

Casper Fabricius
<http://casperfabricius.com>

ActiveRecord

O/RM i Ruby on Rails

Casper Fabricius

- * Freelance webudvikler - casperfabricius.com
- * 9 års erfaring med webudvikling
 - * 6 år med ASP/ASP.NET/C#
 - * 3 år med Ruby on Rails
- * cand.merc.dat fra CBS - DØK'er
- * Lokalebasen.dk, Kiwiminipris.dk, Ideadox.com m. fl.

Ruby

- * Kombinerer det bedste fra Smalltalk, Perl, Python og Lisp
- * Dynamisk stærkt typet, objekt-orienteret, fortolket
- * Implementeret i C, i dag også på JVM (JRuby) og snart også på .NET (IronRuby)
- * Closures, mixins, åbne klasser, reflektion, metaprogrammering

Rails

- * Databasedrevne webapplikationer
- * Convention over configuration
- * Glæde ved at programmere
- * Holdbar produktivitet
- * Smuk kode
- * Skabt til agil udvikling

Ruby-stacken

CONTROLLERS

Action
Controller

Merb

Sinatra

VIEWS

ERb

HAML

Erubis

MODELS

Active
Record

Datamapper

Sequel

DATABASER

MySQL

Post-
greSQL

SQLite

Oracle

SQL
Server

DB2

ActiveRecord

- * Formentlig verdens i øjeblikket bedste O/RM
- * Har en stor del af æren for Ruby on Rails' succes
- * Implementering af Martin Fowlers pattern af samme navn:
"An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data."
- * Princippet om *convention over configuration* gør at AR kan bruges uden nogen form for konfiguration

Demo

Konventioner i AR

- * Hver klasse har sin egen tabel i databasen
 - * Medmindre der bruges nedarvning, hvor flere klasser deler samme tabel
- * Klassens navn er i ental (Person), tabellens navn er i flertal (people)
- * Tabellens primærnøgle er feltet "id", der er et autogenerated, unikt id for rækken
 - * Medmindre er tale om en ren mange-til-mange relationstabel
- * Med en smule kode kan man let afvige fra konventionerne

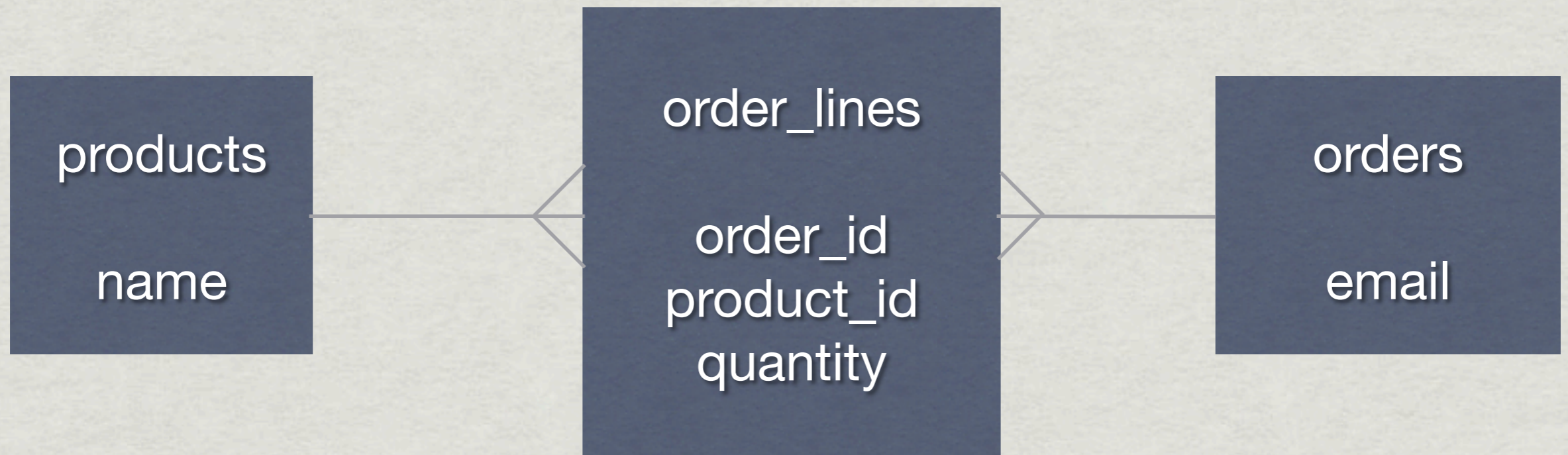
Features i ActiveRecord

- * Automatisk mapping mellem klasser og tabeller, attributter og felter
- * Automatisering af ændringer i databasestrukturen gennem migrations
- * Associationer der kontrolleres vha. simple metaprogrammeringsmakroer
- * Dynamiske opslag på alle attributter
- * Valideringsregler og -håndtering
- * Callbacks og observers til at overvåge og kontrollere et objekts livscyklus
- * Nedarvningshierarkier gennem Single Table Inheritance

Database migrations (1)

- * Kontroller databaseskemaets strukturelle ændringer på tværs af udviklings-, staging- og produktionsmiljøer
- * Løser problemet med hvordan man deler databaseændringer med andre udviklere, og efterfølgende får dem i produktion
- * Hver ændring har sin egen tidsstemplede klasse, der beskriver ændringen i et database-afhængigt sprog
- * Hver udgave af databasen holder selv styr på hvilke ændringer den har fået tilføjet

Database migrations (2)



Associationer (1)

- * Angiver relationerne mellem de forskellige tabeller i databasen, og derved associationerne mellem de forskellige klasser i applikationen.
- * Simpel og letlæselig syntaks
- * Gør det utroligt nemt at tilgå et objekts relaterede objekter vha. metoder på objektet
- * Muliggør også intelligent oprettelse, ændring og sletning af relationer.

Associationer (2)



har mange **order_lines**

har mange ordrer
gennem **order_lines**

tilhører en ordre

tilhører et produkt

har mange **order_lines**

har mange produkter
gennem **order_lines**

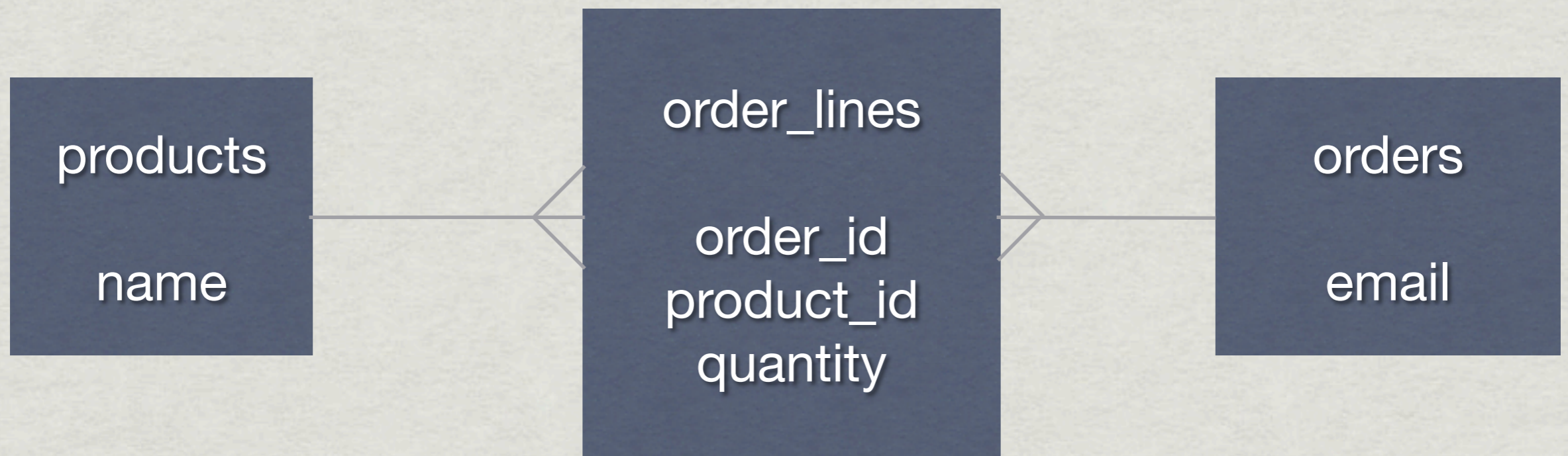
Dynamiske opslag

- * Rubys dynamiske natur tillader at der kan laves opslag gennem navngivne metoder på alle attributter
- * På samme måde kan der laves et kald, som enten returnerer eller opretter en model med de givne attributter, alt efter om den eksisterer i forvejen.

Validering (1)

- * Angiver betingelserne for at et givent objekt er gyldigt og kan gemmes i databasen.
- * Indbyggede valideringsfunktioner til de mest alm. ting:
 - * Må ikke være blank
 - * Må ikke være kortere eller længere end givne tal
 - * Skal være et tal - evt. indenfor et bestemt interval
 - * Skal være unik
 - * Osv.
- * Komplekse forretningsregler kan bygges på vha. custom valideringer.

Validering (2)



skal have et navn

navnet skal være på
mellem 5 og 255
karakterer

**skal have en ordre
skal have et produkt
skal have en mængde
mængden skal være
numerisk**

**skal have en email
emailen skal være
gyldig**

Callbacks og observers (1)

✱ Livscyklus for en ActiveRecord model:

- ✱ *save*
- ✱ *valid*
- ✱ **before_validation**
- ✱ **before_validation_on_create**
- ✱ *validate*
- ✱ *validate_on_create*
- ✱ **after_validation**
- ✱ **after_validation_on_create**
- ✱ **before_save**
- ✱ **before_create**
- ✱ *create*
- ✱ **after_create**
- ✱ **after_save**

Callbacks og observers (2)

- ✱ Callbacks

- ✱ Placeres direkte i modellen

- ✱ Nedarves og kan “chaines”

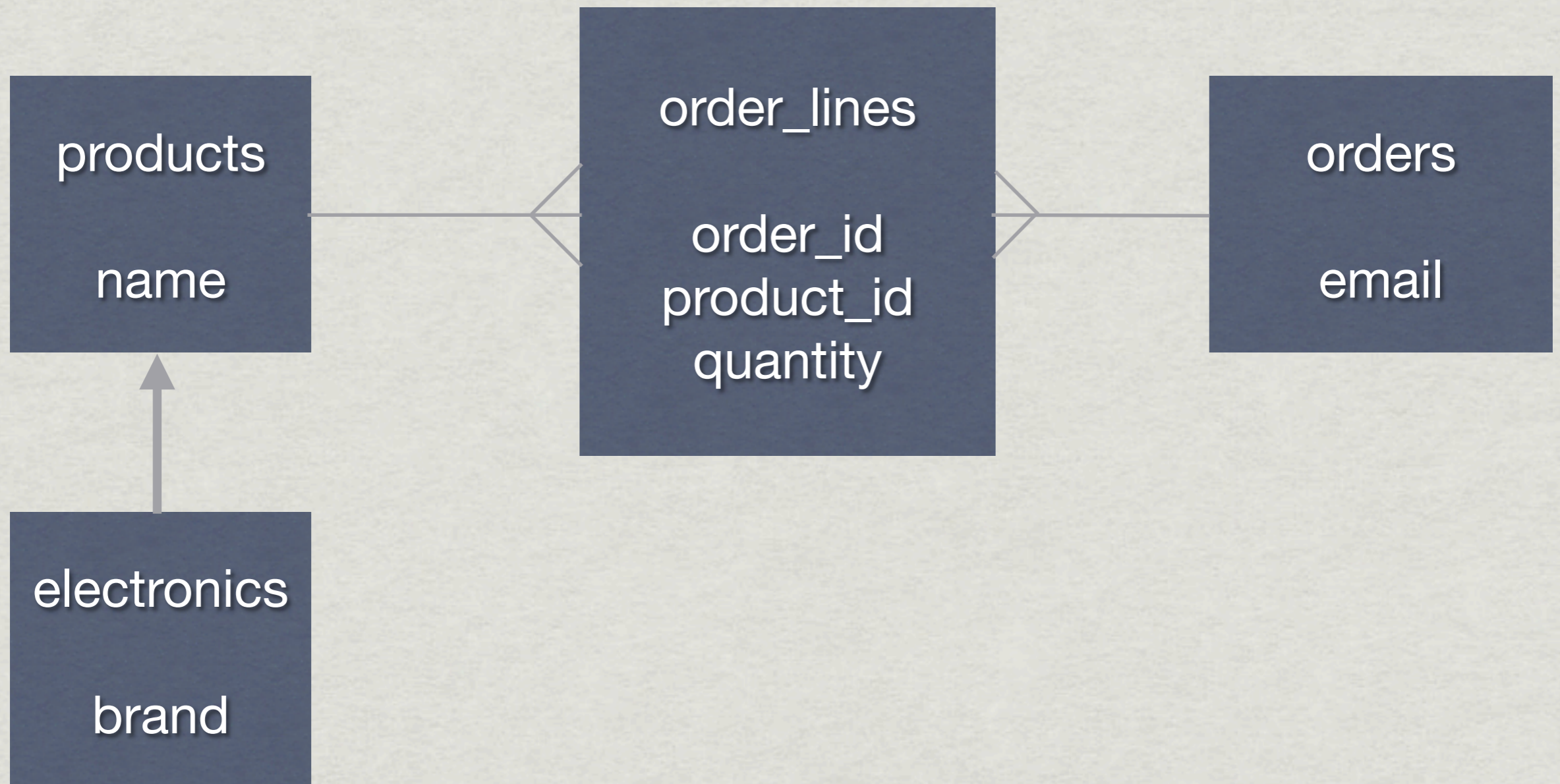
- ✱ Observers

- ✱ Selvstændig klasse der observerer udvalgte hændelser i bestemte modeller

Nedarvning (1)

- * Single Table Inheritance er indbygget er Rails
- * Andre nedarvningsstrategier findes i plugins
- * Bruges ved at:
 - * Tilføje et “type” felt på tabellen
 - * Lave en model der nedarver fra en anden model
 - * “Parent-modellen” skal passe med tabellen

Nedarvning (2)



Spørgsmål?

Casper Fabricius
<http://casperfabricius.com>