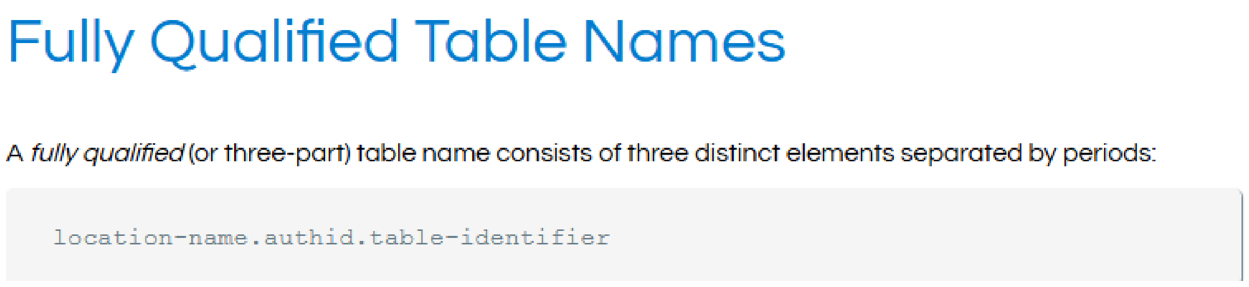
Fahim Tanvir
CSCI- 331 Individual Report/Notes for FQDS, FQTNS and Taxonomy

For this assignment, I decided to look through the internet for more information on Fully Qualified Domains and Table Names. Since I planned on doing the introduction portion of the presentation and wanted to set the base output for my team's slide deck, I decided to learn basic information about both so the rest of my teammates can branch out from there.

It has come to my understanding that both are conventions used to make things easier for SQL as they avoid keeping this vague while improving maintainability in terms of use. What this means is it helps keep things organized and reusable. Fully Qualified Domains(FQD) in a taxonomy can basically be started by creating a domain or type)depends on T-SQL or ANSI) from a data type(usually something that is varchar).

For Fully Qualified Table Names(FQTN), there's like 3 or 4 components as to what forms from location/server name, schema name, (optional) database name and table name:


Fully Qualified Table Names

A *fully qualified* (or three-part) table name consists of three distinct elements separated by periods:

```
location-name.authid.table-identifier
```

Image Source:
https://www.microfocus.com/documentation/enterprise-developer/ed100/ED-VS2022/index.html?t=HRQRRHSQBA0Z.html

Essentially FQTN's are useful to keep things organized, as it lets the user know which table or schema is being referenced every time they run a query. This allows for scalability as well as its easiest to manage with multiple databases or schemas(this is also why I said database name is relatively optional, cause what if you just loaded one into your system).

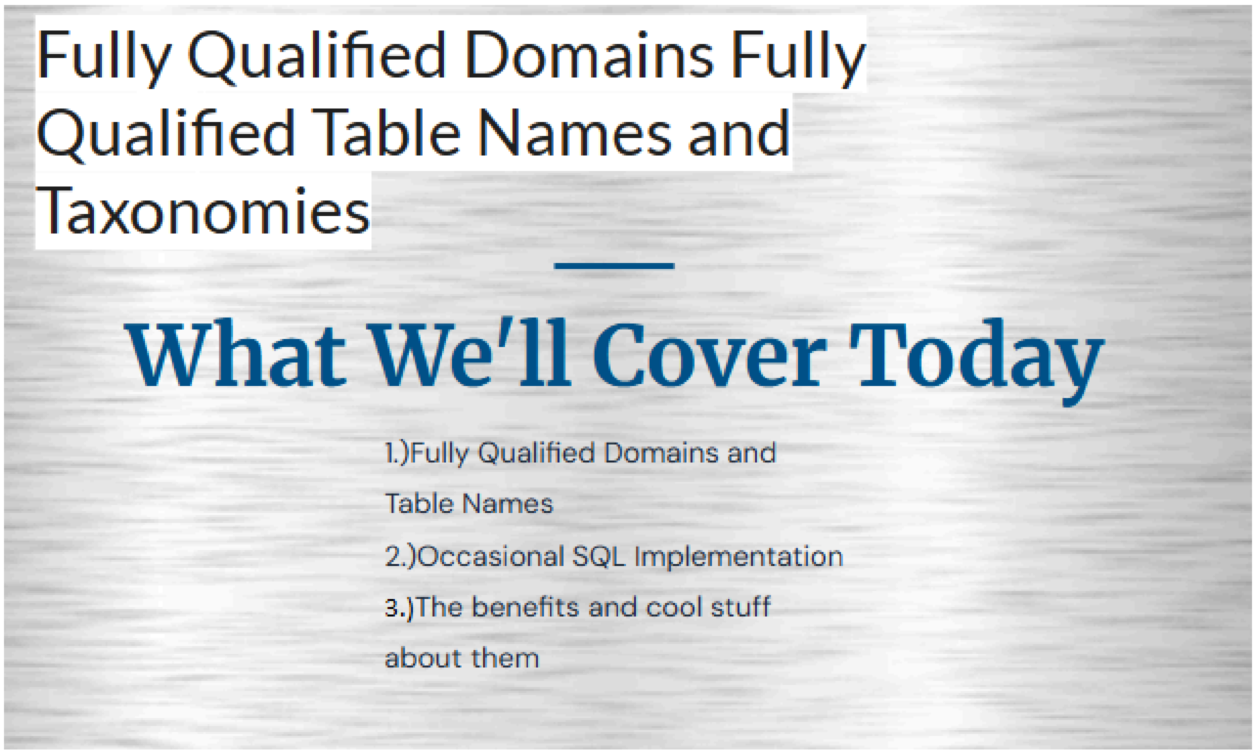
And to wrap a neat bow for everything, taxonomy just means to contextualize everything in your database using a structured hierarchy defining everything.

To give credit, I used the following sources for added clarity and understanding:
Micro Focus Documentation (For FDTN's)

SQL Server 13 - Domain Integrity (I mainly used 1:40- 2:58 and helped contextualize FQD's)
What is a SQL Server Data Dictionary and why would I want to create one?(For Taxonomy)
What is Data Taxonomy? Examples Included | Amplitude(Also for Taxonomy)

If I had to explain in layman's terms(no techno-babble), FQTNs is your mailbox or po-box where everything related to each data is stored in one correct spot. FQD's are basically your mailing address where it sends your mail(data in this analogy) to the correct person or scheme. Taxonomy is a tour guide with a index to help classify and organize every important word.

For the presentation, I introduced FQD's and taxonomy, using the first 6 slides, where I stated a summarized version of what was stated here. For examples to showcase, me and my group used sql implementation from WorldWideImporters database as we used that database the most, so it would've been a simple process to just use that. This can be seen here(not counting any slides that is just a title):

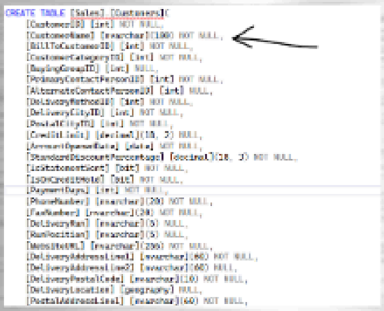

Fully Qualified Domains Fully Qualified Table Names and Taxonomies

What We'll Cover Today

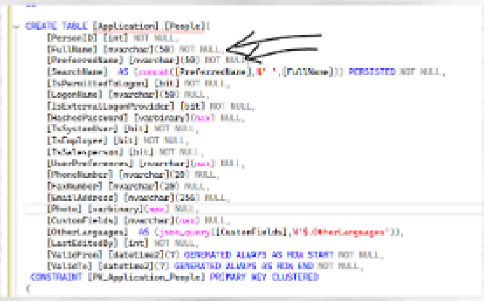
1.)Fully Qualified Domains and Table Names
2.)Occasional SQL Implementation
3.)The benefits and cool stuff about them

Why does it matter?

Why is full names for customers is `NVARCHAR(100)` and for employee's or company affiliations `VARCHAR(50)` in another? Our main enemy here is inconsistency.

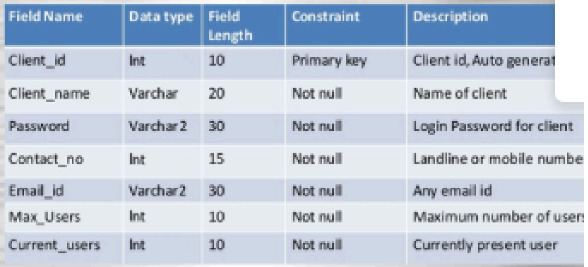


```
CREATE TABLE [Sales].[Customers](
	[CustomerID] [int] NOT NULL,
	[CustomerName] [nvarchar](100) NOT NULL,
	[BillToCustomerID] [int] NOT NULL,
	[CustomerCategoryID] [int] NOT NULL,
	[BuyingGroupID] [int] NULL,
	[PrimaryContactPersonID] [int] NOT NULL,
	[AlternateContactPersonID] [int] NULL,
	[DeliveryMethodID] [int] NOT NULL,
	[DeliveryCityID] [int] NOT NULL,
	[PostalCityID] [int] NOT NULL,
	[CreditLimit] [decimal](18, 2) NULL,
	[AccountOpenedDate] [date] NOT NULL,
	[StandardDiscountPercentage] [decimal](18, 3) NOT NULL,
	[IsStatementSent] [bit] NOT NULL,
	[IsOnCreditHold] [bit] NOT NULL,
	[PaymentDays] [int] NOT NULL,
	[PhoneNumber] [nvarchar](20) NOT NULL,
	[FaxNumber] [nvarchar](20) NOT NULL,
	[DeliveryRun] [nvarchar](5) NULL,
	[RunPosition] [nvarchar](5) NULL,
	[WebsiteURL] [nvarchar](256) NOT NULL,
	[DeliveryAddressLine1] [nvarchar](60) NOT NULL,
	[DeliveryAddressLine2] [nvarchar](60) NULL,
	[DeliveryPostalCode] [nvarchar](10) NOT NULL,
	[DeliveryLocation] [geography] NULL,
	[PostalAddressLine1] [nvarchar](60) NOT NULL,
```



```
CREATE TABLE [Application].[People](
	[PersonID] [int] NOT NULL,
	[FullName] [nvarchar](50) NOT NULL,
	[PreferredName] [nvarchar](50) NOT NULL,
	[SearchName]  AS (concat([PreferredName],N' ',[FullName])) PERSISTED NOT NULL,
	[IsPermittedToLogon] [bit] NOT NULL,
	[LogonName] [nvarchar](50) NULL,
	[IsExternalLogonProvider] [bit] NOT NULL,
	[HashedPassword] [varbinary](max) NULL,
	[IsSystemUser] [bit] NOT NULL,
	[IsEmployee] [bit] NOT NULL,
	[IsSalesperson] [bit] NOT NULL,
	[UserPreferences] [nvarchar](max) NULL,
	[PhoneNumber] [nvarchar](20) NULL,
	[FaxNumber] [nvarchar](20) NULL,
	[EmailAddress] [nvarchar](256) NULL,
	[Photo] [varbinary](max) NULL,
	[CustomFields] [nvarchar](max) NULL,
	[OtherLanguages]  AS (json_query([CustomFields],N'$.OtherLanguages')),
	[LastEditedBy] [int] NOT NULL,
	[ValidFrom] [datetime2](7) GENERATED ALWAYS AS ROW START NOT NULL,
	[ValidTo] [datetime2](7) GENERATED ALWAYS AS ROW END NOT NULL,
 CONSTRAINT [PK_Application_People] PRIMARY KEY CLUSTERED
(
```

What is a Taxonomy?



Field Name	Data type	Field Length	Constraint	Description
Client_id	Int	10	Primary key	Client id, Auto generat
Client_name	Varchar	20	Not null	Name of client
Password	Varchar2	30	Not null	Login Password for client
Contact_no	Int	15	Not null	Landline or mobile numbe
Email_id	Varchar2	30	Not null	Any email id
Max_Users	Int	10	Not null	Maximum number of users
Current_users	Int	10	Not null	Currently present user

Image source: [What is a SQL Server Data Dictionary and why would I want to create]()

A "Dictionary" for Your Data

Think of it as a classification system. It's a "dictionary of dictionaries" that formally defines and names all your business concepts.

- It's the single source of truth.
- It's not a database; it's a design document (or a set of metadata).

  It ensures everyone agrees on what "Customer" or "Email" means within our database(given the context)

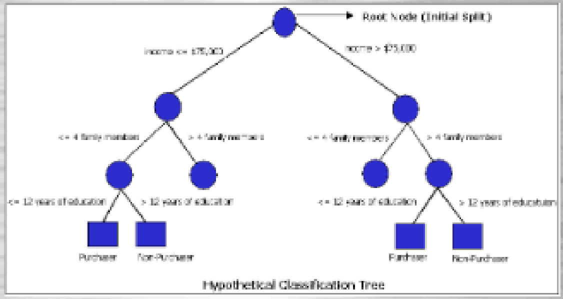
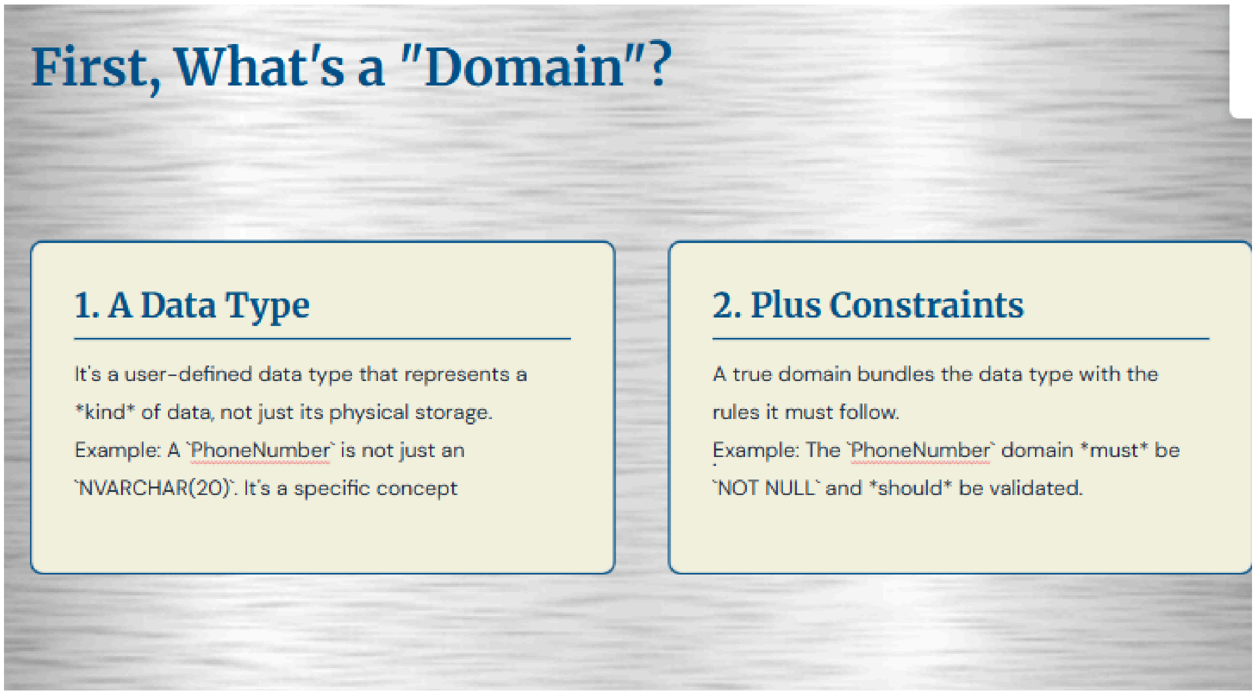


Image source: [Classification Tree | solver]()



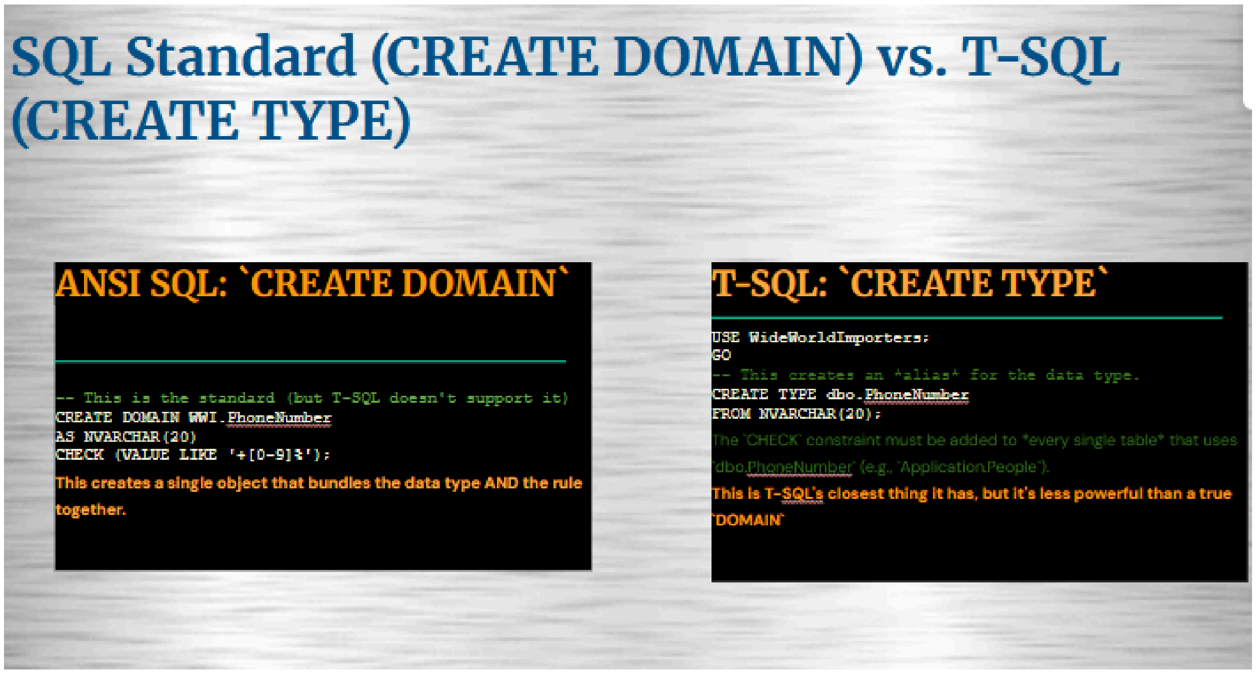
First, What's a "Domain"?

1. A Data Type

It's a user-defined data type that represents a *kind* of data, not just its physical storage.
Example: A `PhoneNumber` is not just an `NVARCHAR(20)`. It's a specific concept

2. Plus Constraints

A true domain bundles the data type with the rules it must follow.
Example: The `PhoneNumber` domain *must* be `NOT NULL` and *should* be validated.



SQL Standard (CREATE DOMAIN) vs. T-SQL (CREATE TYPE)

ANSI SQL: `CREATE DOMAIN`

```
-- This is the standard (but T-SQL doesn't support it)
CREATE DOMAIN WWI.PhoneNumber
AS NVARCHAR(20)
CHECK (VALUE LIKE '+[0-9]%');
```

This creates a single object that bundles the data type AND the rule together.

T-SQL: `CREATE TYPE`

```
USE WideWorldImporters;
GO
-- This creates an *alias* for the data type.
CREATE TYPE dbo.PhoneNumber
FROM NVARCHAR(20);
```

The `CHECK` constraint must be added to *every single table* that uses `dbo.PhoneNumber` (e.g., `Application.People`).

This is T-SQL's closest thing it has, but it's less powerful than a true `DOMAIN`