Is a SQL Database Just a Store?

Rudi Bruchez

SQL SERVER CONSULTANT AND TRAINER

www.linkedin.com/in/rudibruchez/ www.babaluga.com

Summary

What kind of store is a RDBMS?

Where to put the application logic?

Should we use Entity Framework?

Are stored procedures too old-school?

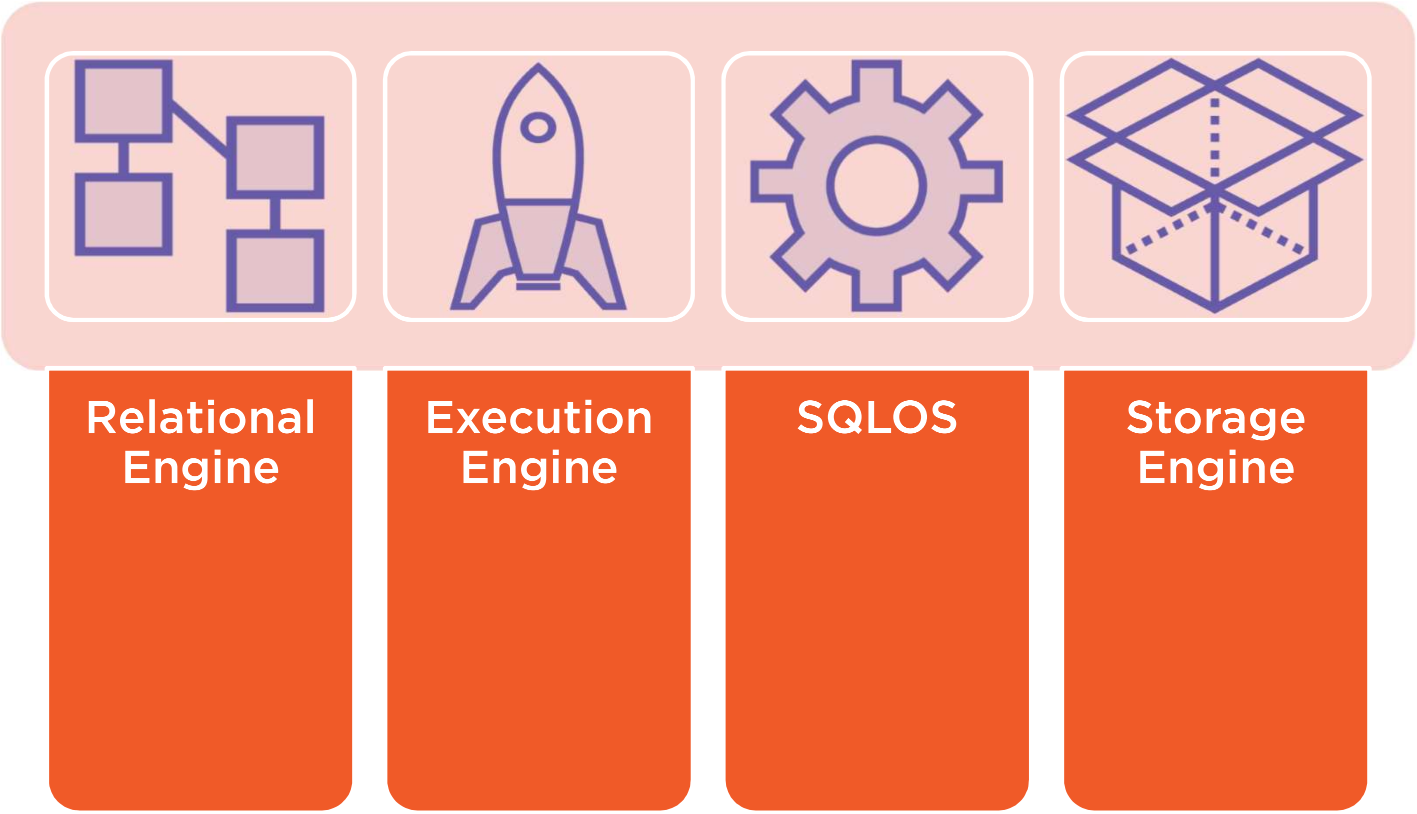
A RDBMS is …

A cupboard?
To store your gigabytes of data

A dummy store
The real intelligence is on the client side

Or could it be…
A well-crafted piece of technology

SQL Server Is Composed Of…



Relational Engine

Execution Engine

SQLOS

Storage Engine

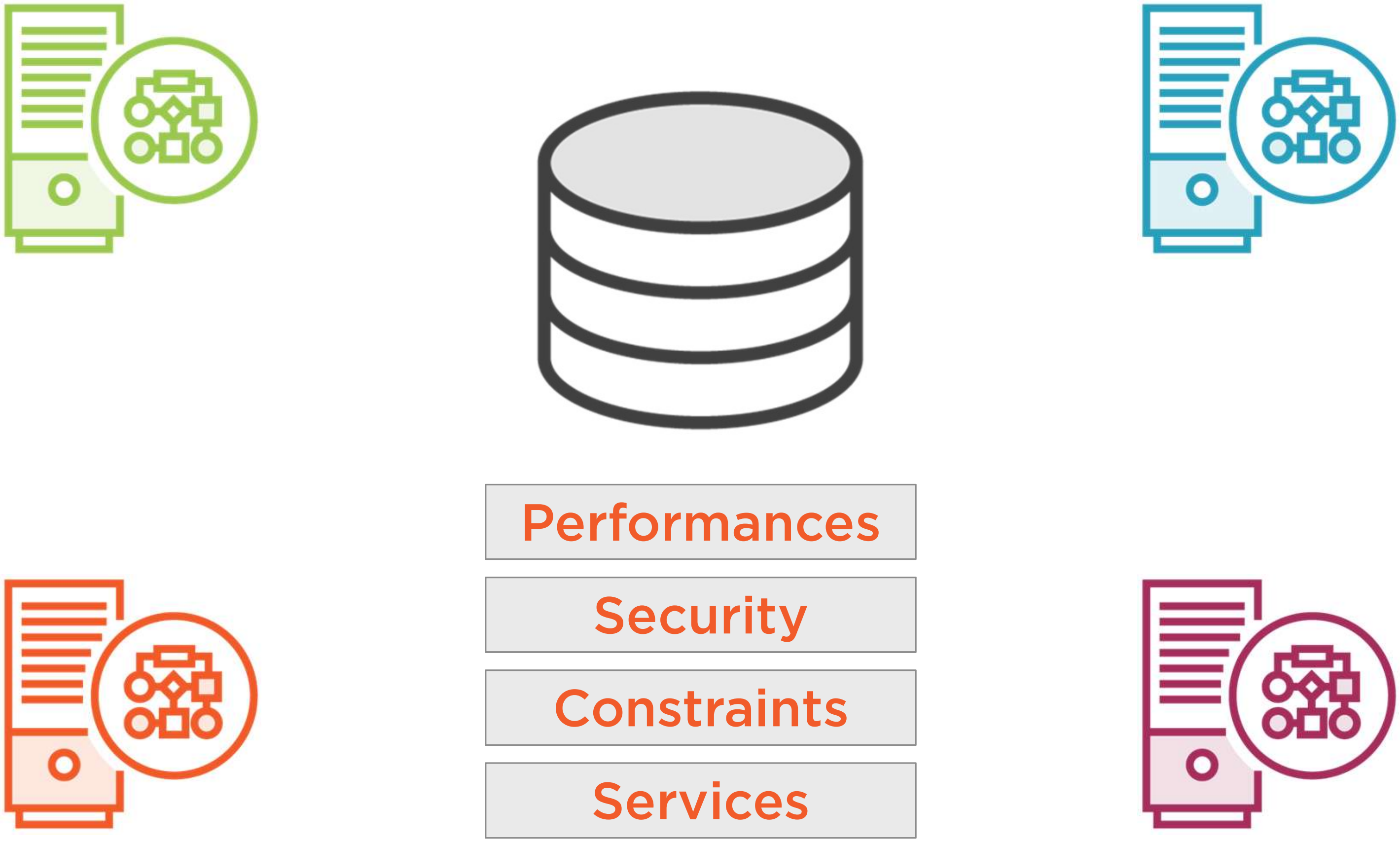
The Business Rules Dispute

The interface
It only shows off

The business layer
The only one working here

The database
Just a box

Ok, but Integration?



Performances

Security

Constraints

Services

Demo

Lack of visibility on the database side

SQL gets encapsulated inside classes

Think declaratively!

Consider the database perspective

Profile in SQL Server

Extended Events

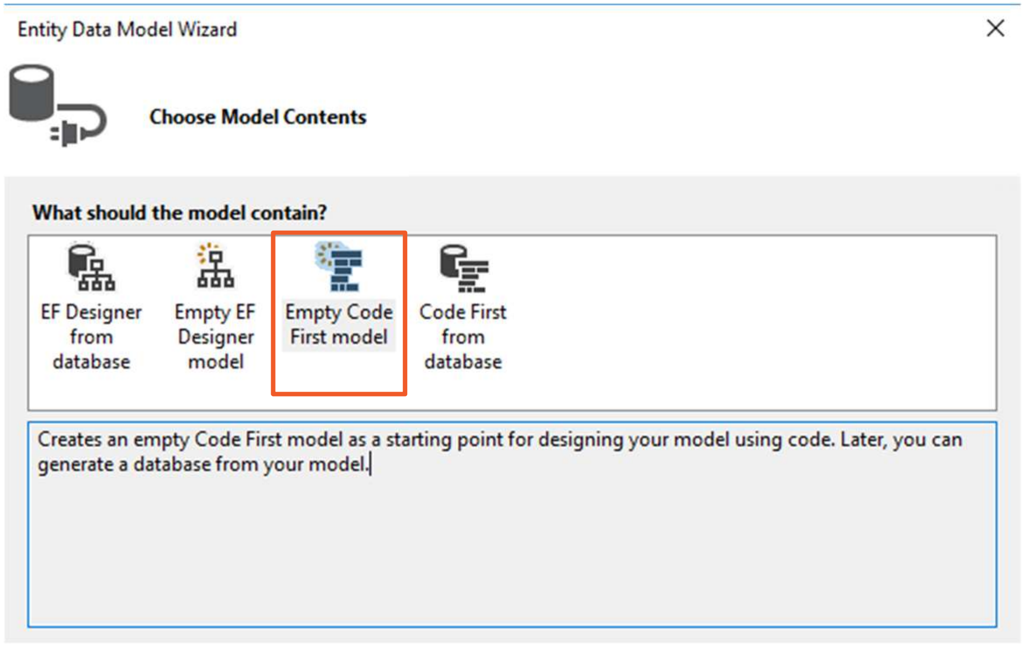
What Is Entity Framework?

ORM
Object-relational
mapper

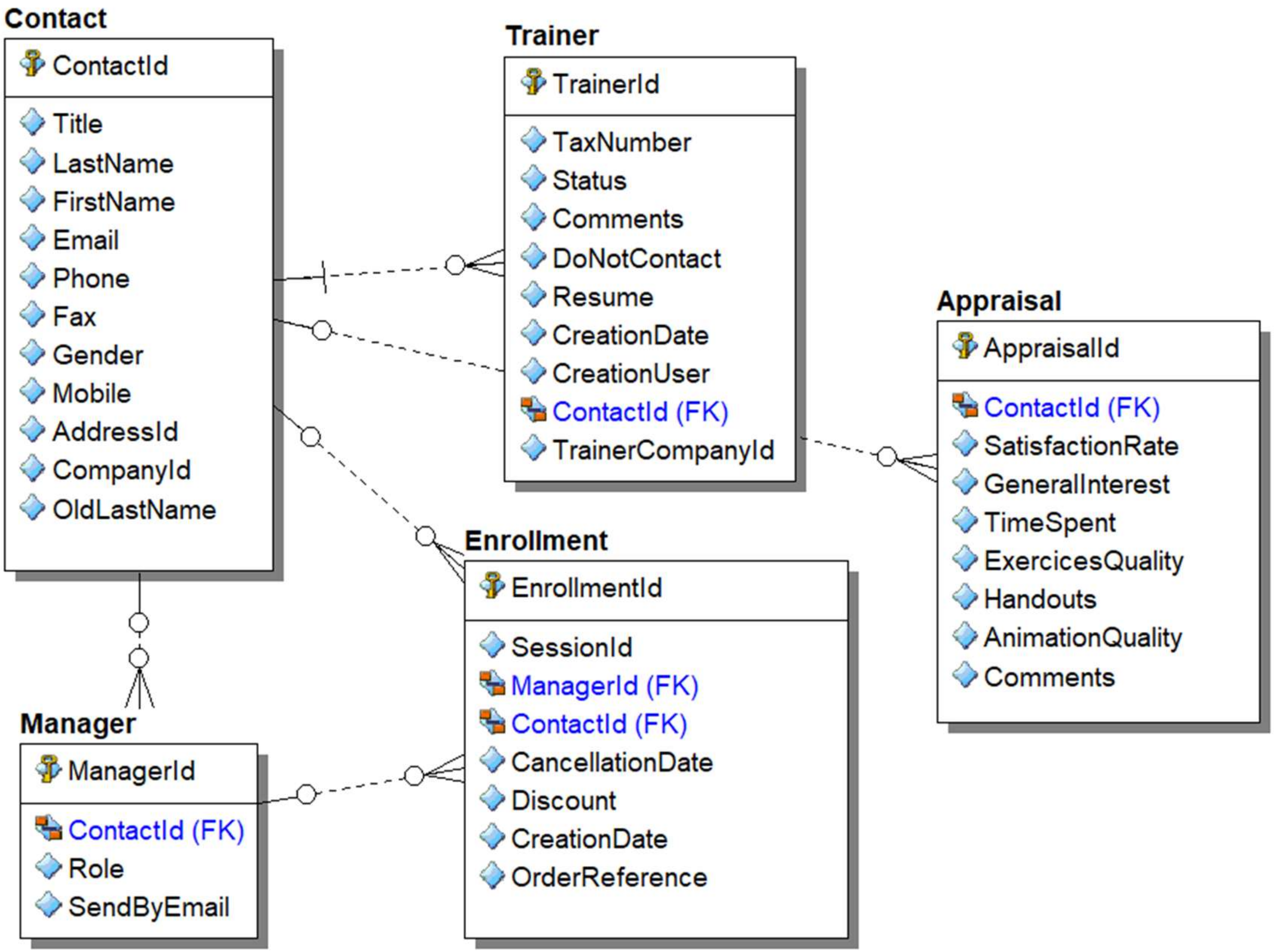
Model generator
Model First
or Code First

Code generator
Free you from
writing SQL

Entity Framework Code First



Creates an empty Code First model as a starting point for designing your model using code. Later, you can generate a database from your model.



Contact

ContactId

- Title
- LastName
- FirstName
- Email
- Phone
- Fax
- Gender
- Mobile
- AddressId
- CompanyId
- OldLastName

Trainer

TrainerId

- TaxNumber
- Status
- Comments
- DoNotContact
- Resume
- CreationDate
- CreationUser
- ContactId (FK)
- TrainerCompanyId

Appraisal

AppraisalId

- ContactId (FK)
- SatisfactionRate
- GeneralInterest
- TimeSpent
- ExercicesQuality
- Handouts
- AnimationQuality
- Comments

Enrollment

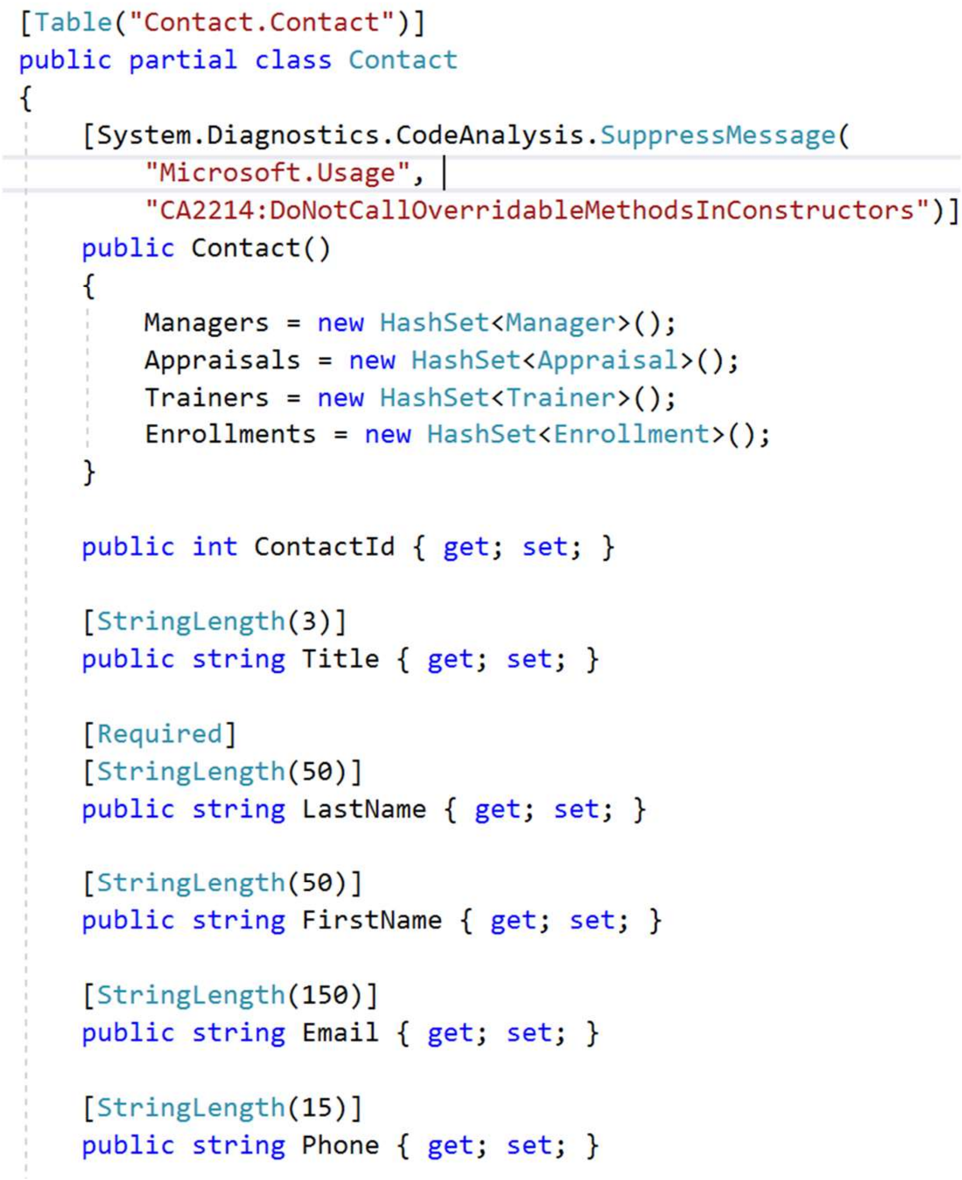
EnrollmentId

- SessionId
- ManagerId (FK)
- ContactId (FK)
- CancellationDate
- Discount
- CreationDate
- OrderReference

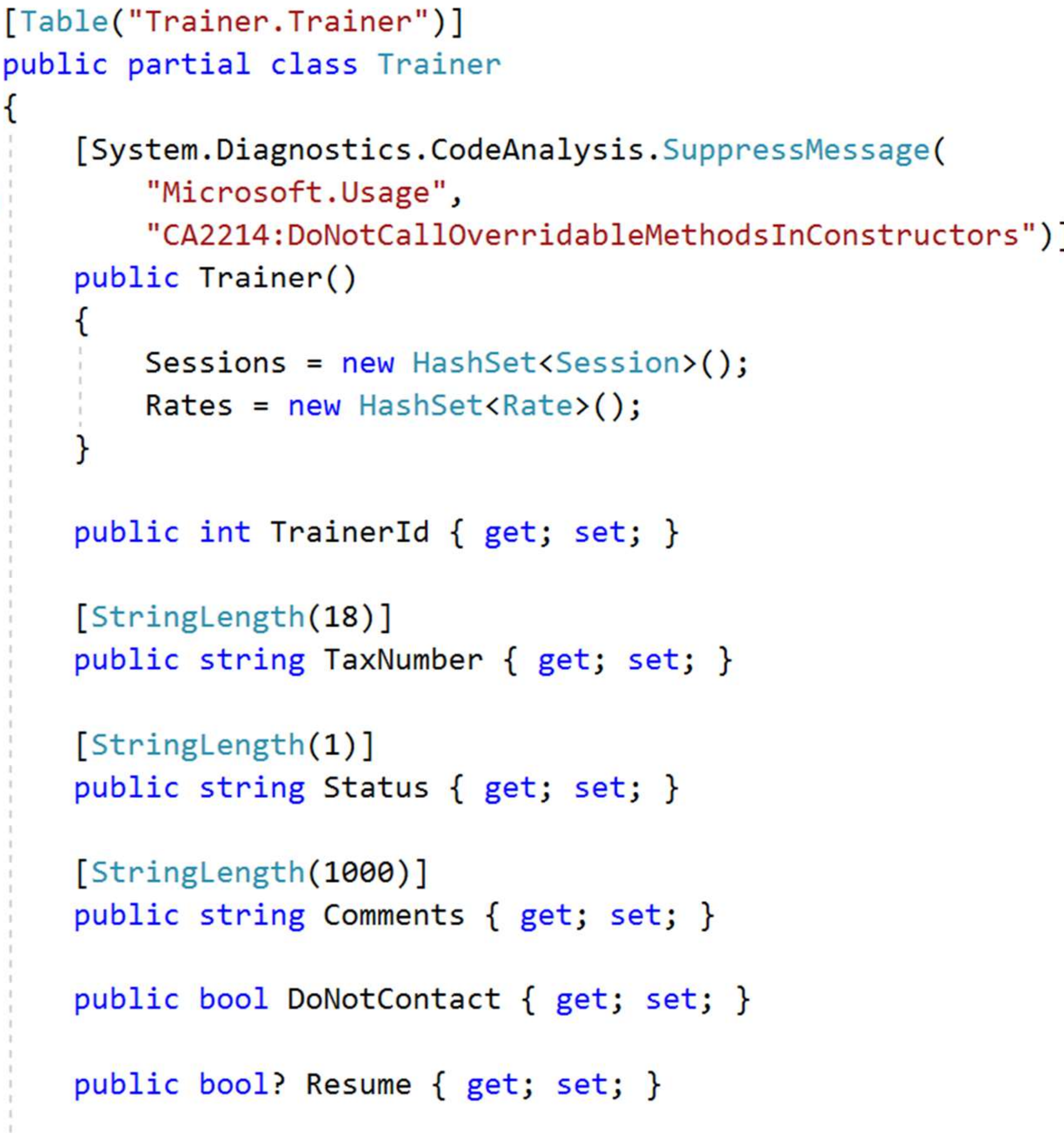
Manager

ManagerId

- ContactId (FK)
- Role
- SendByEmail

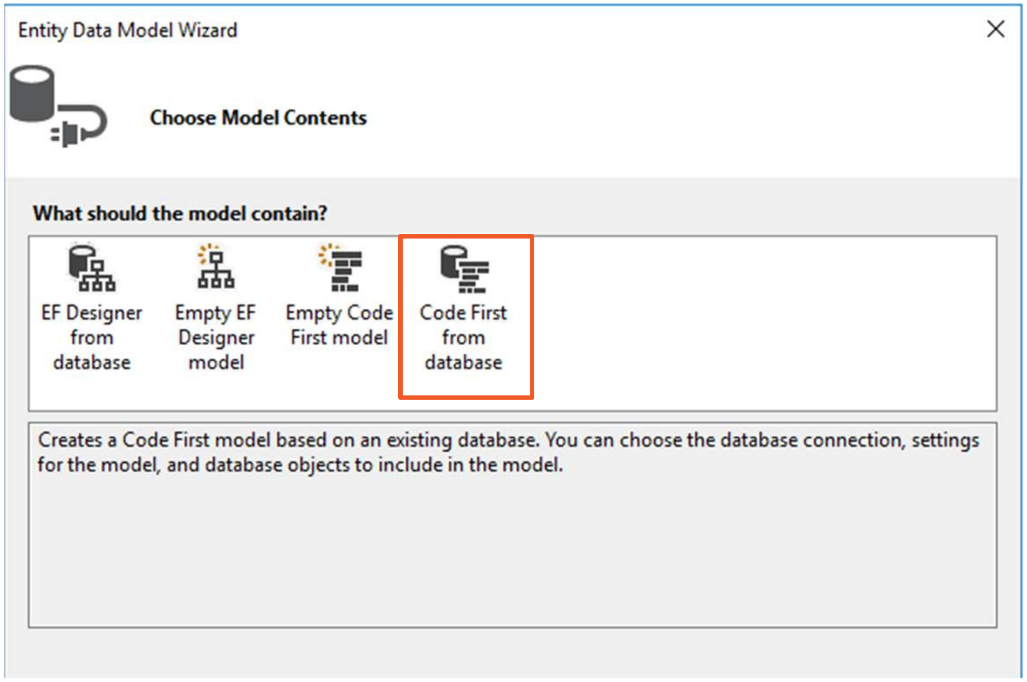


```
[Table("Contact.Contact")]
public partial class Contact
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage(
        "Microsoft.Usage",
        "CA2214:DoNotCallOverridableMethodsInConstructors")]
    public Contact()
    {
        Managers = new HashSet<Manager>();
        Appraisals = new HashSet<Appraisal>();
        Trainers = new HashSet<Trainer>();
        Enrollments = new HashSet<Enrollment>();
    }

    public int ContactId { get; set; }

    [StringLength(3)]
    public string Title { get; set; }

    [Required]
    [StringLength(50)]
    public string LastName { get; set; }

    [StringLength(50)]
    public string FirstName { get; set; }

    [StringLength(150)]
    public string Email { get; set; }

    [StringLength(15)]
    public string Phone { get; set; }
```

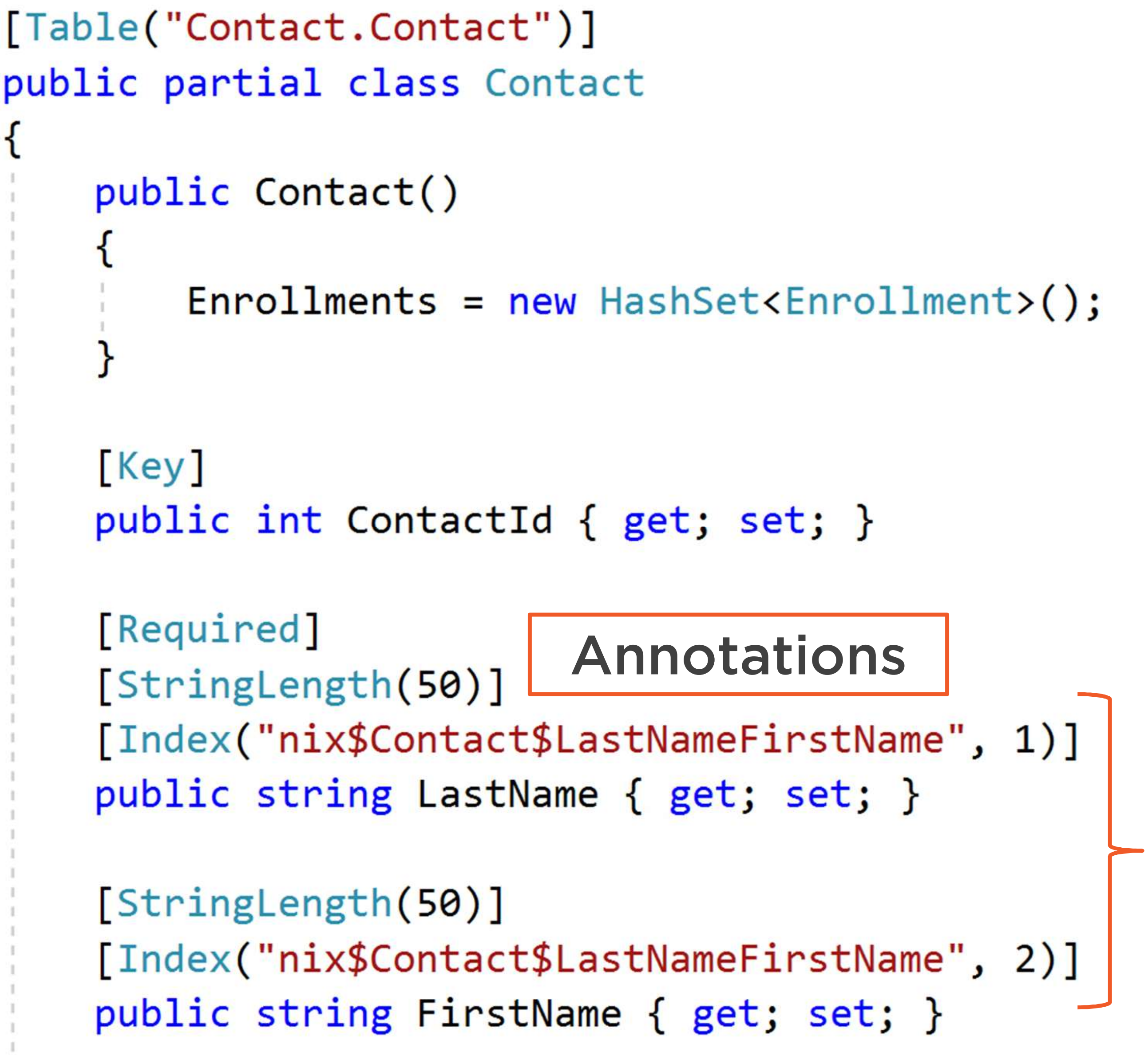


```
[Table("Trainer.Trainer")]
public partial class Trainer
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage(
        "Microsoft.Usage",
        "CA2214:DoNotCallOverridableMethodsInConstructors")]
    public Trainer()
    {
        Sessions = new HashSet<Session>();
        Rates = new HashSet<Rate>();
    }

    public int TrainerId { get; set; }

    [StringLength(18)]
    public string TaxNumber { get; set; }

    [StringLength(1)]
    public string Status { get; set; }

    [StringLength(1000)]
    public string Comments { get; set; }

    public bool DoNotContact { get; set; }

    public bool? Resume { get; set; }
```

Code First from Database



Creates a Code First model based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model.



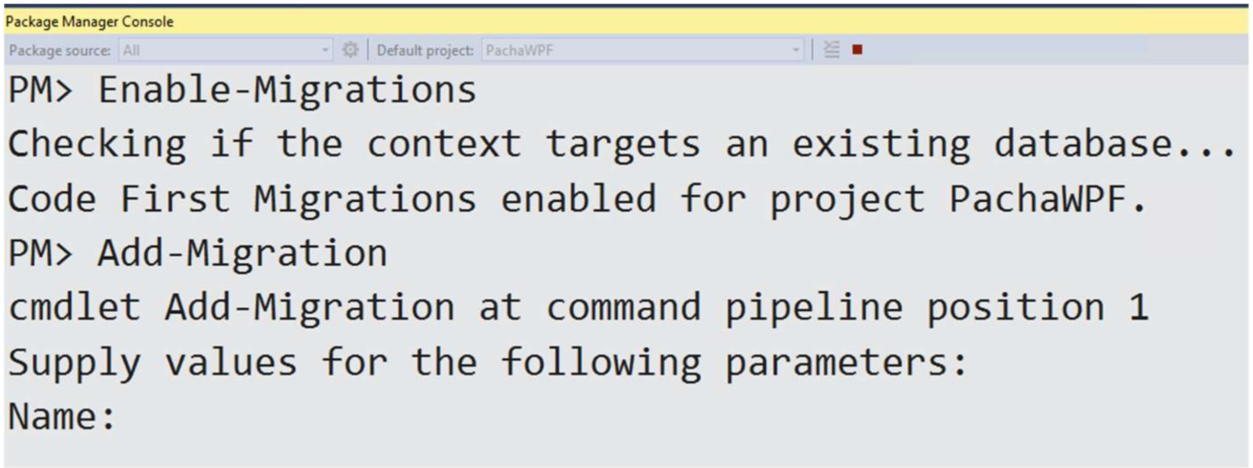
```
[Table("Contact.Contact")]
public partial class Contact
{
    public Contact()
    {
        Enrollments = new HashSet<Enrollment>();
    }

    [Key]
    public int ContactId { get; set; }

    [Required]
    [StringLength(50)]
    [Index("nix$Contact$LastNameFirstName", 1)]
    public string LastName { get; set; }

    [StringLength(50)]
    [Index("nix$Contact$LastNameFirstName", 2)]
    public string FirstName { get; set; }
```

Annotations

Compound index

```
[StringLength(150)]
[EmailAddress(ErrorMessage = "Invalid Email!")]
public string Email { get; set; }

[StringLength(15)]
[RegularExpression(@"((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}",
    ErrorMessage = "Invalid Phone Number!")]
public string Phone { get; set; }
```

Automatic Migrations



```
PM> Enable-Migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project PachaWPF.
PM> Add-Migration
cmdlet Add-Migration at command pipeline position 1
Supply values for the following parameters:
Name:
```

Mixing Layers

The database
Give me the business!

The business layer
Give me the data!

We Love SQL

Tricky at first

Simpler at the end

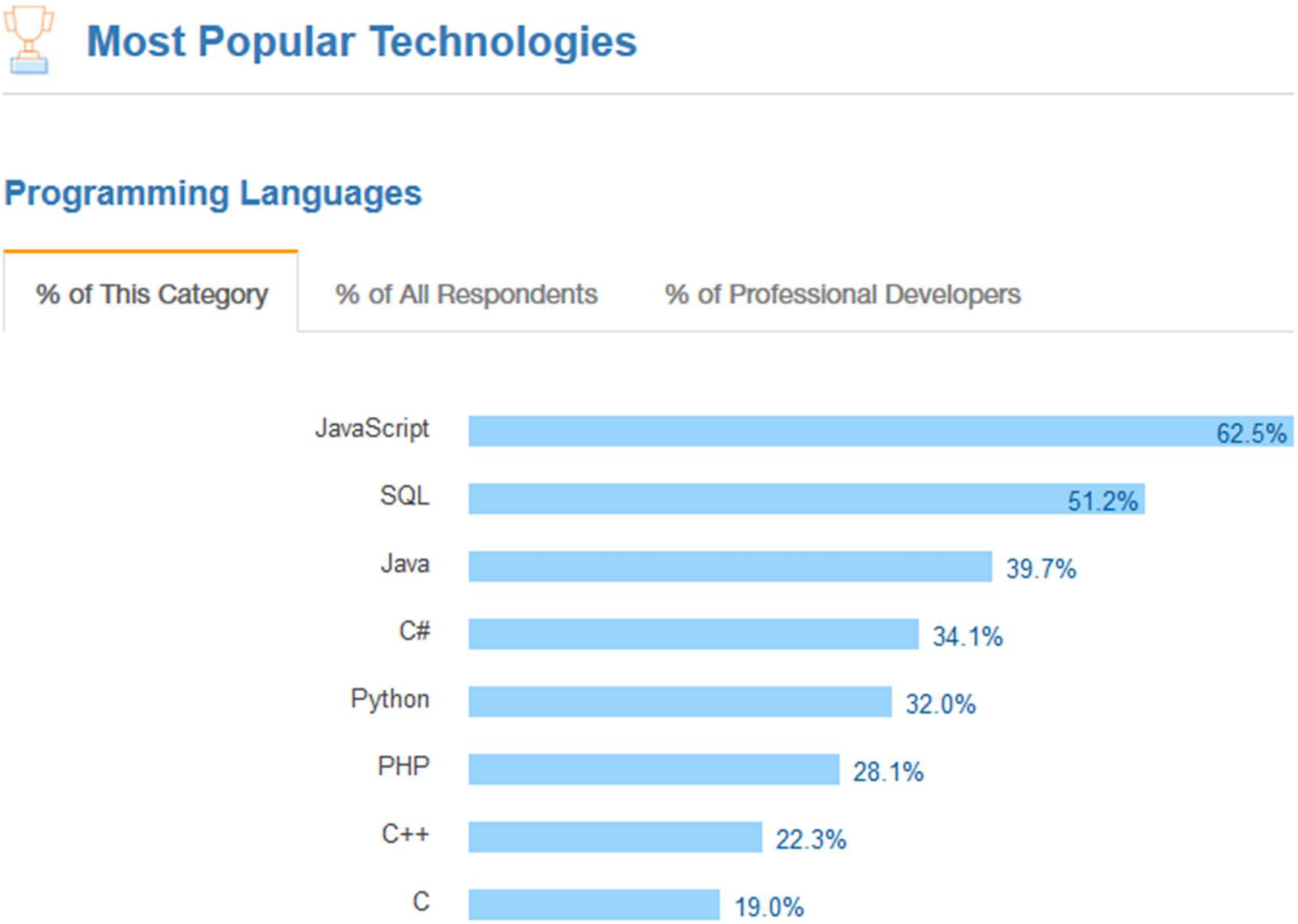
Concentrate on the question

“SQL, Lisp, and Haskell are the only programming languages that I’ve seen where one spends more time thinking than typing”

Philip Greenspun

http://blogs.harvard.edu/philg/2005/03/07/how-long-is-the-average-internet-discussion-forum-posting/

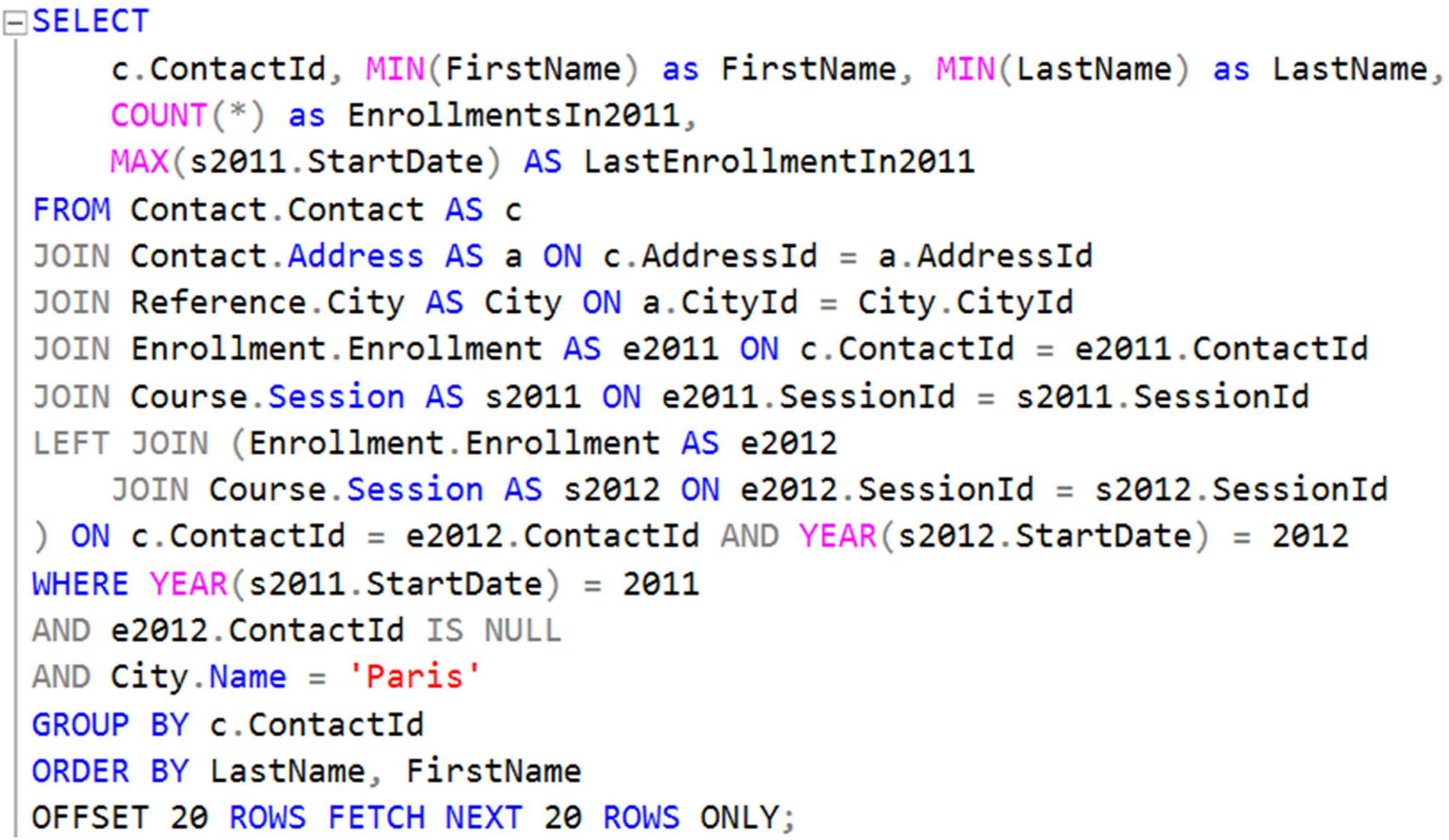
(https://bit.ly/2ujQMXQ)



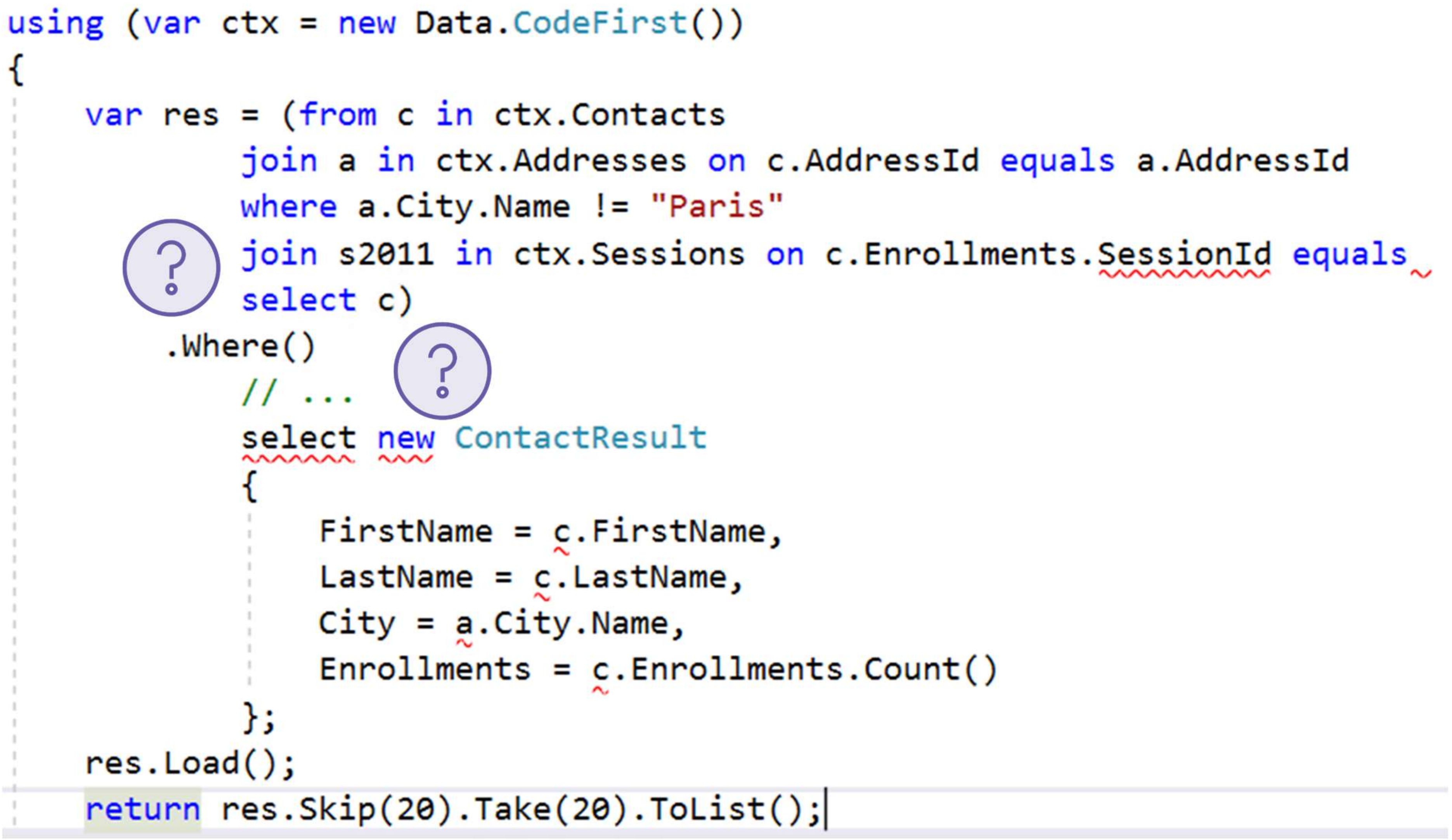
Most Popular Technologies

Programming Languages

https://insights.stackoverflow.com/survey/2017#technology



```sql
SELECT
    c.ContactId, MIN(FirstName) as FirstName, MIN(LastName) as LastName,
    COUNT(*) as EnrollmentsIn2011,
    MAX(s2011.StartDate) AS LastEnrollmentIn2011
FROM Contact.Contact AS c
JOIN Contact.Address AS a ON c.AddressId = a.AddressId
JOIN Reference.City AS City ON a.CityId = City.CityId
JOIN Enrollment.Enrollment AS e2011 ON c.ContactId = e2011.ContactId
JOIN Course.Session AS s2011 ON e2011.SessionId = s2011.SessionId
LEFT JOIN (Enrollment.Enrollment AS e2012
    JOIN Course.Session AS s2012 ON e2012.SessionId = s2012.SessionId
) ON c.ContactId = e2012.ContactId AND YEAR(s2012.StartDate) = 2012
WHERE YEAR(s2011.StartDate) = 2011
AND e2012.ContactId IS NULL
AND City.Name = 'Paris'
GROUP BY c.ContactId
ORDER BY LastName, FirstName
OFFSET 20 ROWS FETCH NEXT 20 ROWS ONLY;
```



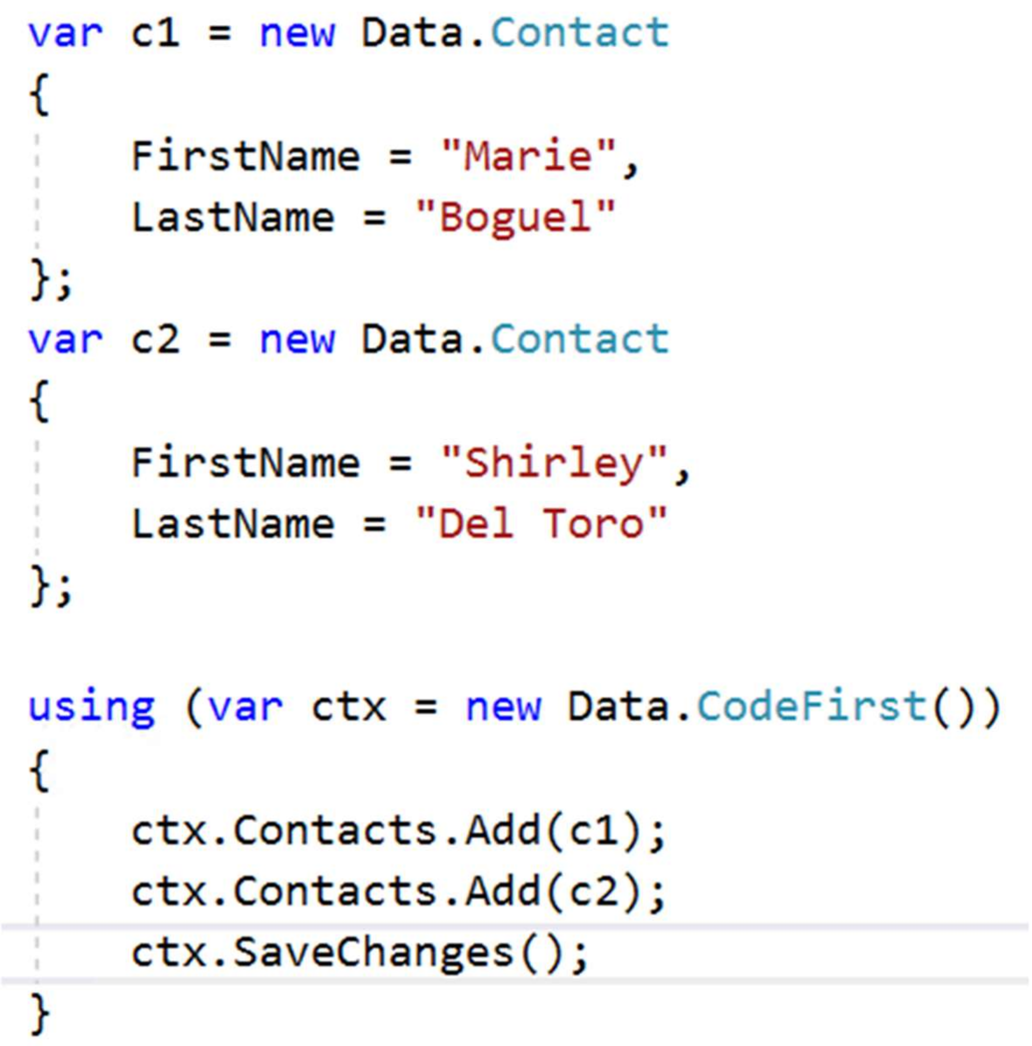
```
using (var ctx = new Data.CodeFirst())
{
    var res = (from c in ctx.Contacts
            join a in ctx.Addresses on c.AddressId equals a.AddressId
            where a.City.Name != "Paris"
            join s2011 in ctx.Sessions on c.Enrollments.SessionId equals
            select c)
        .Where()
            // ...
            select new ContactResult
            {
                FirstName = c.FirstName,
                LastName = c.LastName,
                City = a.City.Name,
                Enrollments = c.Enrollments.Count()
            };
    res.Load();
    return res.Skip(20).Take(20).ToList();
```

> “Any sufficiently complicated C or Fortran program contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of Common Lisp”

Philip Greenspun

https://en.wikipedia.org/wiki/Greenspun%27s_tenth_rule

What About Inserts?



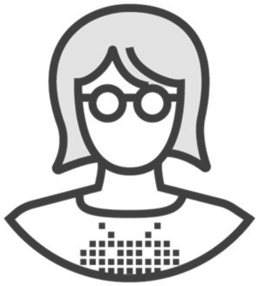
```
var c1 = new Data.Contact
{
    FirstName = "Marie",
    LastName = "Boguel"
};
var c2 = new Data.Contact
{
    FirstName = "Shirley",
    LastName = "Del Toro"
};

using (var ctx = new Data.CodeFirst())
{
    ctx.Contacts.Add(c1);
    ctx.Contacts.Add(c2);
    ctx.SaveChanges();
}
```

How Many Go-betweens Do We Need?

Me

Shirley

Josh



Marie

How Many Go-betweens Do We Need?

LINQ

EF

SQL

SQL Server

Why Are Stored Procedures Awesome?

You centralize logic close to data

You build an abstraction layer

You tighten security and protect the data

You ensure the best performances

The Thick Database Approach

Dr. Paul Dorsey
at the ODTUG conference in 2007

Morten Braten – Fat Database

Leveraging the full potential of the database engine

“If a problem can be solved using the database, it should be solved using the database.”

Conclusion

Pachadata

What is inside the box?

Where to put the logic?

Profile the database

Use ORM wisely

Befriend stored procedures