Databases

Seminar 5

Indexes in SQL Server (I)

- index
  - structure stored on the disk, associated with a table or view
  - optimizes retrieval operations on the table / view
- great indexing

  => fast applications
- poor indexing

  => can slow down the DBMS

- syntax

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

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

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

[ WHERE <filter_predicate> ]

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

- index characteristics
    - clustered versus non-clustered
    - unique versus non-unique
    - search key - single-column versus multicolumn
    - key columns versus non-key columns
    - columns in the index - ascending / descending order
    - non-clustered indexes - full-table versus filtered

- clustered versus non-clustered index
  - clustered index
    - the data rows in the table are kept sorted, based on the values of the search key

```
CREATE CLUSTERED INDEX Index_Name
    ON Schema_Name.Table_Name(Column)
```

  - non-clustered index
    - contains key values and pointers to the data in the table (heap / clustered index)

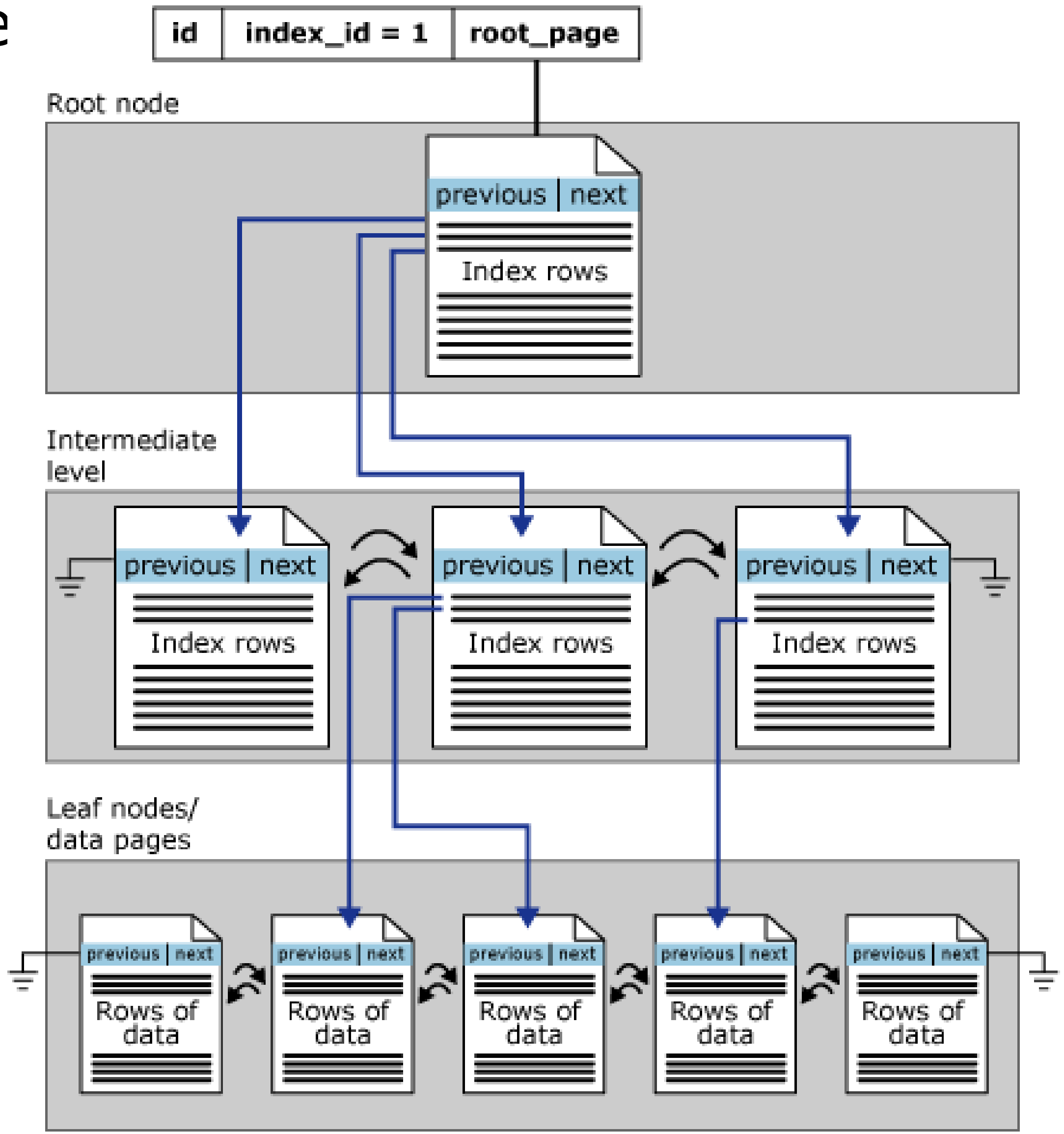
```
CREATE INDEX Index_Name
    ON Schema_Name.Table_Name(Column)
```

- clustered versus non-clustered index
    - data pages in a clustered index always include *all the columns* in the table
    - SQL Server:
        - at most one clustered index per table
        - at most 999 non-clustered indexes per table
    - an index key (clustered / non-clustered)
        - at most 16 columns, 900 bytes*

* version differences

- clustered versus non-clustered index
    - clustered index
        - can be used for frequently executed queries
        - high degree of uniqueness
        - can be used in range queries
        - columns that are part of the search key:
            - shouldn't be frequently changed
            - should be narrow

- clustered versus non-clustered index
  - clustered index
    - organized as a B+ tree



- clustered versus non-clustered index
  - when creating a primary key on a table:
    - if a clustered index is not defined
    - a non-clustered index is not specified

    => a unique clustered index is created on the fields of the primary key

- unique indexes
  - such an index guarantees that the search key contains no duplicate values
  - specifying a unique index makes sense only when there are no entries with identical values in the key columns
  - uniqueness – useful information for the query optimizer

- key versus non-key index columns
  - key columns
    - the columns of the search key
  - non-key columns
    - columns specified in the INCLUDE clause when creating a non-clustered index

```
CREATE INDEX Index_Name
    ON Schema_Name.Table_Name(Column)
    INCLUDE (ColumnA, ColumnB, ColumnC)
```

  - covering index
    - contains all the columns that are necessary in a query

- key versus non-key index columns
  - non-key columns - benefits
    - columns can be accessed from the index
    - data types that are not allowed in key columns can be used in non-key columns
    - non-key columns are not taken into account when computing the size of the key

- filtered indexes
  - optimized non-clustered indexes
  - can be used by queries that select from a certain subset of data

```
CREATE NONCLUSTERED INDEX IDX_eid_pid_f_od
     ON Orders(EmpId, ProdId)
     WHERE OrderDate IS NOT NULL
```

  - benefits
    - better query performance
    - reduced index:
      - maintenance cost
      - storage cost

- index design
    - analyze the characteristics of the:
        - database
            - Online Transaction Processing (OLTP)
            - Online Analytical Processing (OLAP)
        - most frequently executed queries
        - columns used in queries

- index design - guidelines
  - database-related aspects
    - the presence of many indexes on a table deteriorates the performance of INSERT, UPDATE, DELETE, MERGE statements
    - indexing small tables is often useless

- index design - guidelines
  - query-related aspects
    - non-clustered indexes should be created on columns that are often used in WHERE and JOIN
    - covering indexes can significantly improve the performance of queries
    - as many records as possible should be changed in a single statement

- index design - guidelines
  - column-related aspects
    - length of the index key - as short as possible for clustered indexes
    - clustered indexes - better on unique / non-null columns
    - types ntext, text, image, varchar(max), nvarchar(max), varbinary(max) cannot be used for search key fields
    - column uniqueness
    - data distribution in the column
      - avoid indexes on columns with a small number of distinct values
    - consider creating filtered indexes for columns with well-defined subsets
    - order of columns in multicolumn indexes
      - first positions - columns in equality (=), inequality (>, <, BETWEEN) conditions
      - the rest of the columns should be ordered by level of distinctness

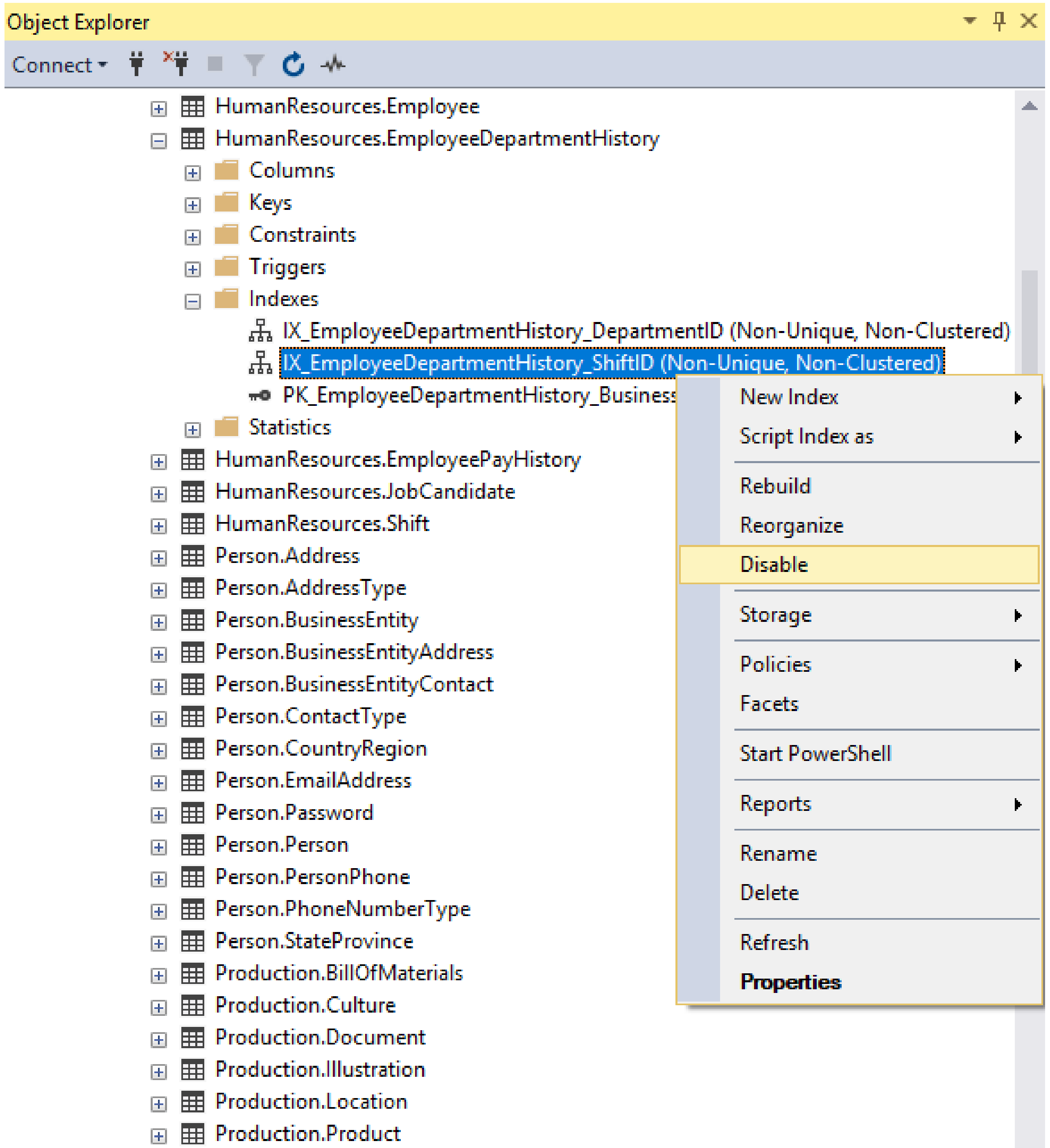
- index design - guidelines
  - column-related aspects
    - consider indexing computed columns

- indexes for deletes
    - when executing a delete statement
        - SQL Server searches for dependent rows by examining all the foreign keys, i.e., when a record r is deleted, the systems checks whether r is referenced by other records
            - if there is an index, SQL Server uses it to check the existence of related data
            - if there isn't an index, the system has to perform a table scan
        - the performance of delete operations can be improved by creating indexes on foreign keys

- disable indexes

```
ALTER INDEX IX_EmployeeDepartmentHistory_ShiftID
  ON HumanResources.EmployeeDepartmentHistory DISABLE
```

- disable indexes


Object Explorer

Connect

HumanResources.Employee
HumanResources.EmployeeDepartmentHistory
Columns
Keys
Constraints
Triggers
Indexes
IX_EmployeeDepartmentHistory_DepartmentID (Non-Unique, Non-Clustered)
IX_EmployeeDepartmentHistory_ShiftID (Non-Unique, Non-Clustered)
PK_EmployeeDepartmentHistory_Business
Statistics
HumanResources.EmployeePayHistory
HumanResources.JobCandidate
HumanResources.Shift
Person.Address
Person.AddressType
Person.BusinessEntity
Person.BusinessEntityAddress
Person.BusinessEntityContact
Person.ContactType
Person.CountryRegion
Person.EmailAddress
Person.Password
Person.Person
Person.PersonPhone
Person.PhoneNumberType
Person.StateProvince
Production.BillOfMaterials
Production.Culture
Production.Document
Production.Illustration
Production.Location
Production.Product

- enable indexes

```
ALTER INDEX IX_EmployeeDepartmentHistory_ShiftID
   ON HumanResources.EmployeeDepartmentHistory REBUILD
```