Подзапросы.
JOIN.

Определение

- **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.
- Альтернативой JOIN могут выступать подзапросы или подзапросы с объединениями.

Подзапросы

- SELECT cat.name as category, c.first_name, c.middle_name, c.last_name, c.age FROM contact c, category cat WHERE c.id_category=cat.id;
- SELECT (***SELECT name from category cat where cat.id = c.id_category***) as category, c.first_name, c.middle_name, c.last_name, c.age FROM contact c;
- SELECT cv.value FROM contact_value cv WHEREcv.id_contact = (***SELECT id FROM contact WHERE age=20***);
- SELECT cv.value FROM contact_value cv WHERE cv.id_contact IN (***SELECT id FROM contact WHERE age<>20***);
- SELECT c.first_name, c.middle_name, c.last_name, c.age FROM contact c WHERE c.age > ***(SELECT avg(age) FROM contact);***

JOIN (Соединение)

Типы JOIN:

- INNER JOIN
- CROSS JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

id	city_name
2	Киев
1	Харьков
3	Одесса

Таблица **A** (**city**)

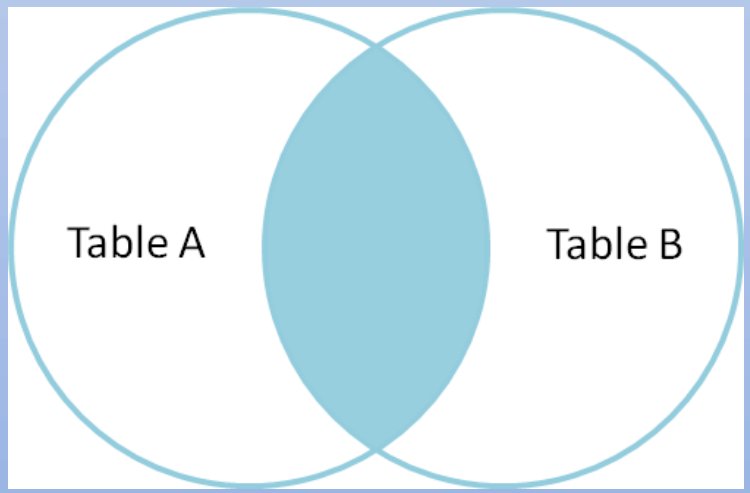
name	id_city
Иван	1
Сергей	2
Николай	1
Дмитрий	4

Таблица **B** (**person**)

INNER JOIN

- SELECT * FROM person p INNER JOIN city c ON p.id_city = c.id
- (SELECT * FROM person p, city c WHERE p.id_city = c.id)

name	id_city	id	city_name
Иван	1	1	Харьков
Сергей	2	2	Киев
Николай	1	1	Харьков



CROSS JOIN

- SELECT * FROM person p CROSS JOIN city c
- (SELECT * FROM person p, city c)

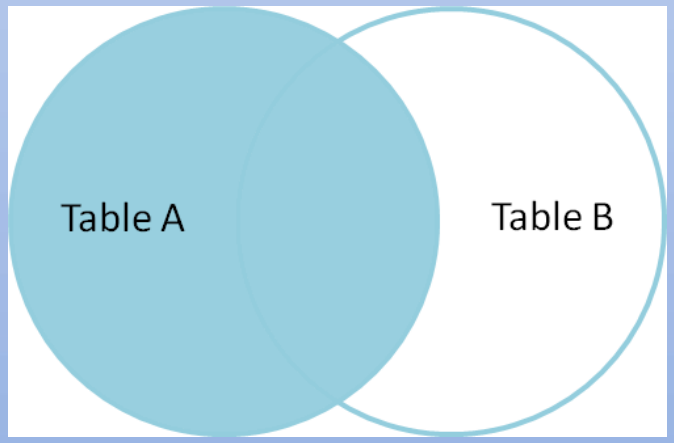
name	id_city	id	city_name
Иван	1	2	Киев
Иван	1	1	Харьков
Иван	1	3	Одесса
Сергей	2	2	Киев
Сергей	2	1	Харьков
Сергей	2	3	Одесса
Николай	1	2	Киев
Николай	1	1	Харьков
Николай	1	3	Одесса
Дмитрий	4	2	Киев
Дмитрий	4	1	Харьков
Дмитрий	4	3	Одесса

LEFT OUTER JOIN

- SELECT * FROM person p LEFT OUTER JOIN city c ON p.id_city = c.id

- (SELECT * FROM person p, city c WHERE p.id_city = c.id UNION
  SELECT p.*, NULL as id, NULL as city_name FROM person p WHERE id_city NOT IN (SELECT id FROM city))

name	id_city	id	city_name
Иван	1	1	Харьков
Сергей	2	2	Киев
Николай	1	1	Харьков
Дмитрий	4	Null	Null

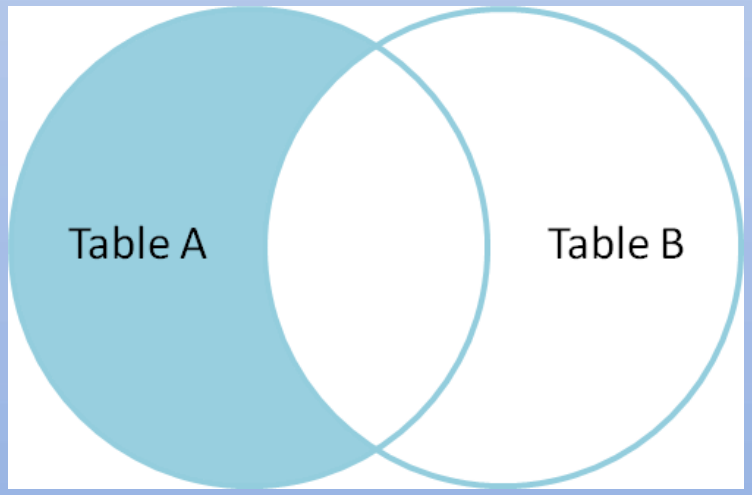


LEFT OUTER JOIN (кроме правой)

- SELECT * FROM person p LEFT OUTER JOIN city c ON p.id_city = c.id WHERE c.id IS NULL

- (SELECT p.*, NULL as id, NULL as city_name FROM person p WHERE id_city NOT IN (SELECT id FROM city))

name	id_city	id	city_name
Дмитрий	4	Null	Null

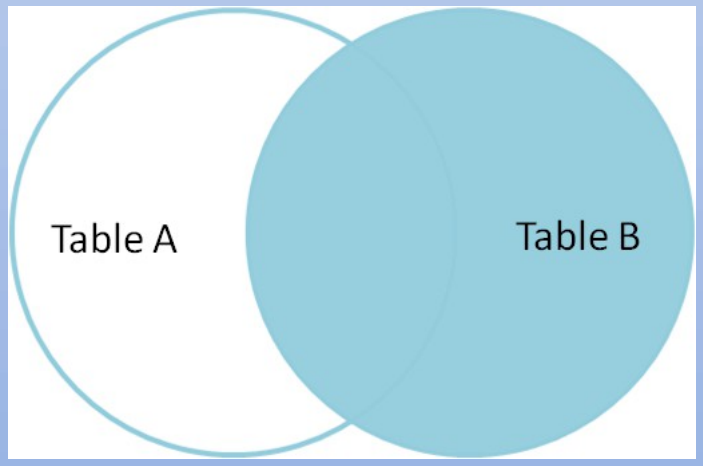


RIGHT OUTER JOIN

- SELECT * FROM person p RIGHT OUTER JOIN city c ON p.id_city = c.id

- (SELECT * FROM person p, city c WHERE p.id_city = c.id UNION
  SELECT NULL as name, NULL as id_city, c.* FROM city c WHERE id NOT IN (SELECT id_city FROM person))

name	id_city	id	city_name
Сергей	2	2	Киев
Николай	1	1	Харьков
Иван	1	1	Харьков
Null	Null	3	Одесса



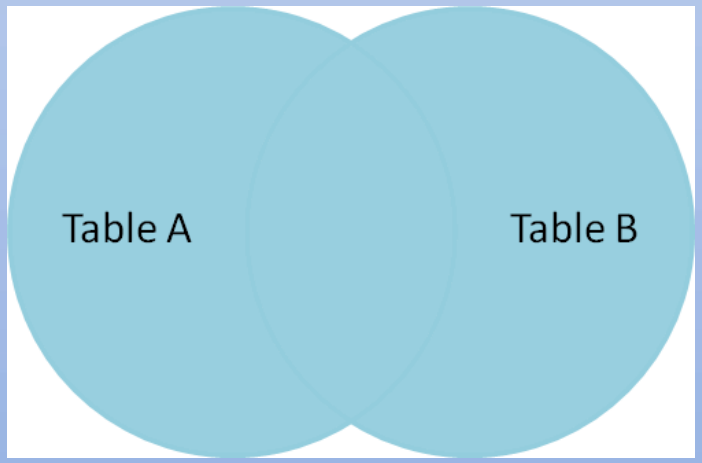
FULL OUTER JOIN

- SELECT * FROM person p FULL OUTER JOIN city c ON p.id_city = c.id
- (SELECT * FROM person p, city c WHERE p.id_city = c.id UNION

  SELECT p.*, NULL as id, NULL as city_name FROM person p WHERE id_city NOT IN (SELECT id FROM city) UNION

  SELECT NULL as name, NULL as id_city, c.* FROM city c WHERE id NOT IN (SELECT id_city FROM person))

name	id_city	id	city_name
Иван	1	1	Харьков
Сергей	2	2	Киев
Николай	1	1	Харьков
Дмитрий	4	Null	Null
Null	Null	3	Одесса



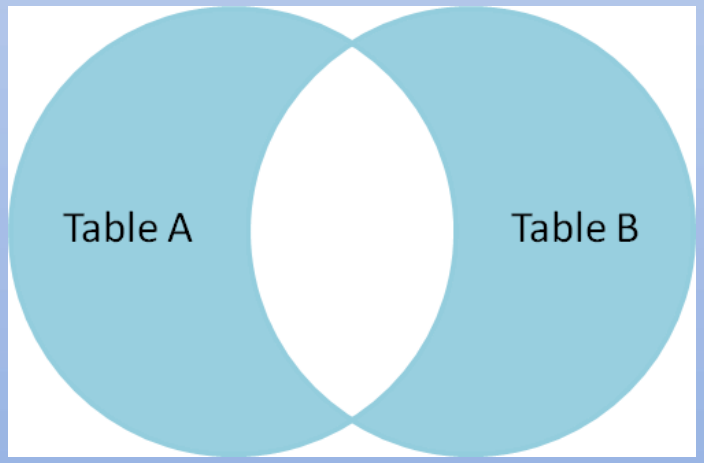
FULL OUTER JOIN (кроме общих)

- SELECT * FROM person p FULL OUTER JOIN city c ON p.id_city = c.id WHERE c.id IS NULL OR p.id_city IS NULL

- SELECT p.*, NULL as id, NULL as city_name FROM person p WHERE id_city NOT IN (SELECT id FROM city) UNION

  SELECT NULL as name, NULL as id_city, c.* FROM city c WHERE id NOT IN (SELECT id_city FROM person)

name	id_city	id	city_name
Дмитрий	4	Null	Null
Null	Null	3	Одесса



Использование JOIN

Найти категории для которых нет ни одного контакта:

- SELECT cat.* FROM category cat LEFT OUTER JOIN contact c ON cat.id = c.id_category WHERE c.id_category IS NULL;
- SELECT * FROM category WHERE id NOT IN (SELECT id_category FROM contact);

Найти категории для которых есть хотя бы один контакт:

- SELECT DISTINCT cat.* FROM category cat, contact c WHERE cat.id = c.id_category;
- SELECT DISTINCT cat.* FROM category cat INNER JOIN contact c ON cat.id = c.id_category;
- SELECT DISTINCT cat.* FROM category cat CROSS JOIN contact c WHERE cat.id = c.id_category;
- SELECT * FROM category WHERE id IN (SELECT id_category FROM contact);

Выводы

1. **JOIN** — оператор языка SQL, который является реализацией операции соединения реляционной алгебры;
2. **Типы JOIN:** INNER JOIN, CROSS JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN;
3. INNER JOIN отображает общие данные, которые присутсвуют в нескольких таблицах;
4. CROSS JOIN выполняет декартовое произведение между таблицами;
5. LEFT OUTER JOIN отображает данные левой таблицы и правой если есть соответсвие или NULL если соответствия нет;
6. RIGHT OUTER JOIN (инвертированная версия LEFT OUTER JOIN, если поменять местами таблицы, то получим LEFT OUTER JOIN) отображает данные правой таблицы и левой если есть соответствия или NULL если соответствия нет;
7. FULL OUTER JOIN отображает данные общие для левой и правой, левую таблицу и NULL справа и правую таблицу и NULL слева;
8. Для того чтобы исключить общие данные из LEFT OUTER JOIN, RIGHT OUTER JOIN или FULL OUTER JOIN используется WHERE и сравнение по IS NULL;
9. Альтернативой JOIN являются подзапросы, которые могут быть между SELECT и FROM, в WHERE и HAVING;
10. Оператор JOIN и подзапросы можно использовать для организации запроса в рамках одной таблицы;
11. Если возникает выбор использовать подзапрос или JOIN, следует выбирать JOIN, так как запросы с JOIN выполняются быстрее чем подзапросы, хотя несомненно с точки зрения чтения запроса человеком, подзапрос читается проще, чем JOIN.