БАЗЫ ДАННЫХ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Ст.преп. каф. ИТ Кеберле Наталья Геннадьевна

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

На прошлых лекциях

Мы познакомились с реляционной алгеброй

- Реляционная алгебра – это математический аппарат для манипулирования данными в реляционной модели данных

Мы также узнали, что недостатки реляционной модели данных – избыточность данных и аномалии обновления – можно исправить, используя процесс нормализации

Мы выяснили, что схема БД в процессе нормализации увеличивается по количеству реляционных отношений – например, в 5НФ чаще всего будут находиться двухатрибутные реляционные отношения (почему ? - подумайте на досуге)

Из практики нам известно, что чаще всего пользователей интересуют данные как раз из нескольких таблиц…

Противоречие

- Возникает противоречие:

  зачем нормализовывать схему БД, если потом всё равно придётся выполнять многократные соединения реляционных отношений?

Оказывается, **принятие решения о необходимости нормализации** того или иного отношения напрямую **зависит** от того, как используется данное отношение ***в запросах***

- На сегодняшней лекции мы познакомимся с языками запросов и механизмами оптимизации производительности СУБД при выполнении запросов

Понятие языка запросов

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

Все запросы должны формулироваться на некотором **языке запросов**.

Примером такого языка является реляционная алгебра.

Пример запроса в языке реляционной алгебры

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

рейсы (

рейс	пункт_ отправл	пункт_ прибытия	время_ вылета	время_ прибыт
83	Нью-Йорк	Чикаго	10:15	12:28
84	Чикаго	Нью-Йорк	14:15	16:30
109	Нью-Йорк	Лос-Анджелес	21:50	2:52
213	Нью-Йорк	Бостон	11:43	12:45
214	Бостон	Нью-Йорк	14:20	15:12
117	Атланта	Бостон	06:30	09:35

)

варианты (

рейс	тип_ самолёта	пилот
83	727	Симонс
83	727	Хилл
83	747	Барт
83	747	Хилл
84	727	Симонс
84	727	Хилл
84	747	Барт
84	747	Хилл
109	707	Симонс

)

Пример запроса в языке реляционной алгебры

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

рейсы (

рейс	пункт_ отправл	пункт_ прибытия	время_ вылета	время_ прибыт
83	Нью-Йорк	Чикаго	10:15	12:28
84	Чикаго	Нью-Йорк	14:15	16:30
109	Нью-Йорк	Лос-Анджелес	21:50	2:52
213	Нью-Йорк	Бостон	11:43	12:45
214	Бостон	Нью-Йорк	14:20	15:12
117	Атланта	Бостон	06:30	09:35

)

варианты (

рейс	тип_ самолёта	пилот
83	727	Симонс
83	727	Хилл
83	747	Барт
83	747	Хилл
84	727	Симонс
84	727	Хилл
84	747	Барт
84	747	Хилл
109	707	Симонс

)

$$r = \pi_{рейс,пилот}\left(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}\left(рейсы \triangleright\triangleleft варианты\right)\right)$$

Это и есть запрос

Типы языков запросов

- Вообще различают процедурные и декларативные языки
- Реляционная алгебра – процедурный язык:
  - 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 сформулировать (продекларировать) искомый результат:
  - запрос на таком языке запросов представляет собой описание того, «что» нужно получить, а не «как» это сделать.

Декларативные языки запросов

- Реляционное исчисление на кортежах
- Реляционное исчисление на доменах
- Язык **SQL** – **Structured Query Language** – язык запросов, основанный на реляционном исчислении на кортежах.
- Язык **QBE – Query By Example** – язык, основанный на реляционном исчислении на доменах.

Эквивалентность языков запросов

- Кодд показал (1971 г.), 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.

Эквивалентность языков запросов

- 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.

Эквивалентность языков запросов

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

- Таким образом,
  - реляционное исчисление кортежей
  - реляционное исчисление доменов
  - реляционная алгебра

  эквивалентны

Реальные языки запросов

- 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 (CREATE TABLE, CREATE DOMAIN,…).

Оптимизация запросов: пример

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

рейсы (

рейс	пункт_ отправл	пункт_ прибытия	время_ вылета	время_ прибыт
83	Нью-Йорк	Чикаго	10:15	12:28
84	Чикаго	Нью-Йорк	14:15	16:30
109	Нью-Йорк	Лос-Анджелес	21:50	2:52
213	Нью-Йорк	Бостон	11:43	12:45
214	Бостон	Нью-Йорк	14:20	15:12
117	Атланта	Бостон	06:30	09:35

)

варианты (

рейс	тип_ самолёта	пилот
83	727	Симонс
83	727	Хилл
83	747	Барт
83	747	Хилл
84	727	Симонс
84	727	Хилл
84	747	Барт
84	747	Хилл
109	707	Симонс

)

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

Оптимизация запросов: пример

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

- Посмотрим на реальную картину:

and international markets. Freighter flight activity was down 34.9%.

THE PORT AUTHORITY OF NY & NJ

APRIL 2009 TRAFFIC REPORT

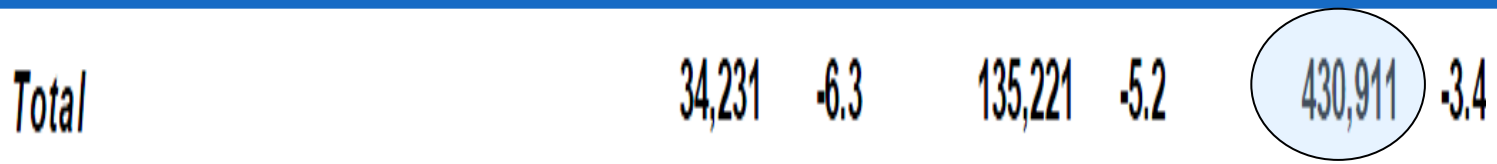
Current month, 12 months ending, year-to-date totals
Showing percentage change from prior year period

Пасса-жиры

Рейсы

JFK	Month		Year-to-date		12 Months Ending	
	Current	%	Current	%	Current	%
PASSENGERS						
Domestic Air Carrier	2,059,125	-5.0	7,563,911	-6.5	24,846,962	-4.7
International Air Carrier	1,766,212	0.0	6,282,575	-7.2	21,932,130	-0.6
Total Revenue Passengers	**3,825,337**	**-2.7**	**13,846,486**	**-6.8**	**46,779,092**	**-2.8**
Non Revenue Passengers	**106,870**	**-0.6**	**424,151**	**-0.7**	**1,351,854**	**2.6**
Note: Commuter - Regional Pax incl. in above	**292,590**	**11.7**	**1,099,138**	**14.7**	**3,726,572**	**13.0**
FLIGHTS						
Domestic Air Carrier	22,731	-5.8	89,701	-4.4	280,986	-4.1
International Air Carrier	10,990	-5.7	43,502	-5.7	141,944	-1.2
General Aviation	510	-31.6	2,018	-24.1	7,981	-16.0
Total	**34,231**	**-6.3**	**135,221**	**-5.2**	**430,911**	**-3.4**
Note:freighter flights included in above	**1,047**	**-34.9**	**4,137**	**-35.7**	**15,274**	**-25.3**

Оптимизация запросов: пример



- За один год через один аэропорт проходит 430000 рейсов (и пассажирских, и грузовых)
  - статистика по общей загрузке аэропорта JFK (New York) http://www.panynj.gov/airports/pdf-traffic/APR2009_LGA.PDF
- Допустим, для перевозок равновероятно используется 10 типов самолётов, и 43000 рейсов (1/10 от 430000) в год выполняется на самолетах типа 727
- Каждый день за время с 07:00 до 12:00 в аэропорту Нью-Йорка взлетает 250 самолётов. В сутки взлетает 500 (= 250*2) самолётов
  - статистика по взлётам/посадкам в аэропорту JFK http://queens.about.com/gi/o.htm?zi=1/XJ&zTi=1&sdn=queens&cdn=citiestowns&tm=8&f=00&su=p284.9.336.ip_p554.12.336.ip_&tt=2&bt=0&bts=0&zu=http://tracker.flightview.com/htNYNJPA/default.asp%3Fcustomer%3DNYNJPA%26mode%3DDEPARTURES%26airport%3DJFK)
- Допустим также, у каждого рейса есть 2 сменных главных пилота, и каждый рейс выполняется на 1 типе самолёта.

Оптимизация запросов: пример вычисления стоимости

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

По реальным данным оценим размеры таблиц «рейсы» и «варианты»:

В таблице «рейсы» для отправления из Нью-Йорка – ½ от 430000 записей – оценим размер таблицы «рейсы» как 430000 записей;

В таблице «варианты» на каждом рейсе используется 1 тип самолёта и 1 из 2 сменных пилотов, при этом 43000 рейсов выполняются на самолёте типа 727, а всего используется 10 типов самолётов – оценим размер таблицы «варианты» как 10*43000*2=860000 записей

Оптимизация запросов: пример вычисления стоимости

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'} (рейсы \triangleright\triangleleft варианты))$$

Вычислим стоимость выполнения запроса «в лоб», сначала – то, что в скобках:

1. Сначала нужно выполнить соединение двух отношений “рейсы” и “варианты”. Это значит, что будет считано 430000 записей о рейсах и для каждой из этих записей будет проведен поиск среди 860000 вариантов. Результат этой операции – отношение из 369 млрд. 800 млн. записей. Предполагаем, что оно будет записано на диск, т.к. оперативной памяти, скорее всего, не хватит (кстати, сколько потребуется памяти?)

Оптимизация запросов: пример вычисления стоимости

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

2. Из 369 млрд. 800 млн. записей, полученных в результате соединения, выбирается 43000*2 записей с нужным типом самолёта и вылетом из Нью-Йорка. Они помещаются в оперативную память. Результат операции - отношение из 86000 записей, операций - 369 млрд. 800 млн.
3. Выбираются столбцы «рейс» и «пилот». Результат операции – отношение из 86000 записей, операций - 86000.
4. Итого было выполнено 2*369 млрд 800 млн + 2*86000 операций.

Оптимизация запросов: пример вычисления стоимости

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

Теперь слегка изменим процедуру выполнения запроса:

1. Сначала выберем из 430000 записей о рейсах те, которые отправляются из Нью-Йорка. Результат – ½ от 430000 записей (=215000 записей), операций - 430000.
2. Затем выберем из таблицы «варианты» те, которые касаются самолётов типа 727. Результат – 86000 записей, операций - 860000.

Оптимизация запросов: пример вычисления стоимости

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

3.Выполняем проекцию на столбцы «рейс, пилот» по таблице «варианты» - в результате получим тоже 86000 записей, но более коротких. Операций – 86000
4.Выполняем соединение таблиц из п.1 и 3 – результат 18 млрд. 490 млн. записей, операций - 18 млрд. 490 млн.
5.Итого было выполнено
18 млрд.490 млн. + 430000 + 860000 + 86000 операций

Оптимизация запросов: сравнение результатов

Показать данные о том, какие пилоты могут выполнять рейсы из Нью-Йорка на самолетах типа 727

$$r = \pi_{рейс,пилот}(\sigma_{n\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

Итак, просто поменяв порядок вычислений в запросе, мы получили сокращение количества операций с 2*369 млрд 800 млн до 18 млрд.491 млн., т.е. в 41 раз.

Оптимизация запросов: как это делать?

Оптимизация запросов позволяет выполнять запросы быстрее и эффективнее (с меньшими ресурсами).

Порядок, в котором выполняются операции в запросе, **можно поменять**, руководствуясь свойствами операций реляционной алгебры.

Процесс оптимизации

- **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.

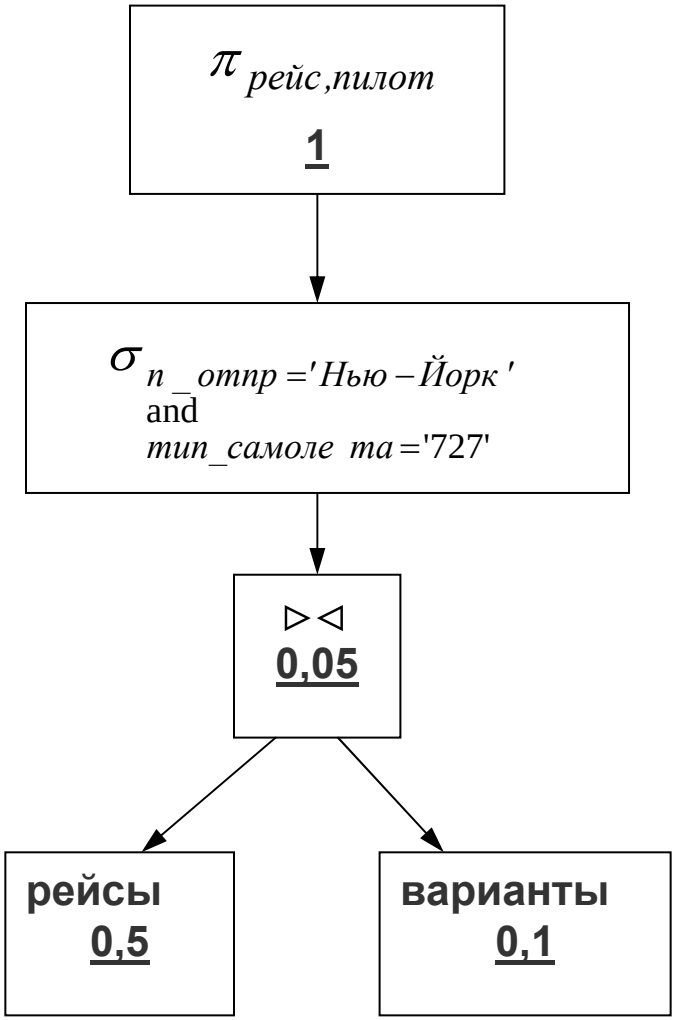
Процесс оптимизации, этап 1: Преобразование запроса в реляц.алгебру

- **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.

```
SELECT варианты.рейс, варианты.пилот
FROM варианты, рейсы
WHERE варианты.рейс=рейсы.рейс
    AND варианты.тип_самолёта=“727”
    AND рейсы.п_отправ=“Нью-Йорк”
```

$$r = \pi_{рейс,пилот}(\sigma_{п\_отпр='Нью-Йорк' \text{ and } тип\_самолета='727'}(рейсы \triangleright\triangleleft варианты))$$

Процесс оптимизации, этап 1: диаграмма запроса



Вершины графа – таблицы

Дуги – соединения между таблицами.

Направление показывает, что соединение гарантированно получит уникальные значения в таблице, на которую указывает связь

Подчеркнутое число – доля строк таблицы, удовлетворяющая условию фильтрации для этой таблицы

Процесс оптимизации: этап 2 оптимизация реляционного выражения

- Оптимизатор запросов использует известные свойства операций реляционной алгебры для построения оптимального выражения

Процесс оптимизации: этап 2 оптимизация реляционного выражения

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. Если $\gamma \in \{\cup, \cap, \setminus\}$, то

$$\sigma_{C1}(r\ \gamma\ \mathrm{s}) = \sigma_{C1}(r)\ \gamma\ \sigma_{C1}(s)$$

Процесс оптимизации: этап 2 оптимизация реляционного выражения

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. Если $\gamma \in \{\cup, \cap, \setminus\}$, то

$$\pi_X(r\ \gamma\ s) = \pi_X(r)\ \gamma\ \pi_X(s)$$

7. «Ранняя проекция»: если проекция выполняется по атрибутам, включающим в себя все атрибуты , по которым выполняется соединение ($X \supset R \cap S$), то

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

Процесс оптимизации: этап 2 оптимизация реляционного выражения

8. Ассоциативность $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

$$r \gamma (s \gamma w) = (r \gamma s) \gamma w$$

9. Коммутативность $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

$$r \gamma s = s \gamma r$$

10. Идемпотентность $\gamma \in \{\cup, \cap, \triangleright\triangleleft\}$

$$r \gamma r = r$$

Процесс оптимизации: этап 2 оптимизация реляционного выражения

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).

Процесс оптимизации: этап 2 оптимизация реляционного выражения

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

Процесс оптимизации: этап 2 оптимизация реляционного выражения

13. Семантические преобразования

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

$$\pi_{пилот}(\ рейсы \triangleright\triangleleft варианты\ )$$

В этом соединении внешнему ключу в таблице «*варианты*» соответствует первичный ключ (в общем случае – потенциальный ключ) из «*рейсы*» - «рейс». Следовательно, строка таблицы «варианты» **уже связана** с какой-то строкой из «*рейсы*» и выполнять соединение нет смысла, т.к. атрибут «пилот» является атрибутом из «*варианты*».

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

$$\pi_{пилот}(\ варианты\ )$$

Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

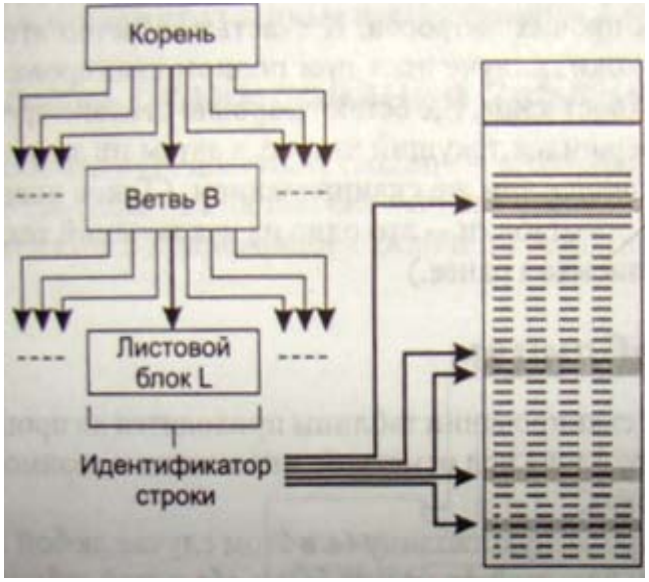
- Получив оптимальный вариант запроса, оптимизатор выбирает способ его выполнения.
- Каждая низкоуровневая операция (соединение, выборка по равенству, выборка по неравенству, выборка для индексированных атрибутов и т.п.) имеет несколько процедур реализации.
- Каждая процедура имеет определенную «**стоимость**».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

Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

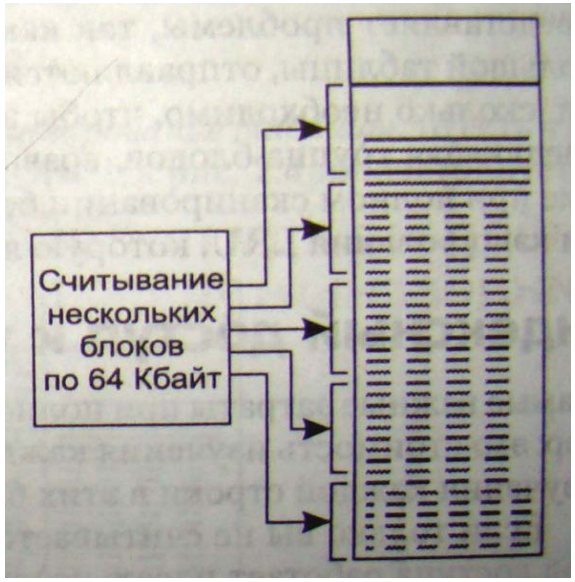
Существуют различные низкоуровневые процедуры выполнения операций над таблицами

Однотабличные запросы могут выполняться двумя способами:

Индексный доступ (**index seek**)



Полнотекстовое сканирование таблицы (**table scan**)



Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

- Выбор между полным сканированием таблицы или индексным доступом зависит от размеров фрагмента таблицы, который должен обработать однотабличный запрос

- Существует несколько приблизительных критериев:

Если запрос должен обработать

- > 20% строк – использовать полное сканирование таблицы;
- <0,5% строк – использовать индексный доступ
- 0,5% - 20% - рассматривать дополнительные условия

Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

Оценка количества строк, которые обрабатывает однотабличный запрос, ещё называется селективностью фильтра.

рейсы
0,5

варианты
0,1

У таблицы *варианты* селективность фильтра =0,1 (т.е. 10%), что лучше чем у таблицы *рейсы* (50%). Однако 10% >0,5%, поэтому нужны дополнительные обоснования использования индекса

Чем больше селективность фильтра (т.е. чем ближе к нулю это значение) – тем лучше использовать индексы и тем раньше нужно применять этот фильтр

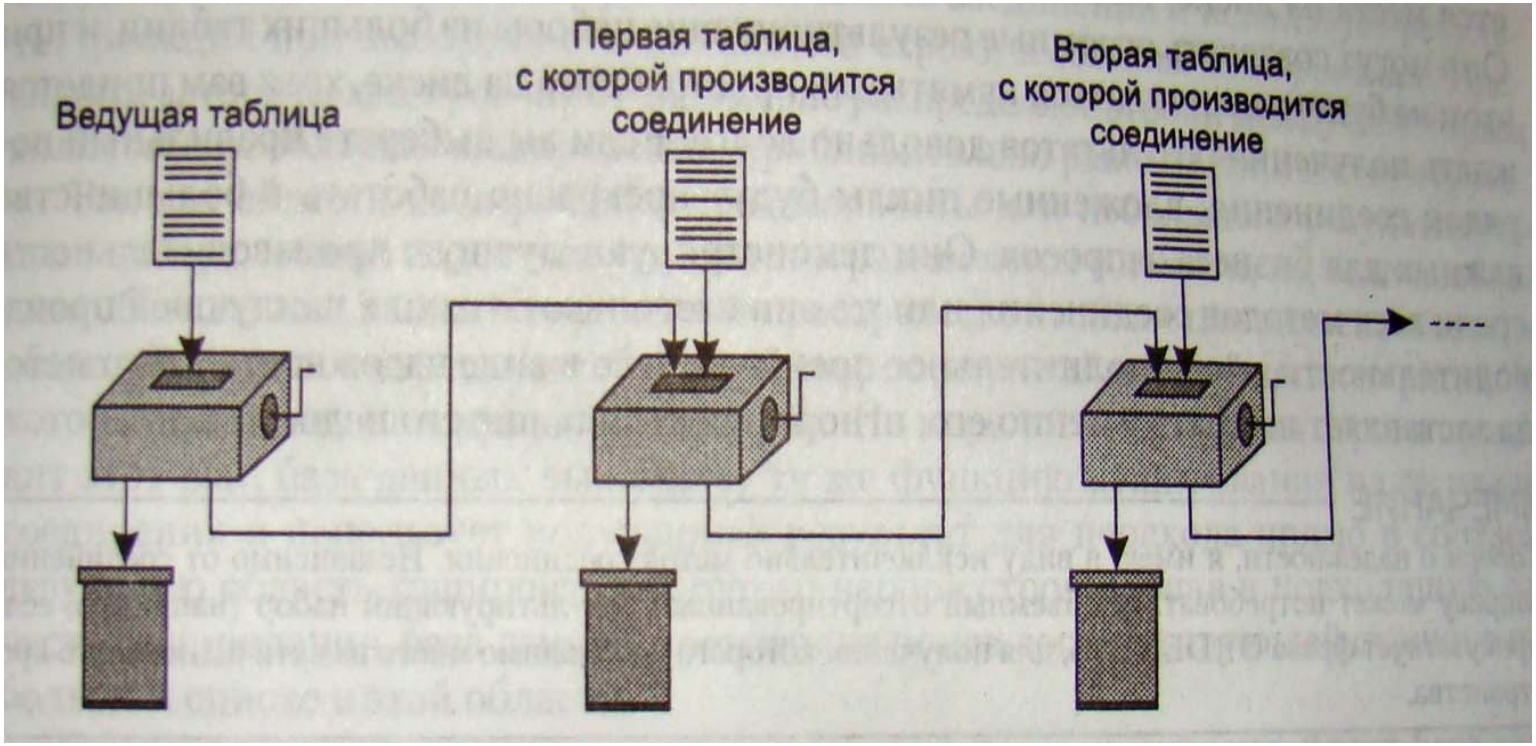
Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

- **Многотабличные запросы** предполагают выполнение соединений – операций, требующих много ресурсов памяти

- Различают такие способы выполнения многотабличных соединений:
  - Вложенные циклы (nested loops)
  - Соединение хэшированием (hash match)
  - Соединение с сортировкой слиянием (merge sort)

Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

- Соединение с помощью **вложенных циклов** (**nested loop**) – наиболее часто используется, т.к. позволяет выполнять многотабличные соединения без риска использования всей памяти



Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

- Соединение хэшированием

Иногда в запросе нужно **обратиться** к соединяемым таблицам **по отдельности**, а затем соединить соответствующие строки и отбросить ненужные.

Тогда используется либо соединение хэшированием…

Процесс оптимизации: этап 3
Выбор низкоуровневых процедур

- Соединение с сортировкой слиянием

Иногда в запросе нужно **обратиться** к соединяемым таблицам **по отдельности**, а затем соединить соответствующие строки и отбросить ненужные.

... либо соединение с сортировкой слиянием

Оптимизация запросов: этап 4
Генерация планов вычисления запросов

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

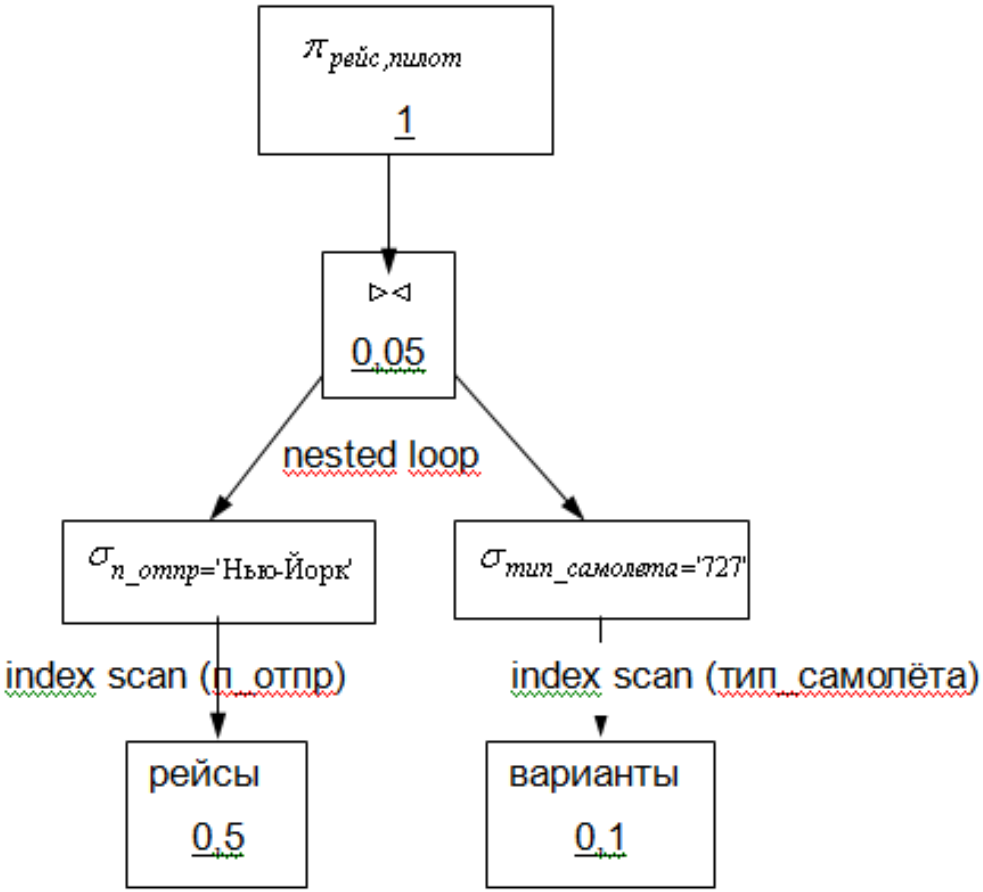
- В СУБД для просмотра плана исполнения запроса (и для проверки «навязанного» плана, специально указанного разработчиком запроса) используются команды SQL:

ORACLE - explain plan

mySQL – explain select … from … where

MS SQL Server – set showplan_text ON

Процесс оптимизации: назад к примеру



Что узнали сегодня?

- На сегодняшней лекции мы познакомились с языками запросов и механизмами оптимизации производительности СУБД при выполнении запросов