***Типы Join:***

**Cross join** – декартово произведение соединенных таблиц (синтаксис без условия определения, Cross так же можно опускать).

**Select** \* **from** <Источник> **join** <Источник>

**Inner** **join** – возвращает строки, где есть совпадения в обеих таблицах по указанному условию (в результирующую выборку не попадут строки, для которых нет совпадений в одной из таблиц).

Условия соединения указываются с помощью ON или USING (если имена атрибутов совпадают).

**Outer join** – помимо строк, имеющих соответствующее значение, так же возвращает строки, которые не имеют соответствий.

*Подразделяются на:*

**Left Outer join** – Возвращает все строки из левой таблицы, строки имеющие соответствия заполняются из правой таблицы, не имеющие заполняются null.

**Right Outer join** – Возвращаются все строки из правой таблицы, если нет совпадений для них, то возвращаются null.

Взаимозаменяемы, по сути можно всегда использовать Left или Right Join, переставляя таблицы в запросе местами.

**Full Outer join** – объединяет в себе Left и Right join, то есть возвращает все строки из обеих таблиц, пустоты заполняются null.

**Natural join** – автоматически объединяет все атрибуты с одинаковыми именами и типами данных (синтаксис без условия). Имеет механику Inner join, если пересекающихся атрибутов нет, то сработает как Cross join (не является видом join, скорее, является особым типом синтаксиса).

Не рекомендуется к использованию в связи с неясностью объединения и возможным непредсказуемым результатам.

**Self join** – Соединение таблицы со своей же копией, не является разновидностью Join (является способом использования).

Используется для таблиц с иерархической структурой (ссылается сама на себя).

**UNION, EXCEPT, INTERSECT –** используются для выполнения операций для наборов данных, полученных с помощью операторов Select для их последующего специфического для каждого оператора объединения.

UNION – убирает дубликаты при выборке, оставляя только уникальные строки. При этом UNION ALL полностью объединяет выборки (включая дубликаты).

EXCEPT – возвращает уникальные строки из главного набора (первого), т.е строки, которых нет в побочном запросе.

INTERSECT – возвращает пересечения из двух выборок (неуникальные значения).

Условия использования:

1. Требуют одинакового кол-ва атрибутов и совместимость типов для всех выборок.
2. Главный набор определяет названия атрибутов в результирующей выборке и условия сортировки (можно сортировать только по главному набору).

**Подзапросы:**

**Виды по количеству возвращаемых данных:**

**Скалярный операнд***:* Результатом запроса будет одно значение подзапроса, с ним используются операторы сравнения ‘>’ ‘<’ ‘=’

**Табличный операнд**: Подзапрос возвращает набор значений, с ним используются in, all (для использования операторов сравнения – вернет true, если все сравнения истины), any

**Некоррелированный подзапрос** – выполняется один раз, независим от основного запроса (может быть исполнен отдельно).

**Select** \* **from** table\_1

**where** salary > (**select** **avg** (salary) **from** table\_2)

выбирает строки, где зарплата выше средней. Сначала выполняется подзапрос и это значение сравнивается с каждой строкой внешнего запроса.

**Коррелированный подзапрос** – зависит от данных внешнего запроса т.к. ссылается на атрибуты основного запроса, исполняется для каждой строки основного запроса, используя значения из этой строки в качестве параметров для подзапроса.

**Select** \*, (**select** **avg**(salary) **from** workers

**where** workers.department\_id = departments.id) **from** departments

Запрос коррелированный, т.к. содержит в себе обращение к внешней таблице departments.

Исполнение такого запроса будет происходить для каждой строки основного, что может сказаться на производительности.

Подзапросы могут быть использованы в качестве источника данных для FROM или JOIN.

Подзапросы чаще всего можно заменить другими конструкциями (join, group by). В большинстве случаев следует избегать использования подзапросов, т.к. они являются менее производительным и менее читаемыми.

В некоторых ситуациях планировщик запросов оптимизирует подзапросы до join, чтобы точнее узнать, каким образом будет исполняться запрос используется EXPLAIN или EXPLAIN ANALIZE.