Структуры хранения, управление памятью, индексирование

Хранение данных

- Операционная система предлагает два способа доступа к диску:
  - на уровне блоков
  - на уровне файлов
- Поскольку дисковые накопители работают очень медленно, используется несколько методов, помогающих сократить время доступа.:
  - кеширование в память
  - цилиндры
  - чередование дисков
- Для повышения надежности, кроме периодического создания резервных копий, используется зеркалирование

Организация хранения

- Для баз данных предпочтительно организовать хранении всех данных в одном или нескольких файлах операционной системы и работе с ними, как если бы они были неструктурированными дисками. То есть система баз данных обращается к своему «диску», используя логические файловые блоки.
- В процессе проектирования методов хранения логических объектов разработчик СУБД должен принимать решения по ряду вопросов, определяющих свойства и характеристики структуры хранения
  - Таблица – файл
  - Таблица – много файлов
  - Файл – много таблиц
  - БД – много файлов

ODS (On-Disk Structure)

- Табличные пространства
- Пространства для временных данных
- Пространство для индексов
- Пространство для схемы БД (метаданных)
- Дополнительные пространства

Отображение логических структур на структуры хранения

- Фрагментация объектов хранения
- Адресация и перемещаемость
- Размещение по значению
- Упорядоченность
- Поведение структуры при внесении изменений

Отображение логических структур на структуры хранения

- должна ли каждая запись целиком размещаться в одном блоке?
- принадлежат ли все записи в блоке одной таблице?
- ограничен ли размер каждого поля заранее определенным количеством
- байтов?
- где в записи располагается значение каждого ее поля?

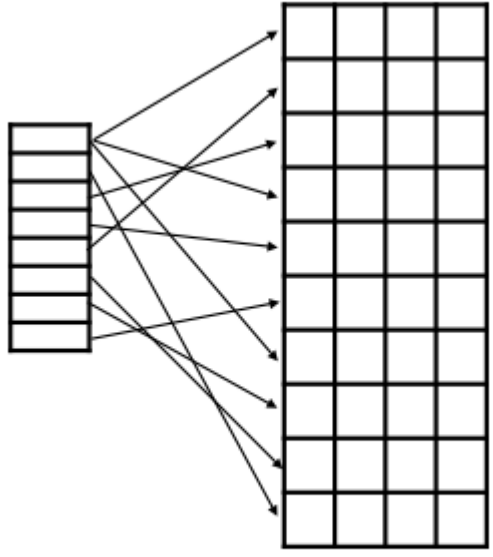
Решения по этим вопросам определяют, какие алгоритмы доступа к данным могут и будут использоваться при работе СУБД

Управление памятью

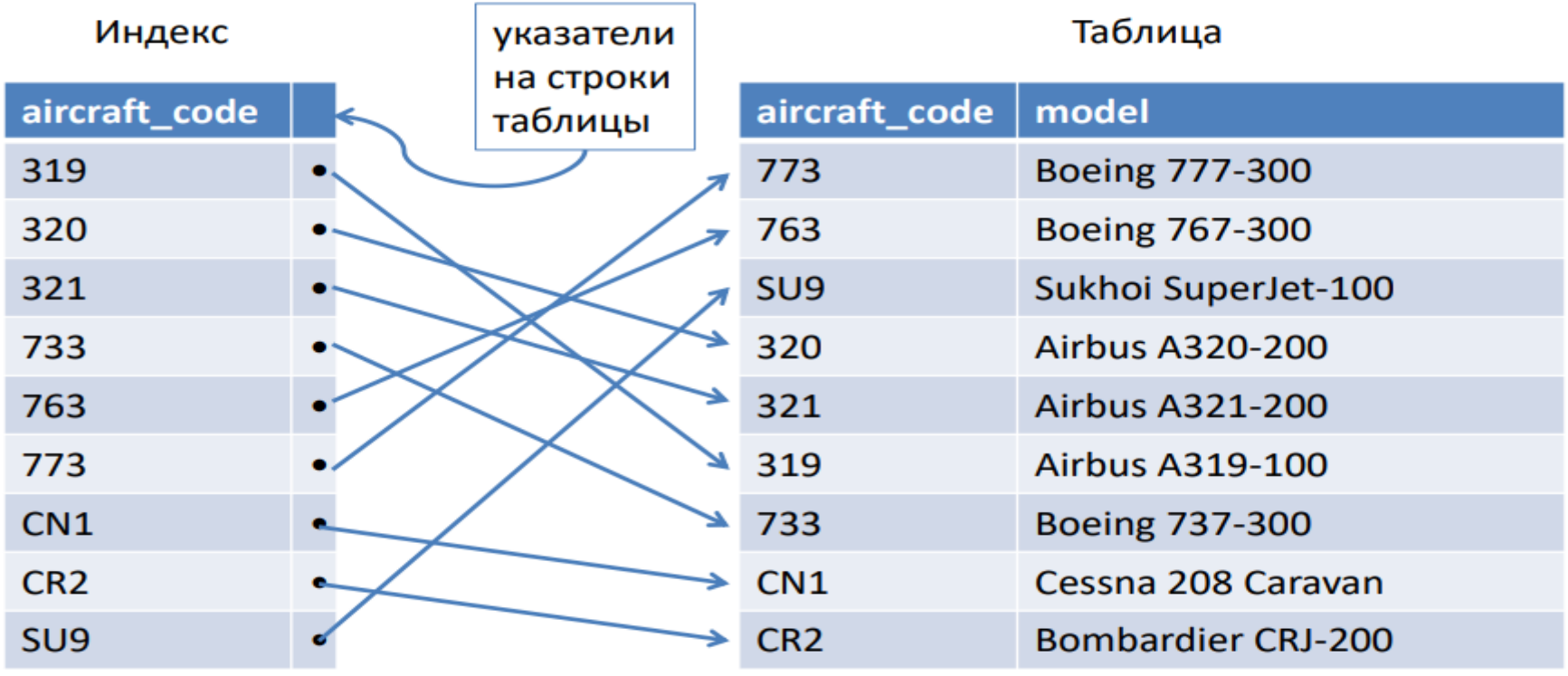
- Необходимо эффективное управление чтением дисковых блоков в оперативную память и записью из оперативной памяти
- Основные принципы
  - Минимизировать число обращений к диску путем предотвращения многократных обращений к одним и тем же блокам данных (кэш)
  - Запись страниц на диск производить только в случае явной необходимости, надеясь одной операцией записи на диск сохранить сразу несколько изменений в странице
  - Самостоятельно управлять страницами кэша, чтобы избежать виртуализации (пул буферов)

Индексирование

- Предназначены для ускорения поиска
- Это избыточная структура
- В общем случае логическая запись индекса состоит из
  - значения атрибута (индексного ключа)
  - списка ссылок на логические записи содержащие это значение

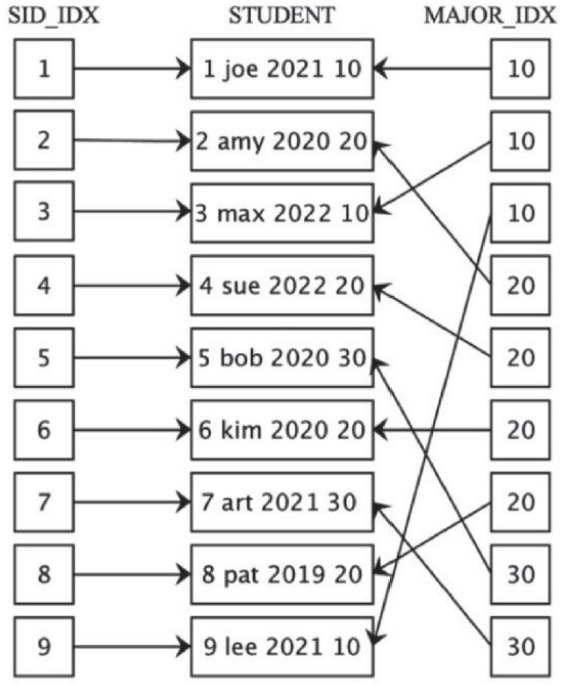


Уникальные индексы



Не уникальные индексы

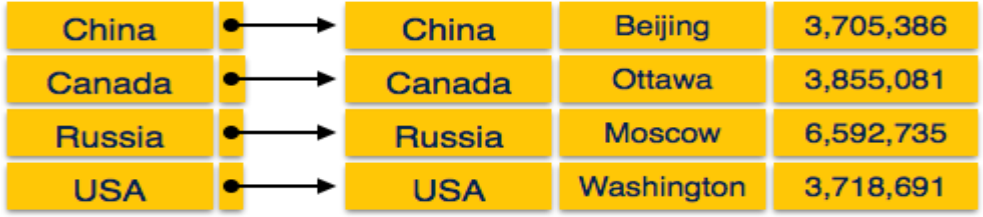
- Можно дублировать индексные записи
  - Возникает избыточность
- Можно хранить списки указателей
  - Потребуются алгоритмы обработки списков
- Важная проблема
  - Индексы и версии данных

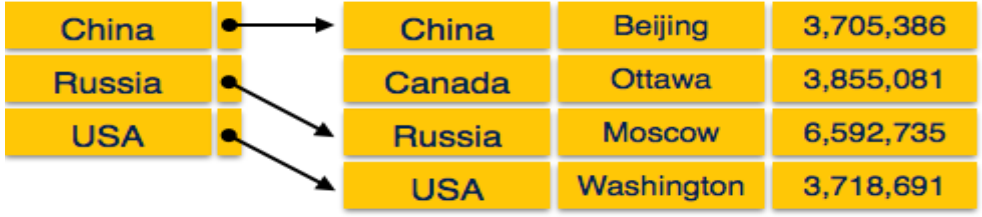


Типы индексов
Кластерные и не кластерные

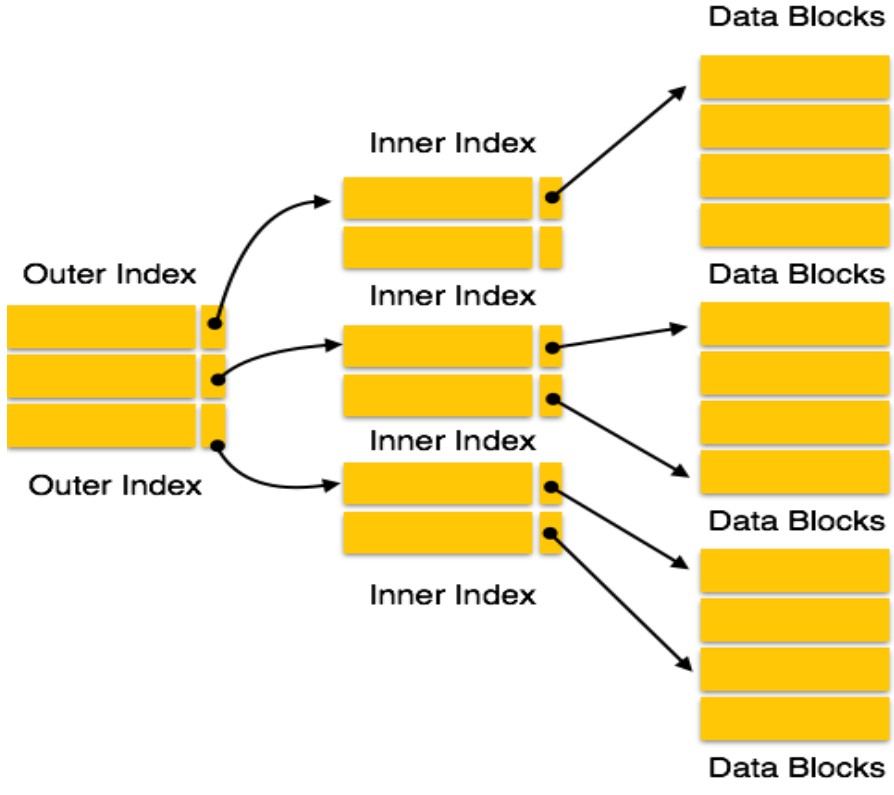
- Способ организации
- Количество для одной таблицы
- Скорость доступа по условию
- Скорость при обновлении
- Фрагментация при удалении

Типы индексов
Плотные и разреженные(dense / sparse)





Типы индексов одноуровневые /многоуровневые

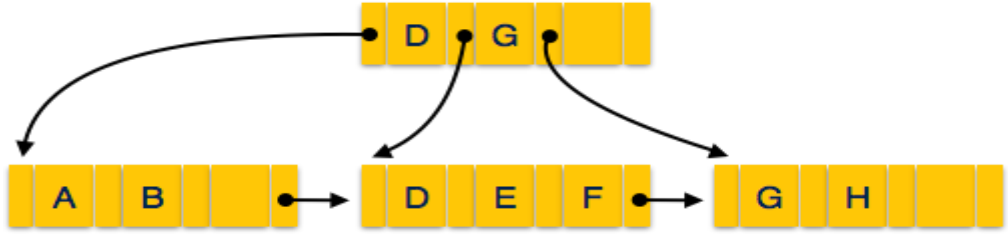


Типы индексов. Одномерные/пространственные

- *Одномерные*, если значения индексного ключа рассматриваются как скалярные. Для них обычно существует естественное упорядочение. Могут формироваться как для одного атрибута, так и для нескольких (составной индекс). Легко реализуются классическими структурами.
- *Пространственные* – ключ индексной записи состоит из нескольких полей, не связанных отношением порядка. Критерии поиска могут задаваться для любого непустого подмножества ключей

Реализация

- Индексы могут быть реализованы различными структурами
  - B* - деревья
  - B+ - деревья
  - B-деревья
  - хеш-таблицы.
  - . . .



Особенность индексов

- Затрагиваются только при выполнении запросов, требующих фильтрацию по ключу
- Подвергаются изменению при выполнении операций вставки/ редактирования/ удаления

Критерии оценки индексов

- Время поиска в индексе
- Сложность модификации
- Соотношение памяти, занимаемой данными и индексом

Производительность

Ни одно доброе дело не должно оставаться безнаказанным

- Увеличение количества индексов замедляет операции обновления
- Не кластерный индекс может требовать последовательные данные с разных страниц
- Для индексов требуется дополнительное место
- При удалении данных в индексе могут возникать «пустоты»

Стратегия индексного доступа

- Найти узел в индексе
- Прочитать данные
- Проблемы
  - переключение страниц: индекс/данные
  - повторное считывание страниц с данными

Индексы и версионность

- Неверсионные базы данных выполняют некоторые запросы (например, count), путем чтения индекса, без фактического чтения данных
- Единственный способ узнать в версионной базе, представляет ли индекс данные, видимые для данной транзакции, состоит в том, чтобы прочитать непосредственно саму запись или иметь вспомогательные структуры (страница видимости версии в PostgreSQL)

Firebird

- Firebird поддерживает только один тип индекса: b-дерево
- Используются только не кластерные индексы
- Индексы могут быть
  - уникальными или неуникальными (позволять дубликаты);
  - построенными на одиночном или составном ключе (но индекс при этом одномерный);
  - упорядоченными по возрастанию или убыванию

Размещение индекса и стратегия доступа (Firebird)

- Размещение
  - сохраняет записи на страницах данных
  - индексы хранятся на страницах индексов и содержат ссылку на местоположение записи
- Стратегия доступа
  - собирает месторасположение записей из индекса, соответствующих фильтру
  - строит битовую карту расположения записей
  - затем читает записи в порядке, в котором они физически хранятся.
  - может использовать несколько индексов для одной таблицы, соединяя битовые карты (И/ИЛИ).

Индексный доступ

- Проблема индексного доступа - это случайный ввод/вывод по отношению к страницам данных
- Сканирование индекса не является конвейерной операцией, а выполняется для всего диапазона поиска, включая полученные номера записей в специальную битовую карту
- Битовая карта – это разреженный битовый массив, где каждый бит соответствует конкретной записи и наличие единицы в нем является указанием для выборки данной записи
- битовая карта по определению отсортирована по физическим номерам записей.
- После окончания скана данный массив служит основой для последовательного доступа через идентификатор записи. Чтение из таблицы идет в физическим порядке расположения страниц, то есть каждая страница будет прочитана только один раз

Представление индексных ключей

- Firebird преобразовывает все ключи индекса в формат, пригодный для побайтового сравнения
- Для недвоичных данных определяются последовательности сопоставления (collation)
- При выполнении индексного поиска, Firebird преобразует входной параметр в тот же самый формат, что и хранимый ключ. Различия в скорости между индексами на полях с типами данных строка, число и дата нет.
- Ключи индекса Firebird всегда хранятся со сжатием

Составные индексы

- Важна последовательность столбцов
- Несколько индексов не должны дублировать друг друга
- При частичном фильтре индекс будет учтен только, если поля участвуют в порядке их следования в ключе

Синтаксис

CREATE [UNIQUE]
[ASC[ENDING] | DESC[ENDING]]
INDEX <имя индекса>
ON <таблица> (<столбец> [, <столбец>] ...);

При создании индекса вместо одного или нескольких столбцов также можно указать одно выражение, используя предложение COMPUTED BY

ALTER INDEX

- ALTER INDEX name {ACTIVE | INACTIVE};

Перевод индекса в активное/неактивное состояние, перестройка индекса

Невозможно перевести в неактивное состояние индекс участвующий в ограничении (primary key/ foreign key)

Ограничения

- Максимальная длина ключа индекса ограничена 1/4 размера страницы
- Для каждой таблицы максимально возможное количество индексов ограничено и зависит от размера страницы и количества столбцов в индексе

Селективность индекса

- Селективность (избирательность) индекса — это оценочное количество строк, которые могут быть выбраны при поиске по каждому значению индекса (версии тут не учитываются)
- Актуальность селективности индекса важна для выбора наиболее оптимального плана выполнения запросов оптимизатором

Сбор статистики

SET STATISTICS INDEX name;

Статистика индекса - это величина в пределах от 0 до 1, значение которой зависит от числа различных (неодинаковых) записей в индексе.

Оптимизатор использует статистику для определения эффективности применения того или иного индекса в запросе

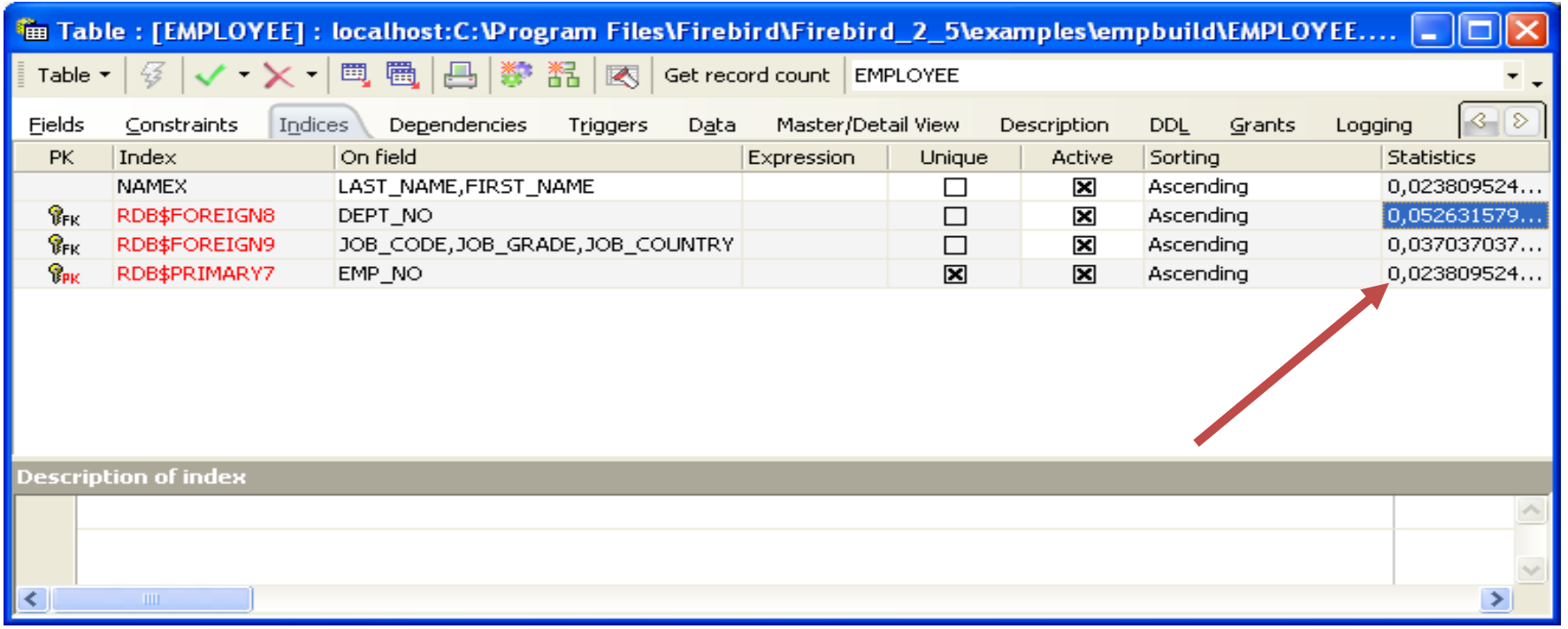
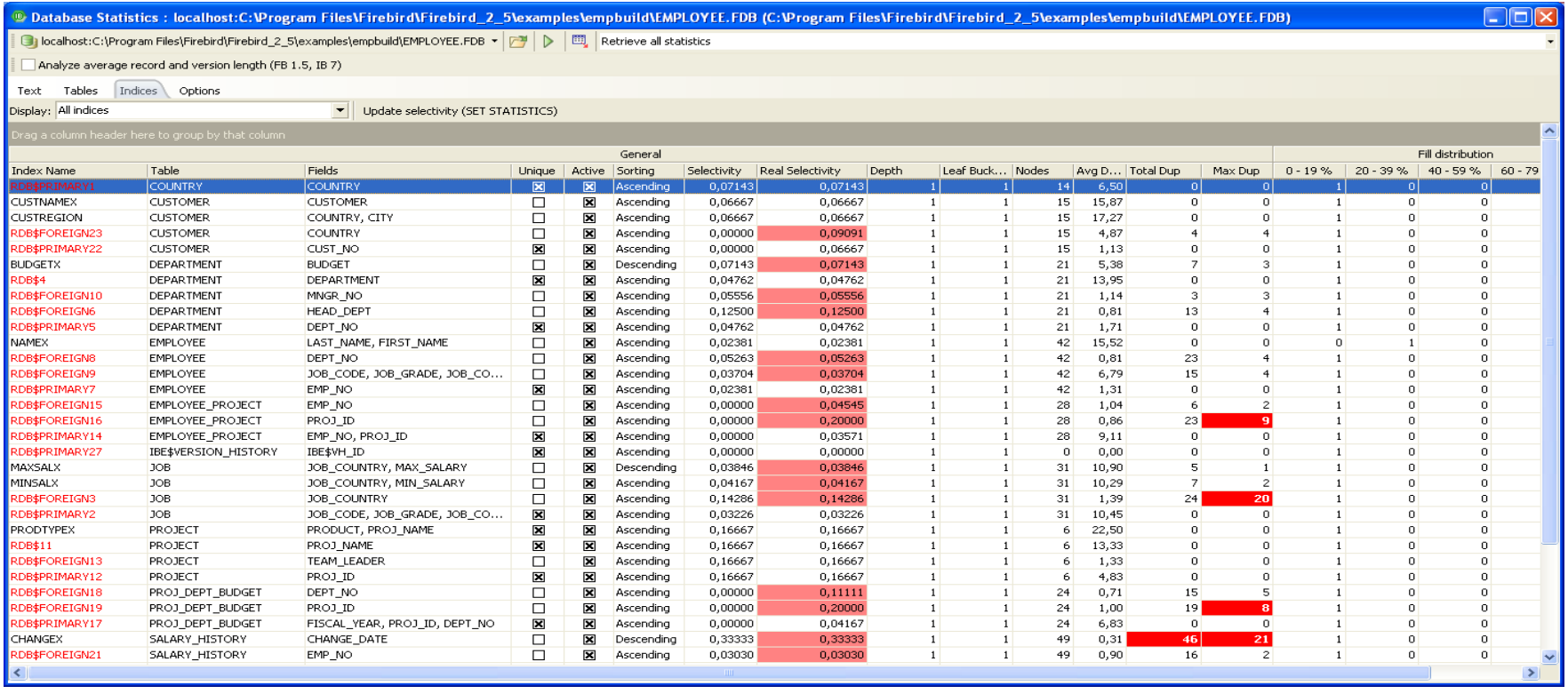


Table : [EMPLOYEE] : localhost:C:\Program Files\Firebird\Firebird_2_5\examples\empbuild\EMPLOYEE....
Table
Get record count
EMPLOYEE
Fields
Constraints
Indices
Dependencies
Triggers
Data
Master/Detail View
Description
DDL
Grants
Logging
PK
Index
On field
Expression
Unique
Active
Sorting
Statistics
NAMEX
LAST_NAME,FIRST_NAME
Ascending
0,023809524...
FK
RDB$FOREIGN8
DEPT_NO
Ascending
0,052631579...
FK
RDB$FOREIGN9
JOB_CODE,JOB_GRADE,JOB_COUNTRY
Ascending
0,037037037...
PK
RDB$PRIMARY7
EMP_NO
Ascending
0,023809524...
Description of index



					General									Fill distribution		
Index Name	Table	Fields	Unique	Active	Sorting	Selectivity	Real Selectivity	Depth	Leaf Buck...	Nodes	Avg D...	Total Dup	Max Dup	0 - 19 %	20 - 39 %	40 - 59 %	60 - 79
RDB$PRIMARY1	COUNTRY	COUNTRY	☒	☒	Ascending	0,07143	0,07143	1	1	14	6,50	0	0	1	0	0	
CUSTNAMEX	CUSTOMER	CUSTOMER	☐	☒	Ascending	0,06667	0,06667	1	1	15	15,87	0	0	1	0	0	
CUSTREGION	CUSTOMER	COUNTRY, CITY	☐	☒	Ascending	0,06667	0,06667	1	1	15	17,27	0	0	1	0	0	
RDB$FOREIGN23	CUSTOMER	COUNTRY	☐	☒	Ascending	0,00000	0,09091	1	1	15	4,87	4	4	1	0	0	
RDB$PRIMARY22	CUSTOMER	CUST_NO	☒	☒	Ascending	0,00000	0,06667	1	1	15	1,13	0	0	1	0	0	
BUDGETX	DEPARTMENT	BUDGET	☐	☒	Descending	0,07143	0,07143	1	1	21	5,38	7	3	1	0	0	
RDB$4	DEPARTMENT	DEPARTMENT	☒	☒	Ascending	0,04762	0,04762	1	1	21	13,95	0	0	1	0	0	
RDB$FOREIGN10	DEPARTMENT	MNGR_NO	☐	☒	Ascending	0,05556	0,05556	1	1	21	1,14	3	3	1	0	0	
RDB$FOREIGN6	DEPARTMENT	HEAD_DEPT	☐	☒	Ascending	0,12500	0,12500	1	1	21	0,81	13	4	1	0	0	
RDB$PRIMARY5	DEPARTMENT	DEPT_NO	☒	☒	Ascending	0,04762	0,04762	1	1	21	1,71	0	0	1	0	0	
NAMEX	EMPLOYEE	LAST_NAME, FIRST_NAME	☐	☒	Ascending	0,02381	0,02381	1	1	42	15,52	0	0	0	1	0	
RDB$FOREIGN8	EMPLOYEE	DEPT_NO	☐	☒	Ascending	0,05263	0,05263	1	1	42	0,81	23	4	1	0	0	
RDB$FOREIGN9	EMPLOYEE	JOB_CODE, JOB_GRADE, JOB_CO...	☐	☒	Ascending	0,03704	0,03704	1	1	42	6,79	15	4	1	0	0	
RDB$PRIMARY7	EMPLOYEE	EMP_NO	☒	☒	Ascending	0,02381	0,02381	1	1	42	1,31	0	0	1	0	0	
RDB$FOREIGN15	EMPLOYEE_PROJECT	EMP_NO	☐	☒	Ascending	0,00000	0,04545	1	1	28	1,04	6	2	1	0	0	
RDB$FOREIGN16	EMPLOYEE_PROJECT	PROJ_ID	☐	☒	Ascending	0,00000	0,20000	1	1	28	0,86	23	9	1	0	0	
RDB$PRIMARY14	EMPLOYEE_PROJECT	EMP_NO, PROJ_ID	☒	☒	Ascending	0,00000	0,03571	1	1	28	9,11	0	0	1	0	0	
RDB$PRIMARY27	IBE$VERSION_HISTORY	IBE$VH_ID	☒	☒	Ascending	0,00000	0,00000	1	1	0	0,00	0	0	1	0	0	
MAXSALX	JOB	JOB_COUNTRY, MAX_SALARY	☐	☒	Descending	0,03846	0,03846	1	1	31	10,90	5	1	1	0	0	
MINSALX	JOB	JOB_COUNTRY, MIN_SALARY	☐	☒	Ascending	0,04167	0,04167	1	1	31	10,29	7	2	1	0	0	
RDB$FOREIGN3	JOB	JOB_COUNTRY	☐	☒	Ascending	0,14286	0,14286	1	1	31	1,39	24	20	1	0	0	
RDB$PRIMARY2	JOB	JOB_CODE, JOB_GRADE, JOB_CO...	☒	☒	Ascending	0,03226	0,03226	1	1	31	10,45	0	0	1	0	0	
PRODTYPEX	PROJECT	PRODUCT, PROJ_NAME	☒	☒	Ascending	0,16667	0,16667	1	1	6	22,50	0	0	1	0	0	
RDB$11	PROJECT	PROJ_NAME	☒	☒	Ascending	0,16667	0,16667	1	1	6	13,33	0	0	1	0	0	
RDB$FOREIGN13	PROJECT	TEAM_LEADER	☐	☒	Ascending	0,16667	0,16667	1	1	6	1,33	0	0	1	0	0	
RDB$PRIMARY12	PROJECT	PROJ_ID	☒	☒	Ascending	0,16667	0,16667	1	1	6	4,83	0	0	1	0	0	
RDB$FOREIGN18	PROJ_DEPT_BUDGET	DEPT_NO	☐	☒	Ascending	0,00000	0,11111	1	1	24	0,71	15	5	1	0	0	
RDB$FOREIGN19	PROJ_DEPT_BUDGET	PROJ_ID	☐	☒	Ascending	0,00000	0,20000	1	1	24	1,00	19	8	1	0	0	
RDB$PRIMARY17	PROJ_DEPT_BUDGET	FISCAL_YEAR, PROJ_ID, DEPT_NO	☒	☒	Ascending	0,00000	0,04167	1	1	24	6,83	0	0	1	0	0	
CHANGEX	SALARY_HISTORY	CHANGE_DATE	☐	☒	Descending	0,33333	0,33333	1	1	49	0,31	46	21	1	0	0	
RDB$FOREIGN21	SALARY_HISTORY	EMP_NO	☐	☒	Ascending	0,03030	0,03030	1	1	49	0,90	16	2	1	0	0	

Оптимизация

Преимущества и недостатки

Преимущества

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

ПРОБЛЕМЫ

- Оптимальность не гарантируется
- Всегда выполняется (требует явного отключения в запросе)

Сравнение двух способов

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

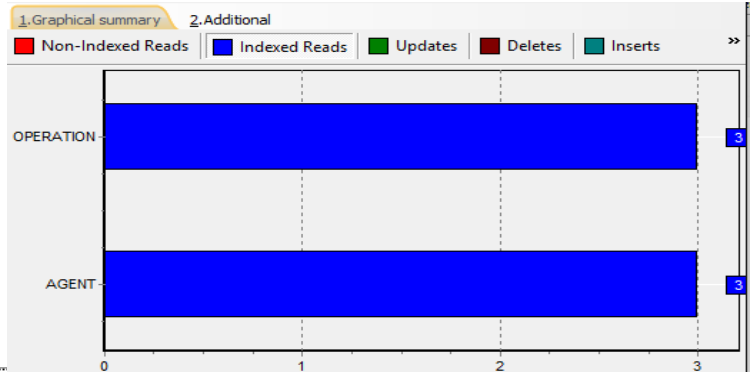
  не более 50 записей

Алгоритм

- Преобразование запроса в формальное 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

```
PLAN JOIN (O INDEX (FK_OP_1), A INDEX (PK_AGENT))
Select Expression
    -> Nested Loop Join (inner)
        -> Filter
            -> Table "OPERATION" as "O" Access By ID
                -> Bitmap
                    -> Index "FK_OP_1" Range Scan (full match)
        -> Filter
            -> Table "AGENT" as "A" Access By ID
                -> Bitmap
                    -> Index "PK_AGENT" Unique Scan
```



PLAN JOIN				
O INDEX (FK_OP_1)	OPERATION			FK
FK_OP_1	OPERATION	ID_GOODS	0,166666671633...	FK
A INDEX (PK_AGENT)	AGENT			PK
PK_AGENT	AGENT	ID_AG	0,142857149243...	PK

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

План запроса

PLAN <выражение>

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

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