Introduction to Databases

CMPE 2400

Introduction To Databases

- A Database is an organized collection of data so as to minimize data redundancy and ensure data consistency. It also provides for efficient data retrieval.
- In this course, we will use a relational Database Management System (DBMS) called SQL Server. In such a system the data is organized as a number of tables. Common data in two tables serve as links between the tables.

Introduction To Databases

- As a concrete example, consider that we want to store data for students and courses in which they are registered.
- Let's say we have the following instances:
  - Student 1: Student Id: 2013425, name: James Smith, Address: Southbrook
    - Courses:
      - CMPE2000 (Web Technologies)
      - CMPE2300 (Object-Oriented Programming)
      - CMPE2400 (Databases)
  - Student 2: Student Id: 2087639, name: Jane Middlestone, Address: Queen Elizabeth Avenue
    - Courses:
      - CMPE2000 (Web Technologies)
      - CMPE2400 (Databases)
      - CMPE2550 (Web Applications)

Introduction To Databases

- We can store the information as one table, as below

Student ID	Last Name	First Name	Address	Course Code	Course Description
2013425	Smith	James	Southbrook	CMPE2000	Web Technologies
2013425	Smith	James	Southbrook	CMPE2300	Object-Oriented Programming
2013425	Smith	James	Southbrook	CMPE2400	Databases
2087639	Middlestone	Jane	Queen Elizabeth Ave	CMPE2000	Web Technologies
2087639	Middlestone	Jane	Queen Elizabeth Ave	CMPE2400	Databases
2087639	Middlestone	Jane	Queen Elizabeth Ave	CMPE2550	Web Applications

Introduction To Databases

- We note that storing the information in one table results in data replication (and redundancy) which results in a number of drawbacks:
  - Waste of space to store the data
  - Have to update the same data in several places
  - Can result in inconsistency on data updates or when adding new data rows.

Introduction To Databases

- The drawbacks mentioned in the previous slide can be avoided by dividing our data in 3 tables as follows:
  - A **Student** table to store students data
  - A **Course** table to store data about courses
  - A **Student-Course** table that links students to courses in which they are registered.

Student Data in Three Tables

- **Student** Table

Student ID	Last Name	First Name	Address
2013425	Smith	James	Southbrook
2087639	Middlestone	Jane	Queen Elizabeth Ave

- **Course** Table

Course Code	Course Description
CMPE2000	Web Technologies
CMPE2300	Object-Oriented Programming
CMPE2400	Databases
CMPE2550	Web Applications

Student Data in Three Tables

- **Student-Course** Table

Student ID	Course Code
2013425	CMPE2000
2013425	CMPE2300
2013425	CMPE2400
2087639	CMPE2000
2087639	CMPE2400
2087639	CMPE2550

Student Data in Three Tables

- Our 3 tables together form an organized collection of data which:
  - reduces redundancy and thus avoids the associated inconsistencies
  - provides for efficient retrieval of data

Introduction To Databases

- Each table in a database is known as an entity.
- A table (or entity) consists of a number of columns.
  - Each column represents an attribute of the entity.
- When defining a table, its columns have to be defined. This constitutes the structure of the table.
- The actual data will form the rows of the table.
- The intersection of a row and a column is called a **cell**.

The Database Management System

- A Database Management System (DBMS) is a software that enables the creation and management of a database and its associated tables.
- It consists of :
  - A database engine- The component that manages the database and its tables
  - Client tools- An interface that provides the functionalities to users to use the database engine
- Examples of DBMS are: MS SQL Server, MySQL, Oracle… etc
- In this course we will use MS SQL Server DBMS. Our database engine will be on a remote data server and our client tool will be SQL Server Management Studio.

SQL

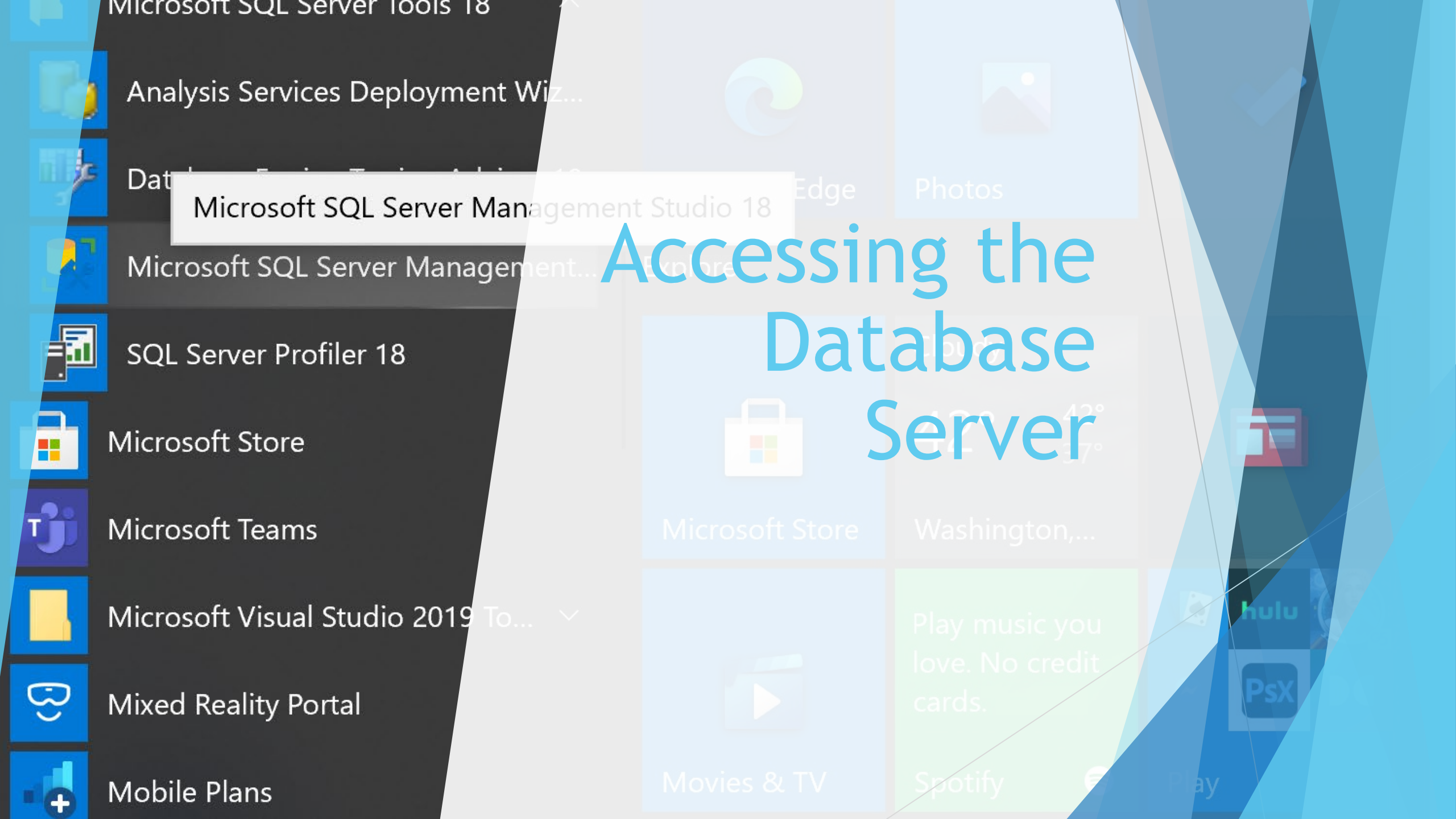
- SQL (Structured Query Language) has been the international standard for managing and manipulating databases since 1987.
- In this course, we will learn how to use SQL for the different tasks that we are required to perform in a database

Accessing the Database Server

- In order to follow along with the simple examples contained in these slides, you will want to make sure you can gain access to the data server.
- First, locate *SQL Server Management Studio Express (latest version)*. Its location is shown on the following slide.
  - You may wish to pin this to the taskbar.
- If you do not have SQL Server Management Studio, download it and install. It's available at

  https://docs.microsoft.com/en-us/sql/ssms

Accessing the Database Server



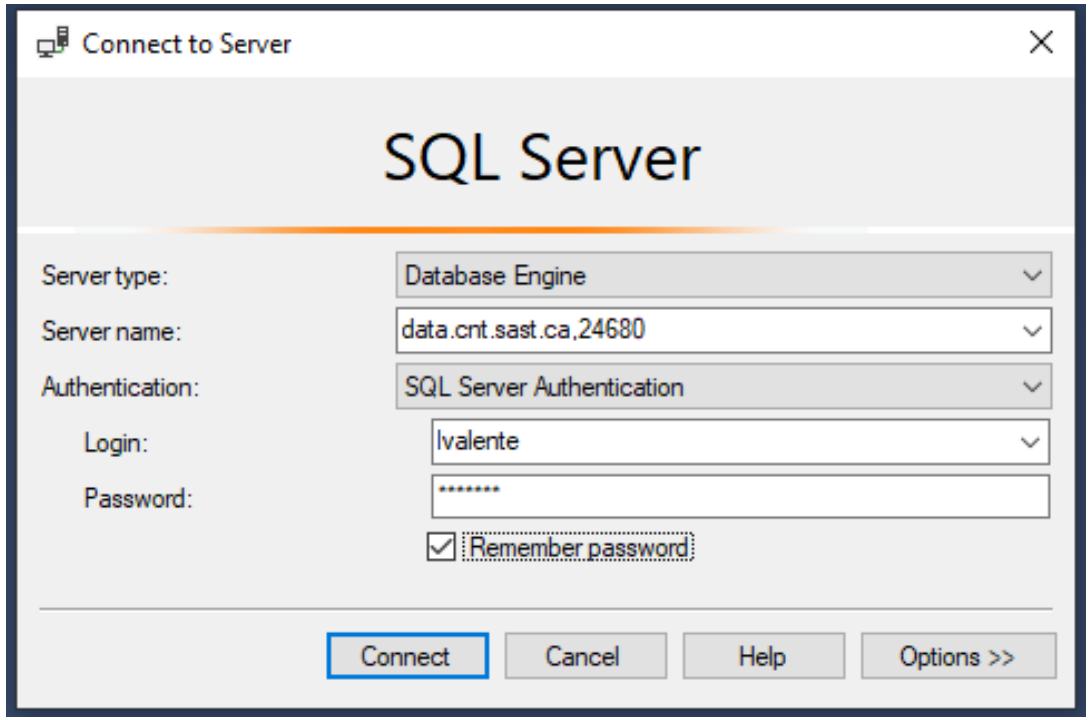
Accessing the Database Server

- Upon opening SQL Server Management Studio, you will be required to login to a Database Engine. The appropriate settings are shown on the following slide.
  - Your Login is the same as that for the NAIT domain, meaning the lab computers.
  - Your initial Password will be *CNT_123*, but you will be required to change this on first login.
    - If you have forgotten your password, ask your instructor to reset it for you.

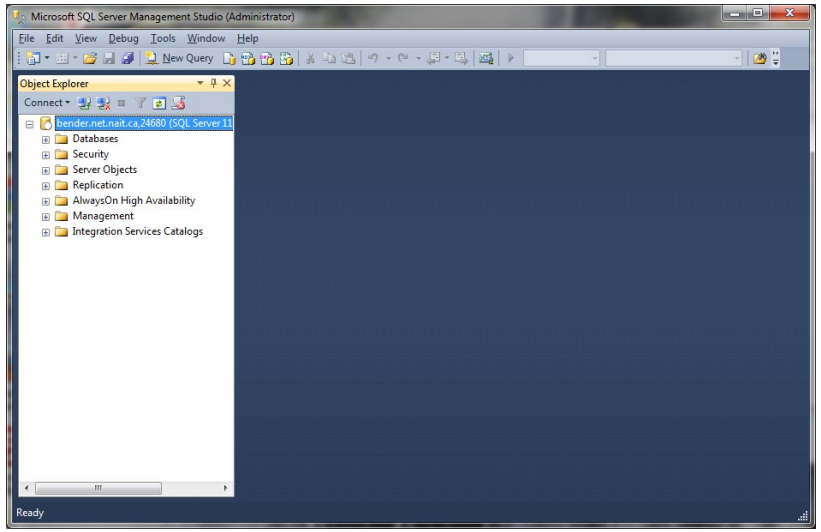
Accessing the Database Server

- For clarity, the Server name is:

  data.cnt.sast.ca,24680 (note the comma)



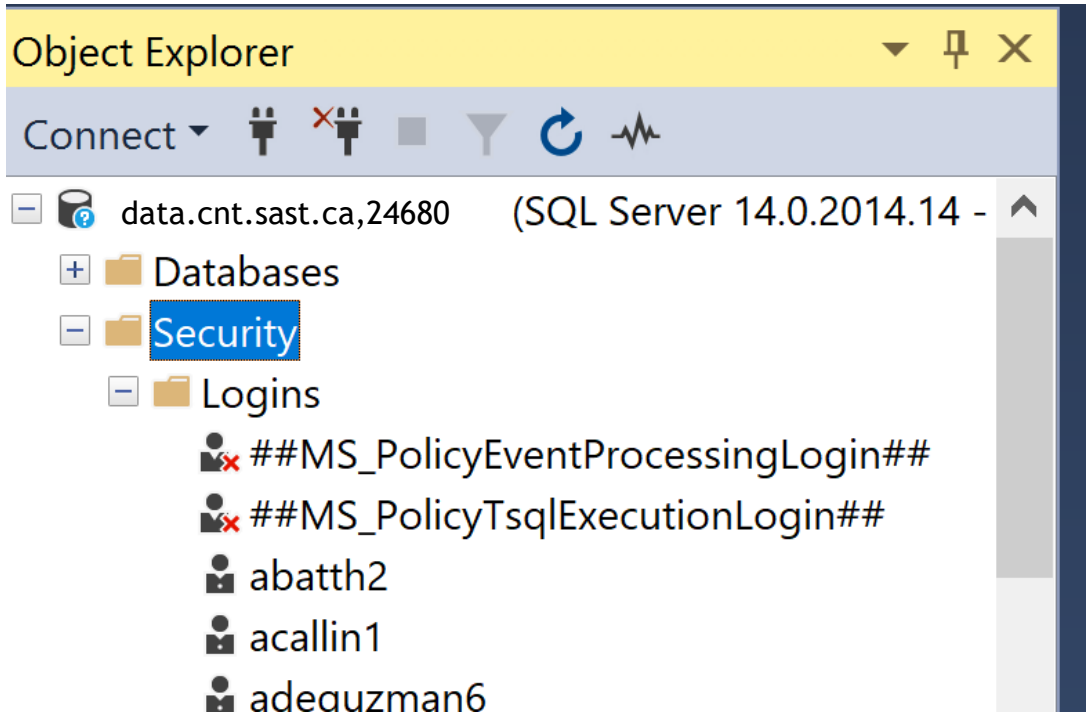
Successful Login



Upon Successful Login, you'll get the following window.

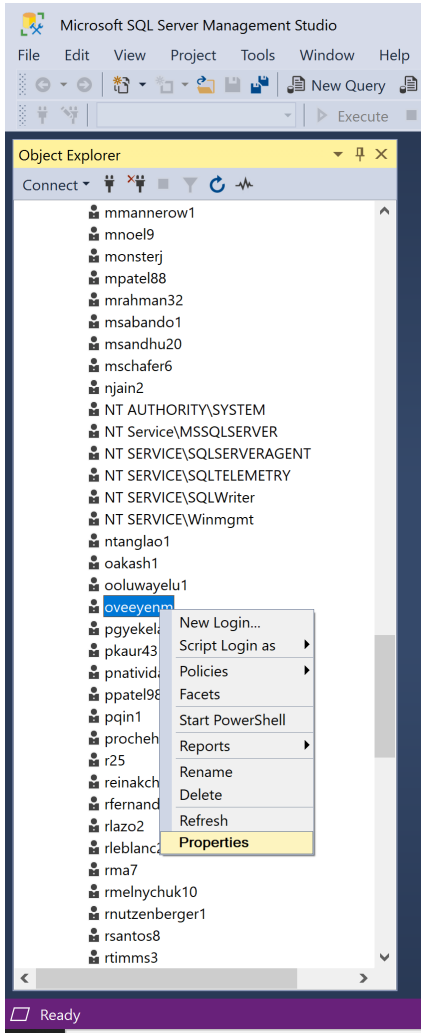
Changing Your Password

- In the Object Explorer pane,
  - expand security, followed by login



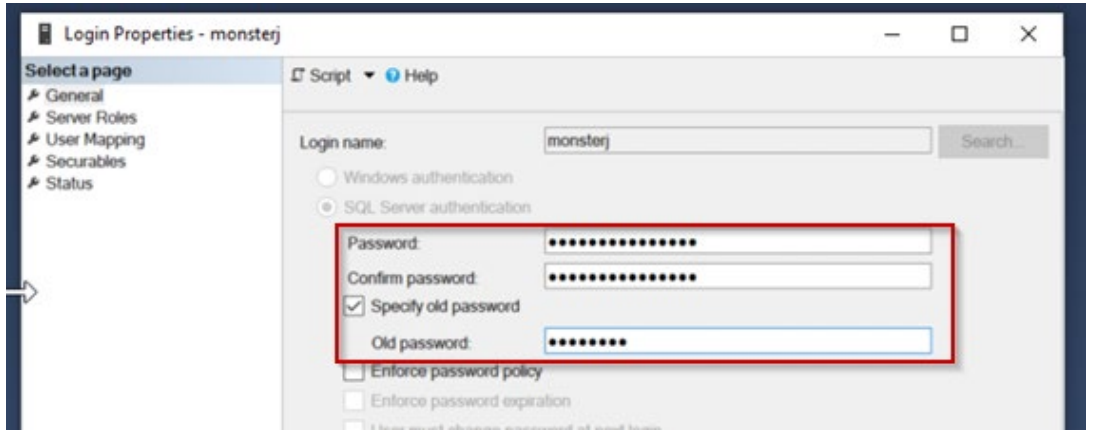
Changing Your Password

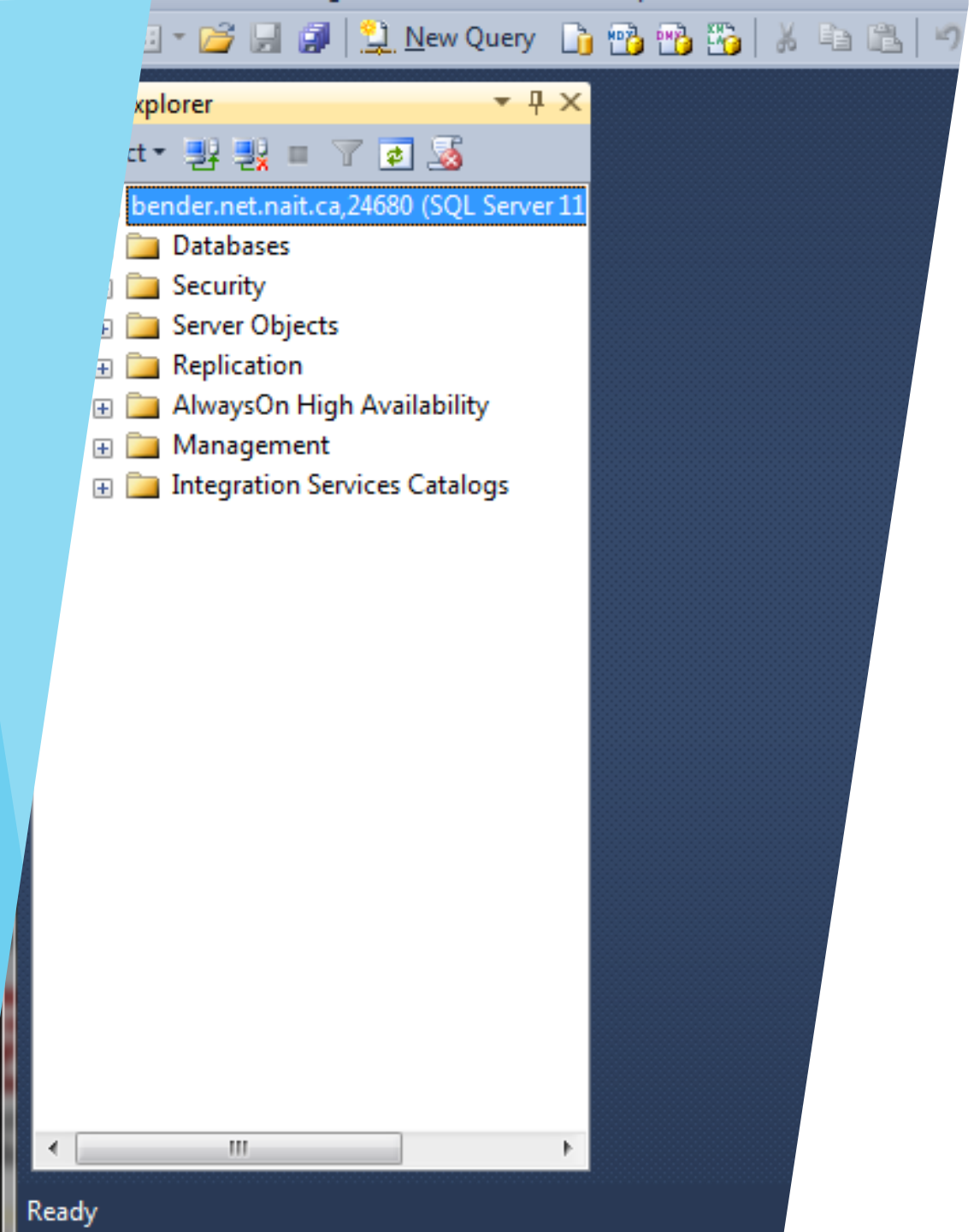
- Right-Click on your user's name



Changing Your Password

- You'll get the following window. You can change the password and click Ok.





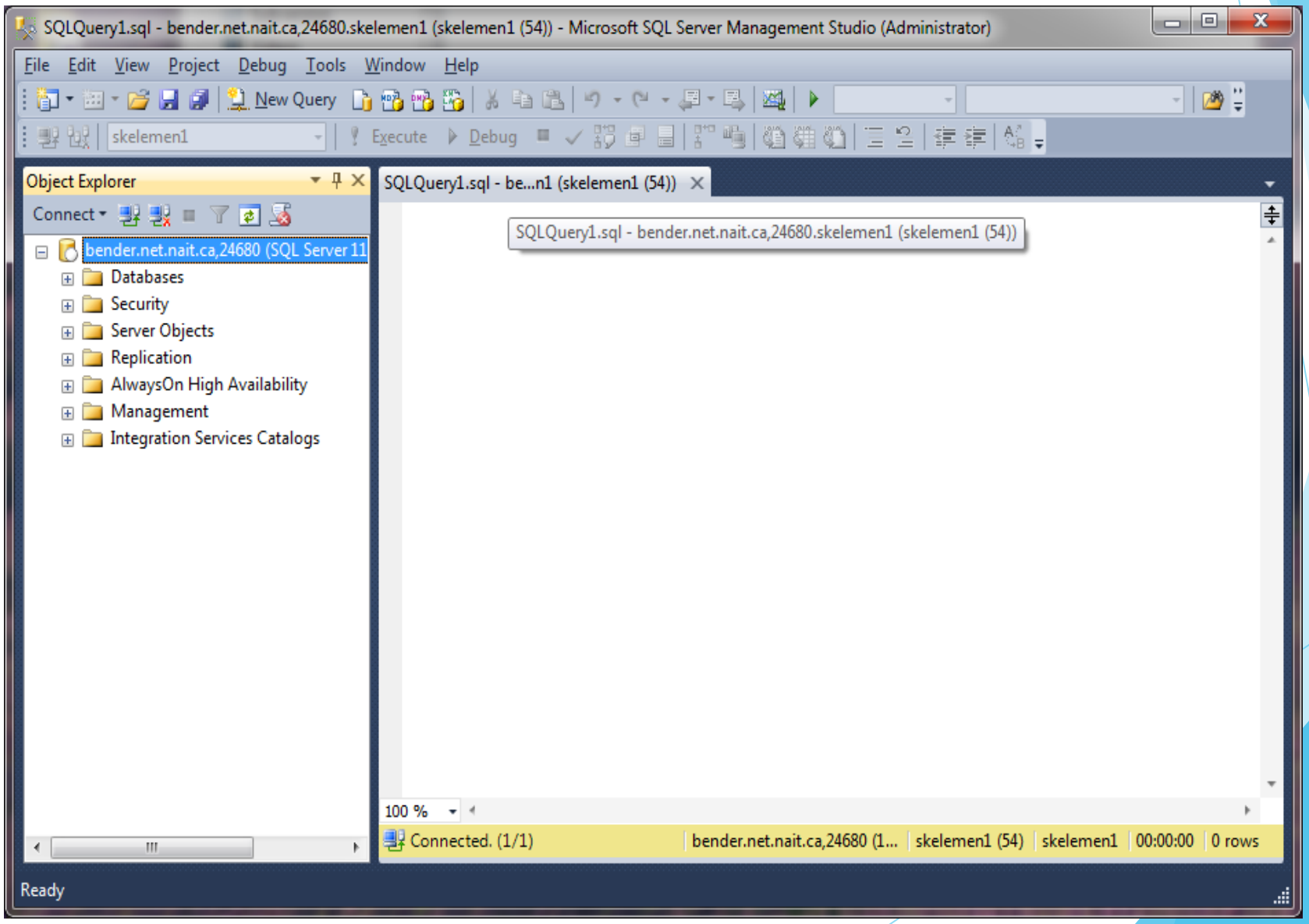
Opening a Query Window

Let's get back to our initial window.
We can now start writing queries

Opening a Query Window

- In order to write SQL queries and execute them against the database engine, you will require a query window.
  - Press the *New Query* button, or Ctrl+N
- Hovering your mouse over the tab of the query window shows you:
  - Name of the query file
  - Server name and port
  - Database name
  - Username in brackets with connection ID

Opening a Query Window



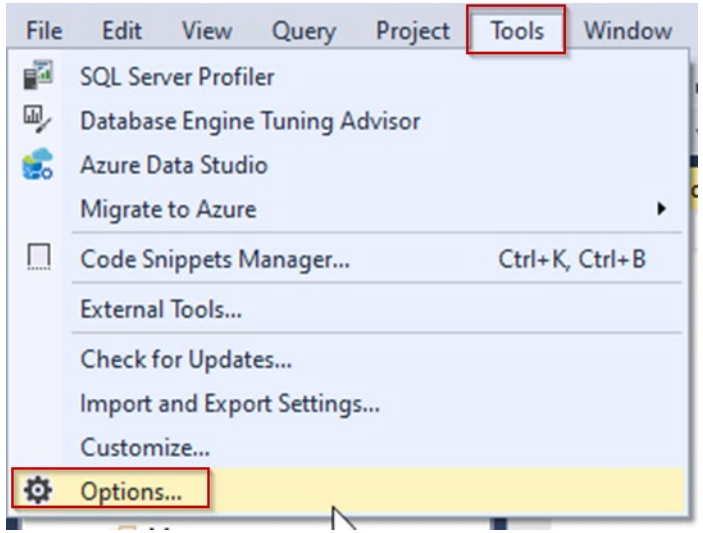
Opening a Query Window

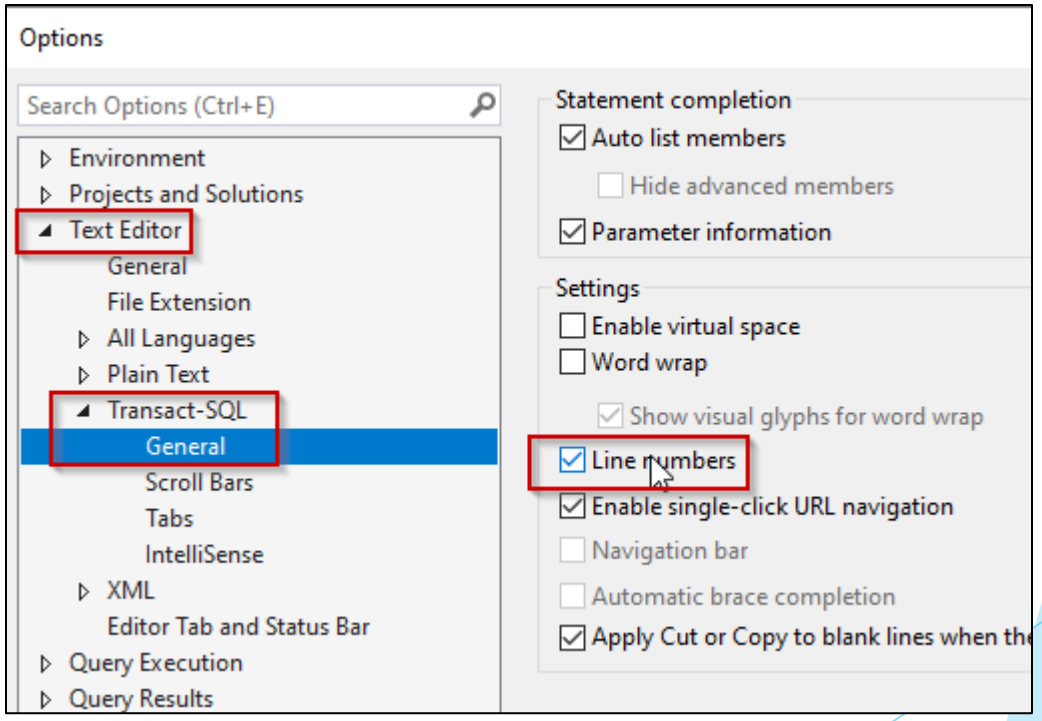
- You will also see a new toolbar that includes a few often used features.
  - Dropdown list populated with database names for quick switching.
  - Execute Script (F5)
  - Cancel Execute (Alt+Break)
  - Results to Text (Ctrl+T)
  - Results to Grid (Ctrl+D)
  - Comment / Uncomment

SQL Statement Properties

- Just as with C# in Visual Studio, your SQL statements will be color coded in the editor.
  - Syntax
    - SQL keywords show up blue
    - SQL operators show up gray, whether a word or symbol
    - SQL built in functions and system variables show up pink
    - SQL keywords and operators are case insensitive
    - SQL statements will ignore whitespace
  - Commenting
    - -- double dash starts a single line comment
    - Enclose block comments using /* comment */

Turn Line Numbering On





1

Ensure that the Database node in the left window is expanded, so that we can see the different databases.

2

Scroll down to the NorthwindTraders database

3

Expand the node, then expand the Tables node. We'll see all the tables.

4

Click on the node for the table **dbo.categories**, then click on the column node to see the columns.

Exercise 1- Object Explorer

Exercise 2 - Executing queries

Choose DB	In the Dropdown box select the database NorthwindTraders
Type	`select * from Products`
Execute	Execute the query
Replace	`select ProductId, ProductName, abs(UnitPrice*UnitsInStock) from Products`
Execute	Execute the query

Exercise 3 – Use <Database>

01

Create a new query window by clicking on the “New Query” button.

02

Type

```
use ClassTrak
go
```

03

Type

```
select * from
Instructors
```

SQL Statement Properties

- For this course, you may use either upper or lower case for your key words, but consistency is required.
  - For the record, we prefer lowercase…

```
-- The following query will show all field data for all records in the
-- jobs table.
select    *
from      jobs

-- The next query will execute with exactly the same results, but will
-- be marked harshly due to the headache it causes your instructor.
seleCT
    *
FRom
    jobs
```

Note: SSMS will provide autocomplete for some names – the case provided by autocomplete is acceptable:

```
4  select OrderID
5  from [Order Details]
```

SQL Statement Properties

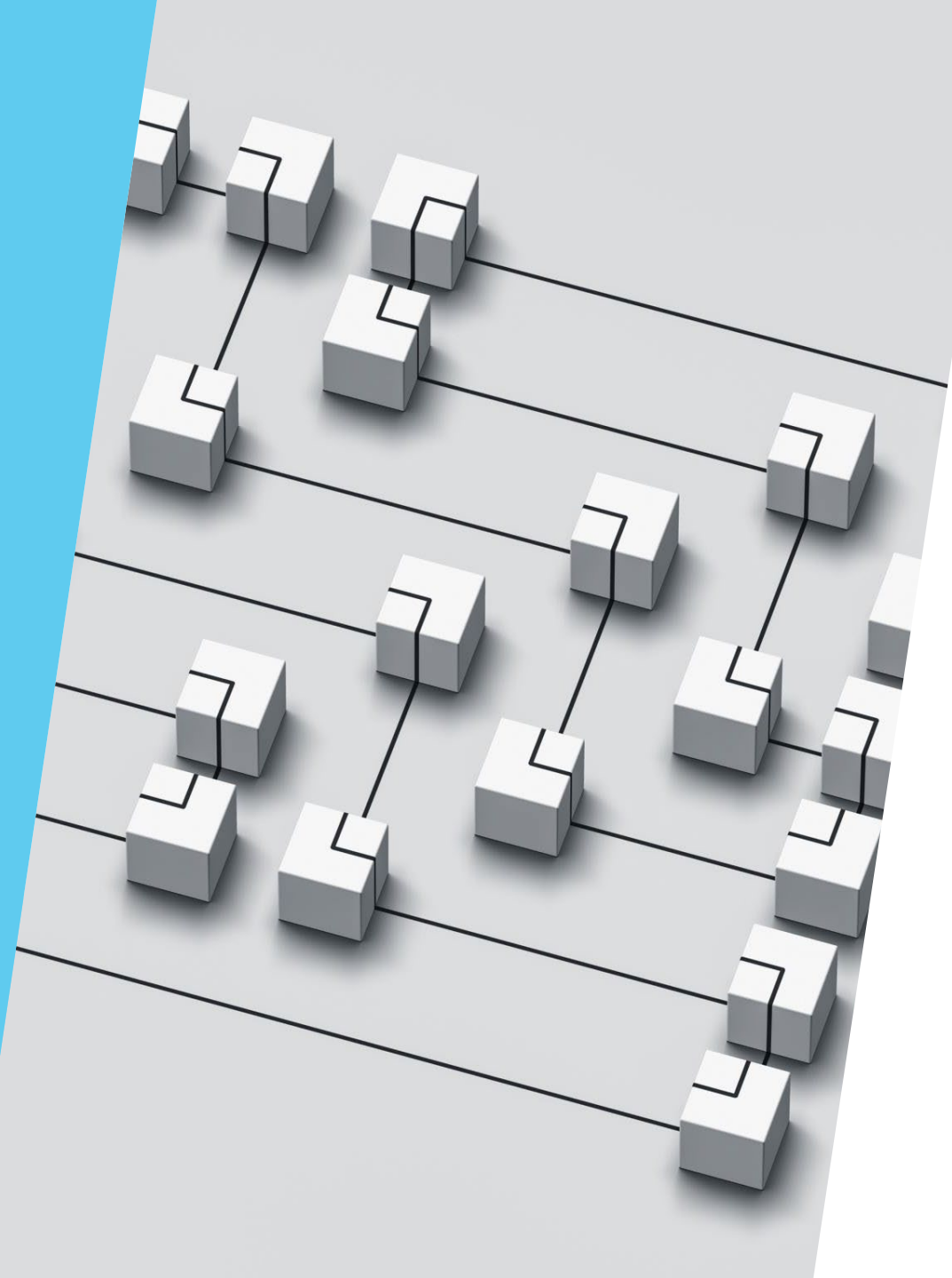
- Text and dates are enclosed in single quotes ‘ ‘
  - By default they will show up red in the editor
- MS SQL Server permits double quotes “ “ to be used as well
  - If double quotes are used, the text will not shade red under default editor setup
  - For consistency, and ease of debugging, we will use single quotes in this course

SQL Statement Properties

- If a table or column name uses a reserved word or contains spaces, double quotes (“ ”) or square brackets ([ ]) may be used to force acceptability
  - We will name all tables and columns using PascalCase just as in C#, and we will avoid keywords

```
-- The following works but is not recommended.
select [First Name] from [Student List]

-- Much cleaner…
select FirstName from StudentList
```



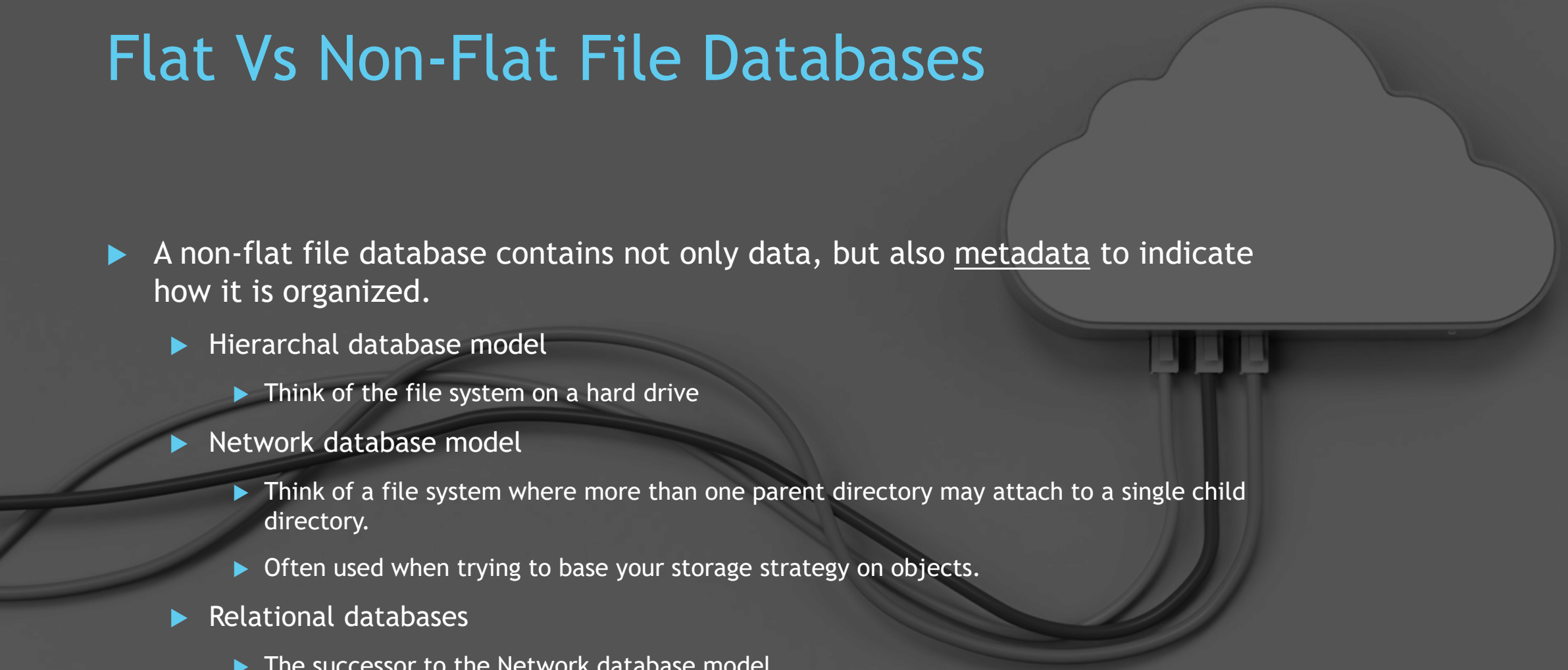
What is a database?

- A database contains records of connected information organized for efficient access

Flat Vs Non-Flat File Databases

- A flat file database contains only the data records
  - Think simple, single page spreadsheets as an example here
- The flat file database does not contain metadata to specify how the data is related
  - **metadata** is **data about data:** descriptive statistical information about the elements of a set of data

Flat Vs Non-Flat File Databases



- A non-flat file database contains not only data, but also metadata to indicate how it is organized.
  - Hierarchal database model
    - Think of the file system on a hard drive
  - Network database model
    - Think of a file system where more than one parent directory may attach to a single child directory.
    - Often used when trying to base your storage strategy on objects.
  - Relational databases
    - The successor to the Network database model
    - Not as operationally efficient, but more flexible

Entity = Table

- An entity may be viewed as a set of information that contains subsets of information.
  - Each *column* of a table describes an *attribute*, or *property*, of the entity. Columns are also called *fields*.
  - Each *row* of a table describes one particular *occurrence*, or *instance*, of an entity. Rows are also called *records*.

Entity = Table

- A relational database consists of data organized into *entities*, or *tables*.
- Consider the following table called StudentInfo

StudentID	**Lastname**	**Firstname**	**Address**
12345678	Horse	Ugly	NAIT
56789043	Head	Meat	4 Rot Lane

- Each row of the StudentInfo table holds the information for a particular student (instance) and each column of the table describes one attribute of the student.

Primary Keys (PK)

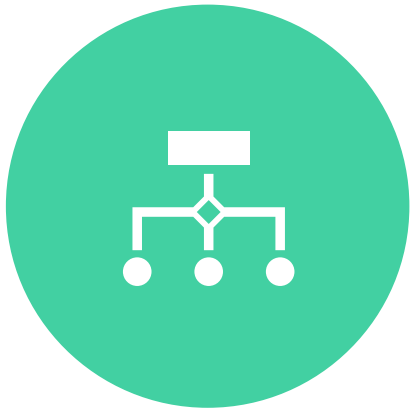
- A unique value is determined which may be used to identify each instance of an entity
  - Sometimes a **Natural Key** is already available, such as a SIN or Student ID number
  - You may need to generate unique values to use as a primary key when none exists – this is known as a **Surrogate Key**
  - ALL entities must have a primary key.

Composite Keys

- A primary key may be the combination of two or more data columns, forming a **Composite Key**
  - The DBMS will guarantee data integrity by ensuring that no duplicates of the key combination exist
  - Example: only one mark allowed per course per student;
    - The Composite Primary key could be: StudentID + CourseID
    - Mark would be a non-key field in that table

StudentId	CourseId	Mark
22123456	CMPE2000	75
22435123	CMPE2000	85
22123456	CMPE2300	90
22435123	CMPE2300	83

THE DATABASE SCHEMA IS ITS STRUCTURE DESCRIBED IN A FORMAL LANGUAGE SUPPORTED BY THE DATABASE MANAGEMENT SYSTEM (DBMS).



THE TERM "SCHEMA" REFERS TO THE ORGANIZATION OF DATA AS A BLUEPRINT OF HOW THE DATABASE IS CONSTRUCTED (DIVIDED INTO DATABASE TABLES IN THE CASE OF RELATIONAL DATABASES).

A Database Schema

Stored Procedures

Sometimes there are group of SQL queries that are executed frequently.

We can save these groups of statements as stored procedures that can be executed as and when we need them.

We execute a stored procedure by using “exec” followed by the procedure name
(e.g. **exec sp_tables**)

Sometimes, the stored procedure name is followed by other parameters

1

Type in the following set of statements in the query window

- **use NorthwindTraders**
- **exec sp_tables**

2

Execute the query

3

Replace the second statement by **exec sp_statistics Products**

4

Execute the query again

Exercise 4