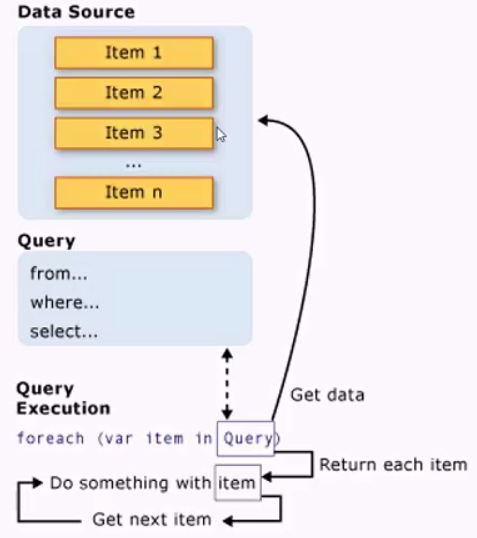
**Entity Framework. LINQ**

**Agenda**

* LINQ to Entities Overview
* CRUD Operations
* Lambda Expressions
* SQL Queries
* Store Procedures
* Table Value functions
* Entity States
* Entity Detach/Attach
* Manage Concurrency
* Validation
* New Features
* Performance

**LINQ to Entities Overview**

* CRUD operation from a data source
* Obtain the data source
* Create the query
* Execute the query

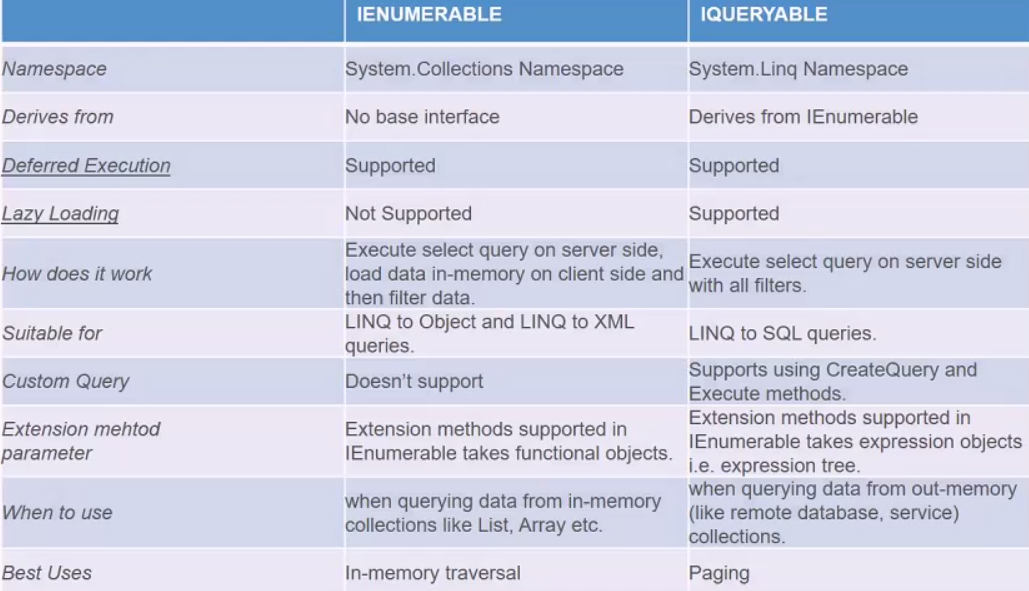


**LINQ to Entities Query**

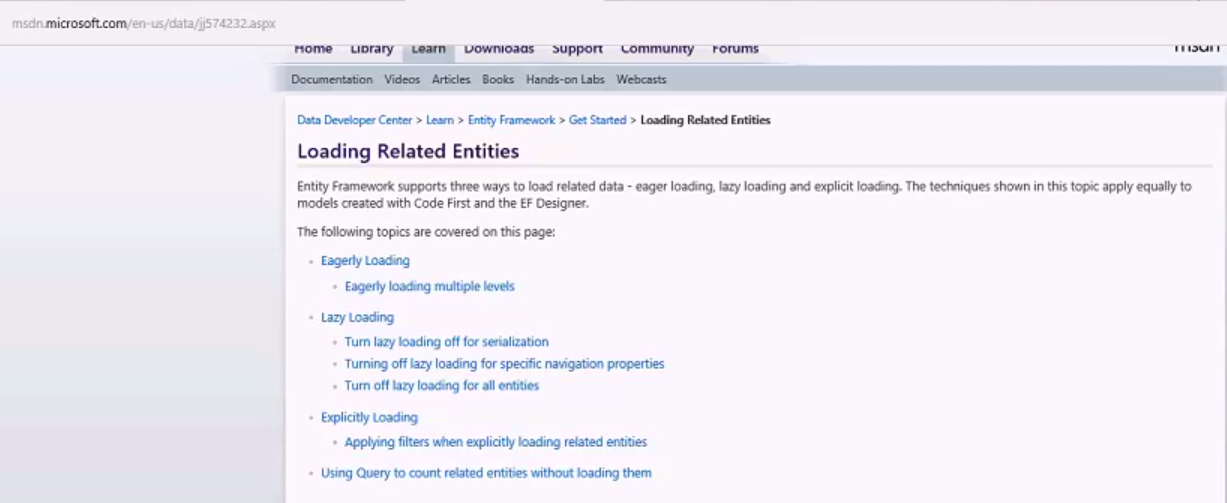
Query Results

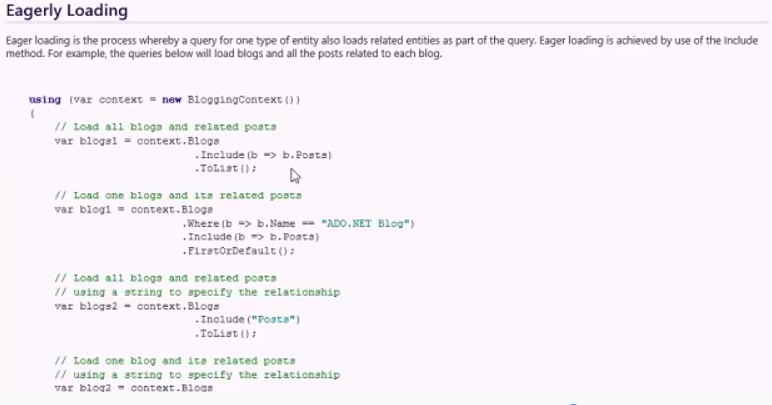
* Data source implicitly supports IEnumerable <T>
* Foreach support queryable types
* IQueryable <T> - derives from IEnumerable <T>
* IEnumerable <T>

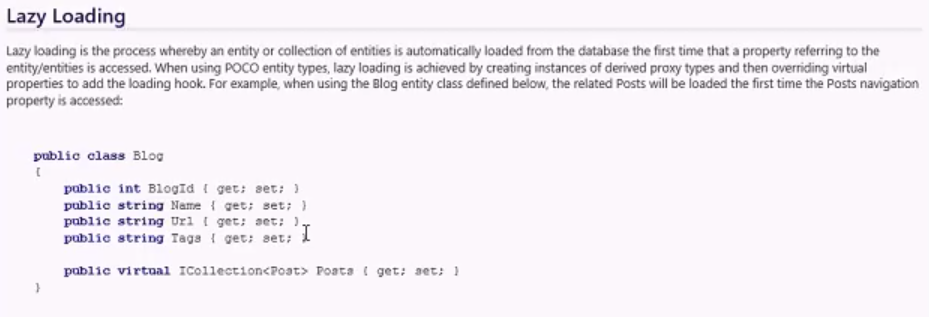
**IEnumerable and IQuery**

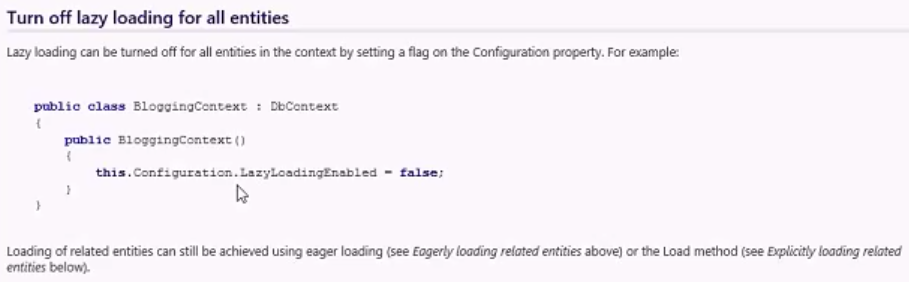


Website: msdn.microsoft.com/en-us/data/jj574232.aspx





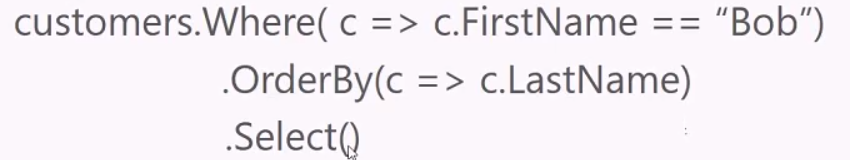




**Expression Trees**

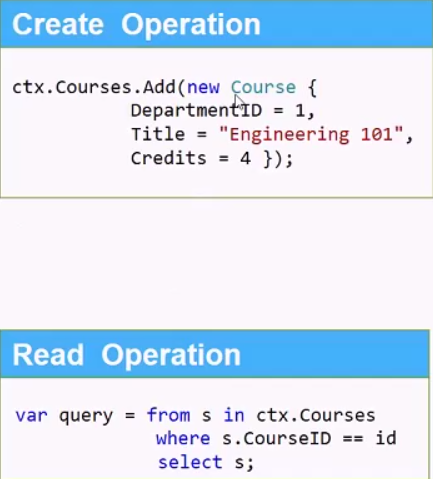
LINQ

* Where
* OrderBy
* Select

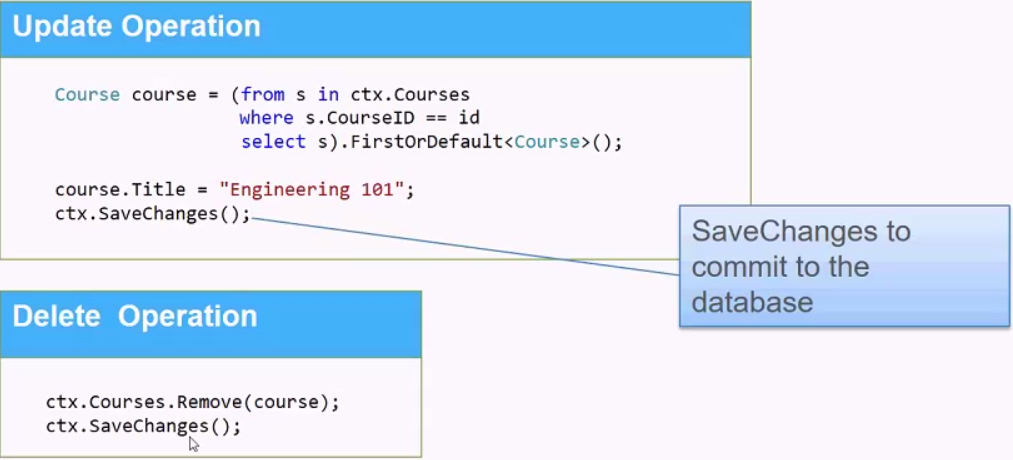


**CRUD Operations**

* Add/AddRange
* Read



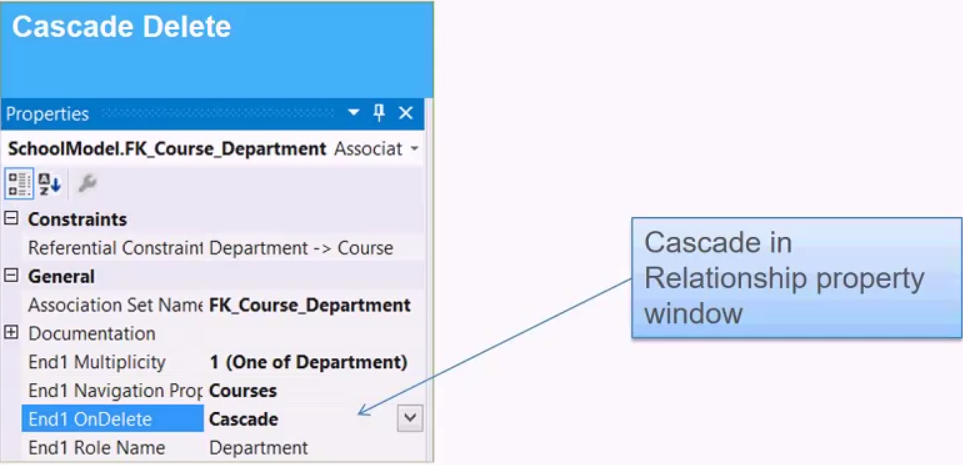
* Update
* Remove/RemoveRange



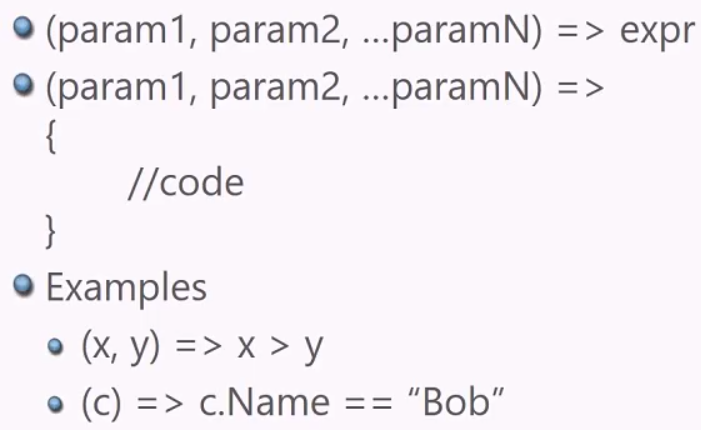
**Cascade Delete**

CRUD Operations

* From Entity Model select the relationship



**Lambda Expressions**



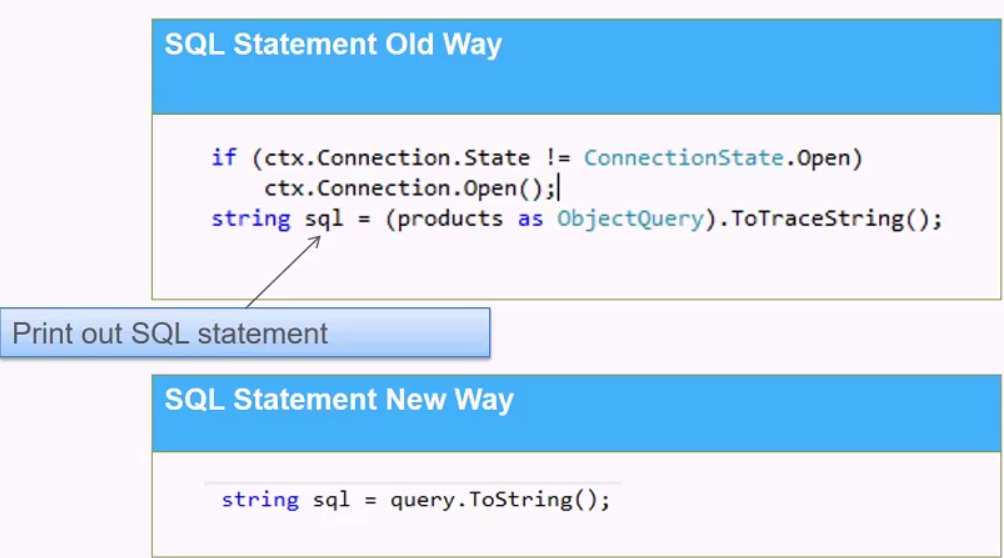
LINQ

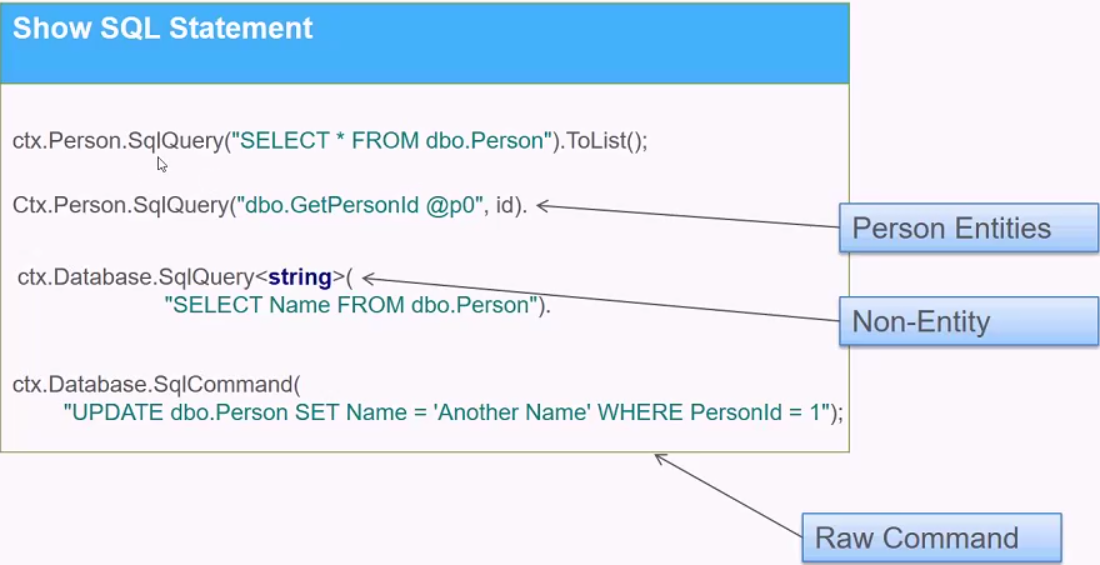
More examples



**SQL Statement**

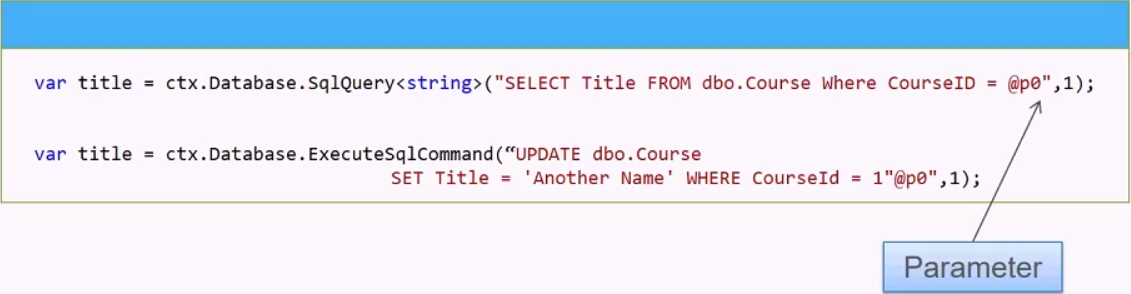
View generated SQL Statement





**Parameterized**

* Classic parameters for ADO.NET
* Replace @ with ? for OLEDB
* SQLQuery for querying – supports Async
* ExecuteSqlCommand for non-queries



**Custom Procedure Performance**

* Performance TPT
* Access property in base class automatically joins



* SQL Injections
* Use SqlParameter for SQL server and OleDBParameter for OLE DB

**Know Custom Procedures Pitfalls**

Need to know

* Columns are automatically mapped based on property-column name match
* Property name is different, solved by renaming the column in the query with AS SQL clause
* Cannot map complex properties



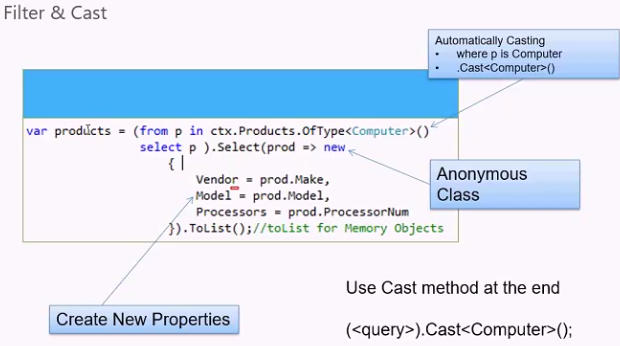
* Bypasses the EDM – Performance TPT
* Not Recommended, use stored procedures.

LINQPad 5:

See samples

**Projection & Casting**

LINQ



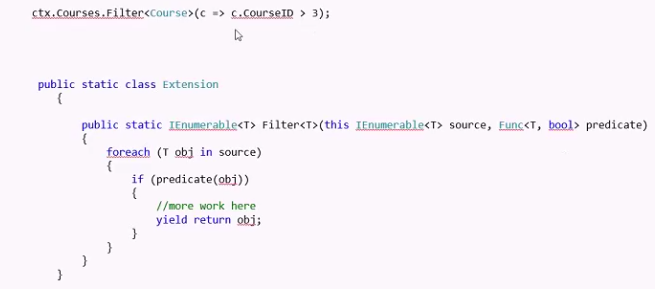
**Entity State**

* Added
* Unchanged
* Modified
* Detached
* SaveChanges
* Unchanged – not touched
* Added – Entity inserted
* Modified – Entity updated
* Deleted – Entity deleted

**Extension Methods**

Advance Features

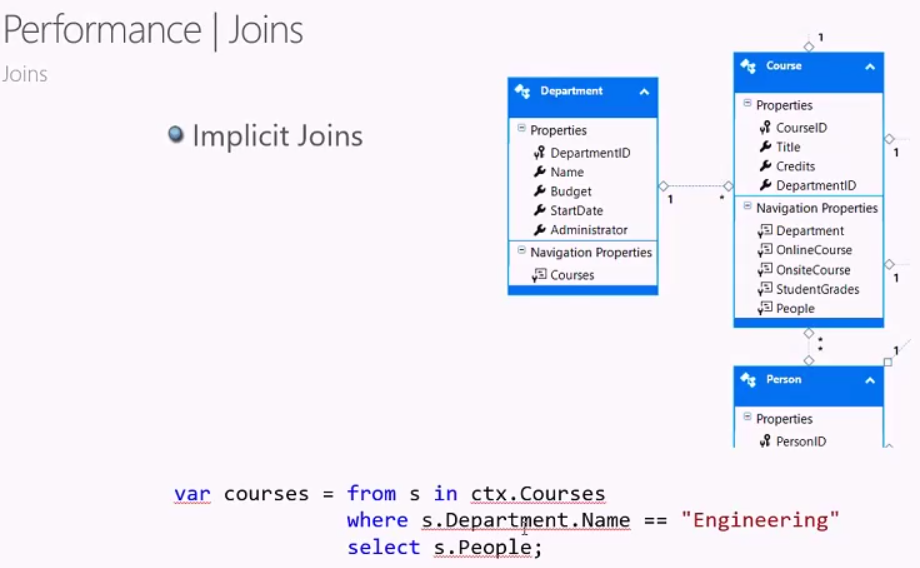
* Looks like a method call on the object

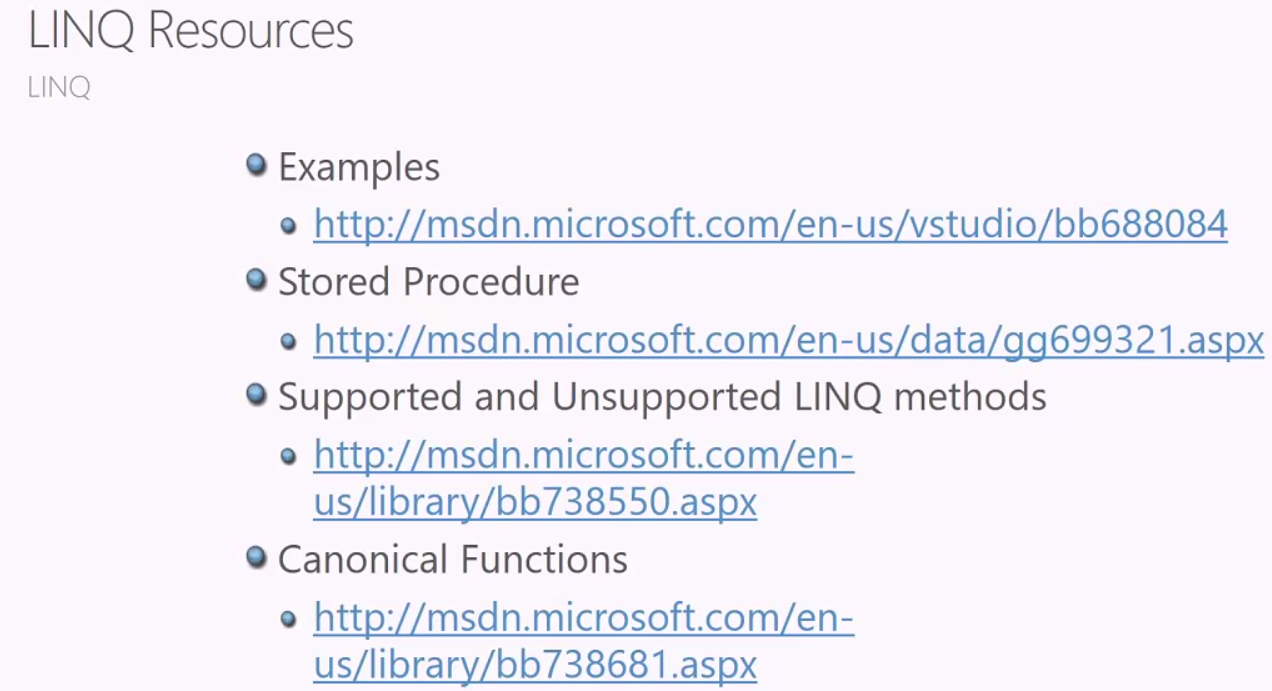


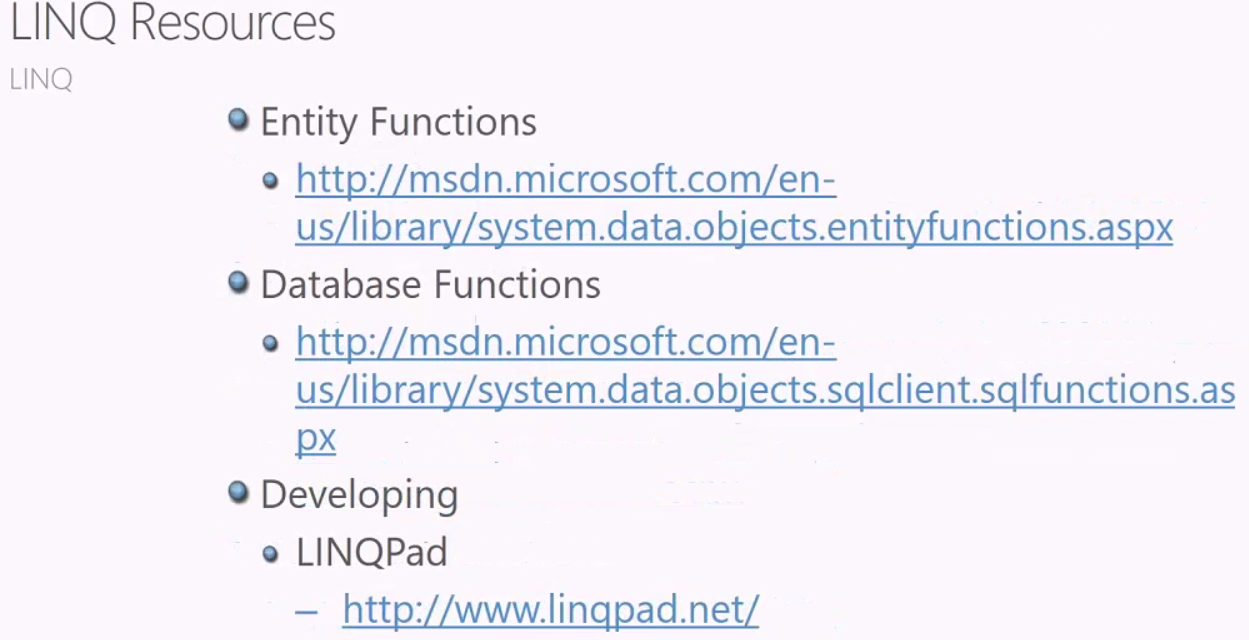
**DatabaseFunciton**

LINQ

* SQL Server functions supported
* Implications
* Tie only to one brand of database

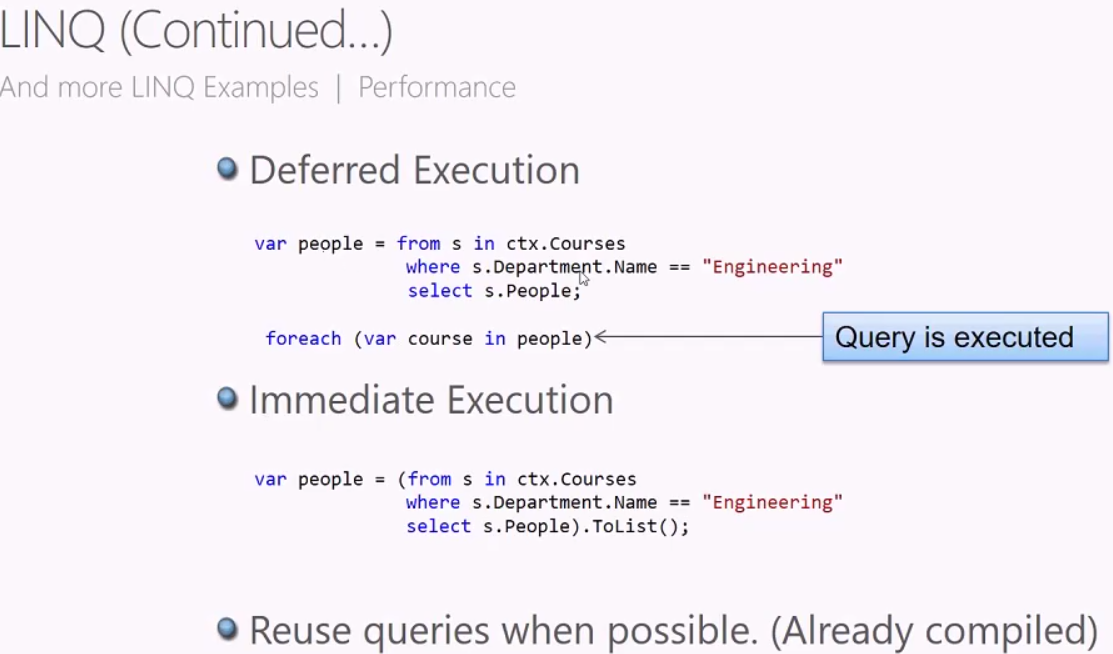






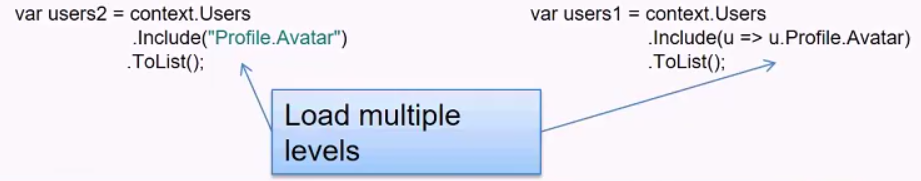
**Lazy Loading**

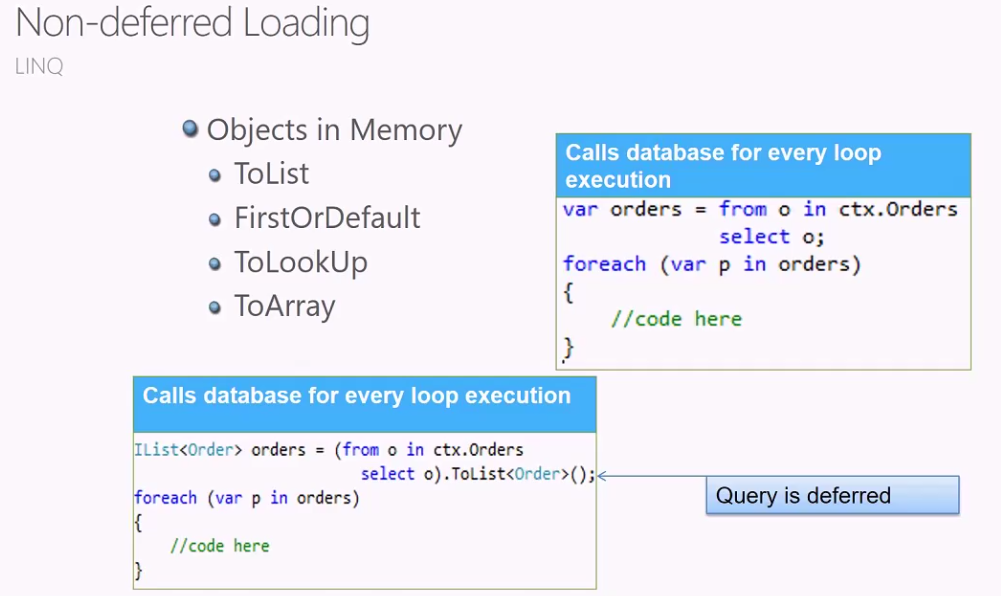
* Load when the data is needed
* Default
* Virtual navigation properties – Not loaded
* Turn Off
* Ctx.Configuration.LazyLoadingEnabled = **false**
* Serialization
* No Proxy – no lazy loading

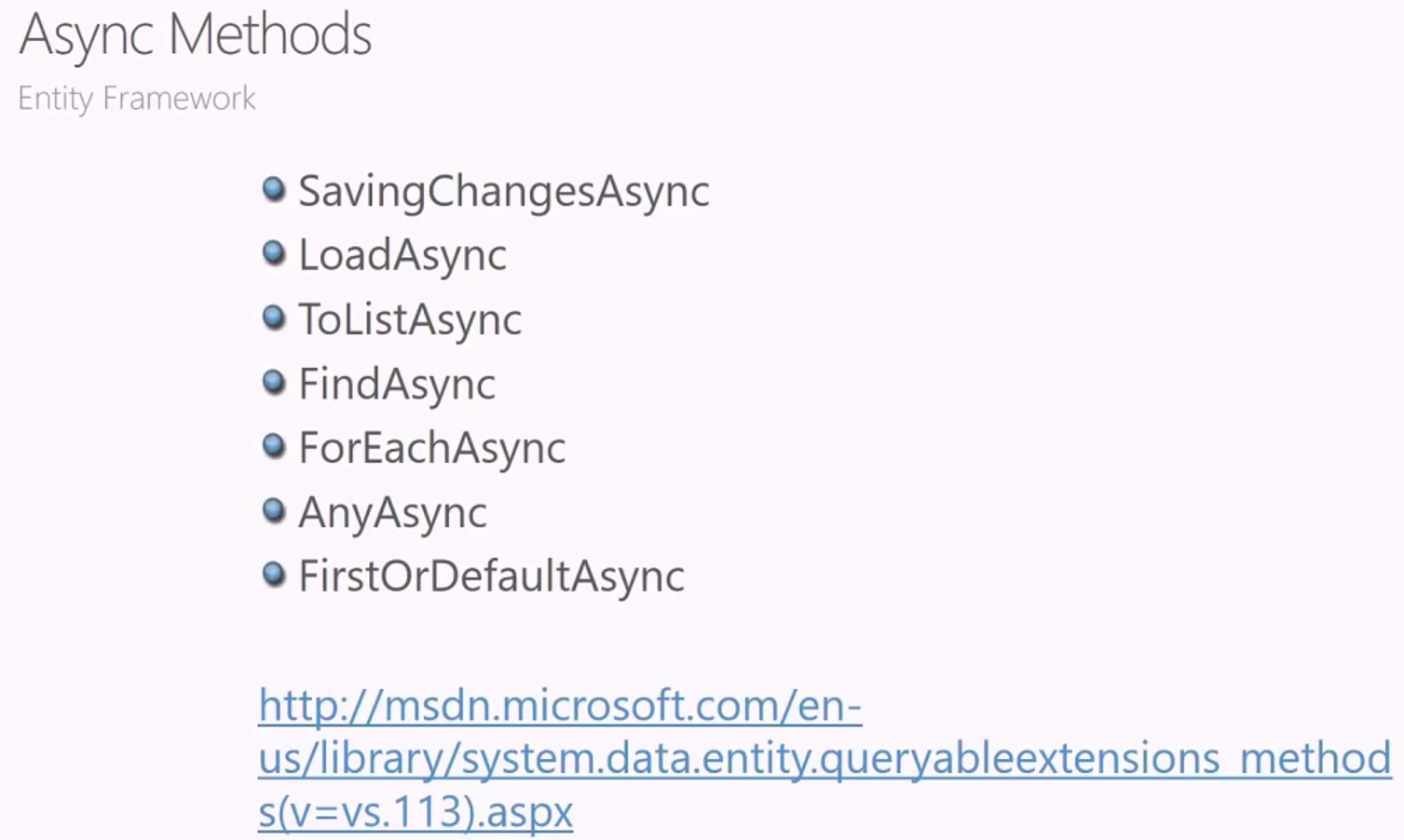


**Eager Loading**

* Load all the data when first called
* Better performance – when multiple calls. 1 RTT
* Include method
* ctx.Orders.Include(“OrderDetails”)
* ctx.Orders.Include(“OrderDetails.Product”)
* ToList()
* FirstOrDefault()







**Concurrency Exception**

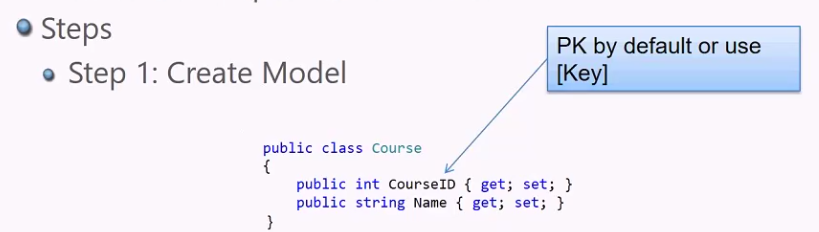
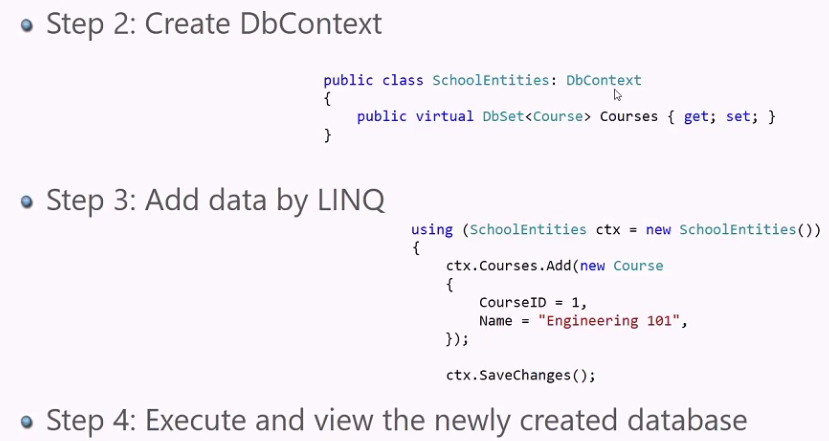
Optimistic Concurrency Patterns

* Exception is thrown when changes have occurred
* DbUpdateConcurrencyException
* Resolve conflict before saving changes
* Database wins
* Client wins
* Custom

**Code First**

New Database Sample

* SQL Express First (VS 2010)
* LocalDb (VS 2012)
* Named fully qualified name
* CodeFirstSample.SchoolContext

**Model Changes**

New Database

* Steps
* Enable – Migrations – Adds Configuration and Initial Create database migration classes.
* EF 6.1
* Add – Migration <name> after modifications
* Update – Database
* Fluent API
* Annotation

