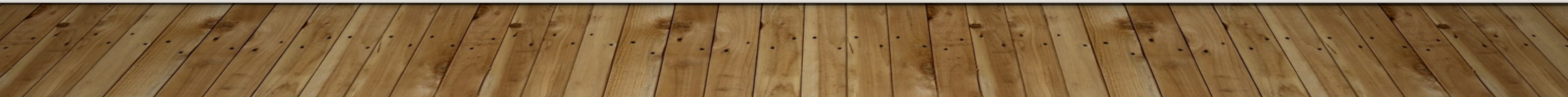
SQL AND DATABASE TRAINING

HOW TO DESIGN A DATABASE AND WRITE EFFICIENT QUERIES



ABOUT ME

- Architect / Programmer / Data Analyst for over 25 years
- I got into programming and software because I enjoy creating things
- I enjoy teaching and have taught Java, Javascript, SQL and C/Assembler

ASSUMPTIONS

- You know what a computer is and that is about it
- It will be helpful if you have Excel on your system
- You will not need to do any programming, this course is all about SQL, Data Wrangling and presentation of the data (using Excel)

WHAT WE WILL COVER

- How to design and create a new database
- How to query, insert and update data
- How to modify tables after the database is created and you have data in your tables
- How to improve performance (time permitting)
- General Q & A – But feel free to ask questions at any time
- If we go over the 2 hours, I'll schedule a follow on class if there is sufficient interest

WHAT YOU WILL NEED

- SQL Server Express or SQL Server Developer Edition (both are free downloads)
- SQL Server Management Studio (free download)
- A laptop running Windows 10 (home or professional)

WHAT IS A DATABASE

- A collection of data organized for easy and fast retrieval of structured data
- Some examples of data by industry
  - Oil and Gas
  - Inventory Control
  - Banking
  - Radioactive Waste Tracking (temporal databases)
  - Benefits Tracking

DATABASE DESIGN CONSIDERATIONS

- How to organize data
- What is an Entity
- What is an Attribute of an Entity
- How to think to get a good design
- Thinking from a programmers perspective
- Thinking from a DBA perspective

NORMAL FORMS

- 1st Normal Form (1NF)
- 2nd Normal Form (2NF)
- 3rd Normal Form (3NF)
- We also have 4th (Boyce Codd Normal Form - BCNF) and 5th normal forms, but they are rarely used in practical designs.
- The goal of normalization is to ensure consistency in data. Lack of consistency can lead to incorrect answers to questions asked of your data
- This however does not preclude machine learning software from providing the same or even better answers

FIRST NORMAL FORM

- Sources:
  - https://en.wikipedia.org/wiki/First_normal_form
  - https://support.microsoft.com/en-us/help/283878/description-of-the-database-normalization-basics
- Goals
  - Eliminate repeating groups in individual tables
  - Create a separate table for each set of related data
  - Identify each set of related data with a primary key
- CJ Date's definition of first normal form
  - A table is in first normal form if and only if it is "isomorphic to some relation"

CJ DATE – FIRST NORMAL FORM RULES

- No top-to-bottom ordering to the rows.
- No left-to-right ordering to the columns.
- No duplicate rows.
- Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
- All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].

SECOND NORMAL FORM

- Source: https://en.wikipedia.org/wiki/Second_normal_form
- A relation is in 2NF if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the relation. **A non-prime attribute of a relation** is an attribute that is not a part of any candidate key of the relation.
- I.e., a relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key.

EXAMPLE NOT IN SECOND NORMAL FORM

Electric Toothbrush Models			
Manufacturer	Model	Model Full Name	Manufacturer Country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

EXAMPLE IN SECOND NORMAL FORM

Toothbrush Manufacturers	
Manufacturer	Country
Forte	Italy
Dent-o-Fresh	USA
Kobayashi	Japan
Hoch	Germany

Toothbrush Models		
Manufacturer	Model	Model Full Name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

THIRD NORMAL FORM

- Source: https://en.wikipedia.org/wiki/Third_normal_form
- A DB design is in 3rd Normal Form if…
  - The relation R (table) is in second normal form (2NF)
  - Every non-prime attribute of R is non-transitively dependent on every key of R.

3NF EXAMPLE

An example of a 2NF table that fails to meet the requirements of 3NF is:			
Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

WHY DOESN'T THIS MEET 3NF

- This design allows a different date of birth for the same person
- We need to ensure each person has only one DOB
- Solution – move the winner and DOB to another table

THIS IS IN 3NF – BUT NOT QUITE….WHY NOT?

Tournament Winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner Dates of Birth

Winner	Date of Birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

FOURTH NORMAL FORM

- **Fourth normal form** (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three **normal forms** (1NF, 2NF and 3NF) and the Boyce-Codd **Normal Form**(BCNF)
- https://en.wikipedia.org/wiki/Fourth_normal_form

FIFTH NORMAL FORM

- **Fifth normal form** (**5NF**), also known as **project-join normal form** (**PJ/NF**) is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships. A table is said to be in the 5NF if and only if every non-trivial join dependency in that table is implied by the candidate keys
- https://en.wikipedia.org/wiki/Fifth_normal_form
- Only in rare situations does a 4NF table not conform to 5NF

LETS DESIGN THE DATABASE

- What are the entities we need to think about
- What are the properties of an entity
- Example Is cat an entity, Is dog an entity, do we need two tables for this, or one table called Animals and it has a property called Animal Type
- Think in abstractions

LAB 1

- We are going to create a simple student management system
- Create a database called StudentManagement

ENTITIES

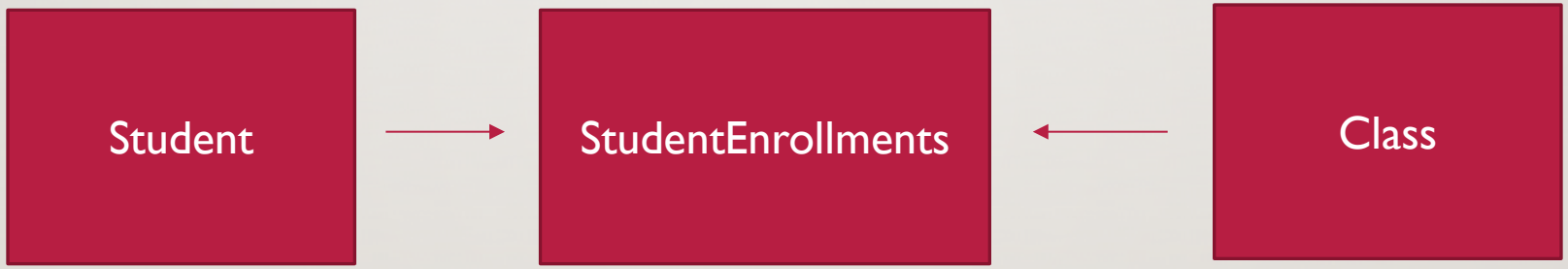
- Students
- Teachers
- Courses

HOW ARE ENTITIES RELATED

- A student enrolls in a class
- A class has a schedule
- A teacher teaches a class
- A teacher can teach more than one class
- A student can enroll in more than one class

STUDENT ENROLLMENTS

- Is this enough?



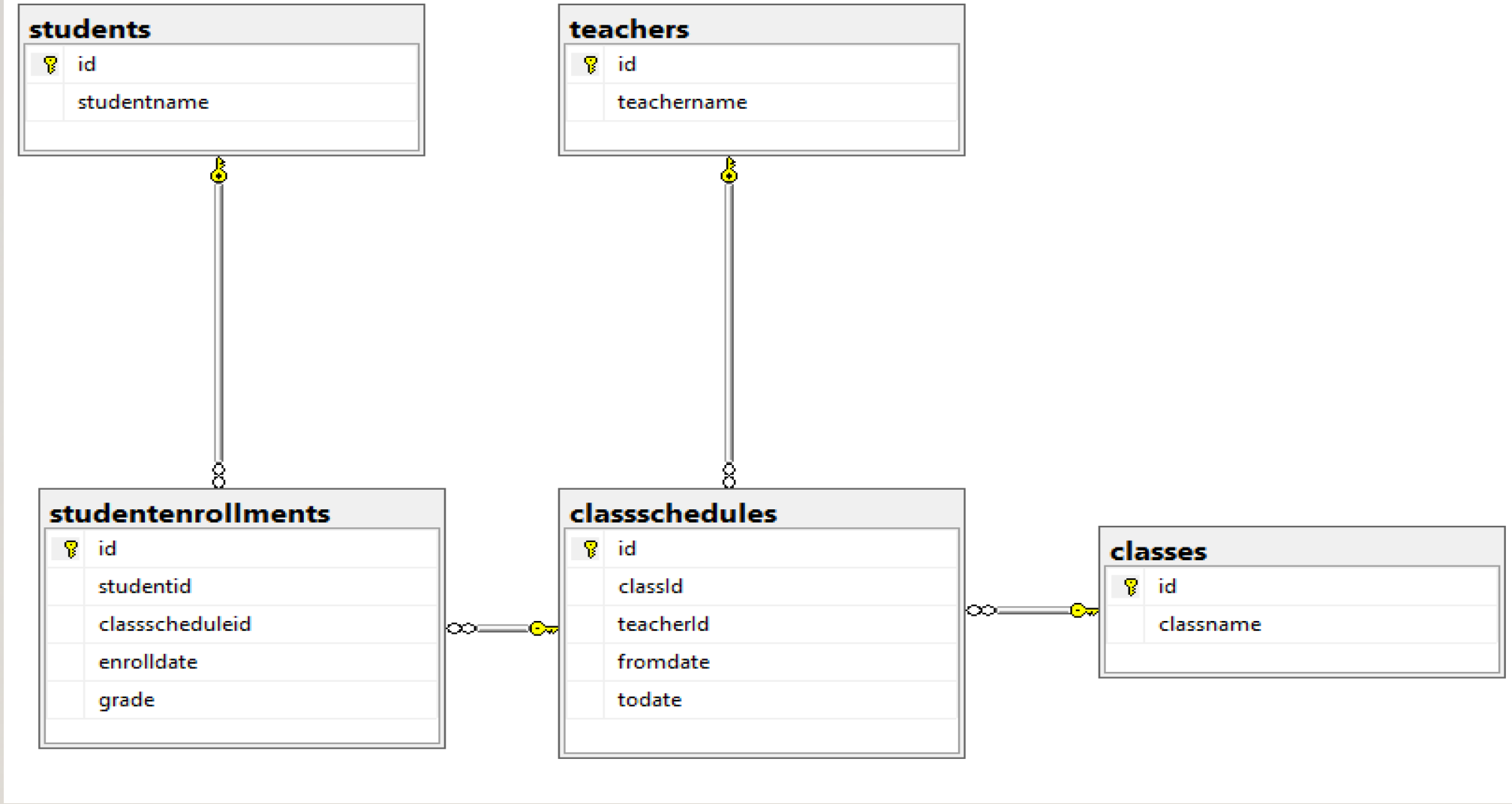
STUDENT ENROLLMENTS

- No, its not enough….Why?
- A class can run more than once, therefore we need class schedule and the student enrolls in a class, but for a particular schedule

WHAT KIND OF QUESTIONS CAN WE ASK

- Which teachers taught physics between the hours of 8am and 12pm on Fridays
- What students were enrolled in math classes in the fall semester of a given year

- Questions like these tell us what attributes we need to track
- Remember, reporting will tell you what you need to track


students
id
studentname
teachers
id
teachername
studentenrollments
id
studentid
classscheduleid
enrolldate
grade
classschedules
id
classId
teacherId
fromdate
todate
classes
id
classname

WHAT IS A QUERY

- A query is asking a question and extracting the answer from your data
- Examples:
  - How many wells with a downtime of at least 12 hours in the last month
  - Give me the total amount of Oil, Water and Gas produced for a given well in the last 2 days
  - Give me a list of the wells that have a downtime of 24 hours but seem to be producing for the last month
  - How many users are there in each role for a given organization including roles with no users
  - Tell me who created a given well, and who last updated the well and when the well was updated

SQL

- Everyone is new to SQL
  - Inner vs outer joins
  - Basic select statements
  - Aggregation (sum, avg etc.,)
  - Group by
  - Having
  - Order by

QUERY EXAMPLES

- Select * from students
- Select * from teachers
- Select * from studentenrollments
- Select * from classschedules

LETS GET MORE INFORMATION

- Select * from classschedule cs
  Join teacher t on t.id = cs.teacherid
  Join class c on c.id = cs.classid
- We can now get c.Name (class name), t.name (teacher name) and cs.DateFrom and cs.DateTo to get which teacher teaches which class

SEARCHING FOR THINGS

- Select…. Where … condition

GROUPED RESULTS

- How many classes did a given student take between 2 dates
- Select s.Name, count(se.id)
  from students s
  join studentenrollments se on se.studentid = s.id
  where se.enrolldate between date1 and date2
  group by s.Name

THE BASIC TOOLKIT

- How to find out information about the database
- Why do we need this?
  - If you can't find a column quickly it will slow you down
  - If you don't know which tables you are pointed at or which tables are pointing at you, you wont know if your insert or delete operations will fail or not
  - If you want to know the datatype of any given column in any given table, how would you find that without using the navigator in the left panel of SSMS
  - If you want to know if the column has a description field associated with it, how would you do that… 2 ways.. I'll show the harder way

FINDING OUT INFO ABOUT A TABLE

- List Columns in this table
- List Indexes on this table
- List Foreign Keys in this table
- List Foreign Keys other tables pointing to this table
- Generating a dictionary and using it in Excel

PRACTICE SESSION - BEGINNER

PRACTICE SESSION - INTERMEDIATE

PRACTICE SESSION – ADVANCED

- Find all columns in a table
- Find all foreign keys in a table
- Find all tables that reference another table
- Find all indexes on a given table
- Generate a data dictionary, copy it to excel and filter the data to show information about a given table
- How many columns does table "xyz" have?
- How many tables reference the "xyz" table?

IDEMPOTENT QUERIES

- Always use these when
  - inserting data into existing tables
  - Updating data in existing tables
  - adding any column
  - removing any column
- Updates and Inserts should happen once and have no action if the update or insert has already been applied
- Do not just rely on the primary key, you might end up with 2 identical rows that only differ in primary key. I.e., duplicate row problems

IDEMPOTENT ADD COLUMN

```
IF NOT EXISTS
(
    SELECT * FROM [information_schema].[columns]
    WHERE   table_name = 'WellProperties'
    AND table_schema = 'dbo'
    AND column_name = 'TestColumn'
)
BEGIN
    ALTER TABLE [dbo].[WellProperties] ADD TestColumn int
END
```

IDEMPOTENT REMOVE COLUMN

```
IF EXISTS
(
    SELECT * FROM [information_schema].[columns]
    WHERE   table_name = 'WellProperties'
    AND table_schema = 'dbo'
    AND column_name = 'TestColumn'
)
BEGIN
    ALTER TABLE [dbo].[WellProperties] DROP COLUMN TestColumn
END
```

IDEMPOTENT INSERTS

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[InterestCodes]') AND type in (N'U'))
BEGIN
select * into #tmpCodes from CDEXInterestCodes where 1 = 2
insert into #tmpCodes (code, description) values('M', 'Miscellaneous')
insert into #tmpCodes (code, description) values('N', 'Net Profit Payment')
insert into CDEXInterestCodes select code, description
from #tmpCodes T where not exists (select 1 from InterestCodes where Code = T.Code and Description = T.Description)
drop table #tmpCodes
END
```

IDEMPOTENT INSERTS

```
IF NOT EXISTS(SELECT 1 FROM [dbo].[ProductionProducts] WHERE Name = 'Combination of Products' AND Code = 'Combination of Products' AND OrgId = @OrgId)

BEGIN

INSERT INTO [dbo].[ProductionProducts] ([Name], [Code], [CDEXCode], [Active], [CreateDate], [UpdateDate], [CreatedById], [LastUpdatedById], [OrgId])

        VALUES ('Combination of Products', 'Combination of Products', 5, 1, GETDATE(), GETDATE(), 1, 1, @OrgId)

END
```

IDEMPOTENT UPDATES

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ProductionProducts]') AND type in (N'U'))

BEGIN

UPDATE [dbo].[ProductionProducts] SET
        Name = newName,
        UpdatedDate = getdate()

WHERE NAME = 'Combination of Products' AND Code = 'Combination of Products' AND
OrgId = @OrgId

END
```

PERFORMANCE TOOLTIP

Nested Loops

For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.

Physical Operation	Nested Loops
Logical Operation	Left Outer Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	24
Actual Number of Batches	0
Estimated Operator Cost	0.0000091 (0%)
Estimated I/O Cost	0
Estimated CPU Cost	0.0000091
Estimated Subtree Cost	0.0520335
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	2.47318
Estimated Row Size	411 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	3

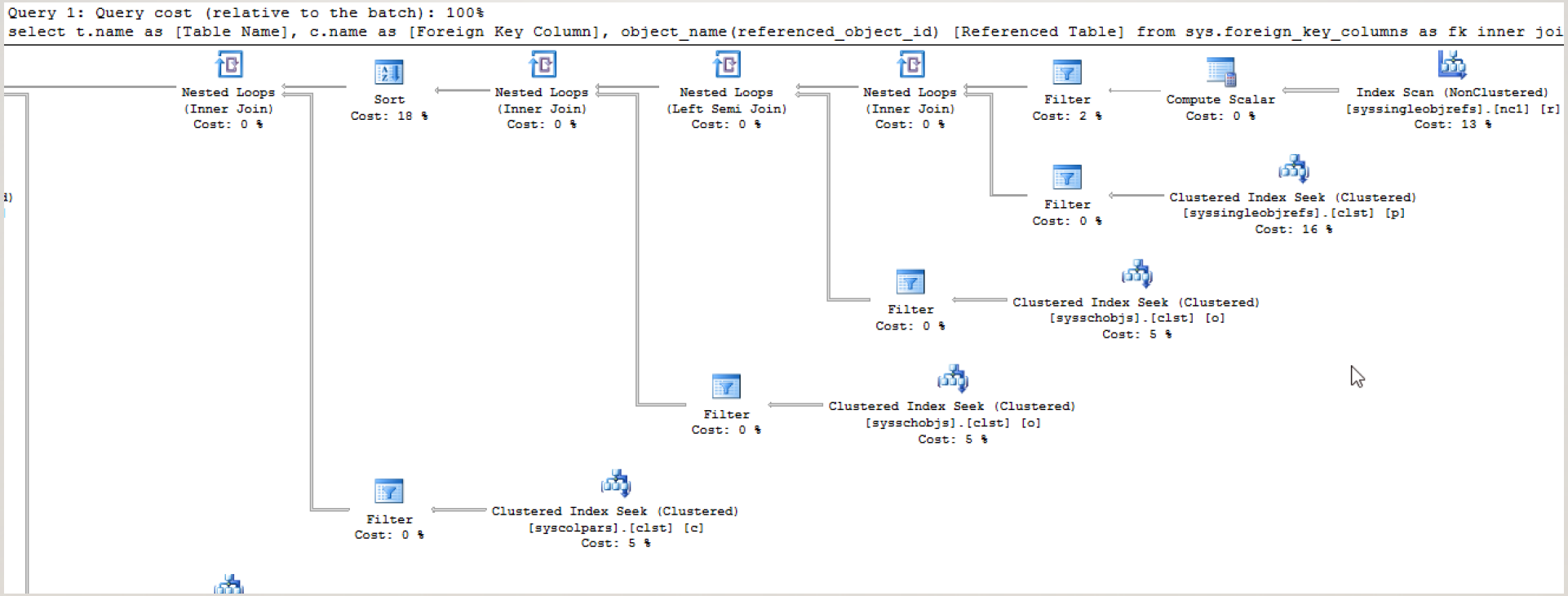
Output List

[NeoFirma7].[sys].[sysschobjs].id, [NeoFirma7].[sys].[sysschobjs].name, [NeoFirma7].[sys].[syscolpars].id, [NeoFirma7].[sys].[syscolpars].colid, [NeoFirma7].[sys].[syscolpars].name, Expr1151

Outer References

[NeoFirma7].[sys].[sysschobjs].id

EXECUTION PLAN



TEST STUBS

- Build out a collection of these for your own use
- They are very handy, so you don't have to repeat steps when testing stored procedures or functions
- It can help to make a list of the stored procs and functions and what they do

TEST STUB EXAMPLE 1

```
DECLARE @BatteryIds IntegerArrayTableType
DECLARE @OrgId int = 98
DECLARE @Active int = 1
DECLARE @UserId int = 1
DECLARE @ProductTypeId INT = NULL
INSERT @BatteryIds(n) select Id from Batteries where OrgId = @OrgID
DECLARE @StartDate varchar(10) = '1/1/2017'
DECLARE @EndDate varchar(10) = '8/10/2017'
DECLARE @GroupBy varchar = 'Daily'
exec RPT_BatteryGroupedProduction @OrgId, @UserID, @BatteryIds, @StartDate, @EndDate, @GroupBy, @Active
```

TEST STUB EXAMPLE 2

```
DECLARE @WellIds IntegerArrayTableType
DECLARE @OrgID int = 91
DECLARE @Active int = 1
DECLARE @UserId int = 1
INSERT @WellIds(n) select distinct Id from Wells where OrgId = 91 and FieldId = 1724
DECLARE @StartDate varchar(10) = '4/1/2017'
DECLARE @EndDate varchar(10) = '8/31/2017'
DECLARE @GroupBy varchar = 'Monthly'
exec [RPT_ExportWellMonthlyProduction] @OrgId, @UserId, @WellIds, @StartDate, @EndDate, @Active
```