
{INNER | {LEFT | RIGHT | FULL} OUTER }
JOIN TableB
```

NATURAL JOIN (ЕСТЕСТВЕННОЕ СОЕДИНЕНИЕ): Пример

```
SELECT *
FROM Students
NATURAL INNER JOIN Groups;
```

Обозначение

Операции имеют свои собственные символы.

Operation	Symbol
Union	$\cup$
Intersection	$\cap$
Set difference	-

Operation	Symbol
Projection	π
Selection	σ
Cartesian product	$\times$
Join	$\bowtie$
Left outer join	$⟕$
Right outer join	$⟖$
Full outer join	$⟗$

Книги

- Коннолли, Томас М. Системы баз данных: практический подход к проектированию, внедрению и управлению / Томас М. Коннолли, Кэролин Э. Бегг.- Пятое.- Соединенные Штаты Америки: Pearson Education, 2010
- Гарсия-Молина, Х. Система баз данных: Полная книга
- / Эктор Гарсия-Молина.- 2. - Соединенные Штаты Америки: Пирсон Прентис Холл, 2009
- Шарма, Н. Основы баз данных: Книга для сообщества, написанная сообществом / Нирадж Шарма, Ливиу Перниу.- Первое издание.- Канада, 2010

- www.postgresql.org/docs/manuals/
- www.postgresql.org/docs/books/