5. SQL
Structured Query Language

к.т.н., доцент кафедры ИиСП

Лучинин
Захар Сергеевич

Оператор DISTINCT

Оператор **DISTINCT** используется для указания на то, что следует работать только с уникальными значениями столбца.

Оператор **DISTINCT** нашел широкое применение в операторе **SELECT**, для выборки уникальных значений. Так же используется в агрегатных функциях.

SELECT DISTINCT column_name **FROM** table_name

Оператор SQL DISTINCT

singer	album	year	sale
The Prodigy	Invaders Must Die	2008	1200000
Drowning Pool	Sinner	2001	400000
Massive Attack	Mezzanine	1998	2300000
The Prodigy	Fat of the Land	1997	600000
The Prodigy	Music For The Jilted Generation	1994	1500000
Massive Attack	100th Window	2003	1200000
Drowning Pool	Full Circle	2007	800000
Massive Attack	Danny The Dog	2004	1900000
Drowning Pool	Resilience	2013	500000

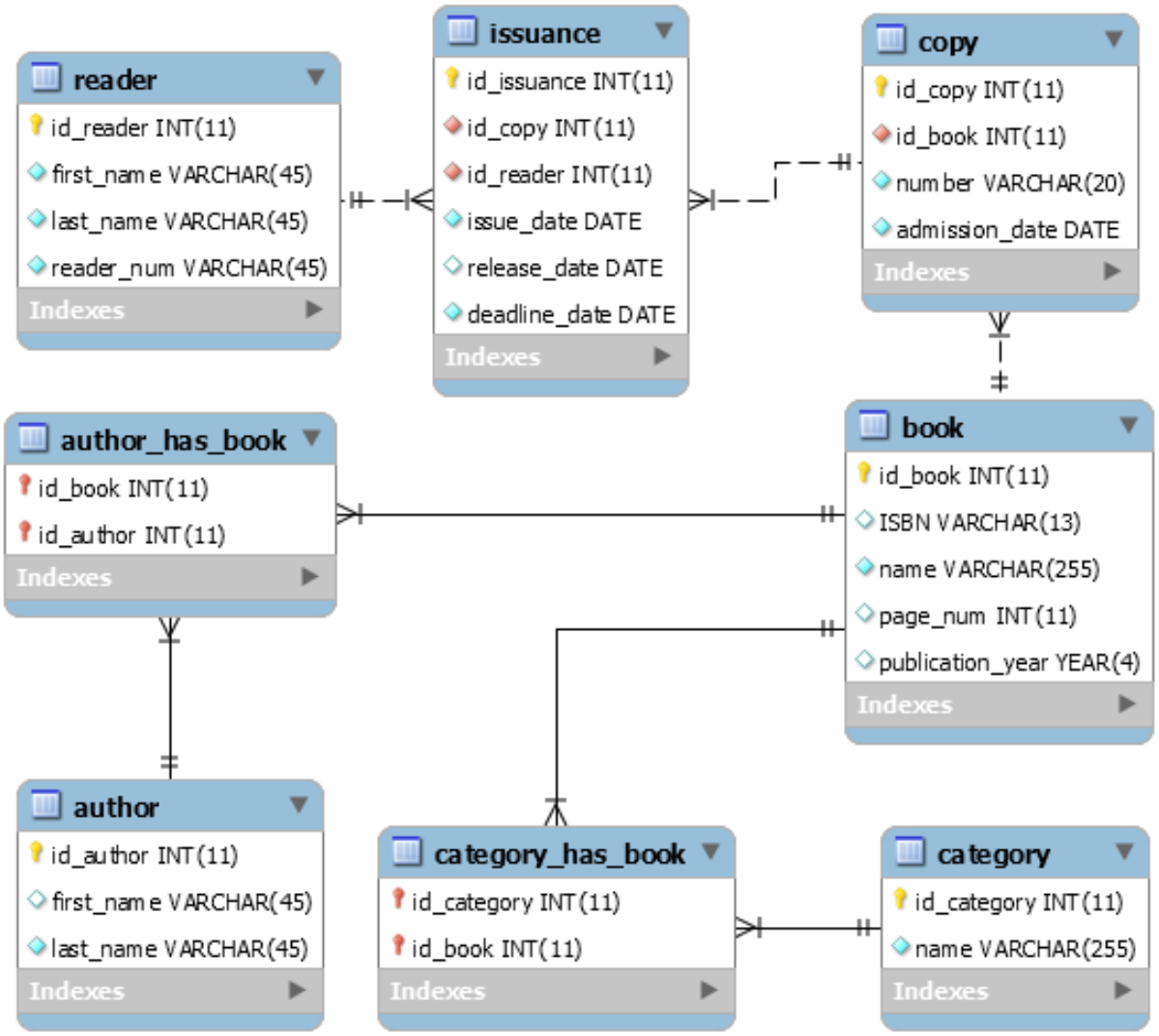
SELECT DISTINCT singer **FROM** artist

singer
The Prodigy
Drowning Pool
Massive Attack

SELECT COUNT(DISTINCT singer**)**
AS count_of_singers **FROM** artist

count_of_singers
3

JOINS. Предметная область “Библиотека”



Оператор JOIN

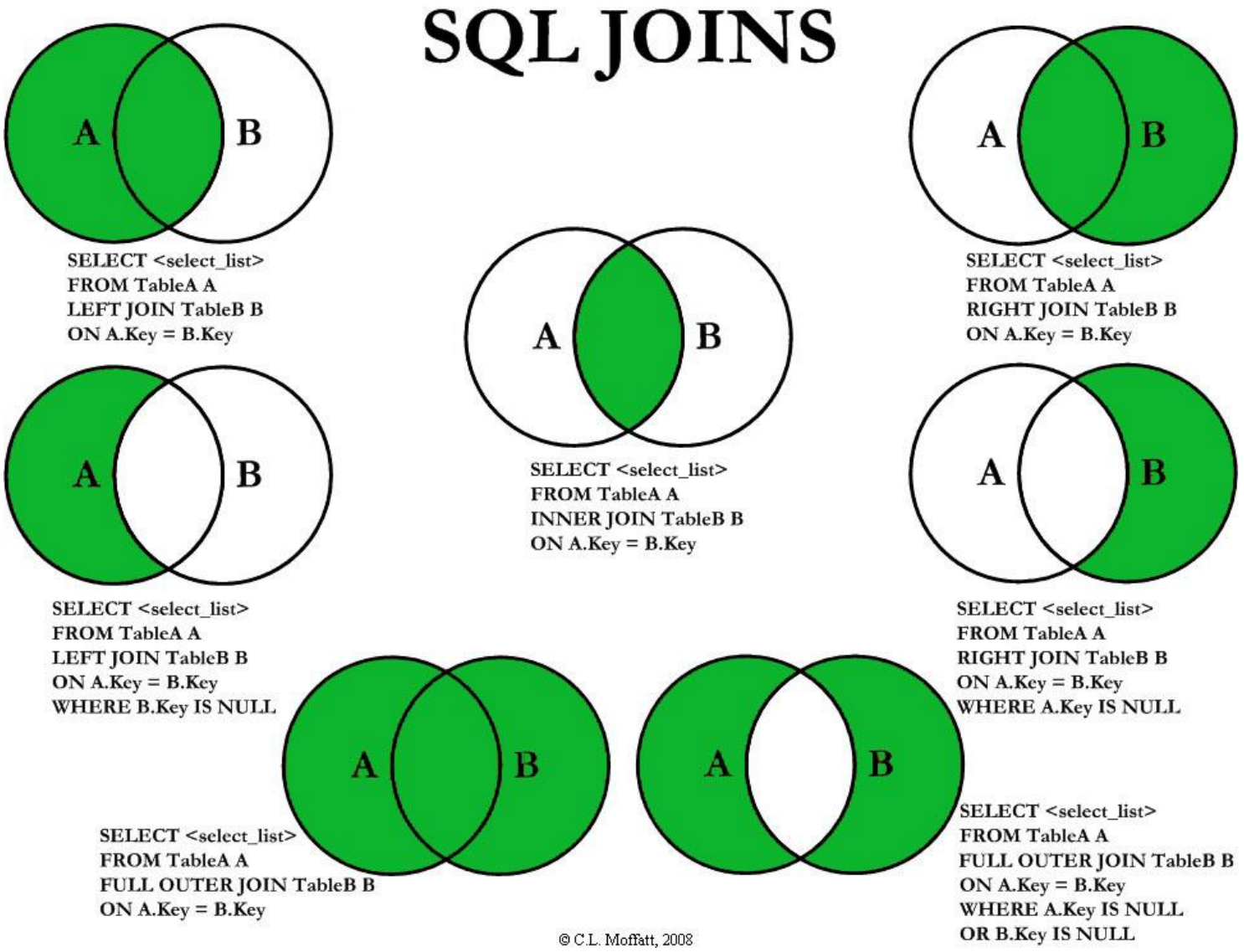
SQL JOIN — оператор языка SQL, который является реализацией операции соединения реляционной алгебры.

Операция соединения, как и другие бинарные операции, предназначена для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор. Отличительными особенностями операции соединения являются следующее:

- в схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-операндов), то есть схема результата является «сцеплением» схем операндов;
- каждая строка таблицы-результата является «сцеплением» строки из одной таблицы-операнда со строкой второй таблицы-операнда.

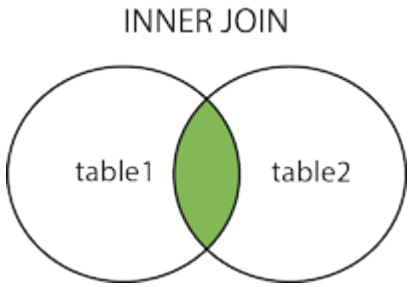
```
SELECT
   column_names [,... n]
FROM
   Table_1
JOIN Table_2 ON condition
```

JOINS


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

SQL INNER JOIN

- **INNER JOIN (CROSS JOIN)** - внутреннее (перекрёстное) объединение.
- Данный тип объединения позволяет извлекать строки, которые **обязательно** присутствуют во всех объединяемых таблицах.




```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

SQL INNER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL INNER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL INNER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL INNER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

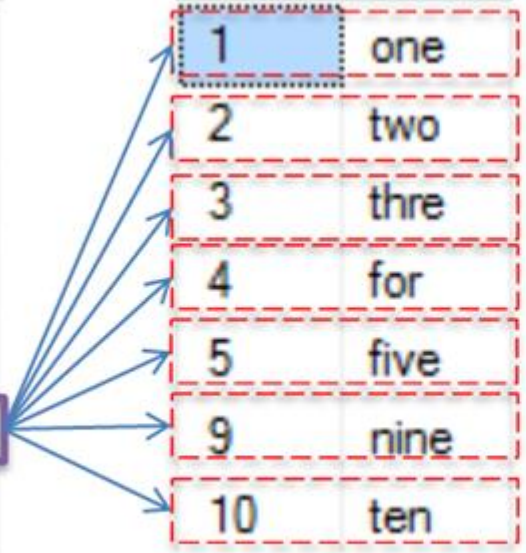


Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL INNER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

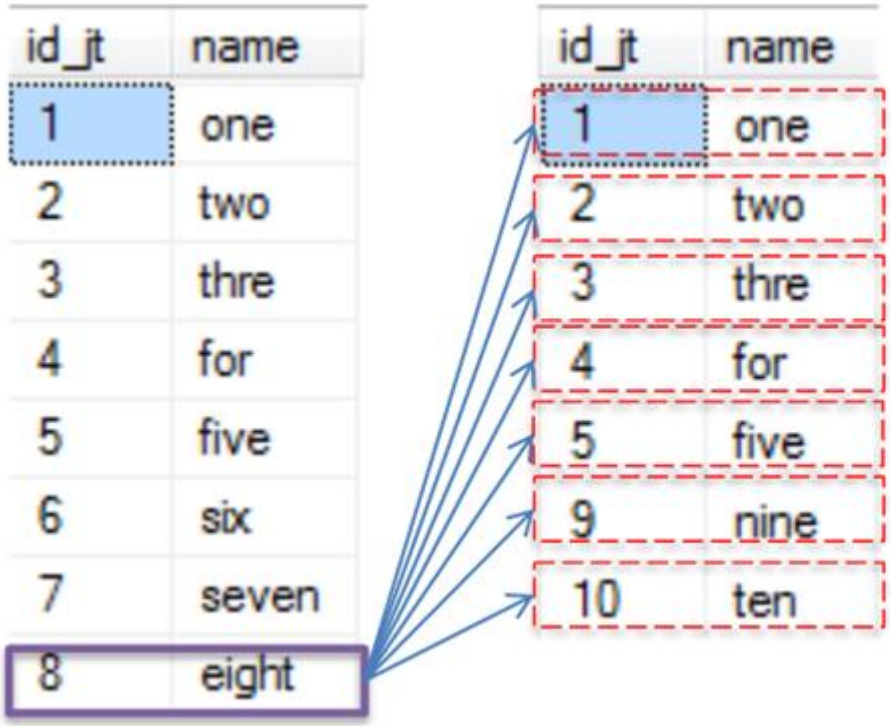


Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL INNER JOIN

Какой результат получим если убрать условие ON ?

```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

Выборка вернёт декартово произведение, в котором каждая строка одной таблицы будет сопоставлена с каждой строкой другой таблицы.

Помимо конструкции **INNER JOIN** внутреннее объединение можно объявить так же через **CROSS JOIN, JOIN** и запятую в объявлении **FROM**.

ON vs USING

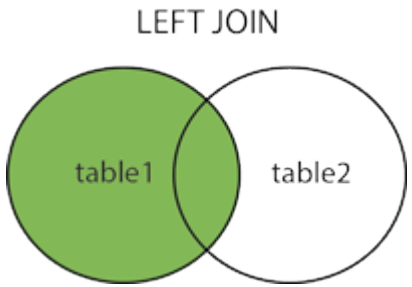
```
SELECT *
FROM join_test1
INNER JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

USING (column_name(s)), по сути, является синтаксическим сахаром над **ON**. Согласно документации служит для указания списка столбцов, которые должны существовать **в обеих** таблицах.

```
SELECT *
FROM join_test1
INNER JOIN join_test2 USING(id_jt);
```

SQL LEFT JOIN

- **LEFT JOIN** - Левостороннее внешнее объединение.
- Левосторонние объединения позволяют извлекать данные из таблицы, **дополняя** их по возможности данными из другой таблицы.




```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

SQL LEFT JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five

```
SELECT *
FROM join_test1
LEFT JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL LEFT JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

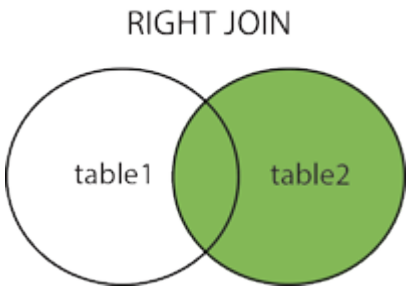
Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five
6	six	NULL	NULL
7	seven	NULL	NULL
8	eight	NULL	NULL

```
SELECT *
FROM join_test1
LEFT JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL RIGHT JOIN

- **RIGHT JOIN** - Правостороннее внешнее объединение.
- Правостороннее объединение позволяет извлекать данные из таблицы (указанной после **JOIN**), **дополняя** их по возможности данными из таблицы, (указанной после **FROM**).




```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

SQL RIGHT JOIN

Таблица - результирующий набор

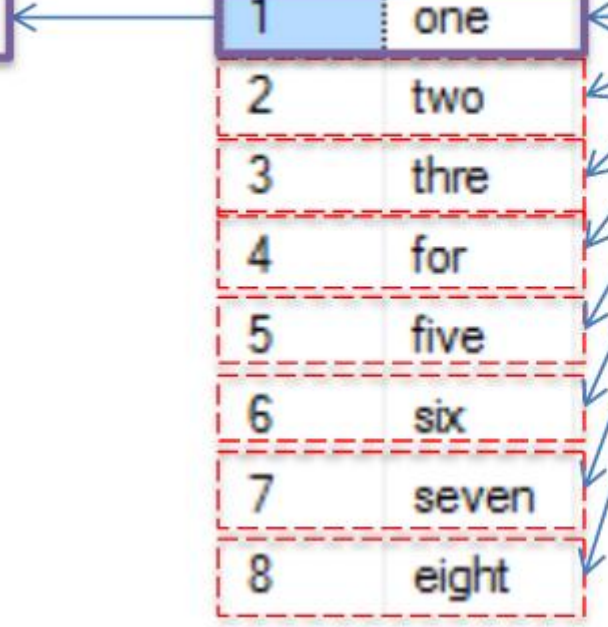
id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five
NULL	NULL	9	nine
NULL	NULL	10	ten

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten



```
SELECT *
FROM join_test1
RIGHT JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL RIGHT JOIN

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five
NULL	NULL	9	nine
NULL	NULL	10	ten

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

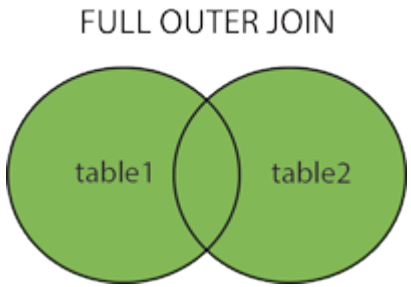
Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

```
SELECT *
FROM join_test1
RIGHT JOIN join_test2 ON join_test1.id_jt = join_test2.id_jt;
```

SQL FULL OUTER JOIN

- **FULL JOIN** - Полное внешнее объединение.
- Порядок таблиц для оператора неважен, поскольку оператор является симметричным.



1. В результат включается внутреннее соединение (INNER JOIN) первой и второй таблиц по предикату p.
2. В результат добавляются те записи первой таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких записей поля, соответствующие второй таблице, заполняются значениями NULL.
3. В результат добавляются те записи второй таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких записей поля, соответствующие первой таблице, заполняются значениями NULL.

SQL FULL OUTER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five

SQL FULL OUTER JOIN

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten

Таблица - результирующий набор

id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five
6	six	NULL	NULL
7	seven	NULL	NULL
8	eight	NULL	NULL

SQL FULL OUTER JOIN

Таблица - результирующий набор

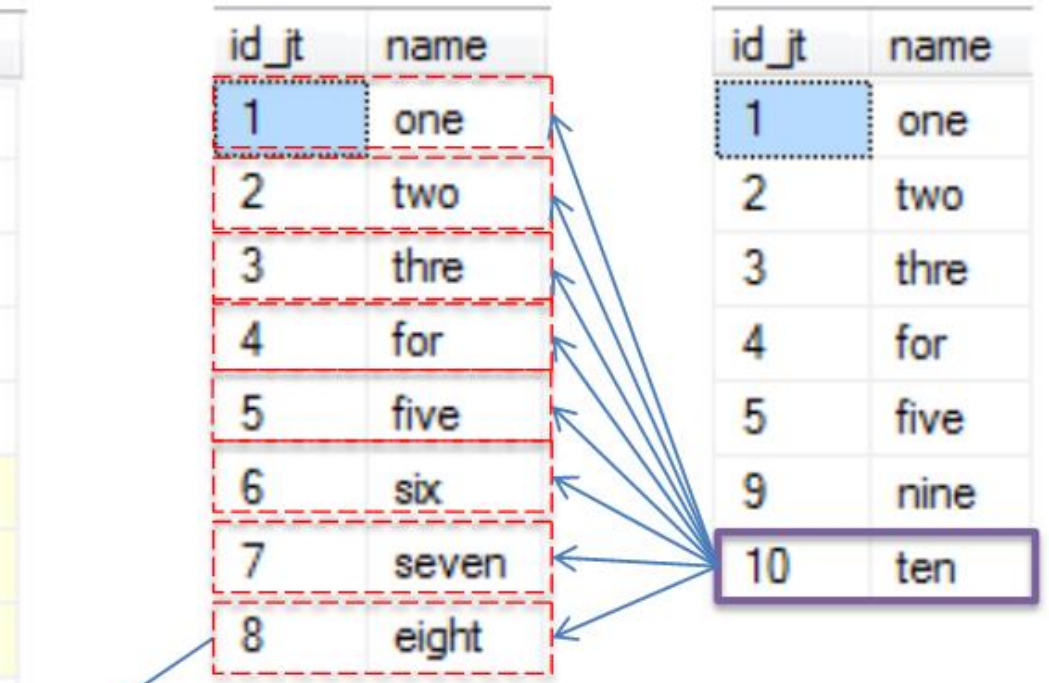
id_jt	name	id_jt	name
1	one	1	one
2	two	2	two
3	thre	3	thre
4	for	4	for
5	five	5	five
6	six	NULL	NULL
7	seven	NULL	NULL
8	eight	NULL	NULL
NULL	NULL	9	nine
NULL	NULL	10	ten

Таблица join_test1

id_jt	name
1	one
2	two
3	thre
4	for
5	five
6	six
7	seven
8	eight

Таблица join_test2

id_jt	name
1	one
2	two
3	thre
4	for
5	five
9	nine
10	ten



MYSQL FULL OUTER JOIN

В MySQL отсутствует **FULL OUTER JOIN** и используется оператор **UNION**

```
SELECT * FROM join_test1
LEFT JOIN join_test2 ON join_test1.id = join_test2.id
```

UNION

```
SELECT * FROM join_test1
RIGHT JOIN join_test2 ON join_test1.id = join_test2.id
```

Два правила, регламентирующие порядок использования оператора **UNION**:

- Число и порядок извлекаемых столбцов должны совпадать во всех объединяемых запросах;
- Типы данных в соответствующих столбцах должны быть совместимы.

UNION

Исходная таблица sale2010

person	amount
Joe	1000
Alex	2000
Bob	5000

Исходная таблица sale2011

person	amount
Joe	2000
Alex	2000
Zach	35000

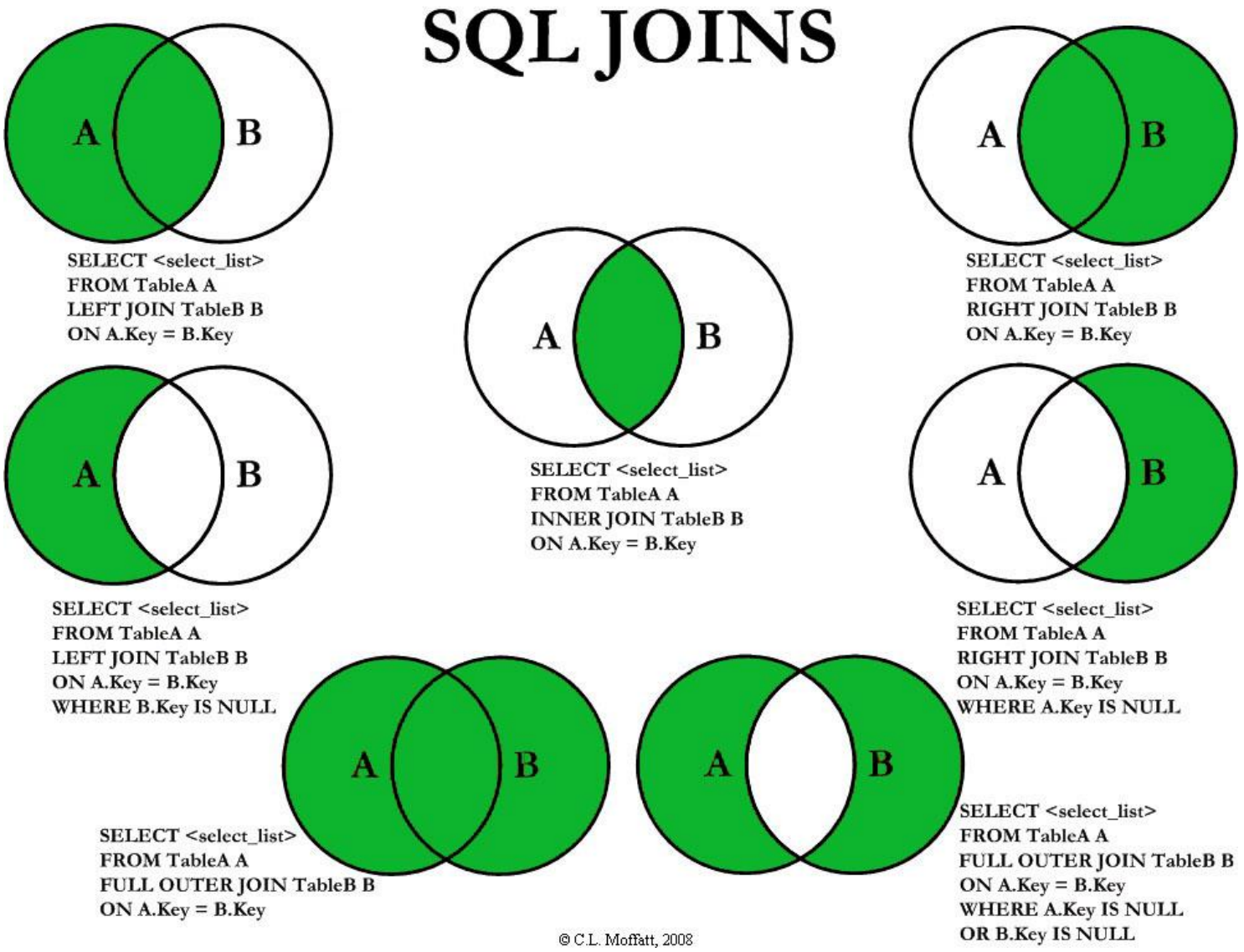
```
SELECT * FROM sale2010
UNION
SELECT * FROM sale2011;
```

person	amount
Joe	1000
Alex	2000
Bob	5000
Joe	2000
Zach	35000

```
SELECT * FROM sale2010
UNION ALL
SELECT * FROM sale2011;
```

person	amount
Joe	1000
Joe	2000
Alex	2000
Alex	2000
Bob	5000
Zach	35000

JOINS


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

SUBQUERIES / ПОДЗАПРОСЫ

- Подзапросы (вложенные запросы) – это запросы, используемые в инструкциях **SELECT, UPDATE, INSERT** и **DELETE**

Подзапросы могут использоваться для следующих целей:

- Для определения множества строк, вставляемых в целевую таблицу выражениями **INSERT** или **CREATE TABLE**
- Для определения одного или более значений, назначаемых существующим строкам в утверждении **UPDATE**
- Для обеспечения необходимых условий в выражениях **WHERE, HAVING** утверждений **SELECT, UPDATE, и DELETE**
- Способ выборки данных из нескольких таблиц

SUBQUERIES

Использование результата подзапроса в качестве атрибута результирующей выборки

```
SELECT
    col0,
    (SELECT col1 FROM table1 WHERE table1.id = 1),
    (SELECT col2 FROM table1 WHERE table1.id = 1)
FROM
    table0
```

Использование результата подзапроса для определения критерия выборки

```
SELECT * FROM `author`
WHERE id_author IN (
  SELECT id_author FROM author_has_book
  GROUP BY id_author
  HAVING COUNT(*) > 3
);
```

SUBQUERIES

Использование результата подзапроса в качестве таблицы для выборки

```
SELECT * FROM ( SELECT * FROM table1 ) AS table
```

Использование результата подзапроса для присоединения к таблице

```
SELECT * FROM table1
    JOIN ( SELECT * FROM table ) AS table ON …
```

SUBQUERIES. [NOT] EXISTS

Предикат **EXISTS** принимает значение **TRUE**, если подзапрос содержит любое количество строк, иначе его значение равно **FALSE**.

Найти всех авторов у которых нет книг

```
SELECT * FROM author
WHERE NOT EXISTS (
   SELECT * FROM author_has_book
   WHERE author.id_author = author_has_book.id_author
);
SELECT * FROM author
WHERE id_author NOT IN (
   SELECT DISTINCT id_author FROM author_has_book
);
```

Найти всех авторов у которых есть книги

```
SELECT * FROM author
WHERE EXISTS (
   SELECT * FROM author_has_book
   WHERE author.id_author = author_has_book.id_author
);
```

SUBQUERIES. IN, ANY, ALL

Операторы сравнения с вложенными запросами могут быть уточнены с помощью ключевых слов **ALL** или **ANY**

- Слово **ALL**, которое следует за оператором сравнения, означает «return **TRUE**, если сравнение верно для ВСЕХ значений в столбце».
- **ALL** – обозначает Например, > **ALL** (1, 2, 3) означает «больше 3»

- Слово **ANY**, которое следует за оператором сравнения, означает «return **TRUE**, если сравнение верно для ЛЮБОГО значений в столбце».
- **ANY** – обозначает Например, > **ANY** (1, 2, 3) означает «больше 1»

SUBQUERIES. IN, ANY, ALL

Найти модели и цены портативных компьютеров, стоимость которых превышает стоимость любого ПК

```
SELECT DISTINCT model, price
FROM laptop
WHERE price > ALL (SELECT price FROM pc);
```

Найти поставщиков компьютеров, моделей которых нет в продаже (то есть модели этих поставщиков отсутствуют в таблице PC)

```
SELECT DISTINCT maker
FROM product
WHERE type = 'pc' AND NOT model = ANY (SELECT model FROM PC);
```