Инновационный Евразийский университет

Кафедра «Автоматизированные системы
обработки информации и управления»

Слайд-лекция

по дисциплине
«Системы управления базами данных»
на тему: Создание индексов
для студентов специальности 5В070400
«Вычислительная техника и программное обеспечение»

Разработала: Ст. преподаватель Третьякова Т.И.

Индексы

Индексы – это внутренний метод организации данных из таблиц, благодаря которому их выборка выполняется оптимальным образом.
В отсутствии индекса поиск данных должен выполнятся путем сканирования таблицы - считывание всех её данных и сравнение с условиями запроса.
С помощью индексов мы значительно сократим количество операций ввода – вывода, ускорим процесс доступа к данным и освободим системные ресурсы.

При создании индексов требуется проанализировать ситуацию выборки данных, которые соответствуют нашей модели СУБД.
Нужно учесть не только операции выборки, но также вставки и обновления.
Приложение, которое занимается вставкой и обновлением информации, и приложение, которое только выбирают информацию – это совершенно разные вещи.
Основная часть работы приходится на обновление индексов.
Поэтому если выполняется много операций вставки и обновления, имеет смысл ограничить количество индексов таблицы.
В целом для таких таблиц рекомендуется использовать не более 5 индексов.

Рекомендации по созданию индексов

Можно создать все необходимые индексы, чтобы повысить производительность запроса индекс не выгодно использовать для поиска большого количества данных.

! Запросы, которые возвращают более 20% записей таблицы, лучше выполнять путем сканирования таблицы.

С помощью индексов наилучшим образом выполняются:

1) запросы, содержащие точный критерий поиска;
2) запросы, удовлетворяющие диапазону значений;
3) запросы, возвращающие данные в определенном порядке –

1. Столбцы, которые указаны в директивах ORDER BY и GROUP BY очень полезно использовать в индексе.
2. Индексы следует использовать в таблицах, для которых редко выполняются операции добавления, удаления, обновления.

! Не создают индексы для небольших таблиц.
Намного выгоднее сканировать небольшие таблицы.

Создание индекса с помощью команды

CREATE [UNIQUE] [CLUSTERED] [NONCLUSTERED]

INDEX имя - индекса ON имя_таблицы (поле1,поле2,…)

[WITH

[IGNORE_DUP_KEY,]

[DROP_EXISTING]]

Опция UNIQUE позволяет создать уникальный индекс.

Правила:

1. В столбце не должно быть повторяющихся значений.
2. Если перед созданием индекса в столбце были повторяющиеся значения, их следует удалить.
3. При использовании опции **IGNORE_DUP_KEY** появляется возможность создать индекс даже в том случае, если в индексированном столбце были повторяющиеся значения.

Метод перестроения индекса

1. Первый метод перестроения индекса –

удаление устаревшего индекса и создание нового.

2. Другой метод заключается в использовании параметра DROP_EXISTING в команде CREATE INDEX.

Второй метод имеет некоторые преимущества при перестроении кластерных индексов. Не кластерные индексы, созданные на кластерной таблице, основаны на кластерных ключах.

При удалении кластерного индекса, не кластерные должны быть перестроены, так как первого больше не существует.

! Если после этого будет построен кластерный индекс, не кластерные индексы необходимо перестроить еще раз.

Таким образом не кластерные индексы перестраивались дважды.

При применении параметра DROP_EXISTING не кластерные индексы придется перестраивать только раз.

В некоторых ситуациях сканирование таблицы более эффективно,
чем поиск по индексу. Например, если мы знаем, что при сканировании индекса будет найдено более 20 % записей таблицы.
Мы также вправе выбрать конкретный индекс, если считаем,
что поиск средствами указанного
индекса будет наиболее эффективным.

В качестве параметра указания индекса выступает либо имя индекса,
либо идентификатор индекса.

На кластерной таблице:

INDEX (0) – сканирование таблицы;

INDEX (1) – явное указание на использование кластерного индекса

На не кластерной таблице можно указать либо имя, либо номер индекса.

Пример 1. Указание индекса с помощью его имени:

SELECT*FROM Tovar (INDEX(Name))

WHERE Name = ‘пальто’ **AND** Firma = ‘Сириус’

Пример 2.Использование первого альтернативного индекса:

SELECT*FROM Tovar (INDEX=2)

WHERE Name = ‘пальто’ **AND** Firma = ‘Сириус’

Пример 3. Использование кластерного индекса:

а) **SELECT*FROM** Tovar **WHERE** Name = ‘пальто’ **AND** Firma = ‘Сириус’

б) **SELECT*FROM** Tovar (INDEX=1)

WHERE Name = ‘пальто’ **AND** Firma = ‘Сириус’

Пример 4. Сканирование таблицы

SELECT*FROM Tovar (INDEX=0)

WHERE Name = ‘пальто’ **AND** Firma = ‘Сириус’

Отображение информации об индексе и удаление индексов

Sp_helpindex имя_таблицы – возвращает первый восемь индексов таблицы;
Sp_statistics имя_таблицы – возвращает всю необходимую информацию об индексах;
Sp_statistics tovar, @ is_unique = ‘Y’ – вернет только уникальные индексы.

Опция **accuracy** (точность).
Чтобы статистика была точной и максимально полной, у становим значение этого параметра равным E.
Sp_statistics tovar, @ accuracy =’E’
Для удаления индексов используется команда DPOP INDEX имя_таблицы. имя_индекса

Определение ключей

Между ключами и индексами в SQL Server имеется некоторая разница. В SQL Server ключи можно определить в таблицах и использовать для поддержания целостности данных.

Первичный ключ – это столбец, содержащий уникальные значения. В таблице может быть только один первичный ключ, в то время как уникальных индексов может быть много.
Кроме того, в первичном ключе не может быть элементов со значением NULL, в то время как в уникальном индексе может быть такой элемент.

Внешние ключи – это столбцы таблицы, которые соответствуют первичным ключам других таблиц.
Соотношение между первичным и внешним ключом определяет область допустимых значений для внешнего ключа.
Эта область значений является подмножеством значений соответствующего первичного ключа.
При определении внешних ключей индексы в таблице не создаются.

Первичные ключи

При создании таблицы с помощью команды:
можно сразу создать первичный ключ, указав поле первичного ключа перед опцией PRIMARY KEY:

CRETE TABLE Kient

(fam CHAR (20) NOT NULL,

adr CHAR (20) NOT NULL,

tel CHAR (8) NULL,

cod INT PRIMARY KEY)

Можно присвоить имя ограничению

CRETE TABLE Kient

(fam CHAR (20) NOT NULL,

adr CHAR (20) NOT NULL,

tel CHAR (8) NULL,

cod INT CONSTRAINT PK_cod **PRIMARY KEY(cod**))

PRIMARY KEY

Для добавления первичного ключа **PRIMARY KEY**
в существующую таблицу используется следующая команда:

ALTER TABLE Kient
ADD CONSTRAINT PK_cod **PRIMARY KEY** (cod)

С помощью ниже приведенной команды можно удалить первичный ключ **PRIMARY KEY :**

ALTER TABLE Kient
DROP CONSTRAINT PK_cod

Внешние ключи

1. Ограничение FOREIGN KEY определяет внешний ключ, обеспечивающий связь между главной и подчиненной таблицами.
2. Внешний ключ в подчиненной таблице ссылается на уникальный ключ в главной таблице.
3. Значение подчиненной таблицы сравниваются со значениями соответствующего поля главной таблицы, если они не совпадают, то запись в подчиненную таблицу не добавляется.
4. Ограничение внешнего ключа проверяется также при удалении записи из главной таблицы, если в подчиненной существуют записи, ссылающиеся на записи главной таблицы.
5. Внешний ключ может ссылаться только на поля, имеющие ограничения PRIMARY KEY или UNIQUE в главной таблице.

Примеры программ

CRETE TABLE Tovar
(cod INT,
name char (20),
firma char (20),
price money,
quant INT,
CONSTRAINT FK_cod FOREIGN KEY (cod) **REFERENCES** Klient (cod))

Опция REFERENCES указывает главную таблицу и поле,
по которому устанавливается связь.
Для того, чтобы изменить ограничения **FOREIGN KEY**,
необходимо удалить старое, а затем создать новое.

ALTER TABLE Tovar
DROP CONSTRAINT FK_cod
ALTER TABLE Tovar
ADD CONSTRAINT FK_cod **FOREIGN KEY** (cod) **REFERENCES** Klient (cod))

Примеры программ

CRETE TABLE Tovar
(cod INT,
name char (20),
firma char (20),
price money,
quant INT,
CONSTRAINT FK_cod **FOREIGN KEY** (cod) **REFERENCES** Klient (cod))

Опция **REFERENCES** указывает главную таблицу и поле,
по которому устанавливается связь.
Для того, чтобы изменить ограничения **FOREIGN KEY**,
необходимо удалить старое, а затем создать новое.

ALTER TABLE Tovar
DROP CONSTRAINT FK_cod
ALTER TABLE Tovar
ADD CONSTRAINT FK_cod **FOREIGN KEY** (cod) **REFERENCES** Klient (cod))

Дополнительныевозможности:

1. Можно **при удалении и обновлении записей** в главной таблице выполнить каскадное удалении и обновление соответствующих записей из подчиненной таблицы.

```
ALTER TABLE Tovar
ADD CONSTRAINT FK_cod FOREIGN KEY (cod)
REFERENSES Klient (cod))
ON DELETE CASCADE
ON UPDATE CASCADE
```

2. При добавление внешнего ключа происходит проверка значения поля cod таблицы *tovar* на наличие такого значения в таблице *client*.
Чтобы избежать такой проверки при создании ограничения, необходимо включить опцию WITH NOCHECK в команду ALTER TABLE.

```
ALTER TABLE Tovar
WITH NOCHECK
ADD CONSTRAINT FK_cod FOREIGN KEY (cod) REFERENCES Klient (cod))
```

! Это позволит добавить ограничение вне зависимости от имеющихся значений.

3 Можно в процессе работы включать или отключать проверку ограничений FOREIGN KEY. Чтобы вставить запись, не удовлетворяющую ограничению, можно временно отключить его, выполнить вставку, а затем снова активизировать ограничение.
Ключевое слово NOCHECK означает, что ограничение следует игнорировать, а ключевое слово CHECK – что его нужно учитывать.

```
ALTER TABLE Tovar
NOCHECK CONSTRAINT FK_cod
```

Сделать операторы вставки и включить ограничение:

```
ALTER TABLE Tovar
CHECK CONSTRAINT FK_cod
```

4 Создав ограничение можно контролировать допустимые значения поля.

Ограничения указываются в опции CONSTRAINT CHECK.

```
CRETE TABLE Tovar
(cod INT,
name char (20),
firma char (20),
price money,
quant INT,
CONSTRAINT FK_cod FOREIGN KEY (cod) REFERENSES Klient (cod))
CONSTRAINT CK_price CHECK ((price>=3000) AND (price<=8000))
```

Ограничения

Пример 1 Изменить диапазон допустимых значений поля следует, удалив ограничение и создав новое.

```
ALTER TABLE Tovar
DROP CONSTRAINT CK_price
```

Создаем новое ограничение

```
ALTER TABLE Tovar
ADD CONSTRAIN CK_price CHECK (price>=5000 AND price<=10000)
```

Если имеющиеся записи не надо проверять
по условию ограничения, то включается опция WITH NOCHECK.

```
ALTER TABLE Tovar
WITH NOCHECK ADD CONSTRAINT
CK_price CHECK ((price>=5000 AND price<=10000))
```

Пример 2. Ограничение отключается, а потом включается:

```
ALTER TABLE Tovar
NOCHECK CONSTRAINT CK_price
```

Выполняем операторы вставки и снова включаем ограничение.

```
ALTER TABLE Tovar
CHECK CONSTRAINT CK_price
```