Лекция 13. Запросы. Языки запросов, их выразительная сила. Язык SQL, язык QBE. Оптимизация запросов.

13.1 Понятие языка запросов

> **Запрос** – обращение к системе баз данных с целью либо получения некоторых данных, либо изменения данных в базе.

Вес запросы должны формулироваться на некотором **языке запросов**. Примером такого языка является реляционная алгебра, т.к. каждое выражение реляционной алгебры возвращает в результате некоторое отношение, а операции обновления данных могут быть выражены с помощью реляционных операций.

Пример 1.
Даны отношения «ПОСТАВЩИКИ» (п_номер, п_адрес, п_название) и «ПОСТАВКИ» (п_номер, код_пст, код_товара, дата, кол-во).
Запрос «выбрать все названия поставщиков, участвовавших в поставках 10.02.1999 г.» в реляционной алгебре можно записать так:

$$r = \pi_{п\_название}(\sigma_{дата='10.02.99'}("ПОСТАВЩИКИ" \triangleright\triangleleft "ПОСТАВКИ"))$$

Как видно, любое выражение реляционной алгебры указывает последовательность действий, процедуру, которую нужно выполнить для получения результата, т.е. «как» построить искомое отношение.
В отличие от реляционной алгебры, существуют другие языки запросов, которые позволяют получать искомый результат, но на непроцедурном уровне, т.е. запрос на таком языке запросов представляет собой описание того, «что» нужно получить, а не «как» это сделать.
К таким языкам запросам относятся реляционное исчисление, основанное на кортежах, и реляционное исчисление, основанное на доменах. Реляционное исчисление представляет собой систему обозначений для определения необходимого отношения на базе уже имеющихся.

Язык **SQL** – **Structured Query Language** – язык запросов, основанный на реляционном исчислении кортежей.

Пример 2.
Запишем запрос из примера 1 на языке SQL:

```
SELECT ПОСТАВЩИКИ.п_название
FROM ПОСТАВЩИКИ, ПОСТАВКИ
WHERE ПОСТАВКИ.дата=10.02.99
     AND ПОСТАВЩИКИ.п_номер = ПОСТАВКИ.п_номер;
```

Этот запрос можно сформулировать теперь так:

Получить значения атрибута «п_название» для поставщиков с таким значением атрибута «п_номер», у которых существует такая поставка в отношении «ПОСТАВКИ», где значение атрибута «дата» равно 10.02.99, а код поставщика совпадает с атрибутом «п_номер»

Как видно, формулировка запроса в реляционном исчислении носит **описательный характер**, а алгебраическая формулировка – **предписывающий характер**.

Рассмотрим реляционные исчисления.

13.2 Реляционное исчисление на кортежах

Типичное выражение реляционного исчисления на кортежах имеет вид

$\{t(R): f(t)\}$,
где t – переменная кортежа на схеме R, f – некоторая формула реляционного исчисления.

Формулы строятся из **атомов**. **Атомы** бывают трех видов:
1. $r(t)$, где r – имя отношения, t- переменная кортежа. Атом эквивалентен утверждению "$t \in r$"
2. $t1(A)\ \theta\ t2(B)$, где r – имя отношения, A, B – атрибуты, θ - знак сравнения.
3. $t(A)\ \theta\ a$, где r – имя отношения, A – атрибут, $a \in dom\ (A)$, θ - знак сравнения.

Формулы строятся из атомов, с использованием логических связок AND, OR, NOT, кванторов EXISTS и FORALL.
1. Каждый атом – формула.
2. Если f - формула, то NOT f - тоже формула
3. Если f, g - формулы, то f AND g, f OR g - тоже формулы
4. Если f - формула, то EXISTS $t(R): f(t)$ и FORALL $t(R): f(t)$ - тоже формулы
5. (f) – формула

Если все элементы кортежа t определены на конечных доменах, то формула $f(t)$ называется **безопасной**.
Это замечание позволяет выполнять запросы типа:
$\{t(R):$ NOT $f(t)\}$, с гарантией, что отношение NOT $f(t)$ будет конечным.

Пример 3.
Пусть домен атрибута A – все множество действительных чисел R (бесконечное). Тогда если одноатрибутное отношение r(A)={1, 2}, то {f(t): NOT r(A)} = R \ r(A) – бесконечное отношение.

Пример 4.
Сформулируем запрос из примера 2 на языке исчисления кортежей.
Пусть r1 – отношение «ПОСТАВЩИКИ», r2 – отношение "ПОСТАВКИ".
Получим выражение для соединения отношений по атрибуту "n_номер":

$$r1 \triangleright\!\triangleleft r2 = q = q(R1 \cup R2) =$$
$$= \{z(R1 \cup R2) : \exists x(R1), \exists y(R2) : r1(x)\ AND\ r2(y)\ AND\ (z(R1) = x(R1))\ AND\ (z(R2) = y(R2))\}$$

Теперь запишем выражение для получения отношения со значением атрибута «дата» равным 10.02.99:

$$\sigma_{дата='10.02.99'}(q) = s = \{w(R1 \cup R2) : q(w)\ \text{AND}\ (w(дата) = '10.02.99')\}$$

«Спроецируем» отношение s на атрибут n_название:

$$\pi_{n\_название}(s) = \{t(n\_название) : \exists w(R1 \cup R2) : s(w) \text{ AND } w(n\_название) = t(n\_название)\}$$

Объединив все записи в одну, получим:

$$\{t(n\_название) : \exists w(R1 \cup R2) : (\exists x(R1), \exists y(R2) : r1(x)\ AND\ r2(y)\ AND\ (z(R1) = x(R1))\ AND\ (z(R2) = y(R2))$$
$$\text{AND } (w(data) = '10.02.99') \text{ AND } (w(n\_номер) = t(n\_номер))\}$$

Кодд показал (в 1971 году), что существует алгоритм преобразования безопасной формулы исчисления на кортежах в эквивалентное выражение реляционной алгебры. Позже (в 1982 году) Ульманом была доказана теорема:

Если E – выражение реляционной алгебры (использующее операции из *O* и *операцию дополнения (dom \ r)*), то существует эквивалентная ему безопасная формула реляционного исчисления на кортежах. Обратное утверждение также верно.

Значит, любое выражение реляционной алгебры можно заменить эквивалентной ему формулой реляционного исчисления.

Факт эквивалентности алгебры и исчисления позволяет определить понятие **выразительной силы** языка запросов к реляционным данным.

> Язык запросов обладает **не меньшей выразительной силой**, чем реляционная алгебра, если любое выражение реляционной алгебры можно заменить выражением на этом языке запросов.

13.3 Реляционное исчисление на доменах

Типичное выражение реляционного исчисления на доменах имеет вид:

$\{r(x\_1,x\_2,\ldots,x\_n): f(x\_1,x\_2,\ldots,x\_n)\}$,

где r –реляционное отношение со схемой $R(X\_1,X\_2,\ldots,X\_m)$, $x\_i \in dom\ (X\_i)$ – переменная или константа из домена, f - формула, использующая переменные $x\_1,x\_2,\ldots,x\_n$.

Атомами являются:

1. $r(x\_1,x\_2,\ldots,x\_n)$, если r –реляционное отношение степени n, $x\_i \in dom\ (X\_i)$ – переменная или константа из домена.
2. $x\ \theta\ y$, где x, y - константы или переменные на соответствующих доменах.

Формулы строятся аналогично формулам исчисления на кортежах.

Пример 5.
Сформулируем запрос из примера 2 на языке исчисления доменов:

$$\{x : \exists \mathbf{z} \exists y \exists a \exists b \exists c : "ПОСТАВЩИКИ"(\mathbf{z}yx) \text{ AND } "ПОСТАВКИ"(\mathbf{z}ab('10.02.99')c)\}$$

В этой формуле: z – " n_номер ", x –" n_название", y –" n_адрес ", a – "код_пст", b – "код_товара", c – "кол-во", т.е. переменные доменов, которым соответствуют имена атрибутов обоих отношений.

Также Ульманом было доказано, что реляционное исчисление на доменах обладает не меньшей выразительной силой, чем реляционное исчисление на кортежах или реляционная алгебра.

Т.е. реляционное исчисление кортежей эквивалентно реляционной алгебре и эквивалентно реляционному исчислению доменов.

При создании нового языка запросов к реляционным данным следует доказать, что его выразительная сила не меньше, чем у реляционной алгебры.
На практике примером языка запросов, основанного на реляционной алгебре, является язык QUEL; язык QBE – Query By Example – запрос по образцу – язык, основанный на исчислении доменов.

Реальные языки запросов, такие как SQL, являются **более чем полными**, т.к. помимо необходимых реляционных операций, в них добавляют арифметические операции +, -, *, /, агрегатные функции типа SUM, COUNT, MAX, MIN, AVG, применимые к одному или нескольким полям таблицы, а также операторы объявления данных.

13.4 Оптимизация запросов

Рассмотрим еще раз запрос из примера 1.
«Получить названия поставщиков, которые участвовали в поставках 10.02.1999 года»
Алгебраическая запись этого запроса такова:

$$r = \pi_{n\_название}(\sigma_{дата='10.02.99'}("ПОСТАВЩИКИ" \triangleright\triangleleft "ПОСТАВКИ"))$$

Предположим, что в реальной базе данных хранится информация о 100 поставщиках, 10 000 поставках, и пусть 10 февраля 1999 года было выполнено всего 10 поставок.

Выполняем алгебраическое выражение “в лоб”:
1) Сначала нужно выполнить соединение двух отношений “ПОСТАВЩИКИ” и “ПОСТАВКИ”. Это значит, что будет считано 10 000 записей о поставках и для каждой из 10 000 записей будет проведен поиск среди 100 поставщиков. Результат этой операции – отношение из 10 000 записей. Предполагаем, что оно будет записано на диск, т.к. оперативной памяти не хватает.
2) Из 10 000 полученных в результате соединения кортежей выбирается 10 кортежей с нужной датой. Они помещаются в оперативную память. Результат операции - отношение из 10 кортежей.
3) Выбирается нужный столбец. Результат – 10 кортежей, состоящих только из названий поставщиков, т.е. итоговый результат алгебраического выражения.

При такой схеме выполнения запроса было проведено по крайней мере 10000 * 100 операций обращения к диску (операций чтения кортежа).

Слегка изменим процедуру выполнения:
1) Сначала выберем из 10 000 кортежей о поставках те, которые сделаны 10.02.1999. Результат – 10 кортежей, помещающихся в оперативную память.
2) Соединяем результат 1-го шага с отношением «ПОСТАВЩИКИ». Результат – 10 кортежей.
3) Выбираем столбец названий.

Итого: 10 000 + 100 операций чтения кортежа.

Очевидно, что второй способ предпочтителен.
Оптимизация запросов позволяет выполнять запросы быстрее и эффективнее.

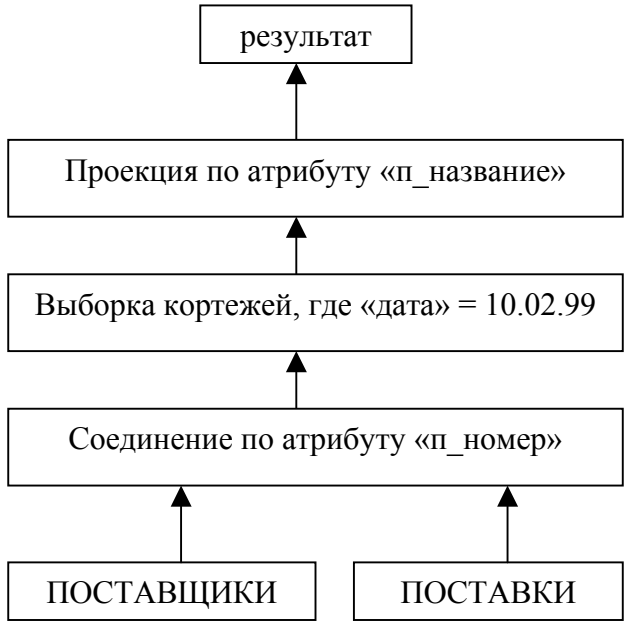
Процесс оптимизации состоит из четырех этапов:

1. Преобразование запроса во внутреннюю форму (т.е. в выражение реляционной алгебры или реляционного исчисления).
2. Оптимизация реляционного выражения.
3. Выбор потенциальных низкоуровневых процедур для выполнения реляционных операций
4. Генерация планов вычисления запросов и выбор плана с наименьшей стоимостью.

Рассмотрим подробно каждый этап процесса оптимизации:

Этап 1.
Преобразование правильного и допустимого в данной системе запроса в эквивалентное выражение реляционной алгебры всегда возможно в силу доказанных теорем о выразительной силе реляционных исчислений и реляционной алгебры.
На данном этапе строится так называемое **дерево запроса.**

Пример 6.



Этап 2.
Пользуясь известными свойствами реляционных операций, оптимизатор запросов формирует оптимальные реляционные выражения

Правила преобразования выражений

1. $\sigma_{C1}(\sigma_{C2}(r)) = \sigma_{C1\,\mathrm{AND}\,C2}(r)$
2. Если $A \subset B \subseteq R$, R - реляционная схема, то $\pi_B(\pi_A(r)) = \pi_B(r)$
3. Если $A \in X$, $X \subseteq R$, R - реляционная схема, то $\pi_X(\sigma_{A=a}(r)) = \sigma_{A=a}(\pi_X(r))$
4. $\sigma_{C1}(r\ \gamma\ \mathrm{s}) = \sigma_{C1}(r)\ \gamma\ \sigma_{C1}(s)$, где $\gamma \in \{\cup, \cap, \setminus\}$
5. «Ранняя выборка»
   Если логическое условие C1 применимо к r и логическое условие C2 применимо к s, то
   $\sigma_{C1\,\mathrm{AND}\,C2}(r \triangleright\triangleleft s) = \sigma_{C1}(r) \triangleright\triangleleft \sigma_{C2}(s)$
6. $\pi_X(r\ \gamma\ \mathrm{s}) = \pi_X(r)\ \gamma\ \pi_X(s)$, где $\gamma \in \{\cup, \cap, \setminus\}$
7. «Ранняя проекция»
   Применяется, если проекция осуществляется по атрибутам, включающим в себя все атрибуты, по которым выполняется соединение.

Множество, использованное в проекции X равно объединению множеств атрибутов в s-части проекции и r-части проекции и включает атрибуты, по которым выполняется соединение.

$\pi_X(r \triangleright\triangleleft s) = \pi_{X \cap R}(r) \triangleright\triangleleft \pi_{X \cap S}(s)$, если $X \supset R \cap S$

8. Операции объединения, пересечения, натурального соединения являются ассоциативными, т.е. $r\gamma(s\gamma w) = (r\gamma s)\gamma w$, где $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

Операции разности и деления не ассоциативны

9. Операции объединения, пересечения, натурального соединения являются коммутативными, т.е. $r\gamma s = s\gamma r$, где $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

8-е и 9-е правило позволяют выбрать в качестве отношения «за скобками» наименьшее по мощности отношение

10. Операции объединения, пересечения, натурального соединения являются идемпотентными, т.е. $r\gamma r = r$, где $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

11. Правило транзитивного замыкания предикатов

Т.к. использование ранней выборки часто оправдано, то и создание условий для ранних выборок тоже полезно. Например, вот что можно проделать со сложным логическим условием C1:

C1=(A>B) AND (B>3) = (A>B) AND (A>3) AND (B>3),

где A, B – атрибуты двух **разных** отношений.

Т.е. дополнение исходного условия C1 условием (A>3) позволяет выполнить раннюю выборку до операции соединения для проверки условия (A>B).

12. Правило преобразования логического условия в конъюнктивную нормальную форму.

C1 OR (C2 AND C3) = (C1 OR C2) AND (C1 OR C3)

Второе выражение (являющееся конъюнктивной нормальной формой) истинно, только если **все** его составные части истинны. Следовательно, если хотя бы одна часть дает «ложь», то ложно и все условие

13. Семантические преобразования

Знание правил целостности для конкретной базы данных позволяет проводить семантические преобразования:

Выполним выражение:

$\pi_{код\_товара}("ПОСТАВКИ" \triangleright\triangleleft "ПОСТАВЩИКИ")$

В этом соединении внешнему ключу в отношении «ПОСТАВКИ» соответствует первичный ключ (в общем случае – потенциальный ключ) из «ПОСТАВЩИКОВ» - «п_номер». Следовательно, кортеж из отношения «ПОСТАВКИ» **уже связан** с определенным кортежем из «ПОСТАВЩИКОВ» и выполнять соединение нет смысла, т.к. атрибут «код_товара» является атрибутом из «ПОСТАВОК».

Исходное отношение эквивалентно (в силу правила целостности по внешним ключам) такому выражению:

$\pi_{код\_товара}("ПОСТАВКИ")$

В общем виде: любое правило целостности (в том числе и функциональные зависимости) может быть использовано для семантической оптимизации.

Этап 3.

Получив оптимальный вариант запроса, оптимизатор выбирает способ его выполнения. Каждая низкоуровневая операция (соединение, выборка по равенству, выборка по неравенству, выборка для индексированных атрибутов и т.п.) имеет несколько процедур реализации. Каждая процедура имеет определенную «**стоимость**». Обычно стоимость операции вычисляется в количестве операций чтения/записи с диска.

Этап 4.

Выбрав несколько путей выполнения запроса, оптимизатор сравнивает их суммарную стоимость и выбирает самый «дешевый» план, в котором (в идеале) учитываются и стоимости получения промежуточных результатов в зависимости от размеров отношений.

Оценка стоимости – достаточно сложная проблема.

Современные коммерческие реляционные СУБД (РСУБД) имеют возможности использования оптимизаторов, однако разработчик может принудительно составить свой план исполнения запроса.

Так, например, в РСУБД ORACLE есть команда Explain Plan, по которой выдается полная информация о стоимости выполнения запроса.