Indexes

Best Practices (I)

S5

Indexes

- An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view.

- Great indexing → application fast & nimble
- Poor indexing → slows entire SQL Server

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]
       INDEX index_name

ON <object> ( column [ ASC | DESC ] [ ,...n ] )

[ INCLUDE ( column_name [ ,...n ] ) ]

[ WHERE <filter_predicate> ]

[ WITH ( <relational_index_option> [ ,...n ] ) ]
```

Index Characteristics

- Clustered versus nonclustered
- Unique versus nonunique
- Single column versus multicolumn
- Ascending or descending order on the columns in the index
- Full-table versus filtered for nonclustered indexes

Clustered vs NonClustered Index

- Clustered index: sorts and stores data rows in a table, based on search key values

```
CREATE CLUSTERED INDEX Index_Name
                    ON Schema.TableName(Column);
```

- NonClustered index: key values and a pointer to data in the heap/clustered index

```
CREATE INDEX Index_Name
                    ON Schema.TableName(Column);
```

Clustered vs NonClustered Index

- The data pages of a clustered index will always include *all columns* in the table

- There is only one clustered index per table.
- SQL Server supports up to 999 nonclustered indexes per table.
- An index key – clustered or nonclustered – can be a maximum of 16 columns and 900 bytes.

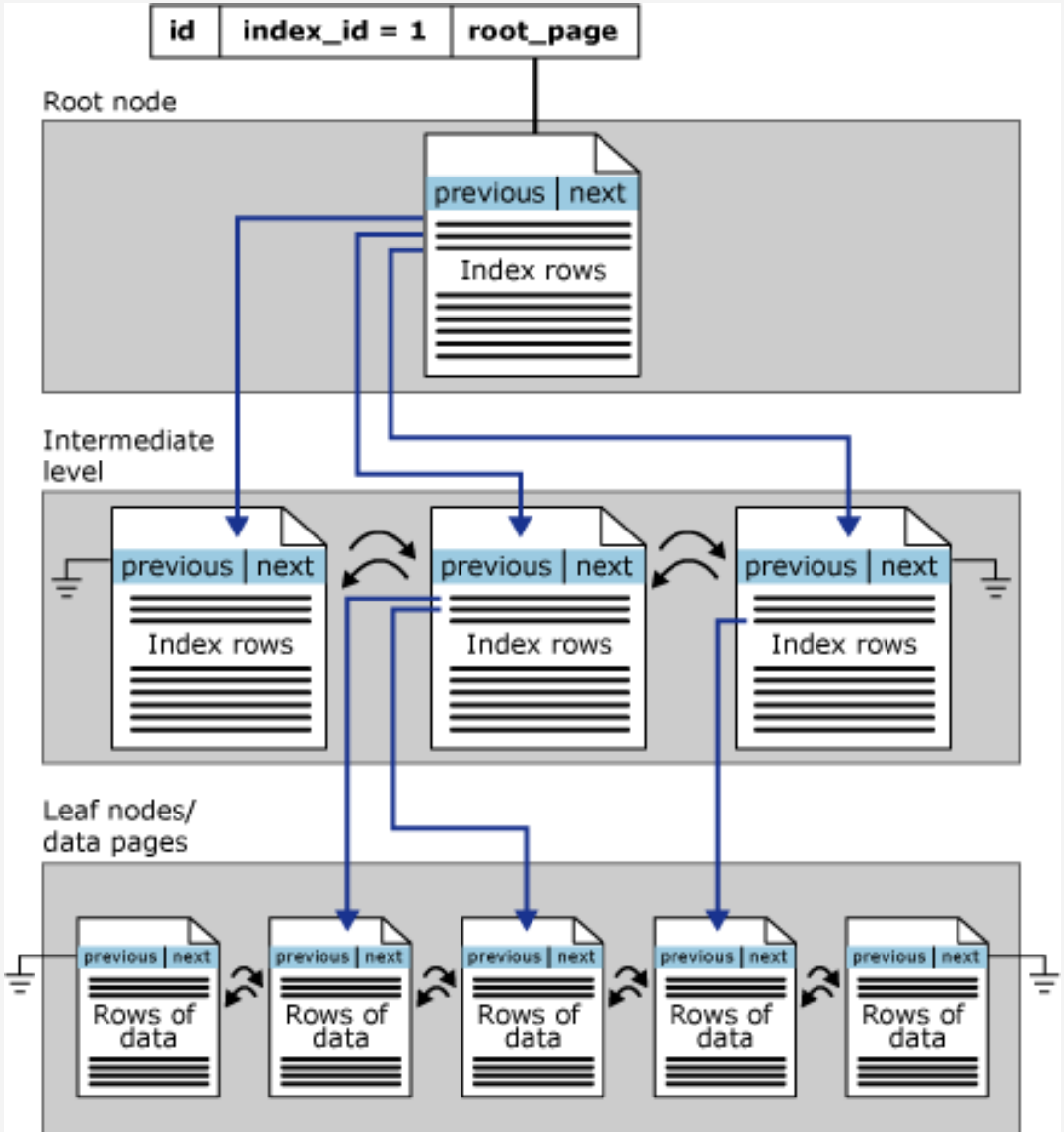
Clustered Index

- Can be used for frequently used queries
- Provide a high degree of uniqueness
- Can be used in range queries

- Clustered indexes are not a good choice for the following attributes:
  - Columns that undergo frequent changes
  - Wide keys

Clustered Index

- indexes are organized as B-trees.



Clustered vs NonClustered Index

- When you create a primary key on a table
  - + if a clustered index is not defined
  - + a nonclustered index is not specified
  - → a unique clustered index is created

- If all columns returned in a query are in the index: covering index

Key vs NonKey Index Columns

- Key columns: the columns specified to create an index.
- Nonkey columns: columns added to the INCLUDE clause of a nonclustered index.

```
CREATE INDEX Index_Name
ON Schema.TableName(Column)
INCLUDE (ColumnA, ColumnB);
```

Key vs NonKey Index Columns

- Benefits to using non-key columns
  - Columns can be accessed with an index scan.
  - Data types not allowed in key columns are allowed in nonkey columns (including text, ntext, and image).
  - Included columns do not count against the 900 byte index key limit enforced by SQL Server.

Index Design Tasks

- Understand the characteristics of the database
  - OLTP - On-line Transaction Processing
  - OLAP - On-line Analytical Processing
- Understand the characteristics of the most frequently used queries
- Understand the characteristics of the columns used in the queries
- Determine the optimal storage location for the index.

General Index Design Guidelines

- Database considerations
  - Too many indexes on a table affect the performance of INSERT, UPDATE, DELETE, MERGE statements
  - Indexing small tables may not be optimal
  - Indexes on views are useful when views contain aggregations and/or table joins

General Index Design Guidelines

- Query considerations
  - Create nonclustered indexes for columns frequently used in WHEREs and JOINs
  - Covering indexes can improve query performance
  - Write queries that insert or modify as many rows as possible in a single statement

General Index Design Guidelines

- Column Considerations
  - Keep the length of index key short for clustered indexes
  - Clustered indexes are better on unique/nonnull cols
  - Columns of **ntext**, **text**, **image**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)** cannot be specified as index key columns
  - Examine column uniqueness
  - Examine data distribution in column (avoid indexes on columns with few unique values) – use filtered indexes
  - Consider the order of the columns for mutliple index. Columns used in an equal to (=), greater than (>), less than (<), or BETWEEN search condition should be placed first. Additional columns should be ordered from the most distinct to the least distinct.
  - Consider indexing computed columns.

Unique Indexes

- A unique index guarantees that the index key contains no duplicate values
- Specifying a unique index makes sense only when key columns are unique
- Uniqueness – helpful information for query optimizer

Filtered Indexes

Filtered Index: an optimized nonclustered index, especially suited to cover queries that select from a well-defined subset of data

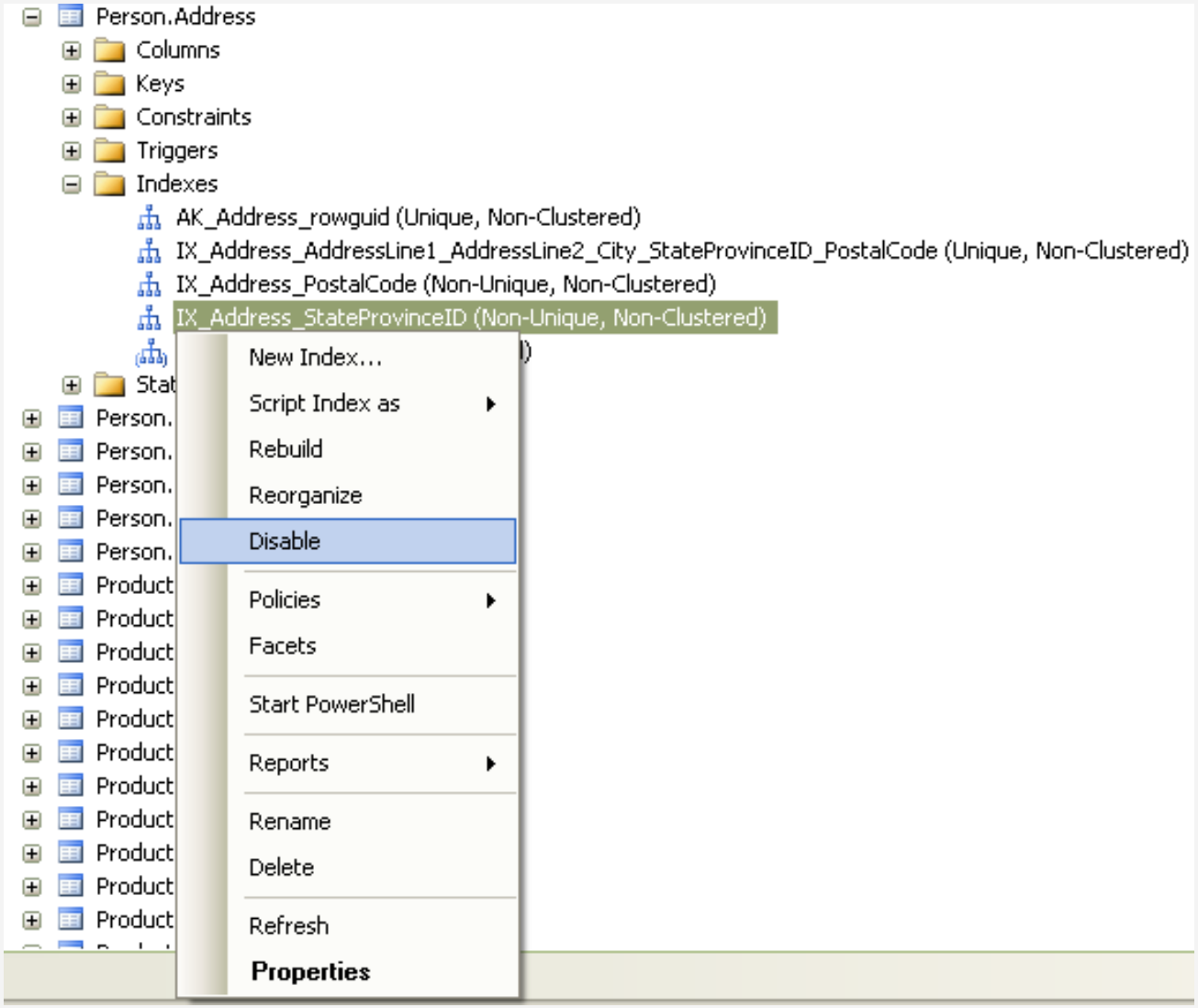
```
CREATE NONCLUSTERED INDEX FI_EndDate ON
Products (ProductID,  EndDate)
 WHERE EndDate IS NOT NULL ;
GO
```

- Improved query performance
- Reduced index maintenance costs
- Reduced index storage costs

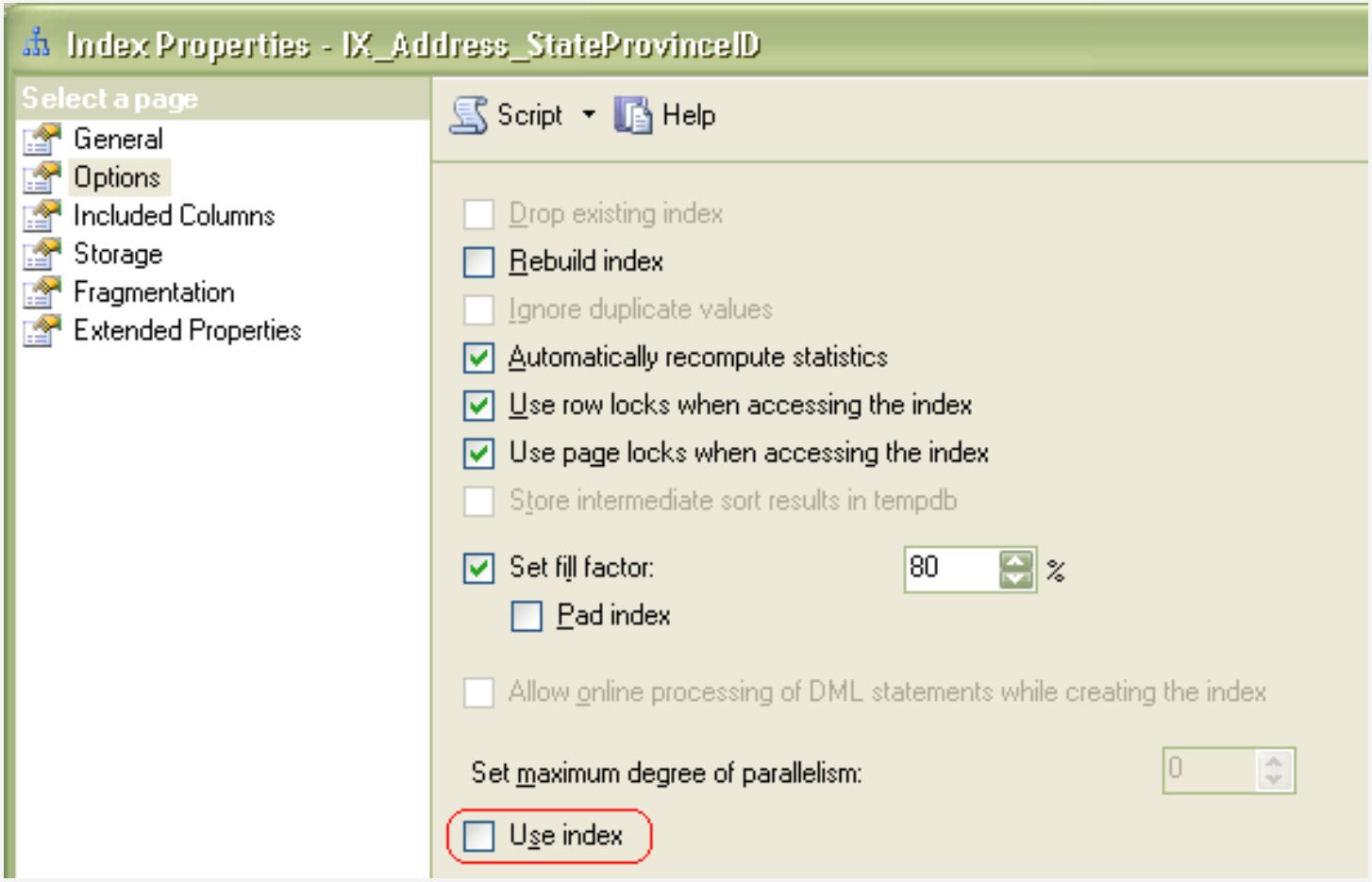
Disable Indexes

```
ALTER INDEX IX_Address_StateProvinceID
        ON Person.Address DISABLE
```

Disable Indexes



Disable Indexes



Enable Indexes

```
ALTER INDEX IX_Address_StateProvinceID
        ON Person.Address REBUILD
```

Indexes for Deletes

At DELETE:

- SQL Server will check for dependent rows by examining all foreign keys
- It will then check any related tables for data.
  - If there is an index, SQL Server will use that index to check for related data
  - If there isn't an index, though, SQL Server will have to scan the table for data.
- Deletes could be very slow if there is no index defined for foreign keys