Оптимизация

Оптимизация и реляционная модель

- + Автоматическая (высокоуровневая)
- - Всегда необходима

Преимущества

- Наличие статистических данных (Системный каталог)
- Оптимизация «здесь» и «сейчас»
- Анализ большого числа альтернатив

Проблемы

- оптимальность не гарантируется

Сравнение двух способов

```
select name_ag
   from agent A join operation O
        on (A.id_ag = O.id_ag)
   where O.id_tovar ='T1'
```

100 поставщиков

10 000 операций

50 с товаром T1

1

- соединение
  - 10 000 * 100 чтений
  - 10 000 записей
- селекция
  - 10 000 чтений, выборка 50 записей
- проекция
  - не более 50 записей

2

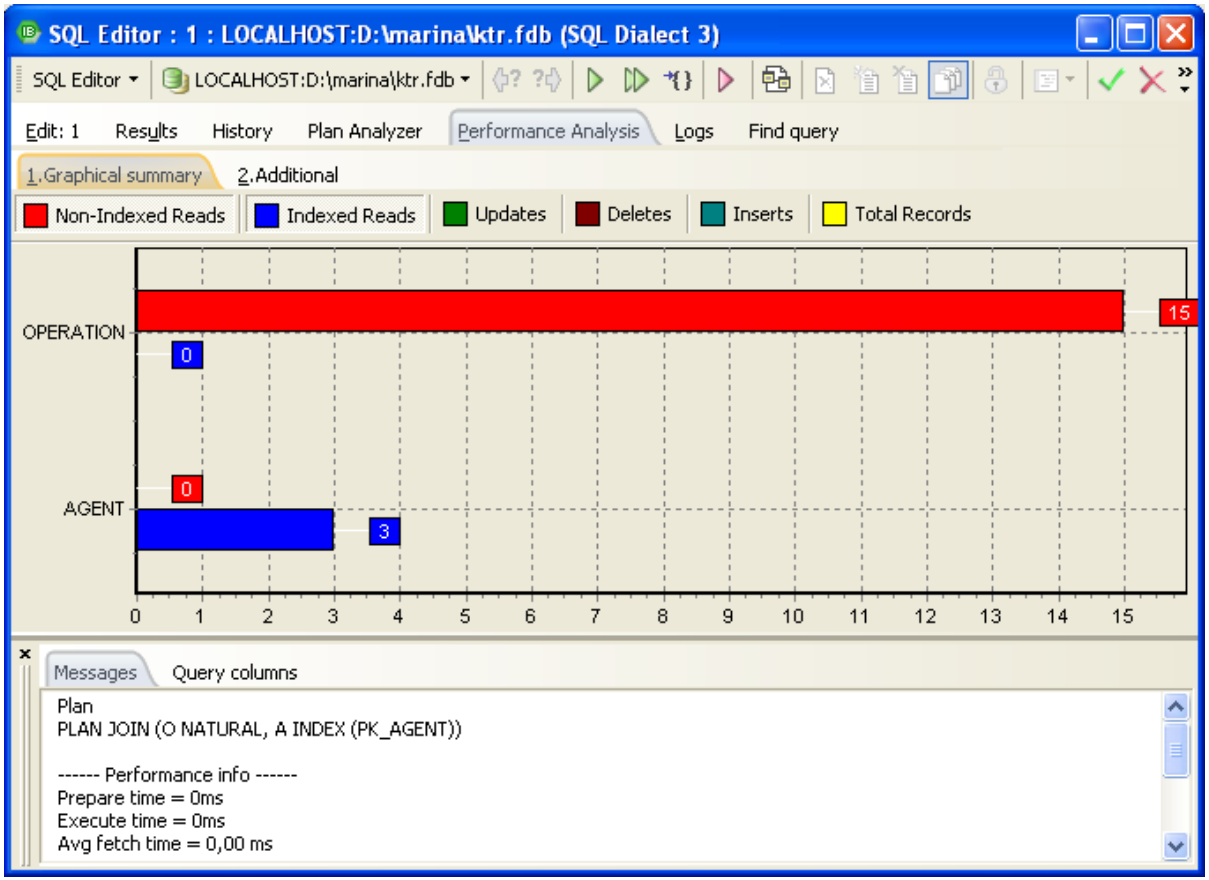
- селекция
  - 10 000 чтений, выборка 50 записей
- соединение
  - 50 * не более 50 записей
- проекция
  - не более 50 записей

Алгоритм

- Преобразование запроса в формальной выражение, например в выражение реляционной алгебры
- Преобразование в каноническую форму на основе законов преобразования
- Выбор низкоуровневых процедур на основе оценки стоимости
- Генерация различных вариантов плана выполнения запроса и выбор плана с минимальными затратами

План выполнения запроса

JOIN (O NATURAL, A INDEX (PK_AGENT))



- **План выполнения запроса** — последовательность операций, необходимых для получения результата SQL-операции в реляционной СУБД
- Производительность
  - кардинальность
  - стоимость

- План в целом разделяется на две стадии:
  - Выборка результатов;
  - Сортировка и группировка, выполнение агрегаций.
- *Сортировка и группировка* — это опциональная стадия, которая выполняется, если не найдено путей доступа для получения результата в запрошенном порядке.
- *Выборка результатов* выполняется следующими способами:
  - Вложенные циклы;
  - Слияние

План запроса

PLAN <выражение>

<выражение> ::= [JOIN | [SORT] [MERGE]]
(<элемент> | <выражение>
[, <элемент> | <выражение> ...])

<элемент> ::= {таблицы | псевдоним}
{NATURAL
| INDEX (индекс [, индекс ...])
| ORDER индекс [INDEX (индекс [, индекс ...])]}

Источники данных

- Классы
  - первичный метод доступа
  - фильтр
  - слияние
- Типы
  - конвейерные
  - буферизованные

Первичные методы доступа

- Создание потока данных на основе таблиц и процедур

Чтение таблицы

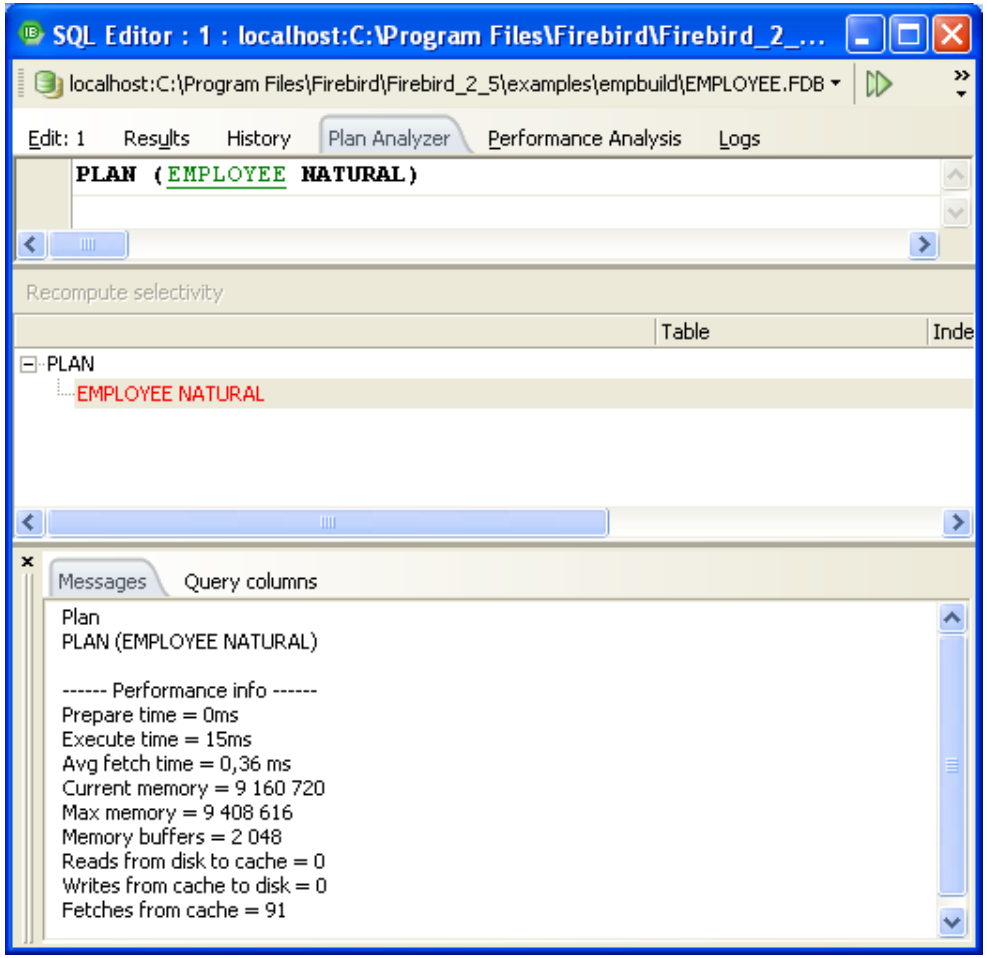
- Полное сканирование

```
select *
FROM employee;
```

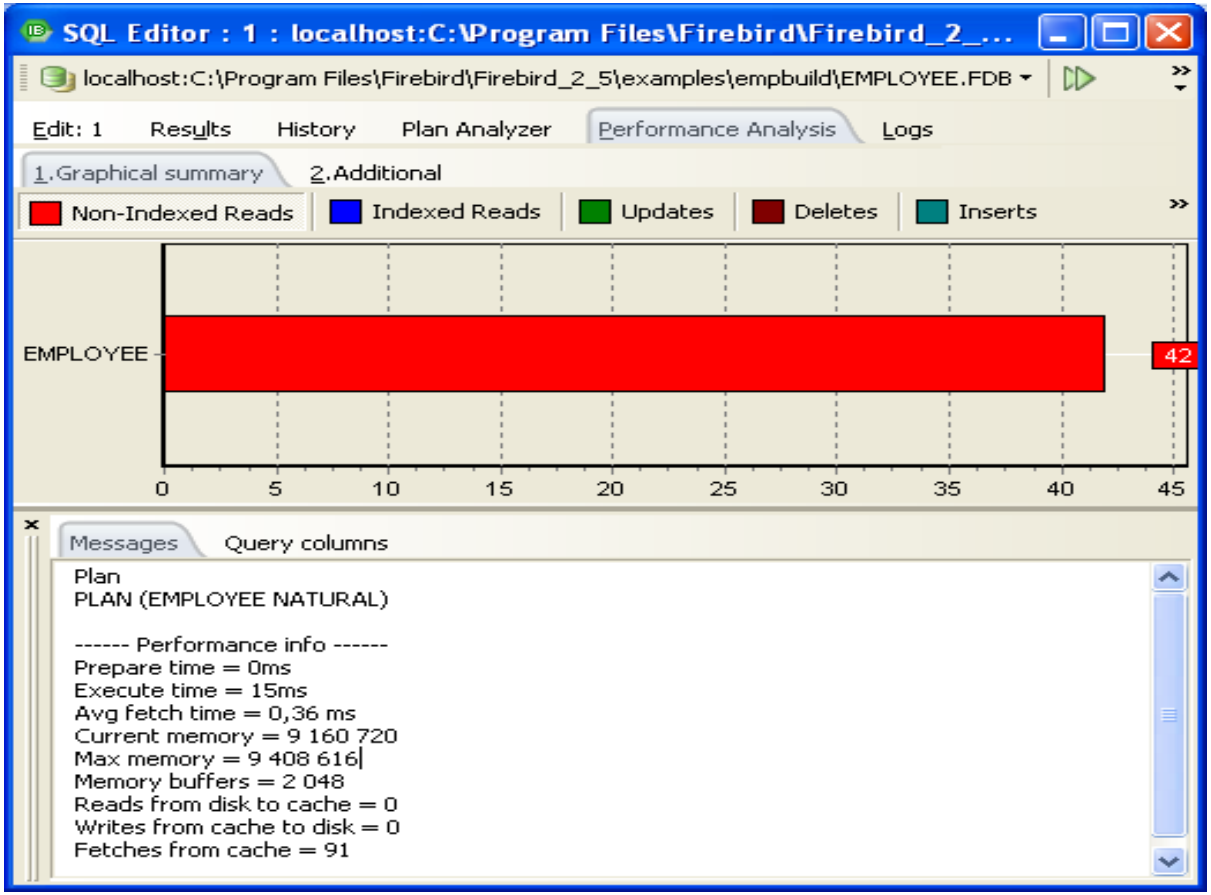
Plan
PLAN (EMPLOYEE NATURAL)

Кардинальное число = 42
стоимость =42

Анализ плана



Анализ плана

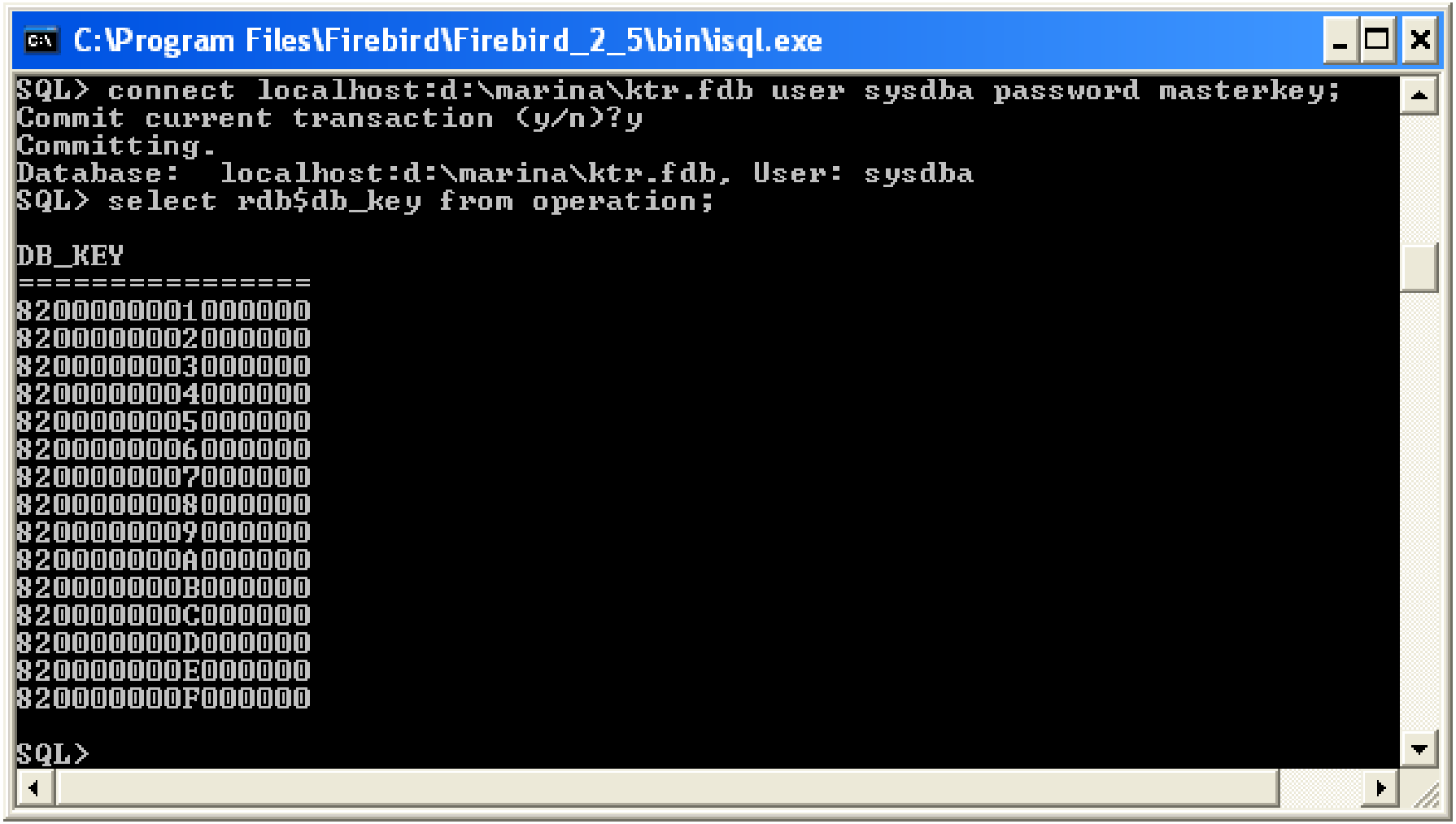


Чтение таблицы

- Доступ через идентификатор записи

Физический номер записи содержит информацию о странице, на которой расположена данная запись, и о смещении внутри этой страницы

Стоимость данного вида доступа всегда равна единице. Чтения записей отражаются в статистике как индексированные.


C:\Program Files\Firebird\Firebird_2_5\bin\isql.exe
SQL> connect localhost:d:\marina\ktr.fdb user sysdba password masterkey;
Commit current transaction (y/n)?y
Committing.
Database: localhost:d:\marina\ktr.fdb, User: sysdba
SQL> select rdb$db_key from operation;
DB_KEY
8200000001000000
8200000002000000
8200000003000000
8200000004000000
8200000005000000
8200000006000000
8200000007000000
8200000008000000
8200000009000000
820000000A000000
820000000B000000
820000000C000000
820000000D000000
820000000E000000
820000000F000000
SQL>

Чтение таблицы

- Позиционированный доступ

команды типа UPDATE и DELETE

(WHERE CURRENT OF <cursor name>)

Позиционированный доступ работает только для активного курсора, т.е. для уже прочитанной записи

Чтение таблицы

- Индексный доступ
  - Основным параметром, влияющим на оптимизацию индексного доступа, является ***селективность*** индекса
  - В расчетах кардинальности и стоимости предполагается равномерное распределение значений ключа в индексе

Индексный доступ

- Составной индекс по полям (A, B, C)
- Используется для предикатов
  - (A=value)
  - (A=avalue and B=bvalue) // or
  - (A= avalue and B=bvalue and C=cvalue)

Индексный доступ

- Используется частично для предикатов
  - (A=avalue and B>bvalue)
  - (A= avalue and B>bvalue and C=cvalue)
- Не используется
  - (A>avalue)
  - (B=bvalue)
  - (B=bvalue and C=cvalue)

Индексный доступ

- Для кластерных индексов возможно использование чтения только индекса
- Если индекс не кластерный, требуется чтение и страниц индекса и страниц данных
- В Firebird основан на битовых картах и учитывает версии
- В Firebird сканирование индекса - однонаправленное

Индексный доступ

- Стоимость доступа через битовую карту – суммарная стоимость индексного поиска для всех индексов, формирующих битовую карту
- Кардинальное число при пересечении битовых карт не больше минимального, при объединении – не меньше максимального

Индексный доступ в диапазоне

- Сканирование диапазона (range scan)
- Сканирование на равенство (equality scan)
- Сканирование на равенство для уникального индекса (unique scan)
- Полное сканирование (full scan)

- При выборе индексов для сканирования оптимизатор использует стратегию на основе стоимости (cost based)
- Стоимость сканирования диапазона оценивается на основании селективности индекса, количества записей в таблице, среднего количества ключей на индексной странице и высоты B+ дерева.

Сканирование на равенство для уникального индекса (unique scan)

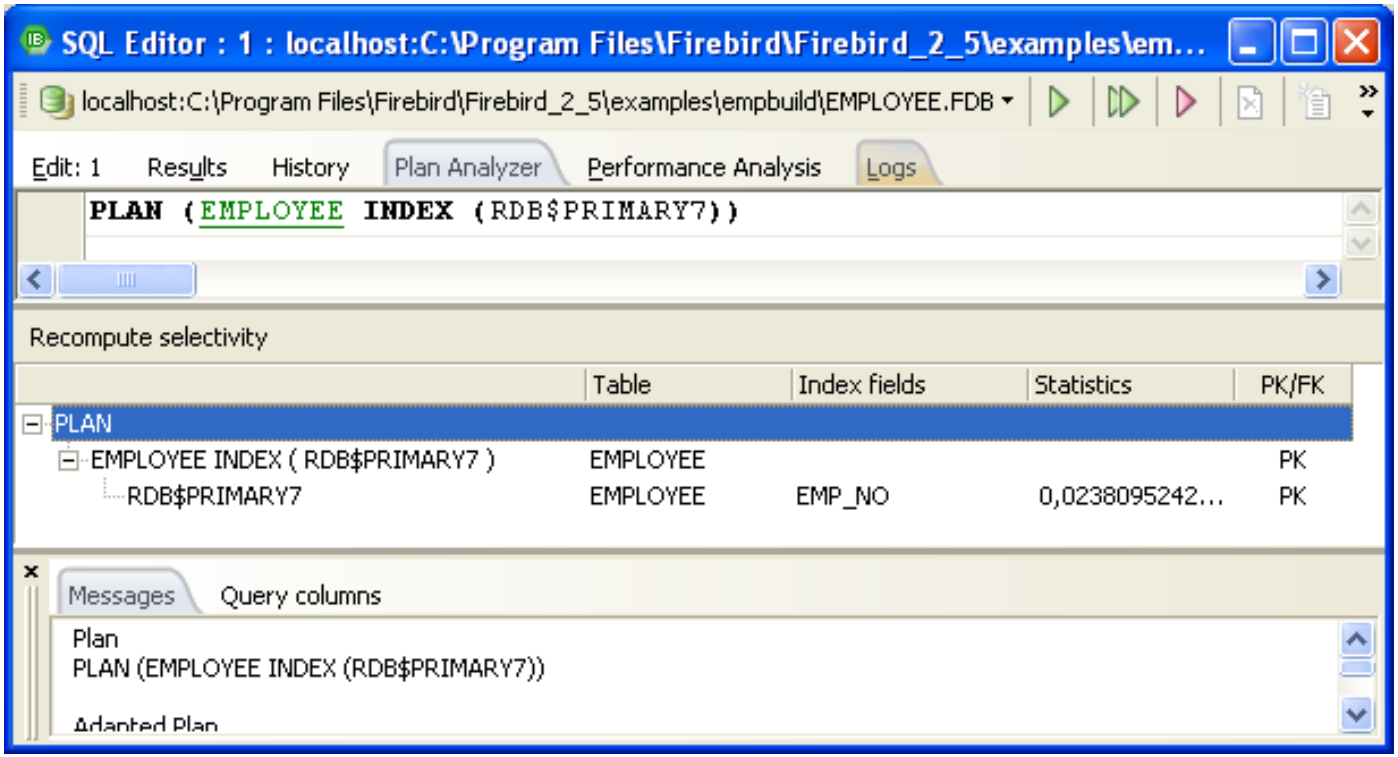
- select * from employee
  where emp_no=15

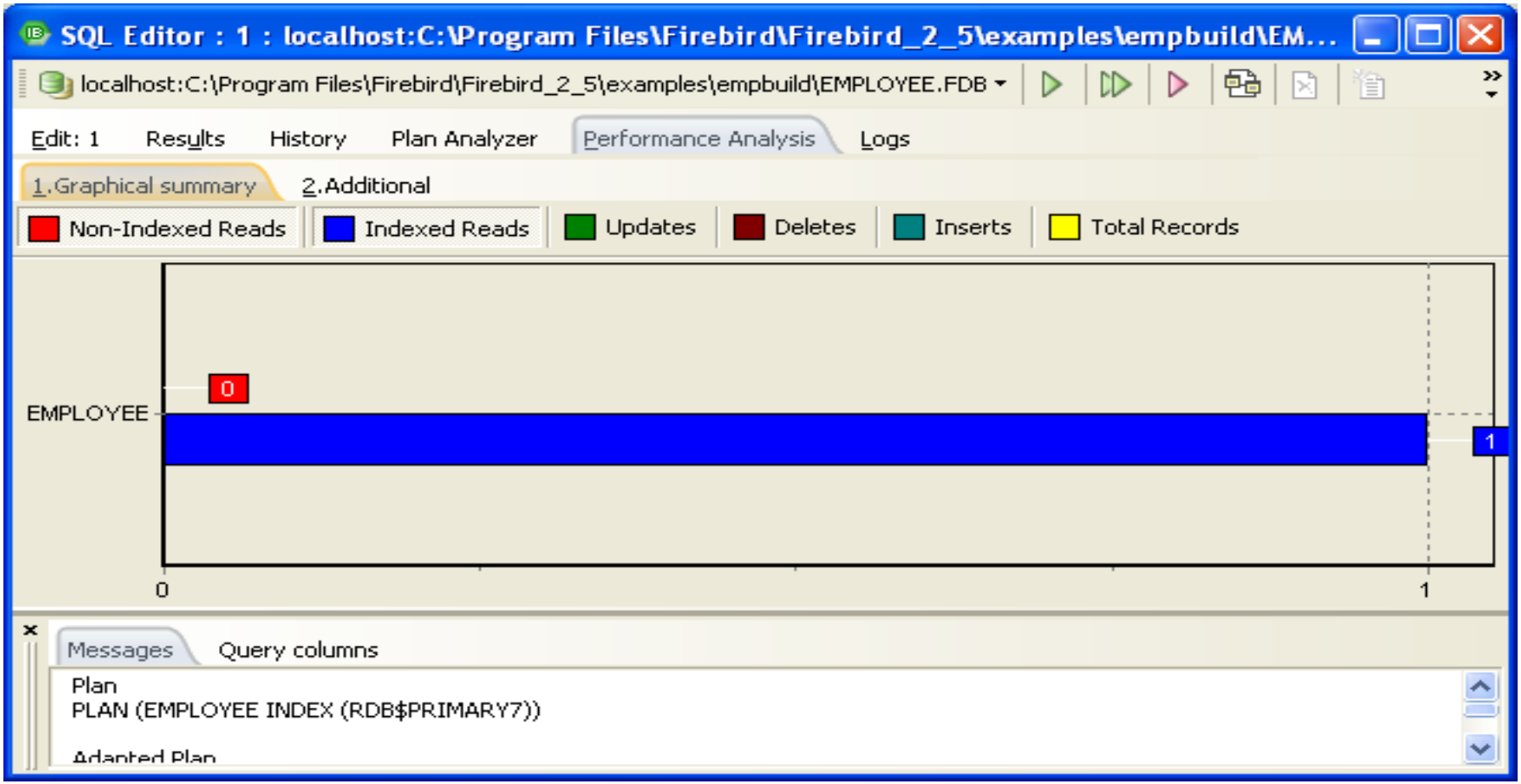
Plan

PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))

Adapted Plan

PLAN (EMPLOYEE INDEX (INTEG_27))


SQL Editor : 1 : localhost:C:\Program Files\Firebird\Firebird_2_5\examples\em...
localhost:C:\Program Files\Firebird\Firebird_2_5\examples\empbuild\EMPLOYEE.FDB
Edit: 1
Results
History
Plan Analyzer
Performance Analysis
Logs
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))
Recompute selectivity
Table
Index fields
Statistics
PK/FK
PLAN
EMPLOYEE INDEX (RDB$PRIMARY7)
EMPLOYEE
PK
RDB$PRIMARY7
EMPLOYEE
EMP_NO
0,0238095242...
PK
Messages
Query columns
Plan
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))
Adapted Plan


SQL Editor : 1 : localhost:C:\Program Files\Firebird\Firebird_2_5\examples\empbuild\EM...
localhost:C:\Program Files\Firebird\Firebird_2_5\examples\empbuild\EMPLOYEE.FDB
Edit: 1
Results
History
Plan Analyzer
Performance Analysis
Logs
1.Graphical summary
2.Additional
Non-Indexed Reads
Indexed Reads
Updates
Deletes
Inserts
Total Records
EMPLOYEE
0
1
0
1
Messages
Query columns
Plan
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))
Adapted Plan

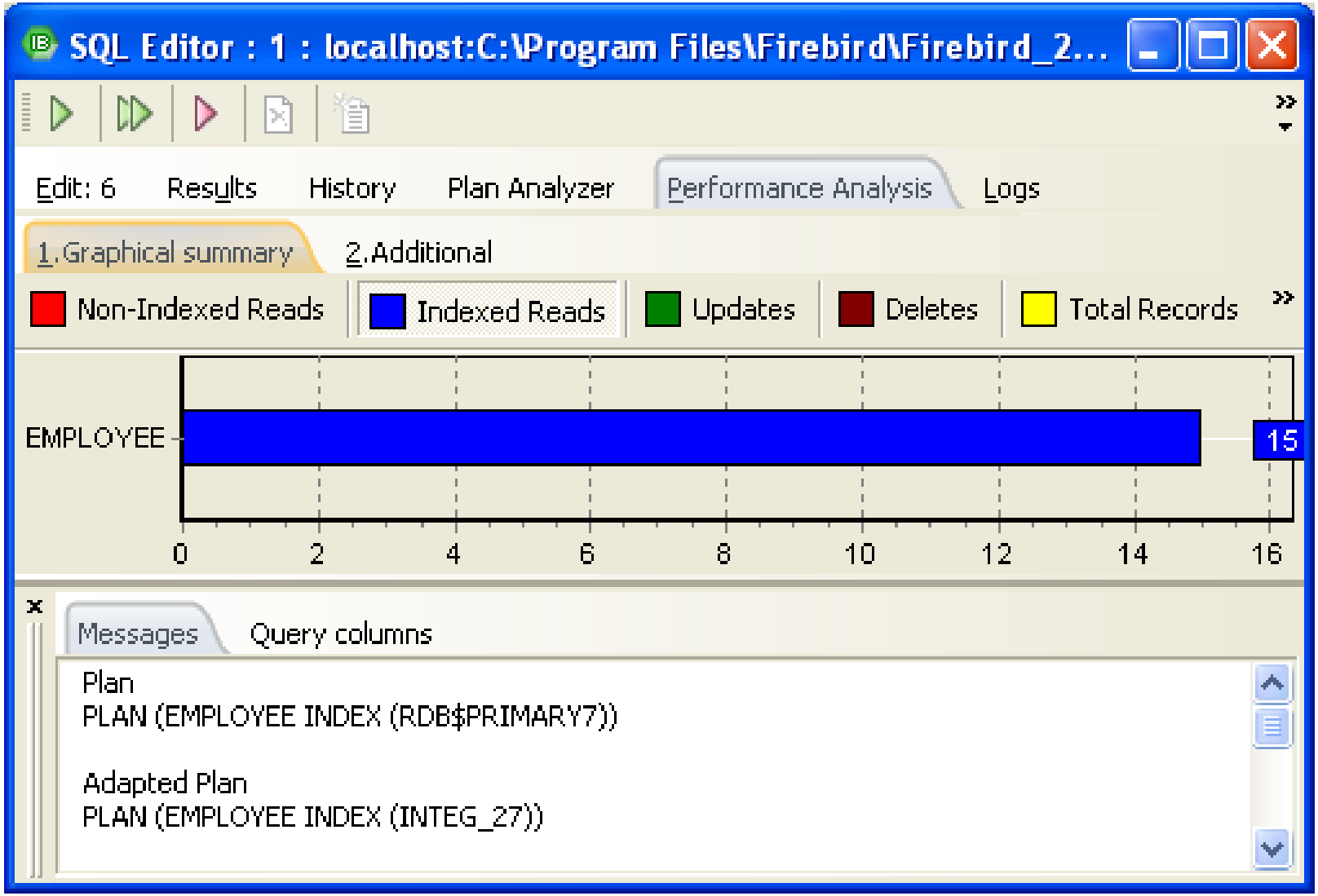
Сканирование диапазона (range scan)

- select * from employee
  where emp_no between 100 and 300

Plan
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))

Adapted Plan
PLAN (EMPLOYEE INDEX (INTEG_27))

Кардинальное число =13


SQL Editor : 1 : localhost:C:\Program Files\Firebird\Firebird_2...
Edit: 6
Results
History
Plan Analyzer
Performance Analysis
Logs
1.Graphical summary
2.Additional
Non-Indexed Reads
Indexed Reads
Updates
Deletes
Total Records
EMPLOYEE
15
0
2
4
6
8
10
12
14
16
Messages
Query columns
Plan
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))
Adapted Plan
PLAN (EMPLOYEE INDEX (INTEG_27))

Полное сканирование (full scan)

```
select * from employee
    where phone_ext=250
PLAN (EMPLOYEE NATURAL)
```

Наложение битовых карт

Тестовая таблица

```
CREATE TABLE PRIME2 (
   ID    INTEGER NOT NULL,
   IDX1  INTEGER,
   IDX2  INTEGER
);
ALTER TABLE PRIME2
        ADD CONSTRAINT PK_PRIME2 PRIMARY KEY (ID);

CREATE INDEX PRIME2_IDX1 ON PRIME2 (IDX1);
CREATE INDEX PRIME2_IDX2 ON PRIME2 (IDX2);
```

Наложение битовых карт

```
select * from prime2
        where idx1=14  AND  idx2=5

PLAN (PRIME2 INDEX (PRIME2_IDX1, PRIME2_IDX2))

STATEMENT (SELECT)
[cardinality=2, cost=9.000]
    => TABLE (PRIME2) ACCESS BY DB_KEY
       [cardinality=2, cost=9.000]
      => BITMAP AND
         [cardinality=2, cost=7.000]
        => INDEX (PRIME2_IDX1) RANGE SCAN
           [cardinality=5, cost=4.000]
        => INDEX (PRIME2_IDX2) RANGE SCAN
           [cardinality=2, cost=3.000]
```

Навигация по индексу

- Используется тогда, когда индекс может помочь в упорядочении результата

```
select * from prime2
    order by idx1

PLAN (PRIME2 ORDER PRIME2_IDX1)


select * from prime2
    order by idx1 desc

PLAN SORT ((PRIME2 NATURAL))
```

```
select min(idx1) from prime2
where idx2>6

PLAN (PRIME2 ORDER PRIME2_IDX1 INDEX
   (PRIME2_IDX2))

   STATEMENT (SELECT)
   [cardinality=100, cost=303.000]
    => TABLE (prime2) ACCESS BY DB_KEY
       [cardinality=100, cost=303.000]
      => INDEX (PRIME2_IDX1) FULL SCAN
         [cardinality=100, cost=203.000]
        => BITMAP
           [cardinality=100, cost=3.000]
          => INDEX (PRIME2_IDX2) RANGE SCAN
             [cardinality=100, cost=3.000]
```

Процедурный доступ

- Процедурный метод доступа представляет собой аналог полного сканирования таблицы
- Стоимость процедурного доступа не рассчитывается и не учитывается
- Кардинальность устанавливается равной нулю

```
SELECT *
    FROM GET_EMP_PROJ (300)

    PLAN (GET_EMP_PROJ NATURAL)
```

Фильтры

- Преобразователи данных
- Имеют только один вход
- Изменяют кардинальное число
- Для фильтров не оценивается кардинальность и стоимость

Проверка предикатов

- WHERE, HAVING, IN, ALL, ANY
- В плане выполнения проверка предикатов не отображается

Сортировка

- ORDER BY, GROUP BY, DISTINCT
- построение B+ дерева индексов
- подготовка данных для однопроходного слияния

Алгоритм сортировки представляет собой многоуровневую быструю сортировку

```
select * from employee
order by salary

PLAN SORT ((EMPLOYEE NATURAL))
```

Агрегация

- Вычисление агрегирующих функций
- В плане выполнения агрегация не показывается

Счетчики

- FIRST/SKIP/ROWS
- В плане выполнения счетчики не показываются

Методы слияния

- всегда оперирует с несколькими входными потоками
- обычным результатом их работы является либо расширение выборки по полям, либо увеличение ее кардинальности
- соединение (join)
- объединение (union)

Соединение (join)

- У любого вида соединений есть два входных потока - левый и правый.
- Для внутреннего и полного внешнего соединений эти потоки семантически равноценны.
- В случае одностороннего внешнего соединения один из потоков является ведущим (обязательным), а второй - ведомым (необязательным).

Соединение (join)

- У каждого соединения помимо входных потоков существует еще один атрибут - условие связи. Именно это условие и определяет результат, то есть как именно будут поставлены в соответствие данные входных потоков.
- При отсутствии данного условия получаем вырожденный случай - декартово произведение (cross join) входных потоков.

Правила выбора потока

- Для односторонних внешних соединений внешний (ведущий) поток всегда должен быть прочитан перед внутренним (ведомым), иначе невозможно будет выполнить требуемую стандартом подстановку NULL-значений в случае отсутствия соответствий внутреннего потока внешнему.

Правила выбора потока

- Для внутренних и полных внешних соединений входные потоки независимы и могут читаться в произвольном порядке, следовательно алгоритм выполнения таких соединений определяется исключительно оптимизатором и никак не зависит от текста запроса.

Методы слияния

- Рекурсивный перебор или соединение посредством вложенных циклов (nested loops join)
- Однопроходное слияние
- Хеширование

Рекурсивный перебор

```
select *
from department d left join employee e
        on d.mngr_no=e.emp_no

PLAN JOIN (D NATURAL, E INDEX
   (RDB$PRIMARY7))
```

Рекурсивный перебор

```
select *
from department d
        join employee e   on d.mngr_no=e.emp_no
       join country  c      on c.country=e.job_country


PLAN JOIN (D NATURAL, E INDEX (RDB$PRIMARY7),
   C INDEX (RDB$PRIMARY1))
```

Однопроходное слияние

- Оптимизатор выбирает данный алгоритм соединения только в случае невозможности или неоптимальности использования рекурсивного алгоритма, то есть в первую очередь при отсутствии индексов по условию связи или их неприменимости, а также при отсутствии зависимости между входными потоками.

```
select *
from department d
        join employee e on
   d.mngr_no+0=e.emp_no+0


PLAN MERGE (SORT (E NATURAL), SORT (D
   NATURAL))
```

Хеширование (общие положения)

- В данном случае входные потоки всегда делятся на ведущий и ведомый, при этом ведомым обычно выбирается поток с наименьшей кардинальностью.
- Сначала меньший (ведомый) поток целиком вычитывается во внутренний буфер.
- В процессе чтения к каждому ключу связи применяется хеш-функция и пара

  {хеш, указатель в буфере} записывается в хеш-таблицу.

Хеширование

- После чего читается ведущий поток и его ключ связи ищется в хеш-таблице.
- Если соответствие найдено, то записи обоих потоков соединяются и выдаются на выход
- Соединение хешированием в Firebird отсутствует.

Объединение (union)

- Существует два режима выполнения этой операции: ALL и DISTINCT.
- В первом случае реализация тривиальна: данный метод просто читает первый входной поток и выдает его на выход, по получении из него EOF начинает читать второй входной поток и так далее.

Объединение (union)

- В случае же DISTINCT требуется устранить полные дубликаты записей, присутствующие в результате объединения. Для этого на выходе метода объединения размещается фильтр сортировки, работающий в "усекающем" режиме по всем полям.

Объединение (union)

- Стоимость выполнения объединения равна суммарной стоимости всех входных потоков, кардинальность также получается суммированием.

```
select job_country
from employee
union
select country
from customer

PLAN (EMPLOYEE NATURAL)
PLAN (CUSTOMER NATURAL)
```

Включение плана в запрос

SELECT [DISTINCT | ALL]

[FIRST <record_number> }|SKIP <record_number>]

<select_list>

FROM <reference_expression_list>

[WHERE <search condition>]

[GROUP BY <group_value_list>]

[HAVING <group_condition>]

[PLAN <plan_item_list>]

PLAN ({ <stream_retrieval> |
<sorted_streams> |
<joined_streams> })

<stream_retrieval> ::= { <natural_scan> |
<indexed_retrieval> | <navigational_scan> }

<natural_scan> ::= <stream_alias> NATURAL

<indexed_retrieval> ::= <stream_alias> INDEX (
<index_name> [, <index_name> ...])

<navigational_scan> ::= <stream_alias> ORDER
<index_name> [INDEX (<index_name> [,
<index_name> ...])]

<sorted_streams> ::= SORT (<stream_retrieval>)

<joined_streams> ::= JOIN (<stream_retrieval>,
<stream_retrieval> [, <stream_retrieval> ...]) |
[SORT] MERGE (<sorted_streams>,
<sorted_streams>)