Fully Qualified Domains Fully Qualified Table Names and Taxonomies

GROUP 4: Fahim Tanvir, Yousuf Ahmed, Esfar Rakin, Sayantan Saha, Justin Zara

Fully Qualified Domains Fully Qualified Table Names and Taxonomies

What We'll Cover Today

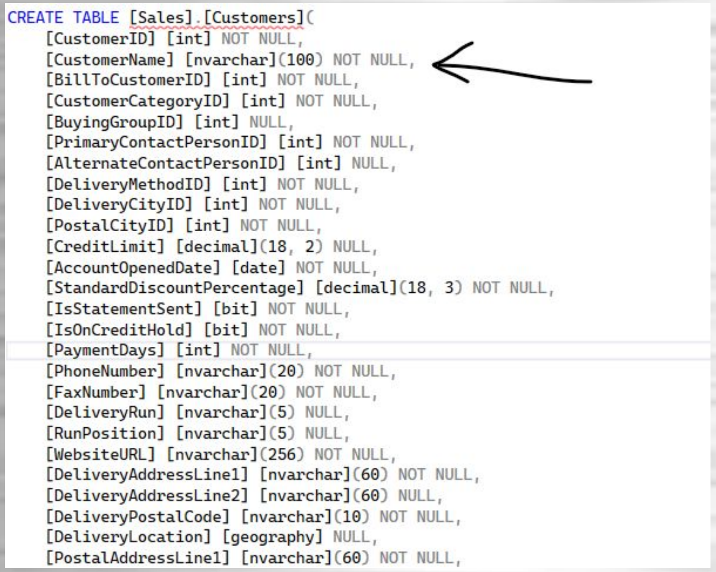
1.)Fully Qualified Domains and Table Names

2.)Occasional SQL Implementation

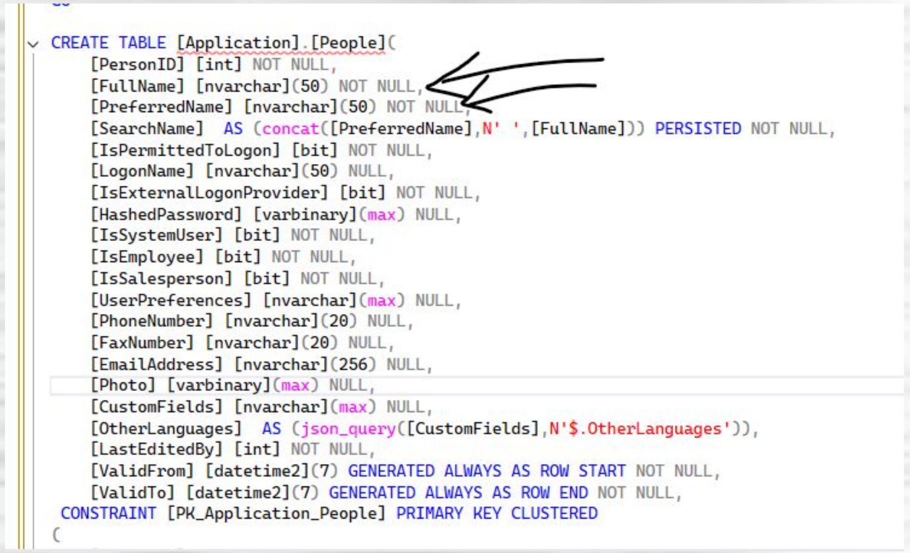
3.)The benefits and cool stuff about them

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
(
```

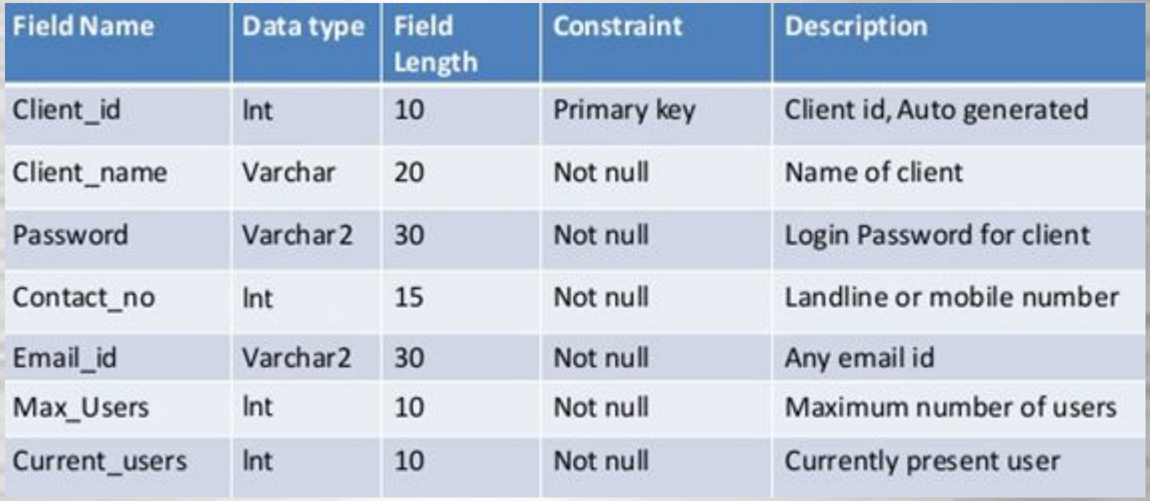
What is a Taxonomy?

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)



Field Name	Data type	Field Length	Constraint	Description
Client_id	Int	10	Primary key	Client id, Auto generated
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 number
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

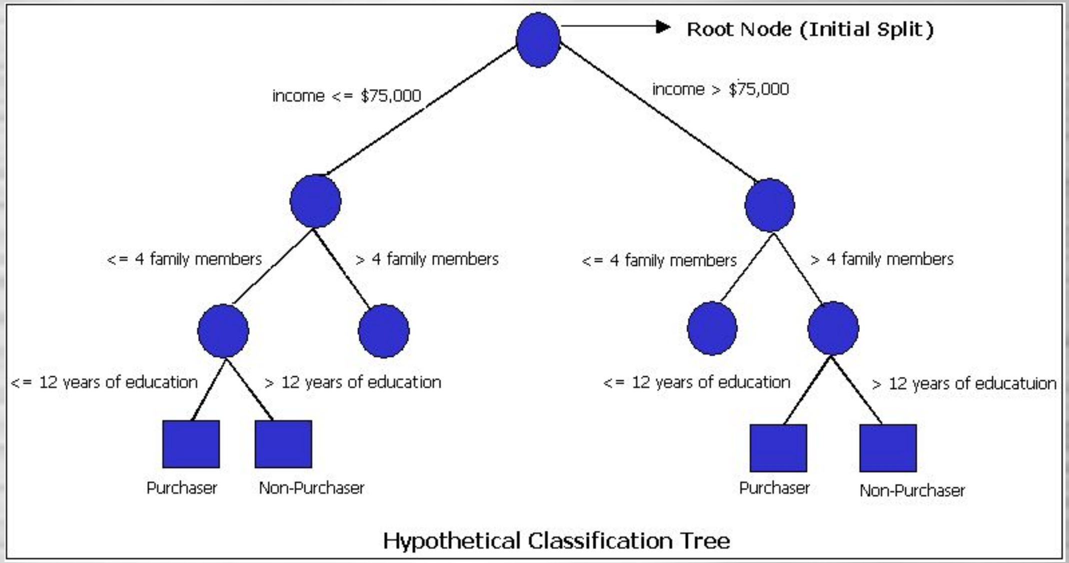


Image source: Classification Tree | solver

Part 1: Fully Qualified Domains

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`

What Makes a Domain "Fully Qualified"?

PhoneNumber

Just a Domain ("What")

Business.Contact.Email

Fully Qualified Domain ("What" + "Where")

It's a Domain + Taxonomy

A "Fully Qualified Domain" (FQD) is a domain whose name connects it directly to your business taxonomy.

The name `Application.Contact.Phone` tells you its exact place in the data dictionary. It's a defined, governed *business concept*.

Benefits of FQDs in a Taxonomy

It’s like a cheatcode or a template, allows for ease of use.

- **Reusable data definitions shared across systems**
- **Consistent rules and formats everywhere**
- **Central control for validation and updates**
- **Better data quality and governance**

Consistency

If `Application.People` and `Sales.Customers` both need a phone, they use the same FQD. No more `NVARCHAR(20)` vs. `VARCHAR(50)`.

Reusability

`Application.Contact.Phone` *once* in your taxonomy, and reuse it everywhere. This saves time and reduces errors.

Governance

Need to change phone number length? You have one central definition (the FQD) to update, making impact analysis simple.

Part 2: Fully Qualified Table Names

We know what a table name is… right?

```
SELECT * FROM Customers;
```

But where does `Customers` live? Is it `Sales.Customers`? Is it `Application.Customers`? Is it `dbo.Customers` in a different database?

WideWorldImporters

Sales Schema	Application Schema
`Sales.Customers` – Buyer and sales data	`Application.People` – Employee information
`Sales.Orders` – Purchase records	`Application.Countries` – Country references
`Sales.Invoices` – Payment tracking	`Application.Cities` – City references

The Anatomy of an FQTN

The Full, Unambiguous Address

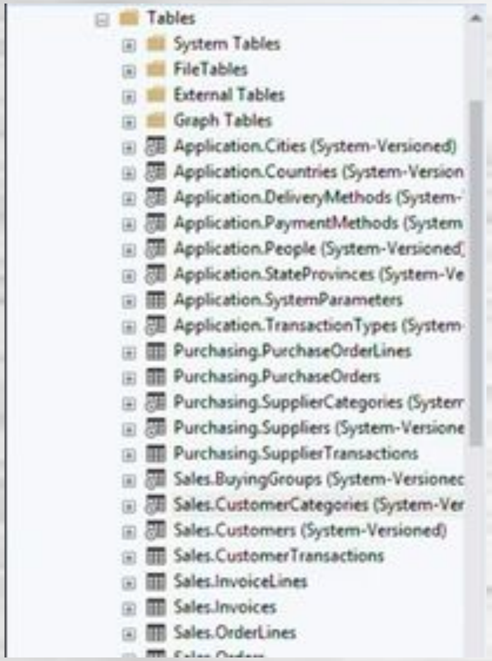
A Fully Qualified Table Name (FQTN) is the complete

"address" for your table, leaving no room for error.

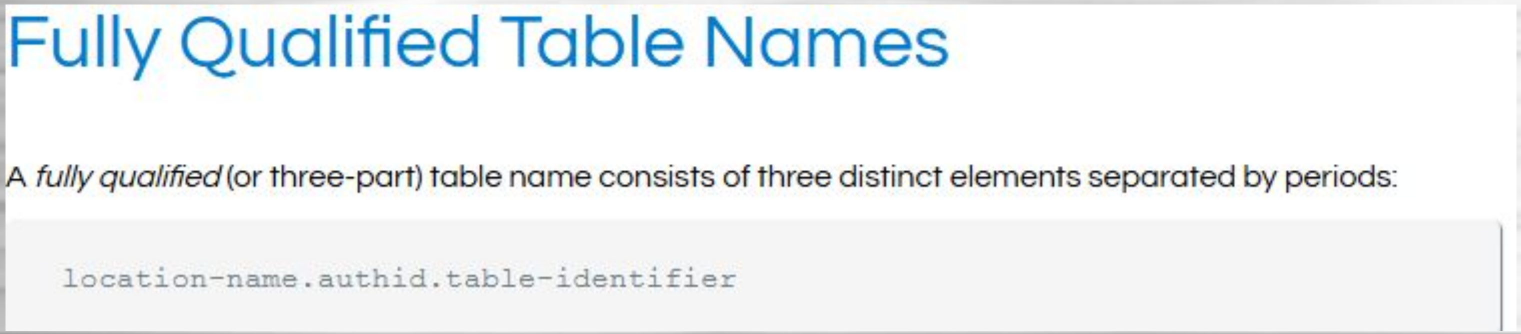
It's typically a three or four-part name:

- Location/Server: `[Docker server]`
- Database: `[WideWorldImporters]`
- Schema: `[Sales]`
- Table: `[Orders]

Database is optional .



```
⊟ Tables
  ⊞ System Tables
  ⊞ FileTables
  ⊞ External Tables
  ⊞ Graph Tables
  ⊞ Application.Cities (System-Versioned)
  ⊞ Application.Countries (System-Version
  ⊞ Application.DeliveryMethods (System-
  ⊞ Application.PaymentMethods (System
  ⊞ Application.People (System-Versioned
  ⊞ Application.StateProvinces (System-Ve
  ⊞ Application.SystemParameters
  ⊞ Application.TransactionTypes (System-
  ⊞ Purchasing.PurchaseOrderLines
  ⊞ Purchasing.PurchaseOrders
  ⊞ Purchasing.SupplierCategories (Systen
  ⊞ Purchasing.Suppliers (System-Versione
  ⊞ Purchasing.SupplierTransactions
  ⊞ Sales.BuyingGroups (System-Versionec
  ⊞ Sales.CustomerCategories (System-Ver
  ⊞ Sales.Customers (System-Versioned)
  ⊞ Sales.CustomerTransactions
  ⊞ Sales.InvoiceLines
  ⊞ Sales.Invoices
  ⊞ Sales.OrderLines
```



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
```

Why FQTNs are So Important

Clarity: `[WideWorldImporters].[Sales].[Orders]` leaves **zero** doubt. This is crucial for scripts and production code.

Scalability (Heterogeneous): Your code must use FQTNs when joining data across different databases or servers.

Referential Integrity: It helps ensure foreign keys are pointing to the *exact* right table, rather than a copy on a different schema.

Real-World Example: Heterogeneous Environments

This query works *because* it uses FQTNs (two-part names) to find tables in different schemas.

```
USE WideWorldImporters;

SELECT o.OrderID, c.CustomerName,
 od.StockItemID, od.Quantity
FROM Sales.Orders AS o
INNER JOIN Sales.Customers AS c
ON o.CustomerID = c.CustomerID
       INNER JOIN Sales.OrderLines AS od
    ON o.OrderID = od.OrderID;
```

↔

This query joins tables from the `Sales` schema. What if you also needed the employee who sold it? You'd join to `Application.People`.

These two-part names (`Schema.Table`) are the most common form of FQTN you'll use.

Real-World Example: Heterogeneous Environments

This is where 4-part FQTNs are *required*. Imagine joining WorldWideImprters to an old archive database on a different server.

```
/* This query CANNOT run without 4-part names
*/
SELECT
    live.CustomerName,
    arc.OldOrderDate
FROM

[ProdServer].[WideWorldImporters].[Sales].[Cust
omers] AS live
INNER JOIN
    [ArchiveServer].[WWI_Old].[dbo].[Orders] AS
arc
    ON live.CustomerID = arc.LegacyCustomerID;
```

↔

Here, you are explicitly telling SQL:

- What server to find?
- What database on that server?
- What schema in that database?
- What table in that schema?

Tying It All Together (Wrap Up)

How It All Connects

To sum things up:

A strong governance strategy uses all three to create a data ecosystem that is consistent, scalable, and easy to manage.

- The Taxonomy is the "Dictionary."
- FQDs are the "Definitions" (the words).
- FQTNs are the "Addresses" (the page numbers).

Conclusion

Fully Qualified Domains are like what Proffesor Heller said before in a lecture where they can be like lego blocks, and Fully Qualified Table Names are the clear addresses/blueprints for what you’re building. This is the foundation that lets you build complex, scalable, and reliable applications within sql no matter what your doing or your skill level at the language.

References/Sources

Big thanks to these articles and videos from the web:

What is a SQL Server Data Dictionary and why would I want to create

What does fully qualified name means in SQL? | Sololearn: Learn to code for FREE!

Micro Focus Documentation

https://www.f5.com/glossary/fully-qualified-domain-name-fqdn
Fully Qualified Table Names

SQL Server 13 - Domain Integrity

What is Data Taxonomy? Examples Included | Amplitude