**Рекомендации по выполнению работы**

*Индексом* называется объект базы данных, позволяющий существенно ускорить доступ к данным, за счет ускорения обработки команд, содержащих сравнительные критерии. Разумеется, что при увеличении скорости одного вида процедур, приходится жертвовать скоростью при других процедурах. Так и в случае с индексами: хранение дополнительной информации, позволяющей ускорить выборки, подразумевает осуществление дополнительных операций при добавлении, удалении и модификации данных. Как правило, индексы создаются для полей первичных и внешних ключей, а также для полей, часто указываемых в условиях поиска.

Индексы создаются командой CREATE INDEX:

CREATE [UNIQUE] INDEX индекс ON таблица [USING тип] (поле [класс] […])

Здесь индекс – имя создаваемого индекса, таблица – имя таблицы, для которой строится индекс, а поле – индексируемые поля. Необязательный параметр тип позволяет выбрать нужную реализацию индекса, а параметр класс описывает операторный класс, используемый для сортировки данных.

Данная команда также может содержать список из нескольких индексируемых полей, разделенных запятыми. В этом случае индекс строится для всех перечисленных полей. Составные индексы используются в PostgreSQL в двух случаях:

* когда при выполнении команды SQL **все** перечисленные в индексе поля участвуют в условии и объединены ключевым словом AND;
* когда при выполнении команды SQL в условии используется только первый (по порядку в составном индексе) атрибут.

Установка ограничений типов UNIQUE и PRIMARY KEY подразумевает автоматическое создание индексов, что позволяет быстро проверять существования того или иного значения первичного ключа (контроль уникальности) при вставки новой записи в таблицу.

В остальных случаях разработчик БД должен самостоятельно принять решение о создании тех или иных индексов. Для того, чтобы оценить эффективность использования тех или иных индексов, в СУБД PostgreSQL существуют инструменты для доступа к статистике использования каждого индекса. Для этого, если у Вас установлен PgAdmin, то Вам необходимо перейти к нужному индексу, вызвать контекстное меню и выбрать среди отчетов интересующий Ваш отчет.

*Ограничения* (constraint) представляют собой особый атрибут таблицы, который устанавливает критерии допустимости для содержимого ее полей. Соблюдение этих правил влияет на непротиворечивость данных в БД.

Ограничения задаются в секции CONSTRAINT при создании таблицы командой CREATE TABLE, а также могут быть добавлены командой ALTER TABLE.

Существуют шесть типов *ограничений полей*, задаваемых при помощи специальных ключевых слов:

* NOT NULL – поле не может содержать псевдозначение NULL;
* UNIQUE – поле не может содержать повторяющиеся записи. Следует однако учитывать, что в поле с ограничением уникальности к различных кортежах могут повторяться значения NULL, так как формально NULL не совпадает ни с каким другим значением;
* PRIMARY KEY – устанавливает для заданного поля ограничения NOT NULL и UNIQUE, а также создает для данного поля индекс;
* DEFAULT– задает значение по умолчанию, которое будет подставляться в случае если при вставке не указано иного;
* CHECK – налагает условие, проверяемое при вставке и обновлении записей в таблице;
* REFERENCES – ограничение, состоящее из нескольких секций, которые перечислены ниже:
  + REFERENCES таблица [(поле)]. Входные значения ограничиваемого поля сравниваются со значениями другого поля в заданной таблице. Если совпадения отсутствуют, команда INSERT или UPDATE завершается неудачей. Если параметр поле не указан, проверка выполняется по первичному ключу;
  + MATCH FULL. Секция MATCH указывает, разрешается ли смешивание значений NULL и обычных значений при вставке в таблицу, у которой внешний ключ ссылается на несколько полей. Конструкция MATCH FULL, запрещает втавку данных, у которых часть полей внешнего ключа содержит псевдозначение NULL;
  + ON DELETE операция. При выполнении команды DELETE для заданной таблицы с ограничиваемым полем выполняется одна из следующих операций: NO ACTION, RESTRIC, SET NULL, SET DEFAULT;
  + ON UPDATE операция. При выполнении команды UPDATE для заданной таблицы выполняется одна из операций, описанных выше.

В *ограничениях таблиц*, в отличие от ограничений полей, могут участвовать сразу несколько полей таблицы. Синтаксис ограничения таблицы следующий:

CONSTRAINT ограничение

{ UNIQUE (поле […]) | PRIMARY KEY (поле […]) | CHECK (условие) |

FOREIGN KEY (поле […])

REFERENCES таблица [(поле […])]

[MATCH FULL | MATCH PARTIAL]

[ON DELETE операция]

[ON UPDATE операция] }

Секция CONSTRAINT ограничение определяет необязательное имя. Разновидности ограничений таблиц следующие:

* PRIMARY KEY – в ограничении могут быть перечислены несколько полей, которые будут являться первичным ключом отношения;
* UNIQUE – данное ограничение для таблицы также как и предыдущее может содержать несколько полей, перечисленных через запятую. При этом любая комбинация полей, перечисленных за ключевым словом UNIQUE, принимает только уникальные значения;
* CHECK – используется аналогично одноименному ограничению полей;
* FOREIGN KEY (поле […]) REFERENCES таблица [ (поле […]) ] – определяет внешний ключ.

**Задание на лабораторную работу**

1. Создать схему istudents
2. Восстановить БД из бэкапа (разархивировать, восстановить в pgAdmin) в схему istudents
3. Замерить скорость выборки и модификации данных в таблице istudents.mark:
   1. Выполнить выборку данных (30 раз с различными условиями, фильтрация по полю **tmark\_fk** – условия могут циклически повторяться) – замерить время – получим ряд t1;
   2. Выполнить вставку данных (30 раз различные данные) – замерить время – получим ряд t2;
   3. Изменения t1 и t2 представить на графиках – объяснить их форму;
   4. Для полученных рядов (t1, t2) посчитать основные статистические показатели – мат. ожидание, дисперсию, СКО и оценить полученные результаты.
4. Создать в таблице istudents.mark первичный ключ (id) и создать индекс для поля tmark\_fk
5. Повторить произведенные в п.3 замеры, построить графики, произвести статистическую оценку полученных данных. Как изменились временные показатели?
6. Написать приложение, которое для каждого значения атрибута «год действия плана» (атрибут plyear в таблице mark) считает количество положительных оценок в таблице (положительными считаются оценки у которых (bal>40). При этом должен быть реализован следующий алгоритм:
   1. Выбор всех возможных значений атрибута plyear из таблицы (например, с помощью запроса SELECT DISTINCT plyear FROM istudents.mark);
   2. Организовать цикл по результатам запроса в п.а и для каждого значения plyear независимо считать количество положительных оценок (например, с помощью запроса SELECT count(id) FROM istudents.mark WHERE bal > 40 and plyear = 1);
   3. Вывести полученные данные (по каждому значению plyear) пользователю, вывести время, за которое значения были получены.
7. Создать индекс по полю plyear и повторить действия, описанные в п.6 – сравнить полученные временные показатели и результаты;
8. Написать приложение, которое для каждого значения атрибута «год действия плана» (атрибут plyear в таблице mark) считает количество положительных оценок в таблице (положительными считаются оценки у которых (bal>40). При этом должен быть реализован следующий алгоритм:
   1. Выполнить запрос с группировкой по полю plyear для расчета значений по каждой группе (SELECT plyear, count(id) FROM istudents.mark WHERE bal>40 GROUP BY plyear);
   2. Полученные данные и время их получения представить пользователю, сравнить эти показатели с результатами, полученными в п.6, 7;
9. Создать первичный ключ в таблице istudents.studplan.
10. Замерить время, требуемое для выполнение внутреннего объединения двух таблиц (istudents.mark и istudents.studplan) по условию объединения: mark INNER JOIN studplan on mark.studplan\_fk = studplan.id. При этом выбирать **не более 100 записей** из результатов объединения (для этого использовать LIMIT 100). Выполнить запрос 30 раз и каждый раз выбирать разные 100 записей – для этого использовать OFFSET (например, так SELECT \* FROM mark INNER JOIN studplan on mark.studplan\_fk = studplan.id LIMIT 100 OFFSET 900).
11. Выполнить запросы из п.10 – 30 раз, замерить время и оценить полученные результаты статистически.
12. Создать внешний ключ в таблице istudents.mark (studplan\_fk -> studplan.id) и для внешнего ключа обязательно **создать индекс**.
13. Повторить запросы из п.10 – 30 раз, замерить время и оценить полученные результаты статистически.
14. Сравнить результаты из п.11 и п.13: отличаются или нет и почему.