

les 1 ADO.NET

ADO.NET = objectgeoriënteerde verzameling van bibliotheken die toelaat om verschillende databronnen te integreren
"DataKlasses"
↳ database, txt, excel, XML, ...

Connection: connectie met database prefix

Command: db command om data terug te geven, wijzigen, ingeven, ...

DataReader: stream of data vd database prefix

DataAdapter: brug tss DataSet en database prefix

DataSet: disconnected representatie van data gen prefix!

```
SqlConnection conn = new SqlConnection ("Data Source = (local); Initial Catalog = postgre;
                                         UserID = kweetnie; Password = password ");
conn. Open ();
conn. Close ();
SqlCommand cmd = ("query.", conn );
RDR = cmd. ExecuteReader ();
while (RDR. Read ()) Console. WriteLine (RDR [0]);
```

- ExecuteReader: voert query uit, returns SqlDataReader object
- ExecuteNonQuery: voert query uit (returns # rijen dat bewerkt is) (bv. bij insert query)
geeft eigen niets terug
- ExecuteScalar: voert query uit, returns 1^e kolom vd 1^e rij
↳ als een object → casten

SqlDataReader: enkel data vragen, forward-only sequential ⇒ zeer snel
read-methode returns boolean
interven v. data mogelijk door kolomindex of kolomnaam
CASTEN + READER SLUITEN

```
DataTable table = new DataTable ();
table. Load (reader, LoadOption. PreserveChanges);
```

SqlCommandBuilder cmdBldr =
new SqlCommandBuilder (dsCustomers);

DataAdapter sluit de connectie vanzelf

CommandBuilder: manier om insert, update, delete queries autom. aan te maken voor
bepaalde dataAdapter
dsCustomers. Fill (dsCustomers, "Customers");

Form: dgCustomers. DataSource = dsCustomers ;
dgCustomers.DataMember = "Customers";

daCustomers.Update(dsCustomers, "Customers") ;

Dataset van meerdere tabellen beratten

elke tabel heeft 1 resultset die gereturnd in odrv SELECT statement
alle objecten in opgeslagen in collections

PARAMETERS om SqlInjectie te voorkomen

- ① SqlCommand cmd = new SqlCommand ("SELECT * FROM Customers WHERE city = @City", conn)
- ② SqlParameter par = new SqlParameter ();
par.ParameterName = "@City";
par.Value = inputcity;
- ③ cmd.Parameters.Add (par);

SqlCommand voor sproc

```
SqlCommand cmd = new SqlCommand ("naam v spec", conn);  
cmd.CommandType = CommandType.StoredProcedure;
```

les 2 SPROCS & transactions

sproc = verzameling van SQL commando's die als 1 geheel bewaard & uitgevoerd in

overzicht v sprocs: show procedure status

code v sproc: show create procedure <naam>

spec is bewaard in mysql.proc systeemtabel

```
delimiter $$  
drop procedure if exists helloworld $$  
create procedure helloworld()  
begin  
    select 'Hello World!';  
end $$  
delimiter;  
call helloworld();
```

variabelen:

- set @val = ?;
- of
- select @dagen := datediff(
 curdate(), '2008-08-08');

parameter meegenen aan sproc:

```
command.Parameters.AddWithValue("@ID", 1);
```

begin / end blocks

< blockname > : begin

commands ;

IF (.) THEN LEAVE < blockname > ; END IF ;

commands ;

end < blockname > ;

user defined variables : @name = . . . ;

lokale variabelen (in proc) declare < name > < type > ; daarna set

altijd ah. begin van
begin-end block !!

IF ... THEN ...
END IF
IF ... THEN ...
ELSE ...
END IF

case < variable >
WHEN 0 THEN ...
WHEN 1 THEN ...
ELSE ...
END CASE

WHILE < loopname > REPEAT
commands ;
OR UNTIL condition END REPEAT < loopname >
WHILE < loopname > WHILE condition DO
commands ;
END WHILE < loopname >
< loopname > LOOP
commands ;
END LOOP < loopname >

leave & iterate
≈ break & continue

ERROR handling :

{ declare < exit/continue > handler for < error nummer / sqlstate/errorstring/condition >
SQL statement

→ ah begin van begin-end block

exit verlaat begin-end block
continuee vervolgt de uitvoer

DECLARE < name > CURSOR FOR SELECT -- ;
OPEN < cursorname > ;
FETCH < . . . > INTO < variabele > ;
CLOSE < . . . >

CURSOR: record per record door
een resultset stappen

Functions : geven altijd een resultaat terug
geen IN en OUT parameters, wel gewone parameters en 'RETURNS'

SPROCS en veiligheid:
CON: → abstractie
→ complexiteit

PRO: → minder network traffic
→ extra laag tss datalaag & logische laag
→ Jt verleent de app geen directe toegang tot de data
→ zelf reguleren hoe de data gebruikt wordt
→ scheiding v data & business logic (ontwikkelaar kan zich concentreren & zijn eigen specialiteit)

tijdelijke tabellen : verwijderd bij sluiten v. connectie. Tabelltype HEAP ; entel in RAM
(⇒ sneller), niet op harde schijf.

- sprocs
- ⊕ • veiligheid
 - scheiding v data & logic
 - network traffic ↓
 - ⊖ • meer abstractie, meer complexiteit

transactie = eenheid van werk die in zijn geheel moet uitgevoerd w

moet voldoen aan **ACID**:

- atomicity**: heel geheel uitgevoerd, heel niet uitgevoerd
- consistency**: creëert nieuwe geldige staat & herstelt de oude
- isolation**: 2 transacties hebben geen zicht in elkaars historie resultaten
- durability**: voltooide "kr niet ongedaan gemaakt"

einde: COMMIT of ROLLBACK
 ↳ cancel
 ↳ make changes permanent

Leng: tijdelijke bescherming v records/blokken/tabellen voor gebruik door andere processen gedurende het bestaan ve transactie/andere data-generatie

```
using (SqlConnection conn) {
    conn.Open();
    SqlTransaction meth = conn.BeginTransaction();
}

```

```
SqlCommand updateCmd = new SqlCommand();
updateCmd.Transaction = meth;
or
SqlCommand updateCmd = new SqlCommand(
    sqlText, conn, transaction);
```

les 3 ADO

```
DataRow eenrij = eenTabel.NewRow();
eenTabel.Rows.Add(eenrij);
```

```
eenrij.Item["ID"] = 8888;
```

veranderingen in batch: eenTabel.AcceptChanges(); & .RejectChanges();

DataRow.RowState

- detached: nog niet toegevoegd @ tabel
- added: toegepast maar nog niet bevestigd
- unchanged: wijz id tabel & niet gewijzigd sinds laatste AcceptChanges
- deleted: verwijderd maar nog niet bevestigd
- modified: wij waaraan iets veranderd is

delete methode v DataRow klasse gebruiken! → wij w nog niet verwijderd maar status wel op deleted. Remove = permanent

- rijversies:
- original: waarde na uitvoeren v laatste AcceptChanges, voor de nieuwe veranderingen
 - proposed: veranderd, maar nog niet bevestigd
 - current: indien bevestigd: zelfde als original. niet bevestigd: zelfde als proposed
 - default: indien attached: zelfde als current. indien detached: " "

MAPPING! indien niet mog is om zelfde tabel- en kolomnamen voor database en dataTable te gebruiken.

adapter.TableMappings.Add
 PassThrough: gebruik naam uit db
 Ignore: geen mapping
 Error: error opsporen

DBPROVIDERFACTORY

machine.config: welke providers zijn aanwezig op je pc (C:\Windows\Microsoft.NET\...)

DataTable tabel = DbProviderFactories.GetFactoryClasses(); → retrieve installed providers & factories
 providerstring opslaan in app config en providername

ConnectionStringSettingsCollection settings = ConfigurationManager.ConnectionStrings;
 ↳ dan in foreach: if (cs.ProviderName == providername)

DbProviderFactory factory = DbProviderFactories.GetFactory(providername);
 connection = factory.CreateConnection();
 connection.ConnectionString = connectionString;

Triggers

Tekens een delete, insert of update gebeurt, gebeurt er ook iets anders

Trigger heeft naam, tabel die het moet trigger, wanneer uitgevoerd, body

CREATE TRIGGER <naam> BEFORE / AFTER INSERT / UPDATE / DELETE
 ON <tabel>

FOR EACH ROW BEGIN

→ [trigger time] - [table time] - [trigger event]
 br before - employees - update

END

les 4

LINQ

LANGUAGE INTEGRATED QUERY

eerst FROM, dan WHERE, dan SELECT

deferred execution: LINQ voert de query niet uit totdat app ze nodig heeft
→ indien + keer over resultaat getereerd wordt, wordt data source
nieuw benaderd en kan resultaat verschillen

→ DOEL: up-to-date data

omzeilen door na LINQ query .ToList te zetten

```
List<string> results = (from string c in colors
                         where c.StartsWith("B")
                         orderby c
                         select c).ToList();
```

object initializers: init van enkele / alle properties in hetzelfde statement
dat het object zelf instantieert

projectie / shaping: transformatie v data ie. LINQ query om enkel de zaken
te krijgen die je nodig hebt

transformatie v originele kolommen in een nieuw subset v kolommen
anonieme types? var carData = from c in getCars() where c.year = 2008 select

lambda var found = cars.Find(c => { int x;

extension methods?

(zie next page)

```
x = c.Year;
return c.Year == x;});
```

LINQ TO SQL

LINQ to DataSet: extensionmethod AsEnumerable gebruiken!

object-relational mapping (ORM) tool: map tss obj's & klassen (app) en data in tabellen
inner join m.b.v. query extension methode:

```
var customers = db.Customers.Join(db.Orders,
                                   c => c.CustomerID,
                                   o => o.CustomerID,
                                   (c,o) => new {c.CustomerID, c.CompanyName, o.OrderID} )
                                   .OrderBy(r => r.CustomerID)
                                   .ThenBy(r => r.OrderID));
```

→ in LINQ: next

extension methods

methode toevoegen aan een type, zelfs als je niet over de source code beschikt.

by. kijker of de waarde ve string numeriek is

mag 1 : helpklasse met isNumeric methode

mag 2 : extension method (in aparte klasse) :

```
public static class stringHelper {  
    public static bool isNumeric (this string str) {  
        double value;  
        return double.TryParse(str, out value);  
    }  
}
```

extension method: altijd in ^{publieke} statische + klasse!

query extension methods:

- all
- any
- average
- cast
- concat
- contains
- count
- distinct
- elementAt
- except
- first
- groupBy
- intersect
- join
- last
- max
- min
- longCount



+ ~~loop~~
lambda expr.

Join mbv LINQ : var customers = from c in customers
join o in orders
on c.CustomerID equals o.CustomerID
orderBy c.CustomerID, o.OrderID
select new {c.CustomerID, c.CompanyName, o.OrderID};

LINQ TO SQL : veranderingen doorgeven @ SQL server

niet alleen data ophalen, maar ook INSERT DELETE UPDATE mogelijk

DataContext bezit identity table. Houdt via PK v record bij welke records uit db gehaald zijn. Daarna is zelfde record uit cache gehaald.

Toestand v objecten is bijgehouden achter status

bv. customer ophalen, naam wijzigen. db.SubmitChanges();

* nieuwe " aanmaken, db.Customers.InsertOnSubmit(customer); , db.SubmitChanges();
DeleteOnSubmit

LINQ TO XML : zie ppt!

les 5 entity framework

relationeel model

↔ ER

↑
datanormalisatie

duplicatie ↑, consistente ↑

↑
data opdelen in dingen (entities) en
relaties daartussen

Entity Data Model Wizard

geen joins, wel navigatieproperties

Elke Klasse (entiteit) erft over van EntityObjectklasse

conceptueel model en storage model en relatie is uitgedrukt in XML schema's :

- conceptual schema def. language CSDL → DATA
- store schema def. language SSDL → LOGICA
- mapping specification language MSL → MAPPING

→ identificeert onderliggende databankelementen die conceptueel model ondersteunen
berat entites & relaties, maar ook queries en spacs

entity = instantie van EntityType, dat de eigenschappen die de structuur vd entiteit beschrijven, beschrijft

TPH table per class hierarchy

Vehicles (Id, Vin, Make, Model, Year)

+ extra veld 'Type' inzelfde tabel

Car \ Boat

TPT table per type

Vehicles (Id, Vin, Make, Year)

↳ Cars (Id, TireSize)

↳ Boats (Id, PropSize)

TPC table per class

tabel voor elke concrete klasse, niet voor abstracte

⊖ dubbele data neemt toe

niet standaard in EF

Cars (Id, Make, Vin, Year, TireSize)

Boats (Id, Make, Vin, Year, PropSize)

} concrete klassen

mysql -u root -p <database name> < <pad ne sql file>

	Model First 1 model in designer	Code first 3 klassen & mapping maken in c# migrations
New db	Db first 2 reverse engineer model in designer	Code first 4 klassen & mapping maken in c# reverse engineer tools

① MODEL FIRST

new Item → Data → Ado.NET Entity Data Model → empty model
 add entities , add association
 generate database (servername (localdb)\...) + execute

② DATABASE FIRST

new Item → Data → Ado.net entity Data Model → Generate from database
 all conn.string : bloggingContext

③ CODE FIRST TO NEW DATABASE

creëer klassen met properties en ^{navigation} virtual properties
 NuGet (rechtermuis op project) → online → entity framework
 creëer context (BlogContext : DbContext) met DbSets < x >
 Package Manager Console (Tools) : enable - Migrations
 Add - Migration Addurl + Update - Database

④ CODE FIRST TO EXISTING DATABASE

new item → Data → Ado.NET Data Model → Code First from Database

Containers

- entityset → instanties van entiteiten
 - relationship set → instanties van relationships
 - entitycontainers → entitySet & relationshipSet
- ≈ relationele tabel
≈ join tabel

Overerving

- table per class hierarchy (TPH) (alle prop van overervende types in 1 tabel opgeslagen)
- table per type (TPT) (elk type wordt in eigen tabel opgeslagen) ⇒ meer joins
- table per concrete class (TPC) (tabel voor elke concrete klasse, niet voor abstracte)

TPH: var db = new TablePerHierarchy.TP-H.Entities();

gr. datasource = (from b in b.VehiclesOfType<TablePerHierarchy.Boat>()
select b).ToList();

Model First: db creeren adhv conceptueel model (conceptueel model vóór db)

Databasefirst: conceptueel model genereren adhv bestaande db

Code First: db laten genereren adhv klassen in c#

```
public class BloggingContext : DbContext {
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}
```

maak model in designer

reverse engineer model in designer

les 6 NoSQL

"meerdere servers"

performantie, grote hoeveelheden data, clusters

Schaalbaarheid = mogelijkh. ve. systeem om de throughput te doen toenemen door extra resources toe te voegen om toenemende belasting te counteren

- verticaal: grotere / snellere computer = SCALING UP
- horizontaal: extra PC's toevoegen @ cluster = SCALING OUT

impedance mismatch: 2 + representaties die vertaling nodig hebben
(proberen glossen mbr document-based NoSQL db's (bv. MongoDB)) (↔ relationele DBs)

polyglot persistente: ≠ databanksystemen voor ≠ omstandighⁿ
integratiedb's → applicatie db's

aggregatie: collectie van gerelateerde objecten die we als 1 geheel willen behandelen
data plaatsen in complexere structuur dan eigen in tabel

⊖ NoSQL: geen ondersteuning voor transacties, overlever geen stroomuitval

kenmerken NoSQL db's:

- gen. scal.
- gemaakt voor op clusters
- open source
- schemaless

Soorten NoSQL databases

- Document based : structuur in aggregate oproeken dmrv wat in "velden zit"
- Key-value based : aggregate (= samengestelde objecten) is een blob bestaande uit bits oproeken dmrv sleutel
- Column-oriented : event logging, CMS, blogs, counters
- graph : niet gemaakt voor op clusters, wel voor relaties tss kleine stukjes data set social networks

bij NoSQL : data-integriteit is verantw.h. vd ontwikkelaar

indien ACID nodig better RDBMS

MongoDB ↗ data die vaak gebruikt & opgestoken wordt
doc based

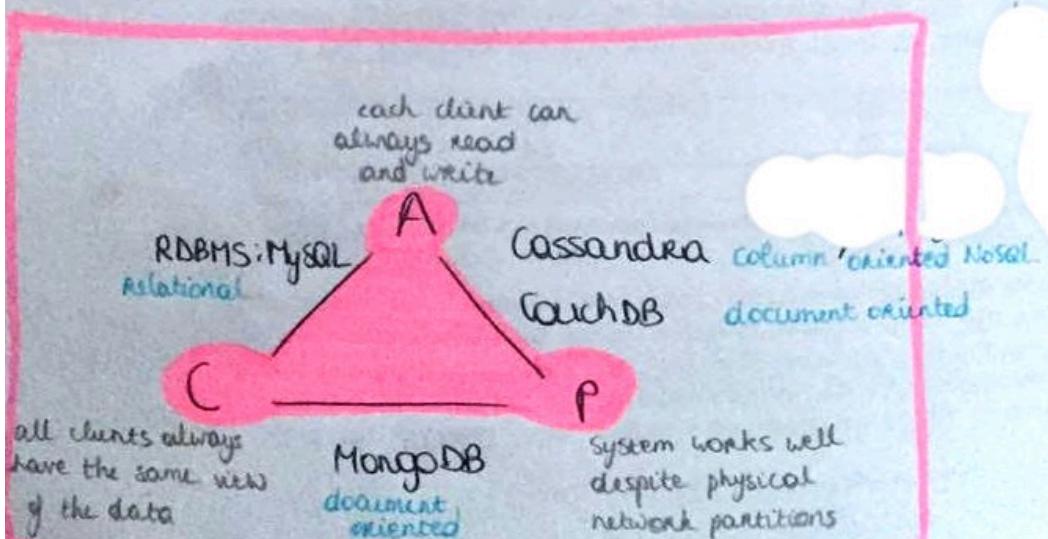
MongoDB & CouchDB : document based

doc based : dataopslag = goedkoop

key-value : cloud apps, ontwikkelde code
vooral voor sessie-info, user profiles, online shopping
niet voor relaties, zoeken op value

Cassandra : event logging, CMS, blogs, counters

↳ column oriented



MongoDB gebruikt BSON → Binary JSON → = JavaScript Object Notation
 om docs op te staan
 MongoClient client = new MongoClient("mongodb://localhost");
 var db = client.getDatabase("test");
 IMongoCollection<Person> personen = db.getCollection<Person>("personen");
 Person hanne = personen.insertOneAsync({name});
 var people = personen.find(x => x.Age > 40).ToListAsync();
 (ACID)
NOSQL

CAP theorema : in NOSQL, consistency afzwakken in gedistribueerde omgeving

CONSISTENCY
 AVAILABILITY
 PARTITION TOLERANCE

van deze
 Max 2^{eig} voor gelijk welk gedistribueerd datasysteem

elke vraag ontvangen van een niet-faaldende node in de cluster moet resulteren in een antwoord

cluster kan onderbrekingen in de communicatielijnen overleven maar hierdoor is de cluster wel in partities verdeeld, waardoor + onderdelen niet met elkaar kunnen communiceren = SPLIT BRAIN
 consistency afwegen tot beschikbaarheid

Map Reduce → er kan geen data na de db geschreven is die de regels voor geldige data overtreedt

cluster : proberen om verkeer tss nodes te verminderen door zoveel mogelijk uit te voeren op de node waar de data zich bevindt

les 7 WPF

XAML: extensible app markup language

→ GDI → DirectX

→ ~~DUI~~ → GPU

→ WPF evalueert videokaart & geeft tier waarde

Databinding = proces dat conn. tss logica & databag opzet

elke binding: 4 componenten:

- target object (bv. textbox)
- target eigenschap (bv. Text-eig. v. textbox)
- binding source (bv. employee object)
- pad naar de waarde in de binding source die gebruikt moet worden (bv. Name v employee)