

SQL Server Developer

SELECT и простые фильтры, JOIN



Проверить, идет ли запись

Меня хорошо видно && слышно?



Ставим "+", если все хорошо
"-", если есть проблемы

SELECT, простые фильтры, JOIN

- Select
- Where
- Union
- Join



Активно
участвуем



Off-topic обсуждаем
в учебной группе
telegram



Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

Select

- синтаксис соответствует английской грамматике

В каком порядке SQL выполняет операторы?

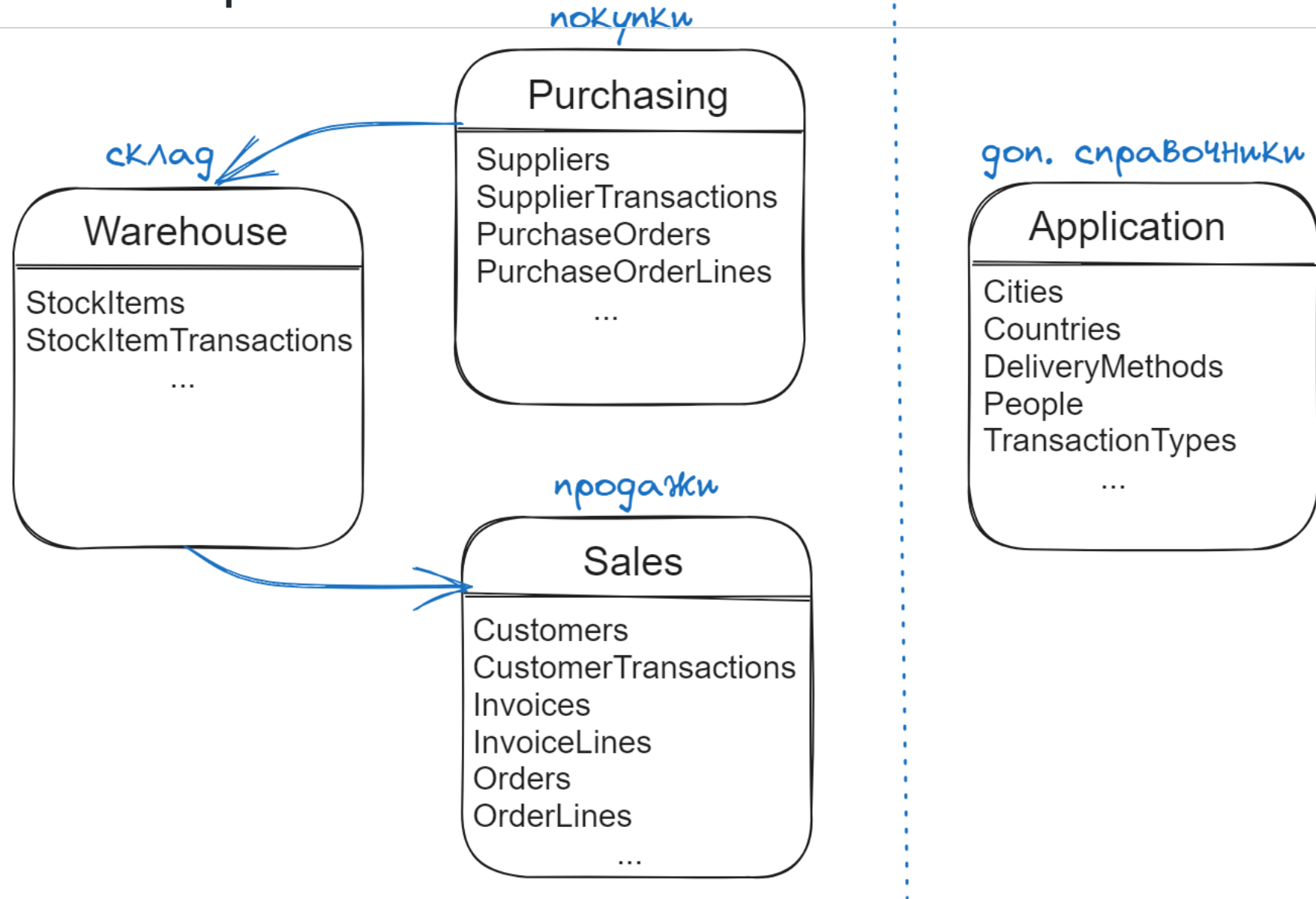
```
(1) select CityName as city, count(*) as qty  
(2) from Application.Cities as c  
(3) where LastEditedBy = 1  
(4) group by CityName  
(5) having count(*) > 25  
(6) order by count(*) desc
```

Порядок выполнения операторов SQL

```
(2) from Application.Cities as c
(3) where LastEditedBy = 1
(4) group by CityName
(5) having count(*) > 25
(1)->(5.5) select CityName, count(*) --! на предпоследнем месте
(6) order by count(*) desc
```

```
-- использование алиаса (псевдонима):
(5) select CityName as city, count(*) as qty
(1) from Application.Cities as c
(2) where LastEditedBy = 1
(3) group by CityName
(4) having qty > 25 --error! qty еще не определен
(6) order by qty desc --qty определен, ошибки нет
```

WideWorldImporters



Select - выбор данных

- `select * from ... where` - с осторожностью на больших таблицах
 - оптимальнее выбирать колонки

Ограничение по кол-ву строк

- `distinct` - удаление дублей
- `top N` без `order by` - порядок не гарантирован
- `order by 1, 2` - по номерам колонок - лучше не использовать
- порядок сортировки зависит от типа данных

Where - условие отбора

- логическое выражение (предикат)
 - проверяется для каждой строки
 - принимает 3 значения: true, false, unknown
- в итог попадут строки, у которых условие выполнено (= true)
- в колонке может быть null (не известно) - не имеет типа
 - сравнение с null всегда не определено (unknown)
- SARGable - предикаты (используют индекс: index seek - в плане запроса)
 - функция в левой части условия - зло! не позволяет использовать index seek

Where - несколько условий

Операции с логическими выражениями

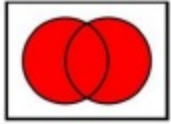
- `OR` - логическое сложение
- `AND` - логическое умножение
- `NOT` (или `!`) - логическое отрицание

Приоритет

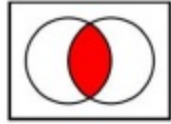
0. `NOT`
1. `(...)` - скобки
2. `AND`
3. `OR`

Операции с множествами

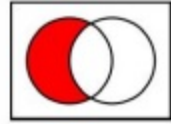
UNION, UNION ALL



INTERSECT



EXCEPT



- **UNION ALL** - объединение 2х таблиц: $t1 + t2$, включая дубли
- **UNION** - объединение: $t1 + t2$, исключая дубли
- **INTERSECT** - общие строки, которые есть в обеих "таблицах"
 - высокий приоритет - всегда выполняется первым
- **EXCEPT** - уникальные строки из $t1$, которых нет в $t2$
- **CROSS JOIN** - "умножение" (декартово произведение)

Операции над множествами - особенности

- количество колонок в таблицах - должны совпадать
- типы совмещаемых колонок должны быть совместимы
- названия колонок - берутся из 1го запроса
- `null` - обрабатывается как обычное значение
- нужен свой порядок выполнения операций - ставим скобки `()`

Типы соединений таблиц

- `CROSS JOIN` - все сочетания строк из соединяемых таблиц, без условий
- `INNER JOIN` - соединение с проверкой условия
- `LEFT JOIN` - вся "левая" таблица, дополненная данными из "правой"
- `RIGHT JOIN` - вся "правая" таблица, дополненная данными из "левой"
- `FULL JOIN` = left join + right join

Cross join - декартово произведение

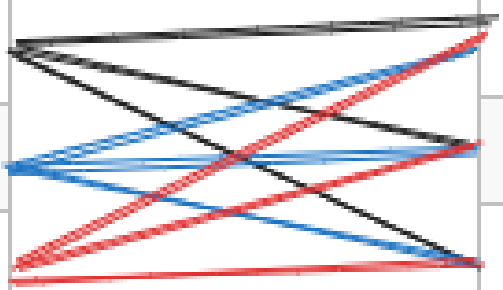
- все возможные комбинации строк 1й и 2й таблицы
- **все** строки из 1й таблицы комбинируются **со всеми** строками из 2й таблицы (получим все сочетания строк из 2х таблиц)
- **нет условий соединения**
- типовая задача: найти все комбинации из 2х таблиц

cafe1

id	name
1	яблоки
2	груши
3	бананы

cafe2

id	name
1	яблоки
2	груши
3	<i>null</i>



cross join

id	name	id	name
1	яблоки	1	яблоки
1	яблоки	2	груши
1	яблоки	3	<i>null</i>
2	груши	1	яблоки
2	груши	2	груши
2	груши	3	<i>null</i>
3	бананы	1	яблоки
3	бананы	2	груши
3	бананы	3	<i>null</i>

Inner join - внутреннее соединение

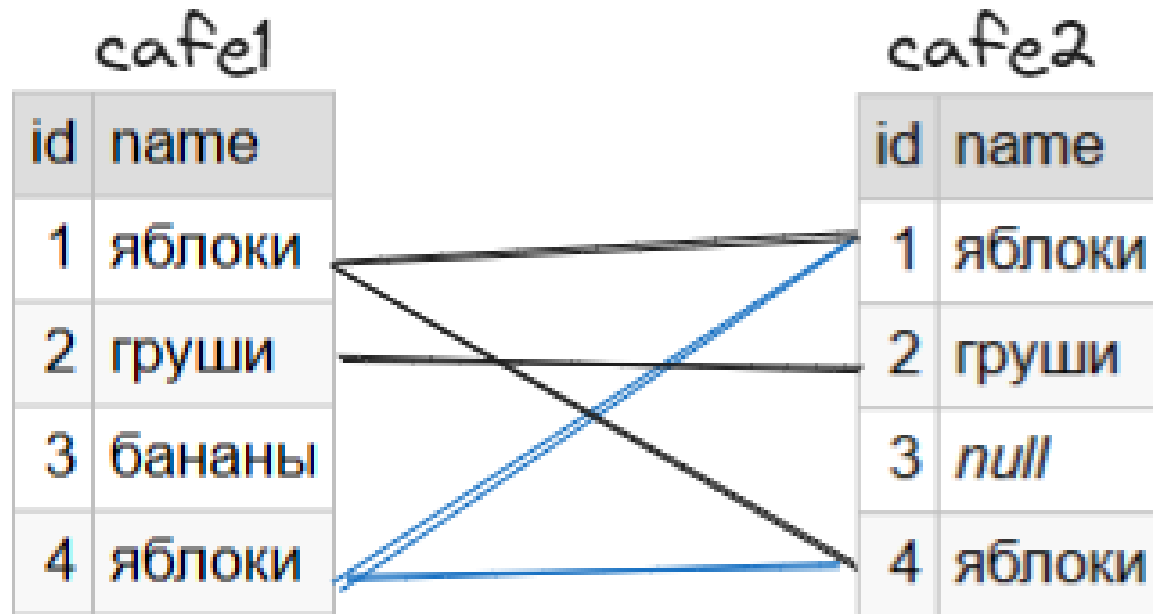
- соединение таблиц по условию
- обычно условие затрагивает поля обеих таблиц
 - `on table1.col1 = table.col2`
- при соединении для каждой пары строк проверяется условие из `ON`
 - совпало? - выводим строку
 - не совпало - пропускаем
- хорошо работает при соединении по **уникальному ключу**
- типовая задача: найти все строки, которые одновременно есть в обеих таблицах

cafe1			cafe2	
id	name		id	name
1	яблоки	—	1	яблоки
2	груши	—	2	груши
3	бананы		3	<i>null</i>

inner join
on cafe1.name = cafe2.name

id	name	id	name
1	яблоки	1	яблоки
2	груши	2	груши

Inner join - по полю с неуникальными записями



inner join
on cafe1.name = cafe2.name

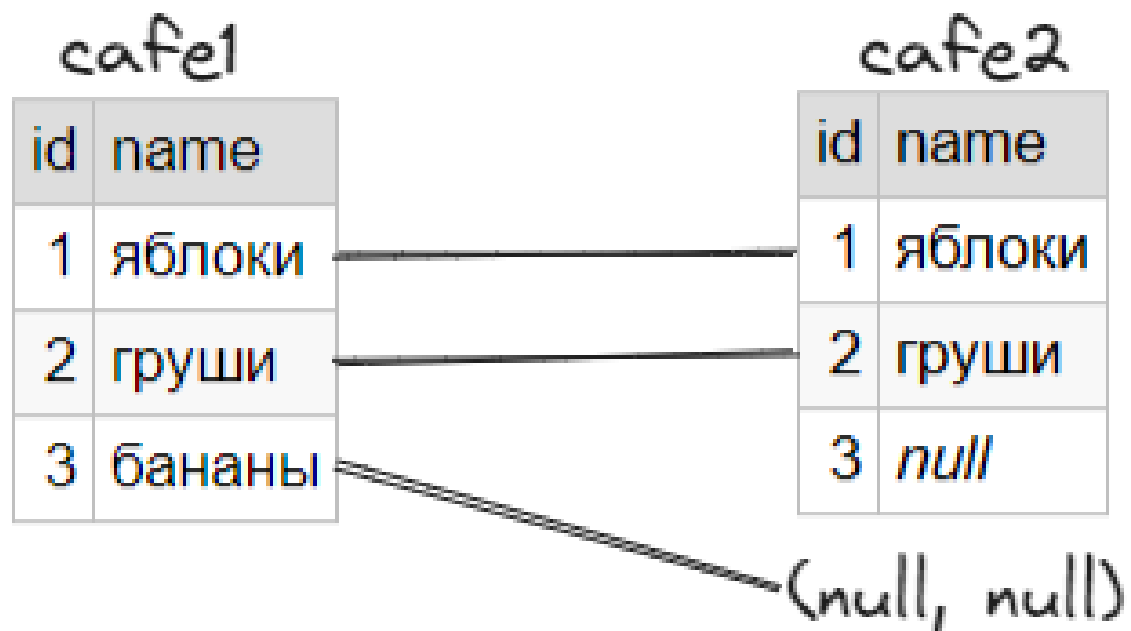
id	name	id	name
1	яблоки	1	яблоки
1	яблоки	4	яблоки
2	груши	2	груши
4	яблоки	1	яблоки
4	яблоки	4	яблоки

Inner join - особенности

- при соединении по полю с неуникальными записями получим дубликаты в итоговой таблице
 - некорректный подсчет количества строк `count(*)`
- 🚩 всегда проверяем уникальность полей из условия соединения
- 🚩 помним про `null`
- порядок соединения не важен

Left join - левое соединение

- есть условие, проверяем:
 - выполнено - строку выводим
 - не выполнено - строку выводим, но данные 2й таблицы заполняются `null`
- порядок таблиц важен
- `LEFT` - указывает, что из указанного направления (СЛЕВА) берем все строки
- типовая задача: найти все строки, которых нет в другой таблице



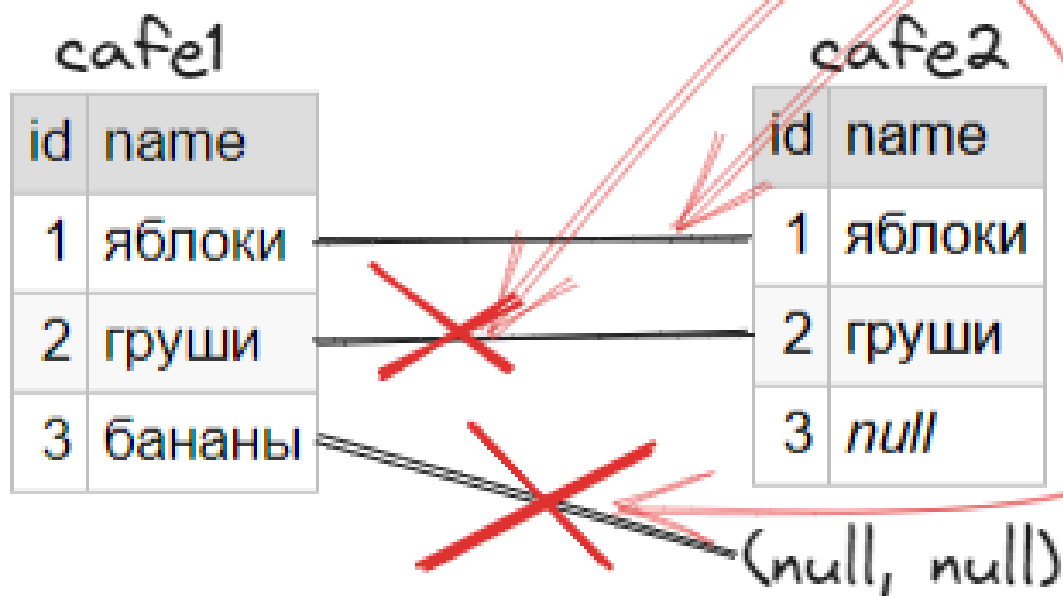
left join
on cafe1.name = cafe2.name

id	name	id	name
1	яблоки	1	яблоки
2	груши	2	груши
3	бананы	<i>null</i>	<i>null</i>

Left join - условие с колонкой из правой таблицы

- дополнить список фруктов из кафе1 данными по фруктам из кафе2, название которых начинается на 'Я'

проверка условия после соединения

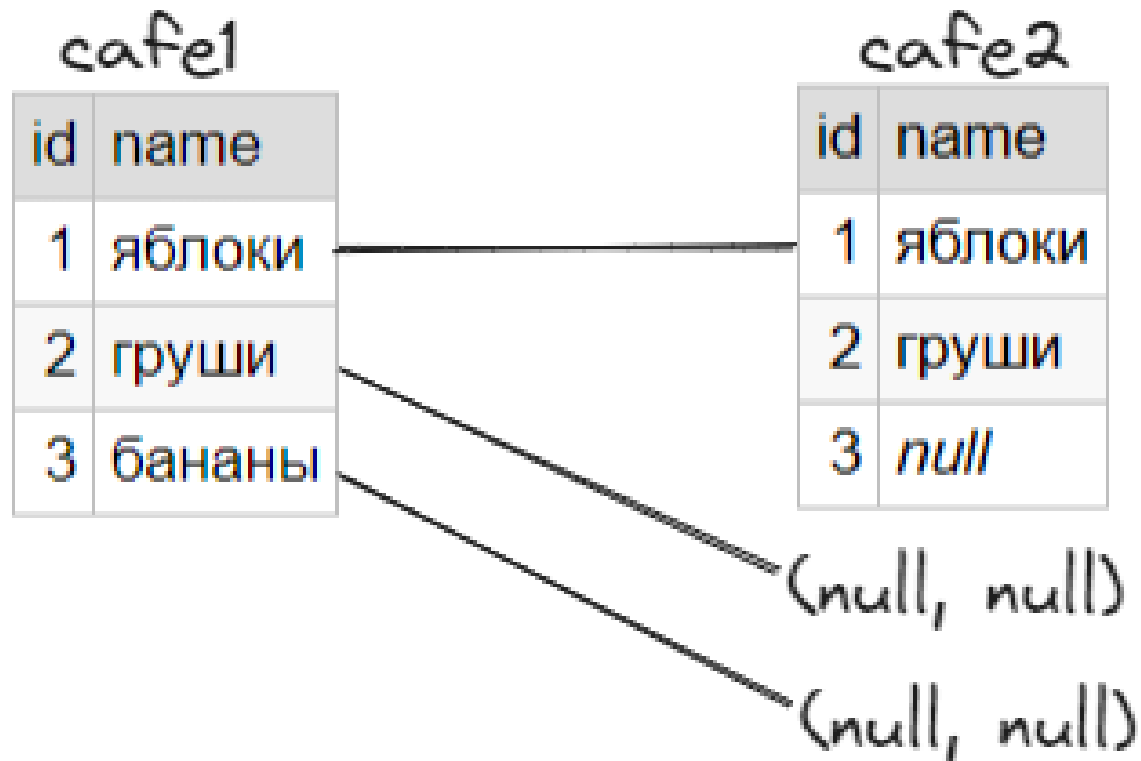


left join
on cafe1.name = cafe2.name
where cafe2.name like N'Я%'

id	name	id	name
1	яблоки	1	яблоки

Вариант решения

- перенести условие из `WHERE` в `ON`
проверка условия при соединении



left join
on `cafe1.name = cafe2.name`
and c2.name like N'я%'

id	name	id	name
1	яблоки	1	яблоки
2	груши	null	null
3	бананы	null	null

Left join - особенности

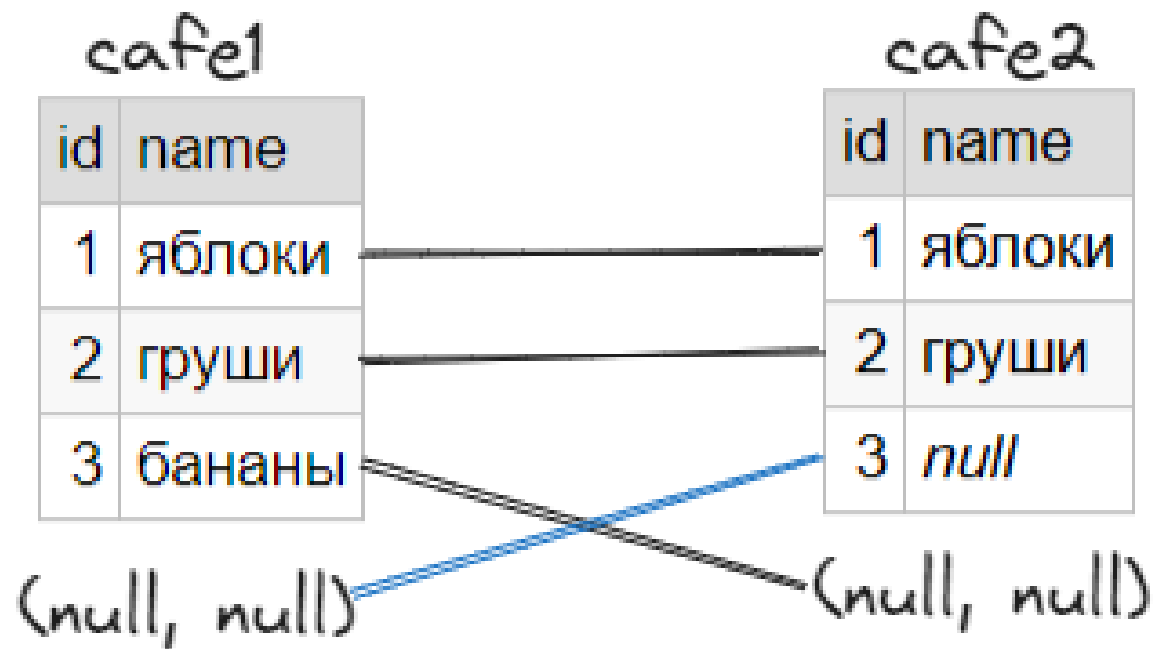
- всегда проверяем уникальность полей из условия соединения
- помним про `null`
- `WHERE <колонка из правой таблицы> = значение` - с осторожностью, возможна потеря данных
 - решение - перенос проверки в условие соединения

Right join - правое соединение

- используется редко
- порядок таблиц важен
- работает как `LEFT JOIN`, но теперь берем все строки СПРАВА
- все особенности - как для `LEFT JOIN`

Full join - полное соединение

- есть условие
- работает как `LEFT JOIN` + `RIGHT JOIN`
- после `LEFT JOIN` выполняется `RIGHT JOIN`, но добавляет данные только в том случае, если их нет после `LEFT JOIN`
- `null` - заполняет столбцы, если не найдено соответствие по условию
- все особенности - как для `LEFT JOIN` и `RIGHT JOIN`



full join
on cafe1.name = cafe2.name

id	name	id	name
1	яблоки	1	яблоки
2	груши	2	груши
3	бананы	<i>null</i>	<i>null</i>
<i>null</i>	<i>null</i>	3	<i>null</i>

Join - резюме

- после соединения доступны все столбцы из соединенных таблиц
- в условии соединения желательно использовать `SARGable` - предикаты
- желательно соединять по уникальному полю (помним про дубликаты)
- соединение по колонке с `null` требует внимания
- `LEFT JOIN` - проверка любого поля из правой таблицы на значение, отличное от `null` приводит к потере строк
- порядок соединения таблиц значения не имеет (определяет сервер), кроме `left & right join`
 - но есть `OPTION (FORCE ORDER)`

Нюансы ДЗ

```
-- поиск по интервалу - between
select s.UnitPrice, *
from Warehouse.StockItems as s
where s.UnitPrice between 10 and 20

-- то же самое, что и
select s.UnitPrice, *
from Warehouse.StockItems as s
where s.UnitPrice >= 10 and s.UnitPrice <= 20
```

Поиск по части наименования

--в названии города встречается burg

```
select * from Application.Cities where CityName like '%burg%'
```

--название города заканчивается на burg

```
select * from Application.Cities where CityName like '%burg'
```

--название города начинается на burg

```
select * from Application.Cities where CityName like 'burg%'
```

isnull vs coalesce - потеря точности

```
-- isnull vs coalesce
;with cte as (
    select 1 as val
    union all
    select null
)
select val
    , isnull(val, 1.4) as [isnull] -- приводит к типу val (потеря точности)
    , coalesce(val, 1.4) as [coalesce] --нет потери точности
from cte
```

Функции работы с датой

```
declare @dt datetime = getdate()

select year(@dt) as [Год]
      , month(@dt) as [Месяц]
      , datepart(quarter, @dt) as [Квартал]
      , datename(month, @dt) as [Месяц]
      , format(@dt, 'MMMM', 'ru-ru') as [Месяц Ru]
      , convert(varchar, @dt, 104) as [Дата]
      , convert(varchar, @dt, 112) as [Дата]
      , datetrunc(month, @dt) as begin_of_month
      , eomonth(@dt) as end_of_month --SQL2022
```

--дата в условии where - удобен формат 'yyyyMMdd': convert(varchar, <дата>, 112)

```
select * from Sales.Orders where OrderDate = '20150527'
```

Постраничная выборка

-- Постраничная выборка

DECLARE

 @pagesize **BIGINT** = 10, -- Размер страницы

 @pagenum **BIGINT** = 3; -- Номер страницы

SELECT

 CityID,

 CityName **AS** City,

 StateProvinceID

FROM Application.Cities

ORDER BY City, CityID

OFFSET (@pagenum - 1) * @pagesize **ROWS FETCH NEXT** @pagesize **ROWS ONLY**;

Top N with ties - топ + строки с данными, попавшими на границу сортировки

- **Top N** + дополнительные строки с одинаковыми значениями сортировки, как у последней строки в ограниченном наборе результатов
- сортировка обязательна
- **WITH TIES** может привести к тому, что будет возвращено больше строк, чем указано в выражении:

```
--товары, входящие в 3ку самых дорогих на складе  
SELECT TOP 3 WITH TIES StockItemID, StockItemName, UnitPrice  
FROM Warehouse.StockItems  
ORDER BY UnitPrice DESC
```

Вопросы для проверки

- Какую часть `SELECT` оптимизатор выполняет первой?
- Какие виды `JOIN` помните?
- Какой тип соединения даст больше всего строк?

Ссылки

- [Бен-Ган, И. Microsoft SQL Server 2012. Создание запросов. - 2012, но актуально](#)

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**