Are Relational Databases Obsolete?

Rudi Bruchez

SQL SERVER CONSULTANT AND TRAINER

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

Summary

What about NoSQL?

Are SQL databases obsolete?

SQL database are not web scale

Joins are slow

The relational schema is too rigid

What is the CAP theorem?

SQL is very old

Can We Predict the Load?

Traditional application
You can predict your load

Web scale
On the web or mobile, you can’t predict the load

Elasticity
You need to be able to scale

Do We Need Scalability?

Is your data big?
Maybe you don't need Big Data

Complexity
Scaling means creating more complexity

NoSQL and Big Data
They are good at scaling

What Is NoSQL?

- It is a movement
- Key-value store
- {JSON} Document database
- Wide-row store
- Graph database
- Search engine

Is It Really No SQL?

No to SQL?
We don't like this language

Not only SQL?
We get more choice

But they have SQL!
So we love this language

Apache Cassandra's CQL

```
CREATE TABLE enrollment
(
    contactid uuid,
    enrollment_date timestamp,
    session text,
    PRIMARY KEY (contactid, enrollment_date)
) WITH CLUSTERING ORDER BY (enrollment_date DESC);

INSERT INTO enrollment (contactid, enrollment_date, session)
VALUES (
    uuid(),
    toTimestamp(now()),
    'Cassandra Fundamentals'
);

SELECT *
FROM enrollment
WHERE contactid = 66fedeb0-69db-41df-b10f-cec2d42a77bd;
```

Couchbase Server’s N1QL

```
SELECT DISTINCT(name) AS Airline, id
FROM `travel-sample`
WHERE type = 'airline’
AND country LIKE '%Fran%' LIMIT 5 OFFSET 5
```

```
SELECT a.name, s.flight, s.utc, r.sourceairport,
       r.destinationairport, r.equipment
FROM `travel-sample` r UNNEST r.schedule s
JOIN `travel-sample` a ON KEYS r.airlineid
WHERE r.sourceairport='SEA’
AND r.destinationairport='MCO' AND s.day=6
ORDER BY a.name
```

It Is Not About the Language

SQL is well-known, we can build on that

In Big Data, Hive or Spark implement SQL

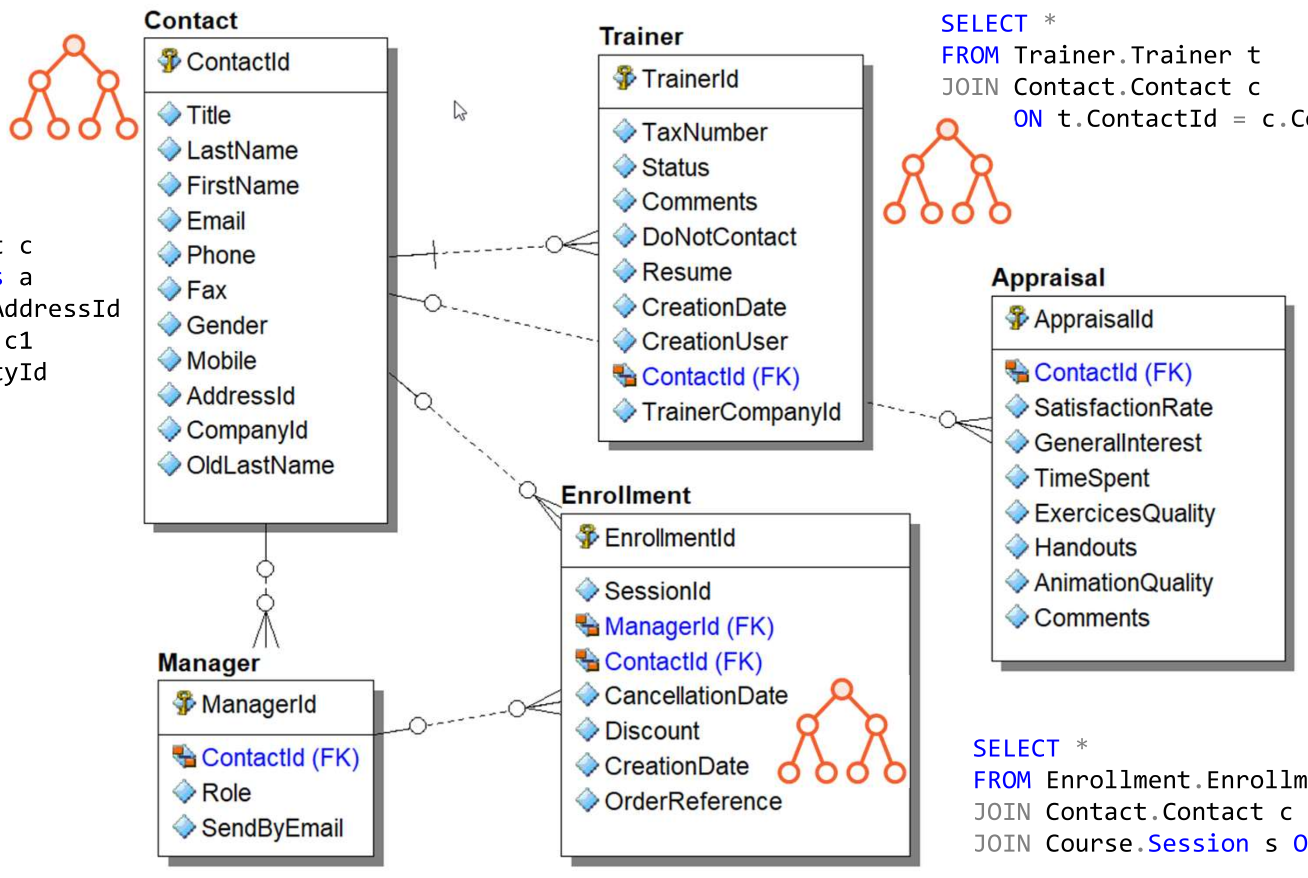
NoSQL really means NoRelational

Different data models

Martin Fowler

The Pachadata Model

```
SELECT c.ContactId
  ,c.Title
  ,c.LastName
  ,c.FirstName
  ,c.Email
  ,a.Address1
  ,c1.Name AS City
FROM Contact.Contact c
JOIN Contact.Address a
ON c.AddressId = a.AddressId
JOIN Reference.City c1
ON a.CityId = c1.CityId
```



```
SELECT *
FROM Trainer.Trainer t
JOIN Contact.Contact c
    ON t.ContactId = c.ContactId
```

```
SELECT *
FROM Enrollment.Enrollment e
JOIN Contact.Contact c ON e.ContactId = c.ContactId
JOIN Course.Session s ON e.SessionId = s.SessionId
```

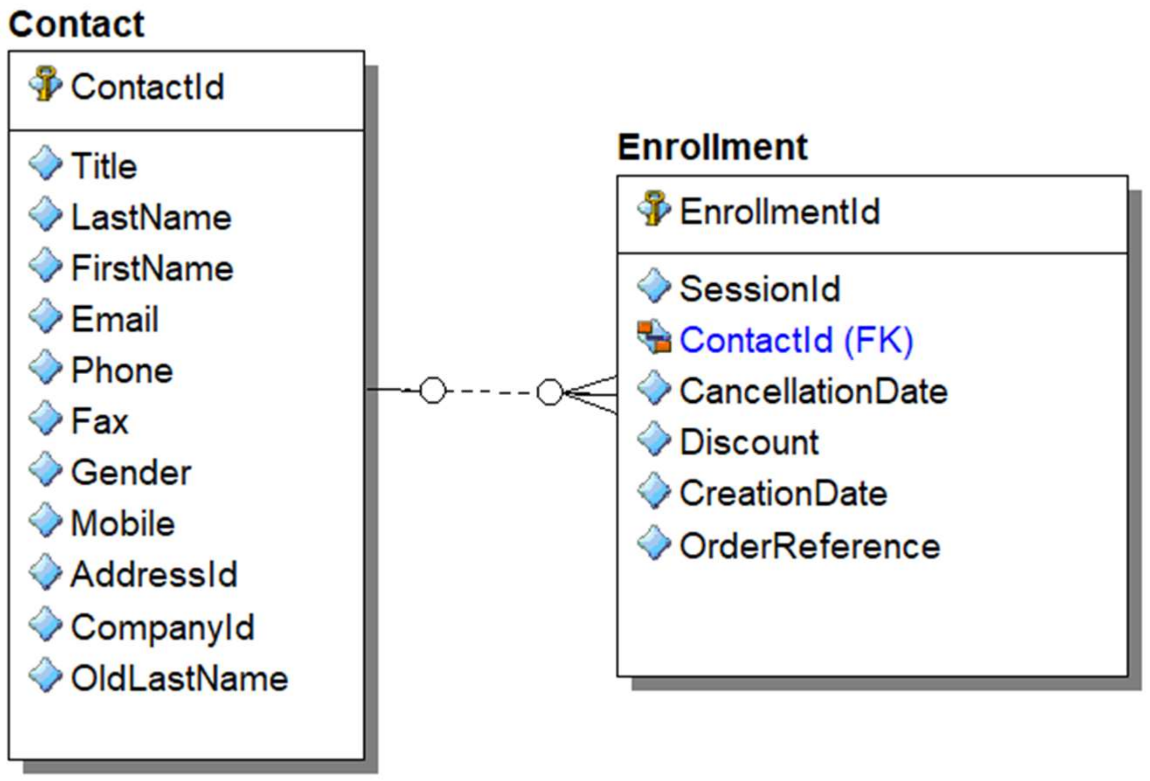
NoSQL Data Model

Faster and Simpler
Handle the load

Key-value store
Great performances

No constraint
Focus on the data

Relational Tables



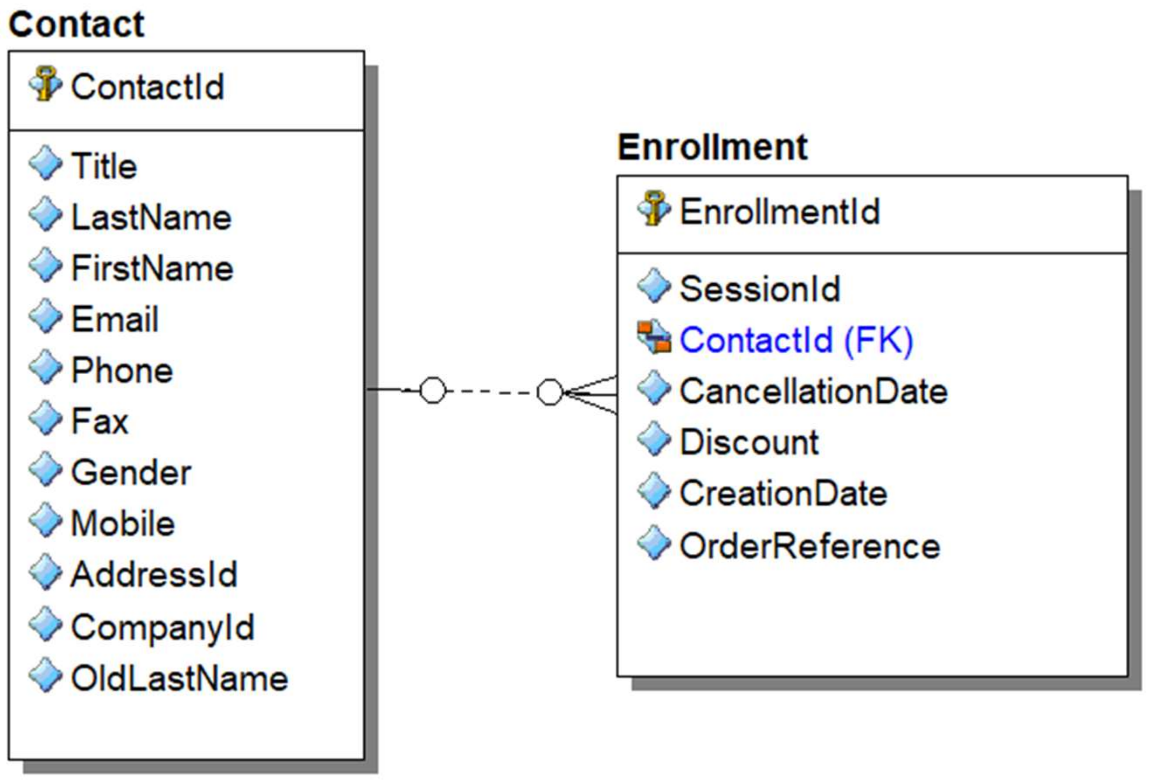
A Document Database

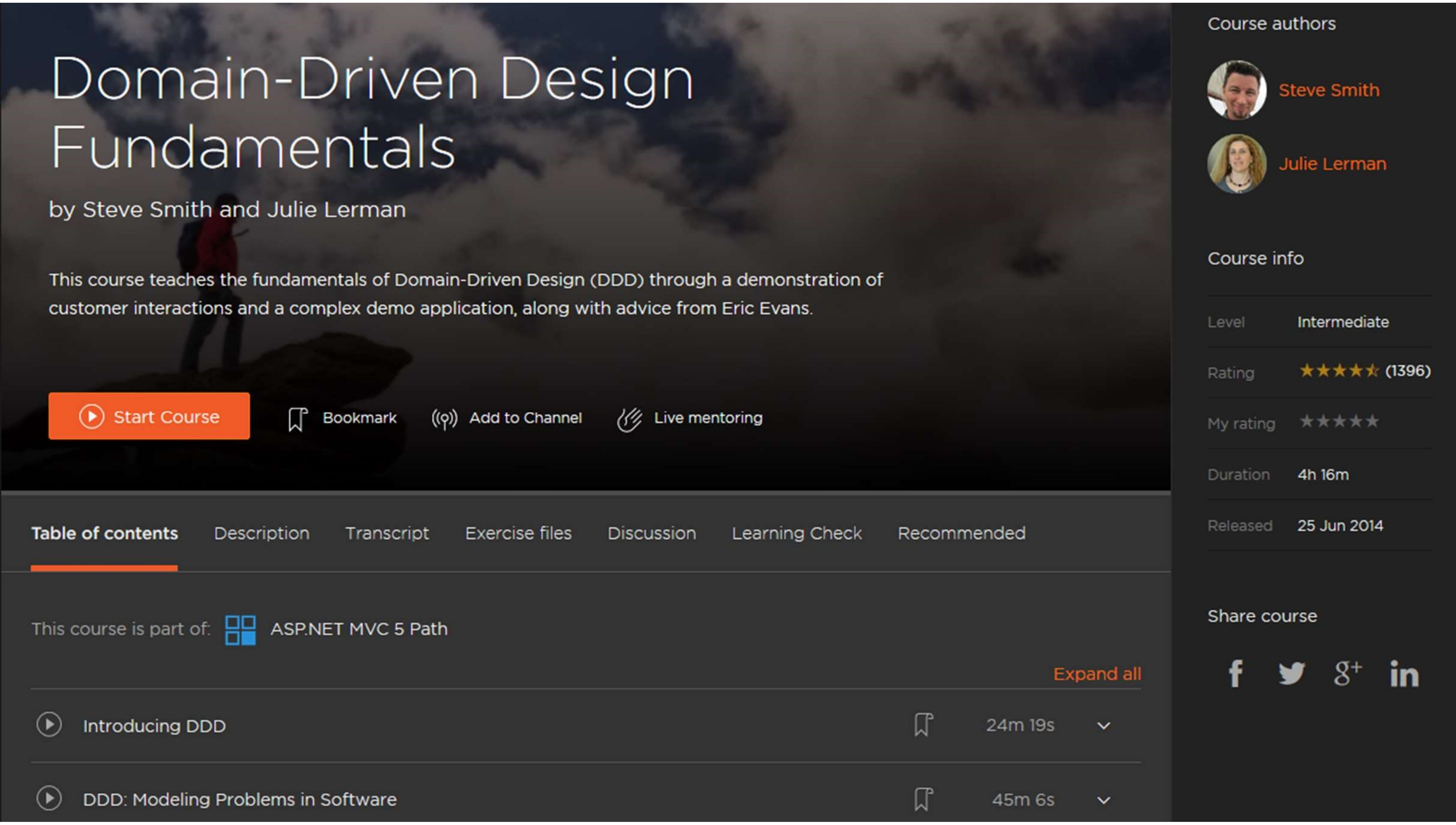
```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Gender":"H",
    "Mobile":"+994997445",
    "Address": {
        "Address1":"Common Street 33",
        "CityName":"San Gommery",
        "ZipCode":"56920"
    },
    "Enrollments": [
        {
            "SessionDate": "2018-07-05",
            "Language":"EN",
            "Class":38,
            "Price":2500.00,
            "Titre":"MySQL 5.7 essential"
        } ]
}
```

A Document Database

```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Gender":"H",
    "Mobile":"+994997445",
    "Address": {
        "Address1":"Common Street 33",
        "CityName":"San Gommery",
        "ZipCode":"56920"
    },
    "Enrollments": [
        {
            "SessionDate": "2018-07-05",
            "Language":"EN",
            "Class":38,
            "Price":2500.00,
            "Titre":"MySQL 5.7 essential"
        },
        {
            "SessionDate": "2018-11-22",
            "Language":"EN",
            "Class":15,
            "Price":2500.00,
            "Titre":"MySQL 5.7 advanced"
        }
    ]
}
```

Relational Tables




Domain-Driven Design Fundamentals

by Steve Smith and Julie Lerman

This course teaches the fundamentals of Domain-Driven Design (DDD) through a demonstration of customer interactions and a complex demo application, along with advice from Eric Evans.

This course is part of: ASP.NET MVC 5 Path

Introducing DDD	24m 19s
DDD: Modeling Problems in Software	45m 6s

Course authors

Steve Smith

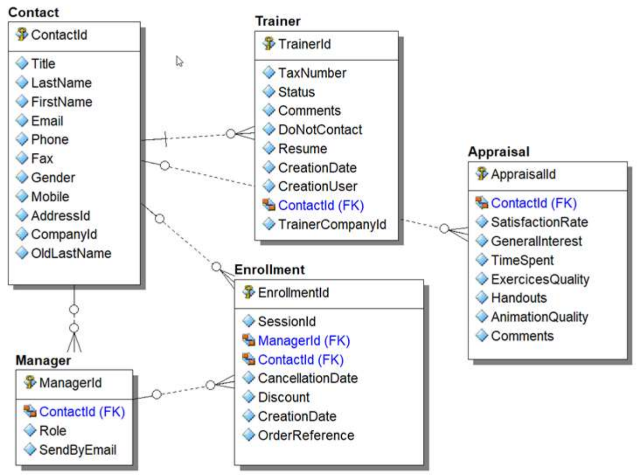
Julie Lerman

Course info

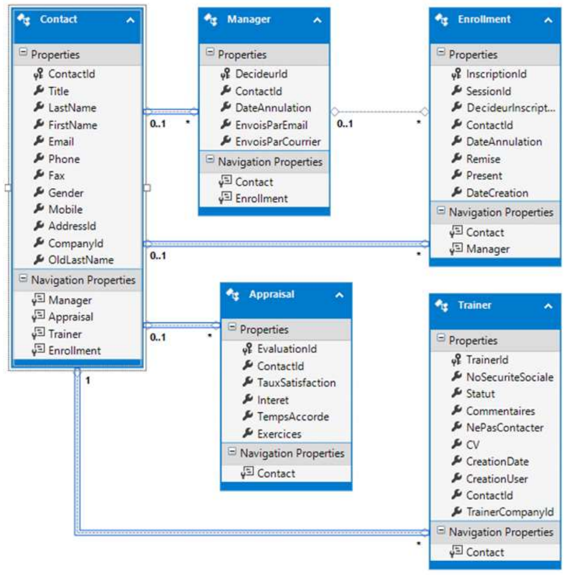
Level	Intermediate
Rating	★★★★½ (1396)
My rating	★★★★★
Duration	4h 16m
Released	25 Jun 2014

Domain Driven Design Aggregates

Relational Database

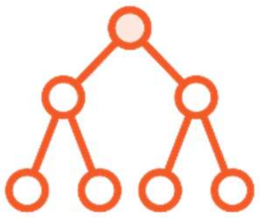


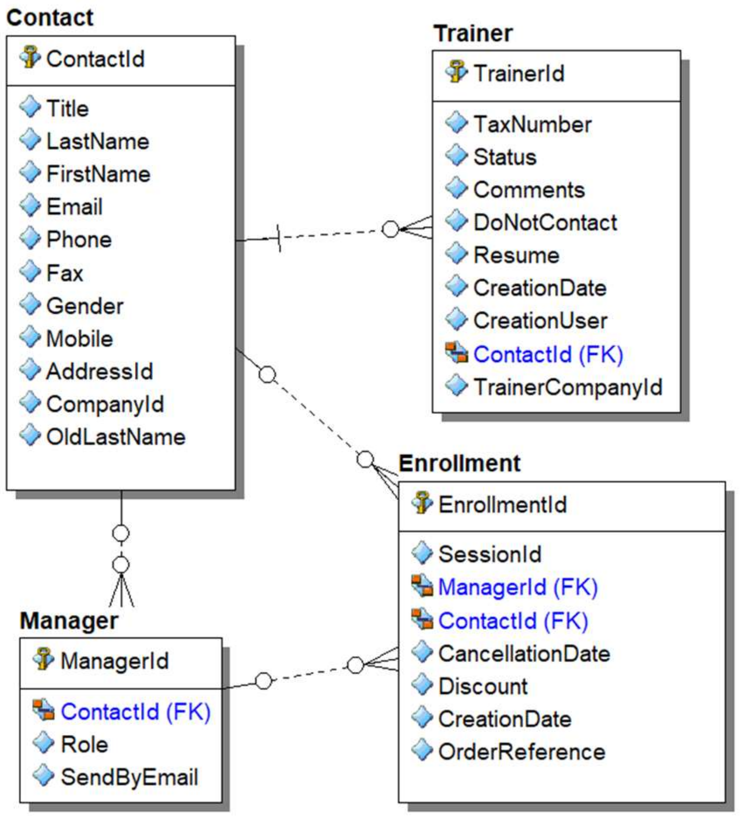
Entity Framework



Joins as Hierarchy

```
SELECT c.Title
   ,c.LastName
   ,c.FirstName
   ,c.Email
   ,s.CourseId
   ,s.LanguageCd
   ,s.RoomId
   ,s.StartDate
FROM Contact.Contact c
JOIN Enrollment.Enrollment e
     ON c.ContactId = e.ContactId
JOIN Course.Session s
     ON e.SessionId = s.SessionId
WHERE c.ContactId = 8608
```





Domain Driven Design Aggregates

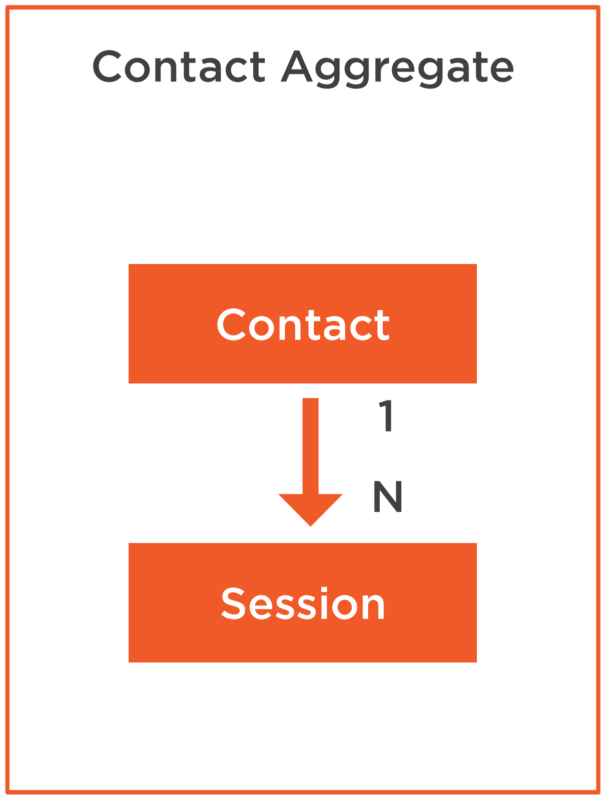
Reduce the complexity of a data model

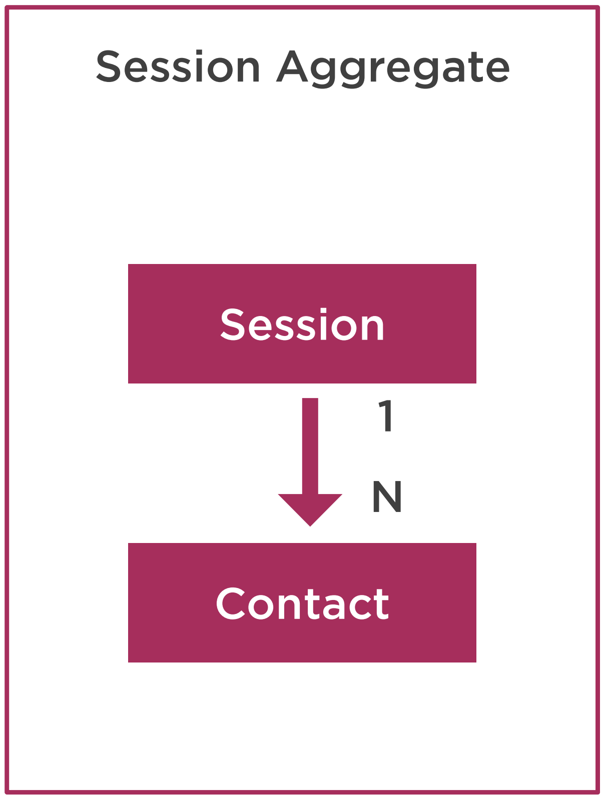
Object hierarchy

Bounded context for the client application

Aggregate root

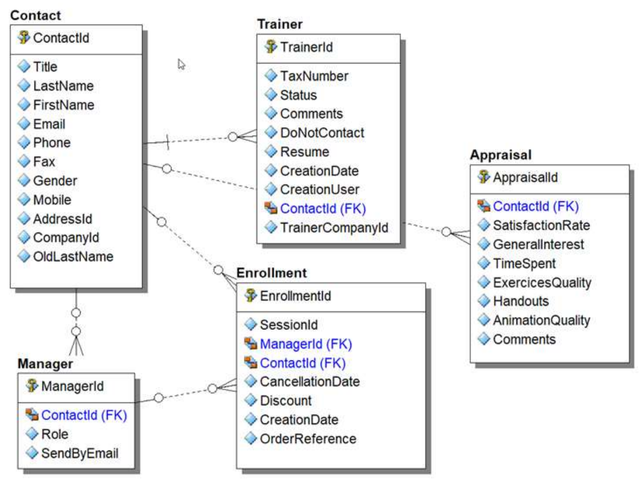
Pachadata Aggregates





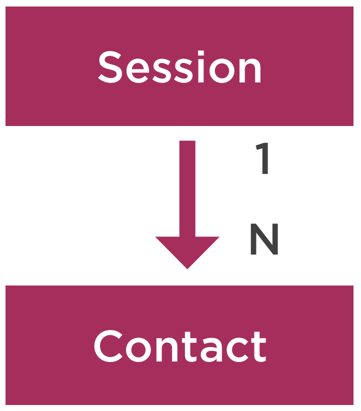
Domain Driven Design Aggregates

Relational Database



```
SELECT
FROM
JOIN
```

Session Aggregate

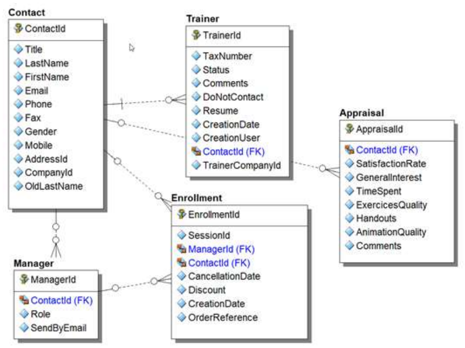


Relational Databases as Integration Stores

Contact Management

Session Management

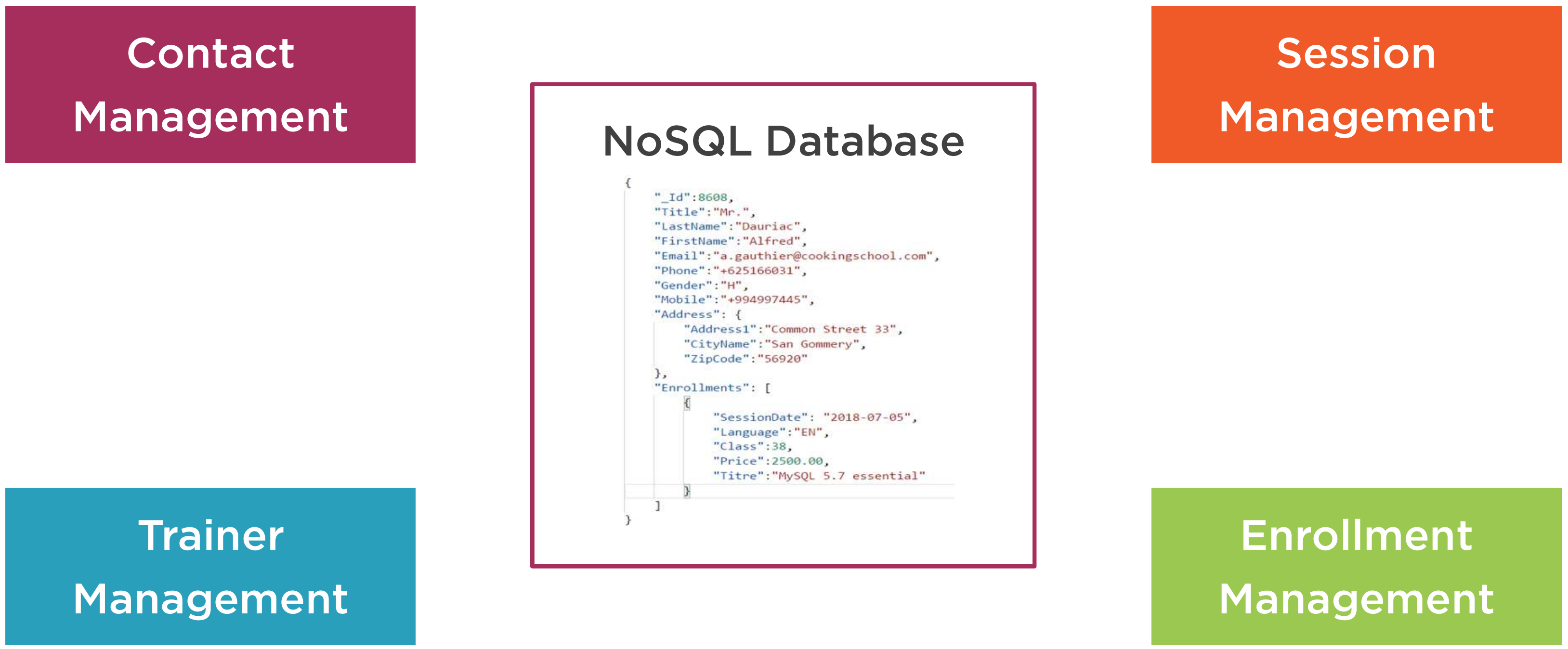
Relational Database



Trainer Management

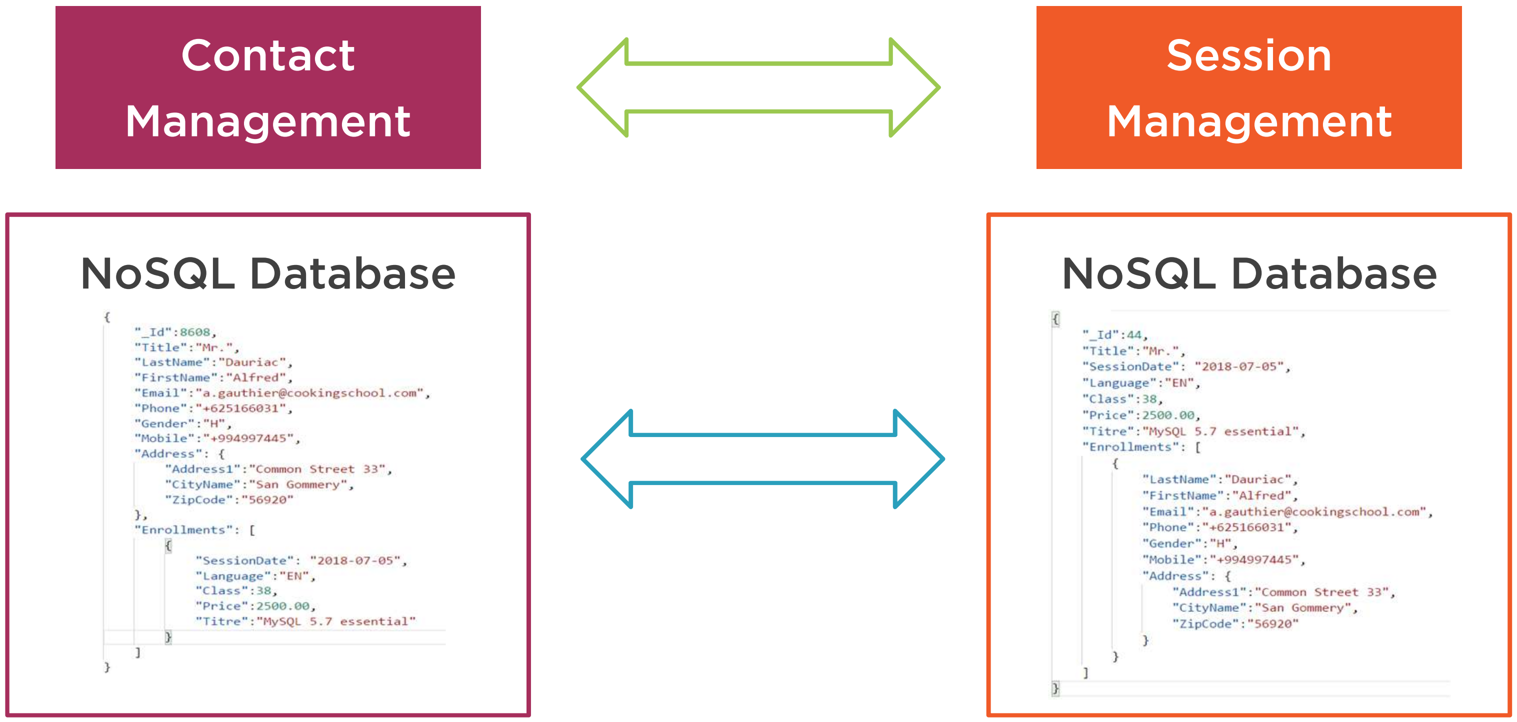
Enrollment Management

NoSQL Databases as Aggregates



```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Gender":"H",
    "Mobile":"+994997445",
    "Address": {
        "Address1":"Common Street 33",
        "CityName":"San Gommery",
        "ZipCode":"56920"
    },
    "Enrollments": [
        {
            "SessionDate": "2018-07-05",
            "Language":"EN",
            "Class":38,
            "Price":2500.00,
            "Titre":"MySQL 5.7 essential"
        }
    ]
}
```

NoSQL Databases as Aggregates



```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Gender":"H",
    "Mobile":"+994997445",
    "Address": {
        "Address1":"Common Street 33",
        "CityName":"San Gommery",
        "ZipCode":"56920"
    },
    "Enrollments": [
        {
            "SessionDate": "2018-07-05",
            "Language":"EN",
            "Class":38,
            "Price":2500.00,
            "Titre":"MySQL 5.7 essential"
        }
    ]
}
```

```
{
    "_Id":44,
    "Title":"Mr.",
    "SessionDate": "2018-07-05",
    "Language":"EN",
    "Class":38,
    "Price":2500.00,
    "Titre":"MySQL 5.7 essential",
    "Enrollments": [
        {
            "LastName":"Dauriac",
            "FirstName":"Alfred",
            "Email":"a.gauthier@cookingschool.com",
            "Phone":"+625166031",
            "Gender":"H",
            "Mobile":"+994997445",
            "Address": {
                "Address1":"Common Street 33",
                "CityName":"San Gommery",
                "ZipCode":"56920"
            }
        }
    ]
}
```

Choosing NoSQL?

Flexibility
With different databases

Complexity
More services, more components

Problems
Get used to it

Choosing NoSQL?

Aggregates
You loose centrality

Scalability
Do you really need it?

Ease of use
Is it a valid argument?

Relational Databases Are Too Rigid

Flexibility
With schemaless databases

With strong coupling?
All tiers need to change

Schema-later
There is always a schema

The Contact Document

```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Mobile":"+994997445"
}
```

The Contact Document

```
{
    "_Id":8608,
    "Title":"Mr.",
    "LastName":"Dauriac",
    "FirstName":"Alfred",
    "Email":"a.gauthier@cookingschool.com",
    "Phone":"+625166031",
    "Mobile":"+994997445“,
    "Gender":"M"
}
```

The Contact Document

```
var client = new MongoClient("mongodb://192.168.1.15:27017");
var database = client.GetDatabase("pachadata");
var collection = database.GetCollection<BsonDocument>("contacts");
var document = collection.Find(new BsonDocument()).FirstOrDefault();
BsonString gender;
if (!document.TryGetElement("Gender", out gender))
{
    // ...
}
```

What Do We Need?

No coupling
Tiers should not be too dependent

Database is a service
Decoupling database from client code

Abstraction layer
Between the database and the client code

Using Views

```sql
CREATE VIEW Contact.ContactAggregate
AS
  SELECT c.Title, c.LastName,
         c.FirstName,c.Email,
         s.CourseId, s.LanguageCd,
         s.RoomId, s.StartDate
  FROM Contact.Contact c
  LEFT JOIN Enrollment.Enrollment e
    ON c.ContactId = e.ContactId
  LEFT JOIN Course.Session s
    ON e.SessionId = s.SessionId;
```

```sql
CREATE VIEW Course.SessionAggregate
AS
  SELECT s.SessionId, s.CourseId,
         s.LanguageCd, s.RoomId,
         s.StartDate, c.Title,
         c.LastName, c.FirstName,
         c.Email
  FROM Course.Session s
  LEFT JOIN Enrollment.Enrollment e
    ON e.SessionId = s.SessionId
  LEFT JOIN  Contact.Contact c
    ON c.ContactId = e.ContactId;
```

“All views that are theoretically updatable are also updatable by the system”

Edgar Codd

Rule 6 – The *view updating rule*

Conclusion

Should we always follow trends?

NoSQL movement

Choice polyglot storage

Choose wisely

We keep SQL Server for now

Predictable performances?

Colors