Содержание

Язык SQL

Язык SQL, предназначенный для взаимодействия с базами данных, появился в середине 1970-х гг. (первые публикации датируются 1974 г.) и был разработан в компании IBM в рамках проекта экспериментальной реляционной системы управления базами данных. Исходное название языка – SEQUEL (Structured English Query Language) – только частично отражало суть этого языка. Позже стали называть язык SQL - Structured Query Language (Язык структурированных запросов). Конечно, язык был ориентирован главным образом на удобную и понятную пользователям формулировку запросов к реляционным базам данных. Но, в действительности, он почти с самого начала являлся полным языком баз данных, обеспечивающим, помимо средств формулирования запросов и манипулирования базами данных, следующие возможности:

1) средства определения и манипулирования схемой базы данных;
2) средства определения ограничений целостности и триггеров (о которых будет упомянуто позднее);
3) средства определения представлений баз данных;
4) средства определения структур физического уровня, поддерживающих эффективное выполнение запросов;
5) средства авторизации доступа к отношениям и их полям.

В языке отсутствовали средства явной синхронизации доступа к объектам баз данных со стороны параллельно выполняемых транзакций: предполагалось, что необходимую синхронизацию неявно выполняет система управления базами данных.

В настоящее время язык структурированных запросов реализован во всех коммерческих реляционных системах управления базами данных и почти во всех СУБД, которые изначально основывались не на реляционном подходе. Все компании-производители провозглашают соответствие своей реализации стандарту SQL.

Базовый набор операторов SQL, включающий операторы определения схемы баз данных, выборки и манипулирования данными, средств ограничения целостности баз данных, авторизации доступа к данным, поддержки встраивания SQL в языки программирования и операторы динамического SQL, в коммерческих реализациях устоялся и более или менее соответствует стандарту.

Специфицированы средства определения первичного и внешних ключей отношений и так называемых проверочных ограничений целостности, которые представляют собой подмножество немедленно проверяемых ограничений целостности SQL. Средства определения внешних ключей позволяют легко формулировать требования так называемой ссылочной целостности баз данных.

В настоящее время язык структурированных запросов – это название не просто одного языка, а название целого класса языков, поскольку, несмотря на имеющиеся стандарты, в различных системах управления базами данных реализуются различные диалекты языка структурированных запросов, имеющие, разумеется, одну общую основу.

1. Select – базовый оператор языка структурированных запросов.

Центральное место в языке структурированных запросов SQL занимает оператор Select, с помощью которого реализуется самая востребованная операция при работе с базами данных – запросы.

Оператор Select осуществляет вычисление выражений как реляционной, так и псевдореляционной алгебры. В данном курсе мы рассмотрим реализацию только уже пройденных нами унарных и бинарных операций реляционной алгебры, а также осуществление запросов, с помощью так называемых подзапросов.

Кстати, необходимо заметить, что в случае работы с операциями реляционной алгебры в результирующих отношениях могут появляться дублирующие кортежи. В правилах языка структурированных запросов нет строгого запрещения на присутствие повторяющихся строк в отношениях (в отличие от обычной реляционной алгебры), поэтому исключать дубликаты из результата не обязательно.

Базовая структура оператора Select достаточно проста и включает в себя следующие стандартные обязательные фразы:

```
Select …
From …
Where … ;
```

На месте многоточия в каждой строчке должны стоять отношения, атрибуты и условия конкретной базы данных и задания к ней. В самом общем случае базовая структура Select должна выглядеть следующим образом:

Select *выбрать такие-то атрибуты*
From *из таких-то отношений*
Where *с такими-то условиями выборки кортежей*

Таким образом, выбираем мы атрибуты из схемы отношений (заголовки некоторых столбцов), при этом указывая, из каких отношений (а их, как видно, может быть несколько) мы производим нашу выборку и, наконец, на основании каких условий мы останавливаем свой выбор на тех или иных кортежах.

Важно заметить, что ссылки на атрибуты происходят с помощью их имен.

Таким образом, получается следующий **алгоритм работы** этого базового оператора Select:

1) запоминаются условия выборки кортежей из отношения;
2) проверяется, какие кортежи удовлетворяют указанным свойствам. Такие кортежи запоминаются;
3) на выход выводятся перечисленные в первой строчке базовой структуры оператора Select атрибуты со своими значениями. (Если говорить о табличной форме записи отношения, то выведутся те столбцы таблицы, заголовки которых были перечислены как необходимые атрибуты; разумеется, столбцы выведутся не полностью, в каждом из них останутся только те кортежи, которые удовлетворили названным условиям.)

Пример 1.1.

Пусть дано следующее отношение r1, как фрагмент некой базы данных книжного магазина:

Код книги	Название книги	Автор книги	Цена книги
1258963	Холодные берега	С. Лукьяненко	186,9
1236954	Мобильник	С. Кинг	201,4

Пусть также нам дано следующее выражение с оператором Select:

Select *Название книги, Автор книги*
From r_1
Where *Цена книги > 200;*

Результатом этого оператора будет следующий фрагмент кортежа:

(Мобильник, С. Кинг).

Далее мы рассмотрим множество примеров реализации запросов с использованием базовой структуры Select и ее применение изучим очень подробно.

2. Унарные операции на языке структурированных запросов.

Рассмотрим, как реализуются на языке структурированных запросов с помощью оператора Select уже знакомые нам унарные операции выборки, проекции и переименования.

Важно заметить, что если раньше мы могли работать только с отдельными операциями, то даже один оператор Select в общем случае позволяет определить целое выражение реляционной алгебры, а не какую-то одну операцию.

Итак, перейдем непосредственно к анализу представления унарных операций на языке структурированных запросов.

2.1. Операция выборки.

Операция выборки — унарный оператор, записываемый как $\sigma_{a\theta b}(R)$ или $\sigma_{a\theta v}(R)$, где:

- a, b — имена атрибутов;
- θ — оператор сравнения из множества {<; ≤; =; ≥; >};
- v — константа;
- R — отношение (в оригинале — relation, однако как видно из примера, подразумевается не столько взаимосвязь таблиц, сколько взаимосвязь/соотношение различных фактов в рядах этих таблиц).

Выборка $\sigma_{a\theta b}(R)$ (или $\sigma_{a\theta v}(R)$) выбирает все наборы значений R, для которых функция a θ b (или a θ v) будет истинна. Условие выборки здесь (и во всех остальных реализациях операций) могут быть составными и записывается в виде логического выражения со стандартными связками not (не), and (и), or (или). На атрибуты отношения ссылаемся посредством их имен.

Операция выборки на языке SQL реализуется оператором Select следующего вида:

Select *все атрибуты*
From *имя отношения*
Where *условие выборки*;

Вместо того, чтобы писать «все атрибуты», можно использовать значок «*».

Пример 2.1.1.

Определим схему отношения (выделенные атрибуты образуют ключ): Успеваемость (***№ зачетной книжки*** , ***Семестр*** , ***Код предмета*** , Оценка, Дата). Составим оператор Select следующего вида, реализующий унарную операцию выборки:

Select *
From *Успеваемость*
Where *№ зачетной книжки = 100* **and** *Семестр = 6* ;

В результате выведется успеваемость студента с номером зачетки 100 за 6 семестр.

Пример 2.1.2.

Пусть дано соотношение Персоны.

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Тогда эквивалентные SQL-запросы и результаты выборок будут такими:

$\sigma_{Возраст \geq 34}(Персоны)$

```
SELECT * FROM Персоны WHERE Возраст >= 34;
```

Имя	Возраст	Вес
Harry	34	80
Helena	54	54
Peter	34	80

$\sigma_{Возраст = Вес}(Персоны)$

```
SELECT * FROM Персоны WHERE Возраст = Вес;
```

Имя	Возраст	Вес
Helena	54	54

2.2. Операция проекции.

Операция выборки — унарный оператор, записываемый как $\pi_{a1,\ldots,an}(R)$ где a1,…,an — список полей, подлежащих выборке. Результатом такой выборки будет набор последовательностей значений отношения R, в котором будут присутствовать только поля, перечисленные в списке a1,…,an с естественным уничтожением потенциально возникающих кортежей-дубликатов.

Операция проекции на языке структурированных запросов реализуется даже проще, чем операция выборки. Напомним, что при применении операции проекции выбираются не строки (как при применении операции выборки), а столбцы. Поэтому достаточно перечислить заголовки нужных столбцов (т. е. имена атрибутов), без указания каких-либо посторонних условий. Итого, получаем оператор следующего вида:

Select *список имен атрибутов*
From *имя отношения* ;

После применения этого оператора машина выдаст те столбцы исходной таблицы-отношения, имена которых были указаны в первой строчке этого оператора Select.

Примечательно, что в SQL для полного соответствия операции проекции необходимо указывать ключевое слово **distinct**, поскольку без него могут появиться повторяющиеся строки, что отличается от результата реляционной операции проекции (сам Эдгар Кодд обвинял SQL в неправильной реализации реляционной теории).

В языке SQL существует специальное обозначение для необязательных элементов выражений – квадратные скобки […]. Поэтому в самом общем виде операция проекции будет выглядеть следующим образом:

Select [**distinct**] *список имен атрибутов*
From *имя отношения* ;

Однако если результат применения операции гарантированно не содержит дубликатов или же дубликаты все-таки допустимы, то опцию **distinct** лучше не указывать, чтобы не загромождать запись и из соображений производительности работы оператора.

Пример 2.2.1, иллюстрирующий возможность 100% уверенности в отсутствии дубликатов.

Пусть дана уже известная нам схема отношений: Успеваемость (***№ зачетной книжки*** , ***Семестр*** , ***Код предмета*** , Оценка, Дата).

Пусть дан следующий оператор Select:

Select *№ зачетной книжки, Семестр, Код предмета*
From *Успеваемость* ;

Здесь, как легко видеть, три возвращающихся оператором атрибута образуют ключ отношения. Именно поэтому опция **distinct** становится излишней, ведь дубликатов гарантированно не будет. Это следует из требования, накладываемого на ключи, называемого ограничением уникальности. Если атрибут ключевой, то дубликатов в нем нет.

Пример 2.2.2.

Пусть дано соотношение Персоны.

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Тогда эквивалентные SQL-запросы и результаты проекций будут такими:

$\pi_{Возраст,Вес}(Персоны)$

```
SELECT Возраст, Вес FROM Персоны;
```

Возраст	Вес
34	80
28	64
29	70
54	54
34	80

```
SELECT DISTINCT Возраст, Вес FROM Персоны;
```

Возраст	Вес
34	80
28	64
29	70
54	54

2.3. Операция переименования.

Операция переименования атрибутов на языке структурированных запросов осуществляется довольно просто. А именно воплощается в действительность следующим алгоритмом:

1) в списке имен атрибутов Select перечисляются атрибуты для переименования;
2) к каждому указанному атрибуту добавляется специальное ключевое слово as;
3) после as указывается то имя, на которое необходимо поменять имя исходное.

Оператор, соответствующий операции переименования атрибутов, будет выглядеть так:

Select *имя атрибута 1* **as** *новое имя атрибута 1, …*
From *имя отношения;*

Пример 2.3.1. Схема: Успеваемость(***№ зачетной книжки***, ***Семестр***, ***Код предмета***, Оценка, Дата);

Пусть необходимо поменять имена некоторых атрибутов, а именно вместо «№ зачетной книжки» должно стоять «№ зачетки» и вместо «Оценка» – «Балл». Запишем, как будет выглядеть оператор Select, реализующий эту операцию переименования:

Select *№ зачетной книжки as № зачетки, Семестр, Код предмета, Оценка as Балл, Дата*
From *Успеваемость;*

Результатом этого оператора будет новая схема отношения, отличающаяся от исходной схемы отношения «Успеваемость» именами двух атрибутов.

3. Бинарные операции на языке структурированных запросов.

Рассмотрим реализацию на SQL бинарных операций, а именно – операций объединения, пересечения, разности, декартового произведения, естественного соединения, внутреннего соединения и левого, правого, полного внешнего соединения.

3.1. Операция объединения.

Для того чтобы реализовать операцию объединения двух отношений приходится использовать одновременно два оператора Select, каждый из которых соответствует какому-то одному из исходных отношений-операндов. И к этим двум базовым операторам Select необходимо применить специальную операцию **Union** . Запишем, как операция объединения будет выглядеть в общем виде с использованием семантики языка структурированных запросов:

Select *список имен атрибутов отношения 1*
From *имя отношения 1*
Union [All]
Select *список имен атрибутов отношения 2*
From *имя отношения 2;*

Важно заметить, что списки имен атрибутов двух объединяемых отношений должны ссылаться на атрибуты совместимых типов и быть перечислены в согласованном порядке, иначе, запрос не сможет быть выполнен, и будет выдано сообщение об ошибке. Но, сами имена атрибутов в этих отношениях могут быть различными. В таком случае результирующему отношению приписываются имена атрибутов, указанные в первом Select.

Пример 3.1.1.

Использование **Union** предполагает автоматическое исключение из результирующего отношения всех дубликатов кортежей. Если вам нужно, чтобы все повторяющиеся строки в конечном результате сохранились, вместо операции Union следует применять её модификацию – **Union All**, в этом случае из результирующего отношения дубликаты кортежей удаляться не будут.

Пример 3.1.2.

Пусть даны соотношения Персоны и Персонажи.

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Имя	Возраст	Вес
Daffy	24	19
George	29	70
Donald	25	23
Scrooge	81	27

Тогда эквивалентные SQL-запросы и результаты объединения будут такими:

Персоны U Персонажи

```
SELECT Имя, Возраст, Вес FROM Персоны
UNION
SELECT Имя, Возраст, Вес FROM Персонажи;
```

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80
Daffy	24	19
Donald	25	23
Scrooge	81	27

3.2. Операция пересечения.

Операция пересечения и операция разности двух отношений на языке структурированных запросов реализуются похожим образом. Рассмотрим способ реализации операции пересечения с использованием **ключей**.

Этот способ предполагает участие двух конструкций Select, но они не равноправны (как в операции объединения), одна из них является как бы «под-конструкцией», «под-циклом». Такой оператор обычно называют **подзапросом**.

Даны две схемы отношений (R1 и R2), приблизительно определенные следующим образом:

R1 (ключ, …)
R2 (ключ, …)

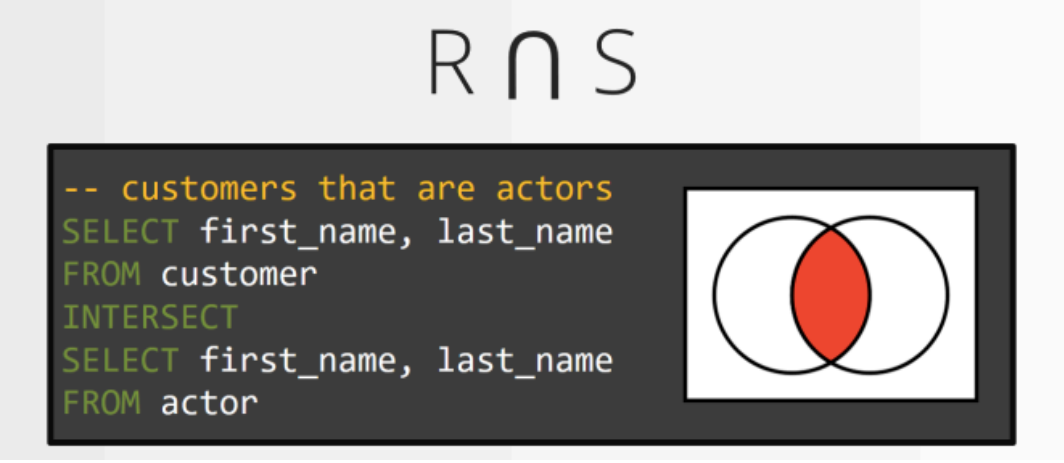
Воспользуемся также при записи этой операции специальной опцией **in** , что буквально означает «в» или (как в данном конкретном случае) «содержится в».

Операция пересечения двух отношений с помощью SQL запишется следующим образом:

Select *
From *R1*
Where *ключ* **in**
(**Select** *ключ* **From** *R2*);

Таким образом, мы видим, что подзапросом в данном случае будет являться оператор в круглых скобках. Этот подзапрос в нашем случае возвращает список значений ключа отношения R2. И, как следует из нашей записи операторов, из анализа условия выборки, в результирующее отношение попадут только те кортежи отношения R1, ключ которых содержится в списке ключей отношения R2. То есть, в итоговом отношении, если вспомнить определение пересечения двух отношений, останутся лишь те кортежи, которые принадлежат обоим отношениям.

Существуют и иные способы записи операции пересечения, например, через INTERSECT:



3.3. Операция разности.

Как уже было сказано ранее, унарная операция разности двух отношений реализуется аналогично операции пересечения. Здесь также, кроме главного запроса с оператором Select, используется второй, вспомогательный запрос – так называемый подзапрос.

Но в отличие от воплощения в жизнь предыдущей операции, при реализации операции разности необходимо использовать другое ключевое слово, а именно **not in** , что в дословном переводе означает «не в» или (как уместно перевести в нашем рассматриваемом случае) – «не содержится в».

Даны две схемы отношений (R1 и R2), приблизительно определенные следующим образом:

R1 (ключ, …) и R2 (ключ, …)

Как видим, среди атрибутов этих отношений снова заданы ключевые атрибуты.

Вид для представления в языке структурированных запросов операции разности:

Select *
From *R1*
Where *ключ* **not in**
(**Select** *ключ* **From** *R2*);

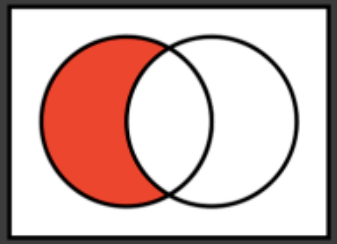
В результирующее отношение выбираются только те кортежи отношения R1, ключ которых не содержится в списке ключей отношения R2. Если рассматривать запись буквально, то действительно получается, что из отношения R1 «вычли» отношение R2.

Два случая применения «метода ключей», которые мы рассмотрели, являются самыми распространенными. Остальные бинарные операции используют другие конструкции.

Существуют и иные способы записи операции разности (Difference), например, через EXCEPT:

$$R - S$$

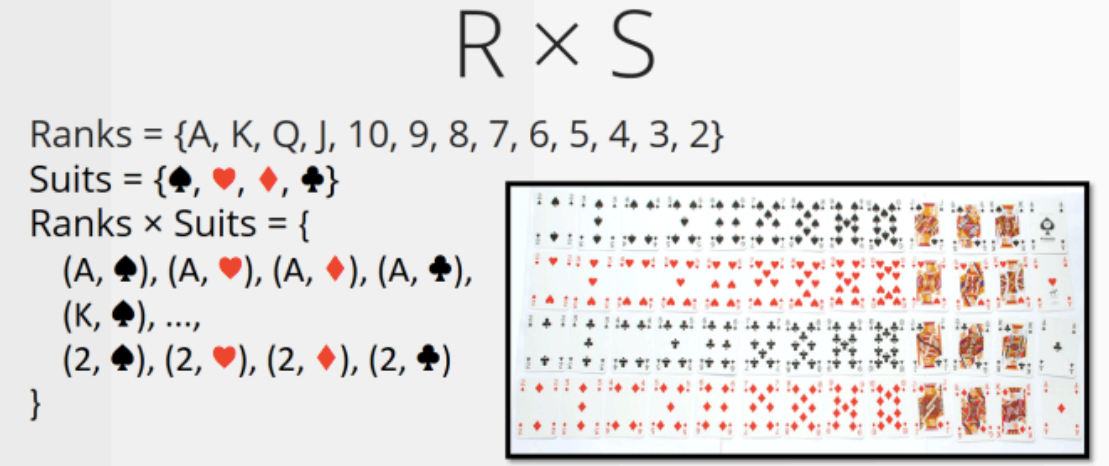
```
-- customers that are not staff
SELECT first_name, last_name
FROM customer
EXCEPT
SELECT first_name, last_name
FROM staff
```



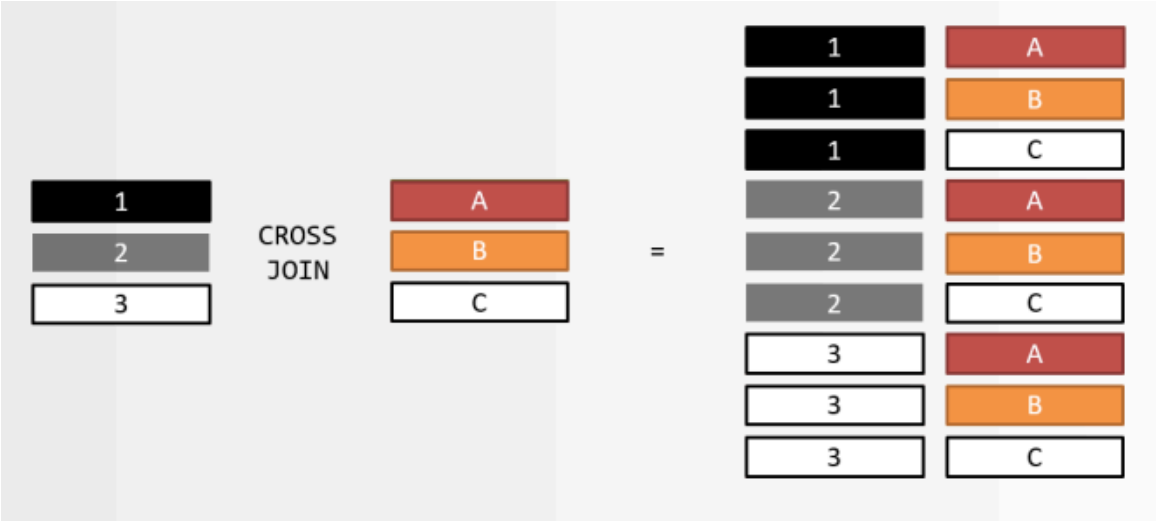
3.4. Операция декартова произведения.

Как мы помним из предыдущих лекций, декартово произведение (Cartesian Product) двух отношений-операндов составляется как набор всех возможных пар именованных значений кортежей на атрибутах.

На SQL операция декартова произведения реализовывается при помощи перекрестного соединения, обозначаемого ключевым словом **cross join**, что буквально и переводится «перекрестное объединение» или «перекрестное соединение».



или ТАК



Оператор Select в конструкции, представляющей декартово произведения имеет вид:

Select *
From *R1* **cross join** *R2*

Здесь R1 и R2 – имена исходных отношений-операндов. Опция **cross join** обеспечивает, что в результирующее отношение запишутся все атрибуты (все, потому что в первой строчке оператора поставлен значок «*»), соответствующие всем парам кортежей отношений R1 и R2.

Очень важно помнить одну особенность воплощения в жизнь операции декартова произведения. Эта особенность является следствием определения бинарной операции декартова произведения. Напомним его:

$$r_4(S_4) = r_1(S_1) \times r_2(S_2) = \{t\,(S_1 \cup S_2) \mid t\,[S_1] \in r_1\ \&\ t\,(S_2) \in r_2\},\ S_1 \cap S_2 = \varnothing;$$

Как видно из приведенного определения, пары кортежей образуются при обязательно непересекающихся схемах отношений. Поэтому и при работе на языке структурированных запросов SQL непременно оговаривается, что исходные отношения-операнды не должны иметь совпадающих имен атрибутов. Но если эти отношения все же имеют одинаковые имена, сложившуюся ситуацию можно легко разрешить с помощью операции переименования атрибутов, т. е. в подобных случаях необходимо просто использовать опцию **as** , о которой упоминалось ранее.

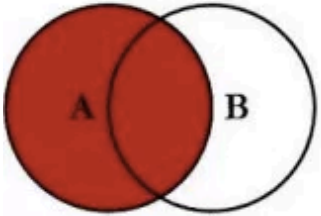
Пример, в котором нужно найти декартово произведение двух отношений, имеющих некоторые имена своих атрибутов совпадающими. Даны два отношения: R1(A, B) и R2(B, C);

Видим, что атрибуты R1.B и R2.B имеют одинаковые имена. С учетом этого оператор Select, реализующий на языке структурированных запросов эту операцию декартова произведения, будет выглядеть следующим образом:

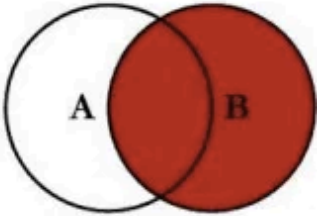
Select *A, R1.B* **as** *B1, R2.B* **as** *B2, C*
From *R1* **cross join** *R2*;

Таким образом, с использованием опции переименования **as**, у машины не возникнет «вопросов», по поводу совпадающих имен двух исходных отношений-операндов.

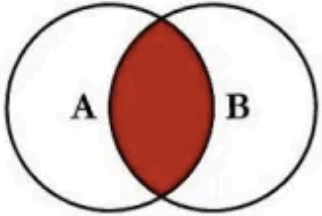
SQL JOINS



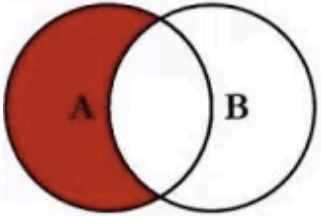

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



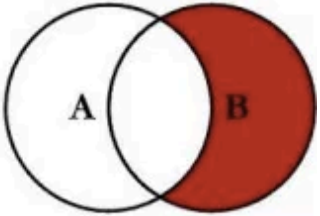

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



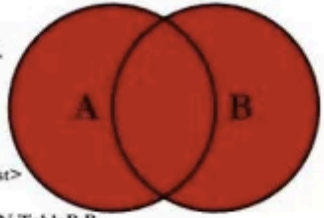

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



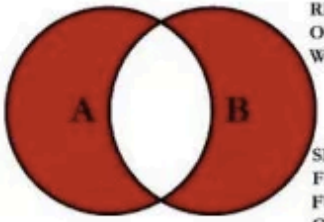

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

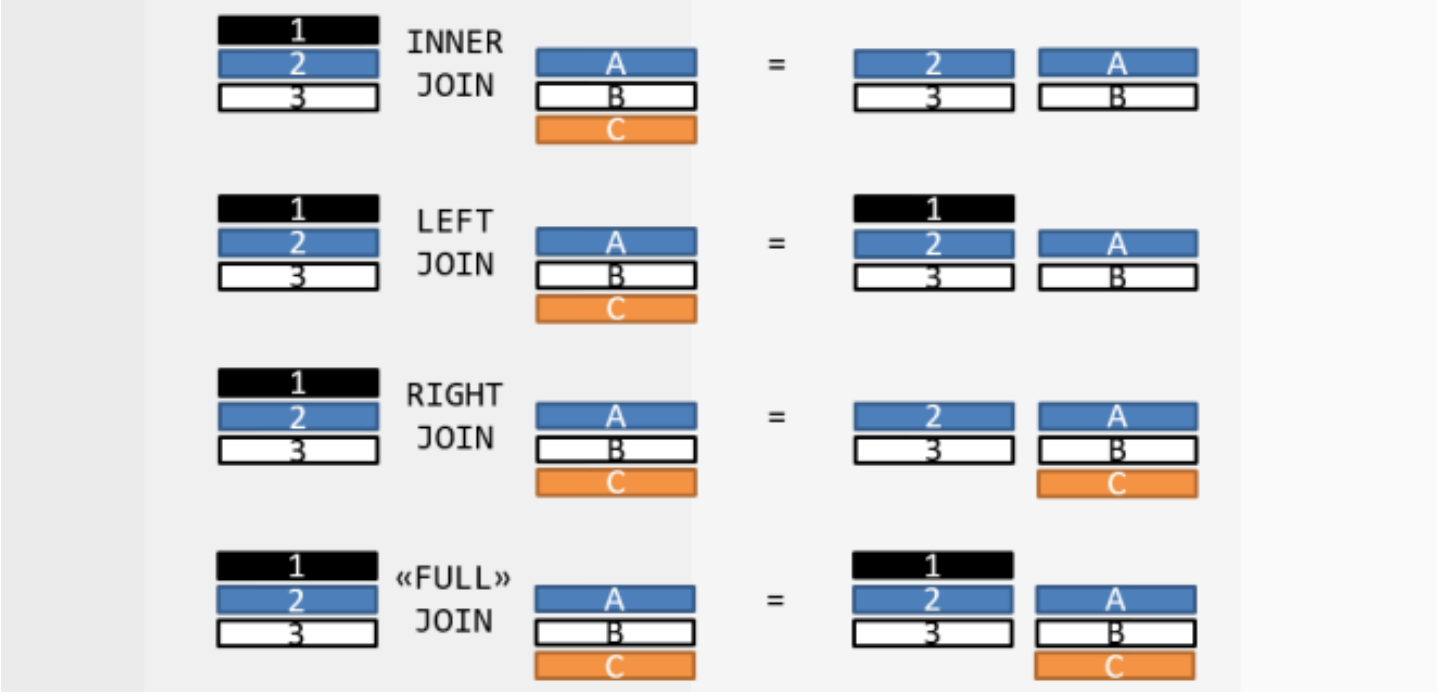



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
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
```

или ТАК


1
2
3
INNER
JOIN
A
B
C
=
2
3
A
B
1
2
3
LEFT
JOIN
A
B
C
=
1
2
3
A
B
1
2
3
RIGHT
JOIN
A
B
C
=
2
3
A
B
C
1
2
3
«FULL»
JOIN
A
B
C
=
1
2
3
A
B
C

3.5. Операция внутреннего соединения.

Операцию внутреннего соединения рассмотрим раньше операции естественного соединения, так как последняя является частным случаем первой.

Итак, для начала вспомним определение операции внутреннего соединения, которое мы проходили раньше:

$$r_1(S_1) \times_P r_2(S_2) = \sigma_{<P>}(r_1 \times r_2),\ S_1 \cap S_2 = \varnothing.$$

Для нас в этом определении особенно важно то, что рассматриваемые схемы отношений-операндов S1 и S2 не должны пересекаться.

Для реализации операции внутреннего соединения в языке структурированных запросов существует специальная опция **inner join**, которая и переводится с английского буквально «внутреннее объединение» или «внутреннее соединение».

Оператор Select для осуществления операции внутреннего соединения будет выглядеть так:

```
Select *
From R1 inner join R2;
```

Здесь, как и раньше, R1 и R2 – имена исходных отношений-операндов. При реализации этой операции нельзя допускать пересечения схем отношений-операндов.

3.6. Операция естественного соединения.

Как мы уже говорили, операция естественного соединения является частным случаем операции внутреннего соединения. Почему? Да потому что при действии естественного соединения кортежи исходных отношений-операндов соединяются по особому условию. А именно по условию равенства кортежей на пересечении отношений-операндов, тогда как при действии операции внутреннего соединения такой ситуации допускать было бы нельзя.

Так как рассматриваемая нами операция естественного соединения является частным случаем операции внутреннего соединения, для ее реализации используется та же опция, что и для предыдущей рассмотренной операции, т. е. опция **inner join**. Но, поскольку при составлении оператора Select для операции естественного соединения необходимо еще учесть условие равенства кортежей исходных отношений-операндов на пересечении их схем, то дополнительно к означенной опции применяется ключевое слово **on**. На английском, это буквально означает «на», а применительно к нашему смыслу, можно перевести как «при условии».

Общий вид оператора Select для выполнения операции естественного соединения такой:

Select *
From *имя отношения 1* **inner join** *имя отношения 2*
on *условие равенства кортежей*;

Рассмотрим пример. Пусть даны два отношения:

R1(A, B, C) и R2(B, C, D);

Операцию естественного соединения этих отношений можно реализовать с помощью следующего оператора:

Select *A* , *R1.B* , *R1.C* , *D*
From *R1* **inner join** *R2*
on *R1.B* = *R2.B* and *R1.C* = *R2.C*

В итоге этой операции в результат выведутся атрибуты, указанные в первой строке оператора Select, соответствующие кортежам, равным на указанном пересечении.

Заметим, что здесь мы обращаемся к общим атрибутам B и C не просто по именам. Это необходимо делать не по той причине, что и в случае реализации операции декартова произведения, а потому, что в противном случае будет не ясно, к какому отношению они относятся.

Интересно, что использованная формулировка условия соединения (R1.B = R2.B and R1.C = R2.C) предполагает, что общие атрибуты соединяемых отношений Null-значений не допускают. Это изначально встроено в систему языка структурированных запросов.

3.7. Операция левого внешнего соединения.

Выражение на языке структурированных запросов SQL операции левого внешнего соединения получается из реализации операции естественного соединения заменой ключевого слова **inner** на ключевое слово **left outer**.

Таким образом, на языке структурированных запросов эта операция запишется так:

Select *
From *имя отношения* 1 **left outer join** *имя отношения* 2
on *условие равенства кортежей*;

3.8. Операция правого внешнего соединения.

Выражение для операции правого внешнего соединения на языке структурированных запросов получается из осуществления операции естественного соединения заменой ключевого слова **inner** на ключевое слово **right outer**.

На языке структурированных запросов SQL операция правого внешнего соединения запишется так:

Select *
From *имя отношения* 1 **right outer join** *имя отношения* 2
on *условие равенства кортежей*;

3.9. Операция полного внешнего соединения.

Выражение на языке структурированных запросов операции полного внешнего соединения получается, как и в двух предыдущих случаях, из выражения для операции естественного соединения путем замены ключевого слова **inner** на ключевое слово **full outer**.

Таким образом, на языке структурированных запросов эта операция запишется так:

Select *
From *имя отношения 1* **full outer join** *имя отношения 2*
on *условие равенства кортежей*;

Очень удобно, что в семантику языка структурированных запросов SQL изначально встроены эти опции, ведь иначе каждому программисту приходилось бы выводить их самостоятельно и вводить в каждую новую базу данных.

4. Использование подзапросов.

Понятие «подзапрос» в языке структурированных запросов является понятием базовым и довольно широко применимым (иногда, кстати, их еще называют SQL-запросами. Практика программирования и работы с базами данных показывает, что составление системы подзапросов для решения различных сопутствующих задач является эффективным приемом работы со структурированной информацией. Рассмотрим пример для лучшего понимания действий с подзапросами, их составлением и использованием.

Пусть имеется следующий фрагмент некой базы данных, которая вполне может использоваться в каком-либо учебном заведении:

Предметы(***Код предмета***, Имя предмета);
Студенты(***№ зачетной книжки***, Фамилия, Имя, Отчество);
Сессия(***Код предмета***, ***№ зачетной книжки***, Оценка);

Сформулируем SQL-запрос, возвращающий ведомость с указанием номера зачетной книжки, фамилии и инициалов студента и оценки для предмета с наименованием «Базы данных». Такую информацию в университетах необходимо получать всегда и своевременно, поэтому приведенный далее запрос является едва ли не самой востребованной единицей программирования с использованием таких баз данных.

Для удобства работы, дополнительно предположим, что атрибуты «Фамилия», «Имя» и «Отчество» не допускают Null-значений и не являются пустыми. Это требование вполне объясни-

мо и закономерно, ведь в базу данных любого учебного заведения первыми из данных на нового ученика вводятся именно данные о его фамилии, имени и отчестве. И само собой разумеется, что не может быть записи в подобной базе данных, в которой присутствуют данные на ученика, но при этом неизвестно его имя.

Заметим, что атрибут «Имя предмета» схемы отношения «Предметы» является ключом, поэтому, как следует из определения (подробнее об этом будет сказано дальше), все наименования предметов являются уникальными. Это тоже понятно и без пояснения представления ключа, ведь все преподающиеся в учебном заведении предметы должны иметь и имеют различные имена.

Теперь, прежде чем мы приступим к составлению текста самого оператора, рассмотрим две функции, которые нам пригодятся в дальнейшем.

Во-первых, нам будет полезна функция **Trim**, записывается Trim (“строка“), т. е. аргументом этой функции является строка. Что делает эта функция? Они возвращает сам аргумент без пробелов, стоящих в начале и в конце этой строки, т. е., эту функцию применяют, например, в случаях: Trim (“Богучарников “) или Trim (“ Максименко“), когда после или до аргумента стоят по несколько лишних пробелов.

А во-вторых, необходимо также рассмотреть функцию **Left**, которая записывается Left(строка, число), т.е. функцию от уже двух аргументов, одним из которых является, как и раньше, строка. Второй ее аргумент – число, оно показывает, сколько символов из левой части строки следует вывести в результат.

Например, результатом операции:

Left (“Михаил”, 1”) + “.“ + Left(“Зиновьевич“, 1) + “.“

будут инициалы «М.З.». Именно для выведения инициалов студентов мы и будем использовать эту функцию в нашем запросе.

Итак, приступим к составлению искомого запроса. Но, сначала составим небольшой вспомогательный запрос, который потом используем в основном, главном запросе:

Select № *зачетной книжки, Оценка*
From *Сессия*
Where *Код предмета* = (**Select** *Код предмета*
From *Предметы*
Where *Имя предмета* = “*«Базы данных»*“)
as “*Оценки «Базы данных»*“;

Применение здесь опции **as** означает, что мы присвоили этому запросу псевдоним “*Оценки «Базы данных»*“. Сделали мы это для удобства дальнейшей работы с этим запросом.

Далее, в этом запросе подзапрос:

Select *Код предмета*
From *Предметы*
Where *Имя предмета* = “*«Базы данных»*“;

позволяет выделить из отношения «Сессия» те кортежи, которые относятся к рассматриваемому предмету, т. е. к базам данных.

Интересно, что этот внутренний подзапрос может возвращать не более одного значения, так как атрибут «Имя предмета» является ключом отношения «Предметы», т. е. все его значения уникальны.

А весь запрос “Оценки «Базы данных»“ позволяет выделить из отношения «Сессия» данные о тех студентах (их номера зачетных книжек и оценки), которые удовлетворяют условию, оговоренному в подзапросе, т. е. информацию о предмете под названием «База данных».

Теперь составим основной запрос, используя уже полученные результаты.

Select *Студенты.* № *зачетной книжки*,
Trim(*Фамилия*) + « » + **Left** (*Имя*, 1) + «.» + **Left** (*Отчество*, 1) + «.» **as** *ФИО*, *Оценки «Базы данных».Оценка*
From *Студенты* **inner join**
(
Select № *зачетной книжки, Оценка*
From *Сессия*
Where *Код предмета* = (**Select** *Код предмета*
From *Предметы*
Where *Имя предмета* = ”*«Базы данных»*”)
) as *”Оценки «Базы данных»“.*
on *Студенты.*№ *зачетной книжки* = *Оценки «Базы данных».*№ *зачетной книжки.*

Итак, сначала мы перечисляем атрибуты, которые будет необходимо вывести, после окончания работы запроса. Необходимо упомянуть, что атрибут «№ зачетной книжки» из отношения Студенты, оттуда же – атрибуты «Фамилия», «Имя» и «Отчество». Правда, два последних атрибута выводим не полностью, а только первые буквы. Также мы упоминаем атрибут «Оценка» из запроса *Оценки «Базы данных»*, которое ввели раньше.

Выбираем мы все эти атрибуты из внутреннего соединения отношения «Студенты» и запроса *«Оценки „Базы данных“* . Это внутреннее соединение, как мы можем видеть, берется нами по условию равенства номеров зачетной книжки. В результате этой операции внутреннего соединения, к отношению «Студенты» добавляются оценки.

Надо заметить, что так как атрибуты «Фамилия», «Имя» и «Отчество» по условию не допускают Null-значений и не являются пустыми, то формула вычисления, возвращающая атрибут «ФИО» (**Trim** (Фамилия) + « » + **Left** (Имя, 1) + «.» + **Left** (Отчество, 1) + «.» **as** ФИО), соответственно не требует дополнительных проверок, упрощается.