Indices

Indices: Tabelas especiais de consulta, indexadas de forma a aceder mais rapidamente a dados.

Um indice é utilizado para aumentar a velociade de pesquisa numa base de dados. Um indice pode ser usado para efetivamente encontrar todas as entradas que correspondam a uma coluna numa query e depois apenas trabalhar nesse subset da tabela para encontrar resultados exatos. Se não tens indices em nenhuma coluna na WHERE clause, o SQL sever vai ter de procurar em toda a tabela e procura por cada entrada para ver se é o que procura, o que se pode tornar lento para operações em tabelas grandes.
O indice pode ser também única, o que significa que não pode haver valores duplicados nessa coluna (tipo chave primária).

(Os indices primarios sao os aplicados sobre as chaves, todos os outros sao secundarios)

Os indices podem ser simples ou compostos.

Uma tabela ou view pode ter os seguintes tipos de indices:

- Clustered

Os indices clustered ordenam e armazenam os registos de dados na tabela ou view baseados nos seus valores chave. Estes são as colunas incluidas na definição do indice. Apenas pode haver um indice clustered por tabela, porque os registos apenas podem ser ordenados por uma ordem.

A unica vez que os registos numa tabela sao ordenados é quando a tabela contem um indice clustered. Quando uma tabela tem um indice clustered, a tabela chama-se clustered. Se a tabela não possui indice clustered, então os registos são armazenados de forma não ordenada numa estrutura denominada heap.

- Nonclustered

Resposta 1:

With a clustered index the rows are stored physically on the disk in the same order as the index. There can therefore be only one clustered index.
With a non clustered index there is a second list that has pointers to the physical rows. You can have many non clustered indexes, although each new index will increase the time it takes to write new records.
It is generally faster to read from a clustered index if you want to get back all the columns. You do not have to go first to the index and then to the table.
Writing to a table with a clustered index can be slower, if there is a need to rearrange the data.

Resposta 2:

A clustered index means you are telling the database to store close values actually close to one another on the disk. This has the benefit of rapid scan / retrieval of records falling into some range of clustered index values.

For example, you have two tables, Customer and Order:

```
Customer ----------
ID
Name
Address

Order ----------
ID
CustomerID
Price
```

If you wish to quickly retrieve all orders of one particular customer, you may wish to create a clustered index on the "CustomerID" column of the Order table. This way the records with the same CustomerID will be physically stored close to each other on disk (clustered) which speeds up their retrieval.
P.S. The index on CustomerID will obviously be not unique, so you either need to add a second field to "uniquify" the index or let the database handle that for you but that's another story.
Regarding multiple indexes. You can have only one clustered index per table because this defines how the data is physically arranged. If you wish an analogy, imagine a big room with many tables in it. You can either put these tables to form several rows or pull them all together to form a big conference table, but not both ways at the same time. A table can have other indexes, they will then point to the entries in the clustered index which in its turn will finally say where to find the actual data.

You should use extreme care when picking a clustering key - it should be:
- narrow (4 bytes ideal)

- unique (it's the "row pointer" after all. If you don't make it unique SQL Server will do it for you in the background, costing you a couple of bytes for each entry times the number of rows and the number of nonclustered indices you have - this can be very costly!)

- static (never change - if possible)

- ideally ever-increasing so you won't end up with horrible index fragmentation (a GUID is the total opposite of a good clustering key - for that particular reason)

- it should be non-nullable and ideally also fixed width - a varchar(250) makes a very poor clustering key

Para determinar a utilidade de um indice podemos utilizar as seguintes estrategias:
- examinar a clausula WHERE ou a clausula JOIN definidas nas intruçoes de interrogaçao. Cada atributo incluido em qualquer uma dessas clausulas é um possivel candidato a um indice.
- experimentar o novo indice para verificar o seu efeito no desempenho do sistema na satisfaçao das interrogaçoes.
- considerar o nmr de indices ja criados no sistema sobre a tabela em questão, é conveniente evitar um grande nmr de indices sobre uma unica tabela.
- examinar as definiçoes dos indices ja criados sobre uma tabela. deve-se evitar definiçoes de indices sobrepostas - index overlapping - que contenham atributos partilhados.
- examinar o nmr de valores de dados unicos num atributo e comparar esse numero com o nmr de registos da tabela. o resultado é a seletividade desse atributo, que é um factor que pode ajudar na decisão se esse atributo é um candidato a indice e, se sim, que tipo de indice deverá ser definido.

```
CREATE INDEX B ON T1(B);
CREATE UNIQUE INDEX C ON T1(C);

CREATE INDEX PIndex
ON Persons (LastName, FirstName)

ALTER TABLE T1 DROP INDEX B, DROP INDEX C;

DROP INDEX B ON T1;
DROP INDEX C ON T1;
SHOW INDEX FROM table_name.G
```