Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

ORM & OLAP

Object-oriented Enterprise Application Programming Model for In-Memory Databases

Sebastian Oergel

Probleme

- Datenbanken sind elementar für Business-Anwendungen
  - Gängiges Datenbankparadigma: relational
    - Ausgereift, bewährt
  - Gängige Programmiersprachen & -paradigmen basieren auf Objekten (OO)
    - Entwickler agieren auf Business Objects

- OR-Mapper wollen Welten miteinander verbinden
- Viele Probleme
    - Identität von Objekten, Vererbung, Data Retrieval (C**R**UD), ...

ORM & OLAP

- ORM beherrschen *Create*, *Update*, *Delete* recht gut
- Bieten oft auch verschiedene Data-Retrieval-Möglichkeiten (*Read*) an
  - Oft nicht optimal (OLAP)

- Beispiel:

ORM & OLAP

- ORM beherrschen *Create, Update, Delete* recht gut
- Biete
  - C
- Beis

OLTP-Query (*Login*)

SQL
```
SELECT `hash_password` FROM `user` WHERE
`user`.`id` = 324
```
OO:
```
User user = User.findById(324);
user.hash_password;
```

SQL:
```
INSERT INTO `user` VALUES („Eric
Miller“, ...)
```
OO:
```
User user = new User(„Eric Miller“, ...);
user.save();
```

OLAP-Query (*Dunning*)

SQL:

```
SELECT gbi.customerid AS customerid, COUNT(gbi.invoiceid) AS
numberOfInvoices, AVG(DSO) AS averageDSO
FROM (
 (
  SELECT so.customerid AS customerid,iv.id AS invoiceid,
pr.paymentdate AS paymentdate, iv.billingdate AS billingdate,
AVG(DATEDIFF(paymentdate, billingdate)) AS DSO
    FROM SalesOrder AS so, SalesOrderLineItem AS soli, Invoice AS iv,
PaymentReceipt AS pr
    WHERE so.customerid = 1
        AND so.id = soli.salesorderid
        AND soli.outboundid = iv.outboundid
        AND iv.id = pr.invoiceid
        AND iv.billingdate BETWEEN '2000-01-01' AND '2013-01-01'
        AND pr.paymentdate BETWEEN '2000-01-01' AND '2013-01-01'
    GROUP BY iv.id
 ) AS gbi # grouped by invoices
) GROUP BY gbi.customerid
```

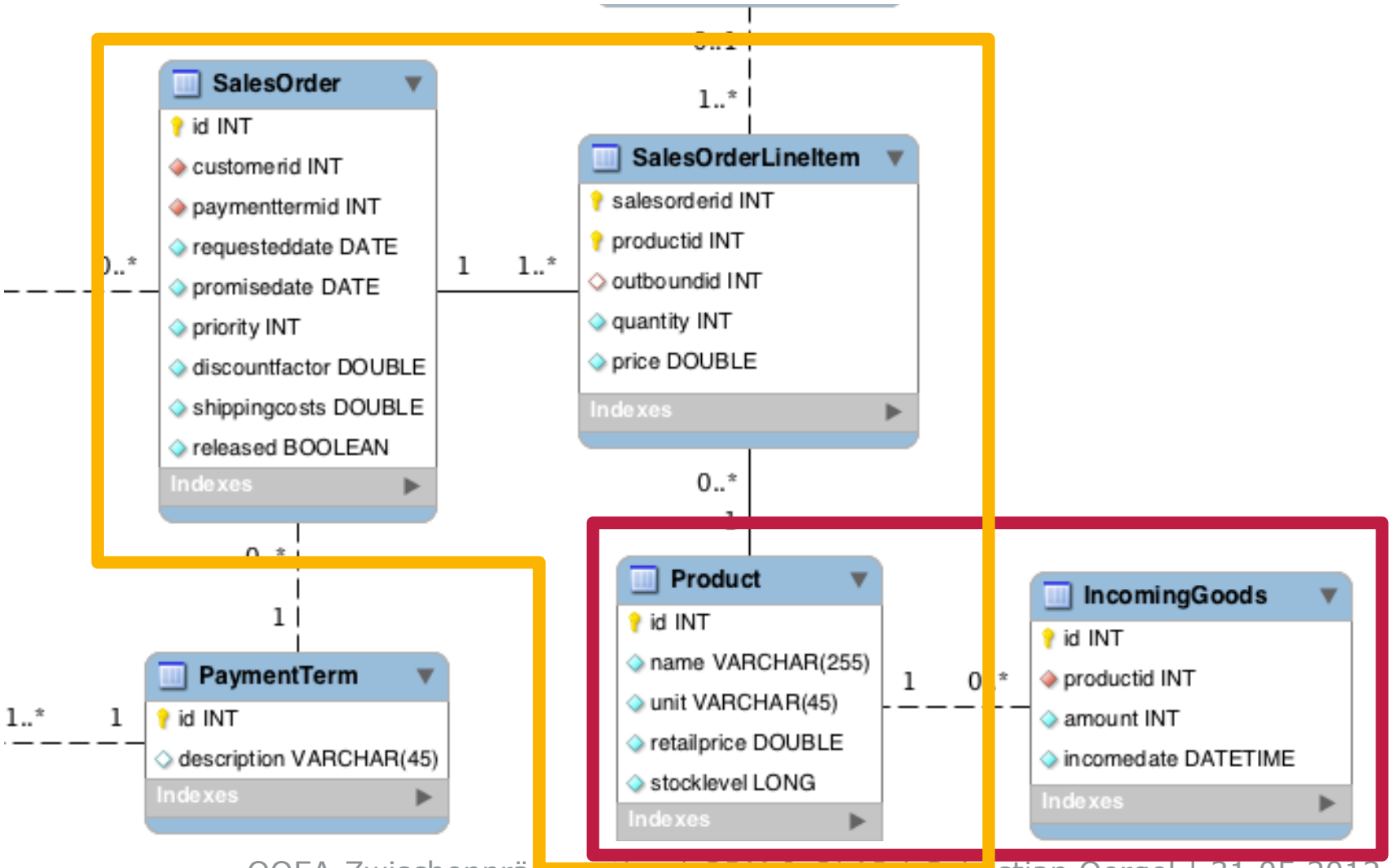
OO:

```
List<SalesOrder> salesOrder = SalesOrder.all();
...
```

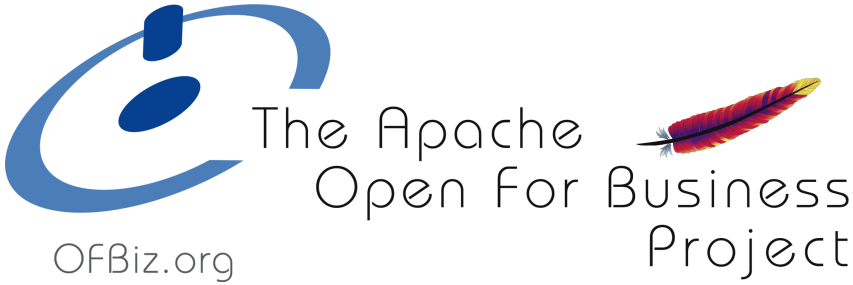
Einige Probleme:

- (meist) keine Unterstützung für Aggregate (Summen, ...)
  - Eigene Implementierung notwendig
- Keine Join-Optimierung wie SQL
- SQL: deklarativ (*was?*), OO: imperativ (*was? wie?*)

Problematische Verbindungen



Evaluierte Open-Source-ERP-Systeme



Evaluierte Open-Source-ERP-Systeme

OpenERP
OPEN SOURCE MANAGEMENT SOLUTION

2.3 Do not bypass the ORM

You should never use the database cursor directly when the ORM can do the same thing! By doing so you are bypassing all the ORM features, possibly the transactions, access rights and so on.

```
86      product = self.browse(cr, uid, rec_id, context=c)
87      qty = product.qty_available
88      diff = product.standard_price - new_price
89      if not diff: raise osv.except_osv(_('Error!'), _("Could not find any di
90      if qty:
91          company_id = location.company_id and location.company_id.id or Fals
92          if not company_id: raise osv.except_osv(_('Error!'), _('Company is
93          #
94          # Accounting Entries
95          #
96          if not journal_id:
97              journal_id = product.categ_id.property_stock_journal and produc
98          if not journal_id:
```

OpenERP – Verfügbarkeitsprüfung


```
256             where.append(tuple(date_values))
257         if 'in' in what:
258             # all moves from a location out of the set to a location in the set
259             cr.execute(
260                 'select sum(product_qty), product_id, product_uom '\
261                 'from stock_move '\
262                 'where location_id NOT IN %s '\
263                 'and location_dest_id IN %s '\
264                 'and product_id IN %s '\
265                 '' + (prodlot_id and ('and prodlot_id = ' + str(prodlot_id)) or '') +
266                 'and state IN %s ' + (date_str and 'and '+date_str+' ' or '') +' '\
267                 'group by product_id,product_uom',tuple(where))
268             results = cr.fetchall()
269         if 'out' in what:
270             # all moves from a location in the set to a location out of the set
271             cr.execute(
272                 'select sum(product_qty), product_id, product_uom '\
273                 'from stock_move '\
274                 'where location_id IN %s '\
275                 'and location_dest_id NOT IN %s '\
276                 'and product_id  IN %s '\
277                 '' + (prodlot_id and ('and prodlot_id = ' + str(prodlot_id)) or '') +
278                 'and state in %s ' + (date_str and 'and '+date_str+' ' or '') + ' '\
279                 'group by product_id,product_uom',tuple(where))
280             results2 = cr.fetchall()
```

OpenERP – Mahnungen


```
cr.execute("""

        SELECT
            l.id as id,
            l.partner_id AS partner_id,
            min(l.date) AS date_move,
            max(l.date) AS date_move_last,
            max(l.followup_date) AS date_followup,
            max(l.followup_line_id) AS followup_id,
            sum(l.debit) AS debit,
            sum(l.credit) AS credit,
            sum(l.debit - l.credit) AS balance,
            l.company_id AS company_id,
            l.blocked as blocked,
            l.period_id AS period_id
        FROM
            account_move_line l
            LEFT JOIN account_account a ON (l.account_id = a.id)
        WHERE
            a.active AND
            a.type = 'receivable' AND
            l.reconcile_id is NULL AND
            l.partner_id IS NOT NULL
        GROUP BY
            l.id, l.partner_id, l.company_id, l.blocked, l.period_id
    """)
```

Lösungsansätze

- Existieren
- Bringen aber oft neue Probleme mit sich
- Z.B.:
  - *ORM-Query-Languages*
  - Erlauben mehr Kontrolle über das Verhalten des ORM
    - Code wird sehr schnell wieder „technisch“
    - „verpacktes SQL“
    - Kein reines Arbeiten auf Objekten

```
User.select(„name“, „password“).where(„login_number
> 100“).fetchFirst(5)
```

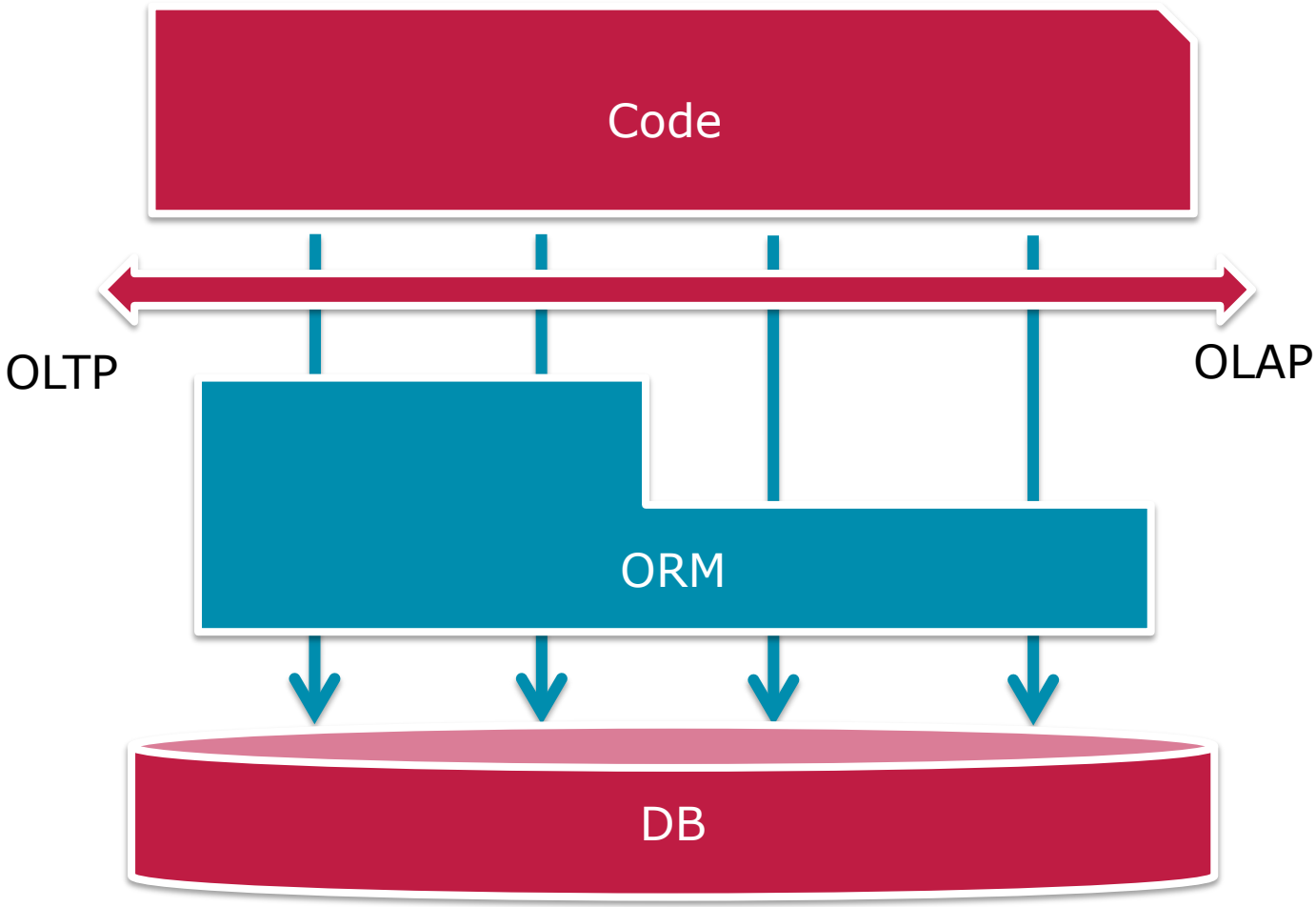
```
ORM.execute(„FIND user.{name, password} OF User
WHERE user.login_number > 100 FETCH FIRST 4“)
```

Future Work

- Weitere Evaluierung anhand von OpenERP
  - Alternative Implementierung von SQL-Queries über ORM/Objekte
  - Vergleiche:
    - Laufzeit
    - Code
      - Wie technisch?
      - Wie fehleranfällig?

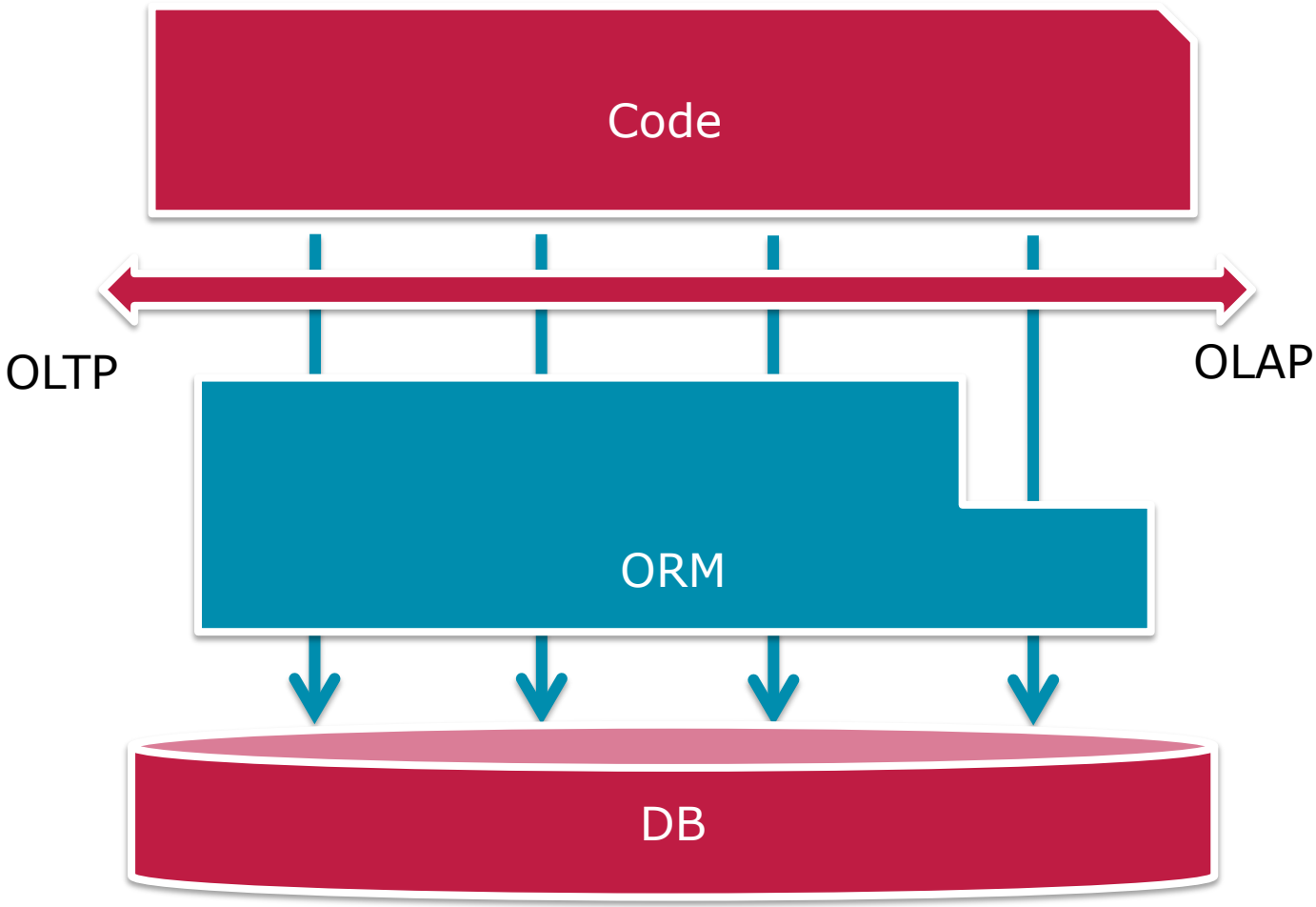
- Wie kann man gängige ORM zur Umgehung **einiger business-relevanter Probleme** erweitern?
  - Arbeit auf Objekten
  - Weniger technischer Code
  - Bessere Laufzeiten als bisherige reine ORM-Lösungen
  - StoredProcedures?

Übersicht



Use Case: Business-Anwendungen, Analytics

Übersicht



Use Case: Business-Anwendungen, Analytics

Weitere Beispielqueries

Forecast

```
cr.execute("SELECT period.date_stop, forecast.product_qty, forecast.product_uom \
    FROM stock_sale_forecast AS forecast \
    LEFT JOIN stock_period AS period \
    ON forecast.period_id = period.id \
    WHERE (forecast.user_id = %s OR forecast.create_uid = %s OR forecast.write_uid = %s) \
        AND forecast.warehouse_id = %s AND forecast.product_id = %s \
        AND period.date_stop < %s \
    ORDER BY period.date_stop DESC",
    (uid, uid, uid, f.warehouse_id.id, p.id, f.period_id.date_stop) )
```

Weitere Beispielqueries

MRP

```
cr.execute("SELECT SUM(mrp_production_workcenter_line.hour) AS hours, SUM(mrp_production_workcenter_line.cycle) AS cycles, \
        resource_resource.name AS name, mrp_workcenter.id AS id \
    FROM mrp_production_workcenter_line, mrp_production, mrp_workcenter, resource_resource \
    WHERE (mrp_production_workcenter_line.production_id=mrp_production.id) \
       AND (mrp_production_workcenter_line.workcenter_id=mrp_workcenter.id) \
       AND (mrp_workcenter.resource_id=resource_resource.id) \
       AND (mrp_workcenter.id=%s) \
       AND (mrp_production.date_planned BETWEEN %s AND %s) \
    GROUP BY mrp_production_workcenter_line.workcenter_id, resource_resource.name, mrp_workcenter.id \
    ORDER BY mrp_workcenter.id", (workcenter['id'], dates[date]['start'] + ' 00:00:00', dates[date]['stop'] + ' 23:59:59'))
```

Weitere Beispielqueries

Geo-Location

```
cr.execute(
"""SELECT id, distance
    FROM  (select id, (point(partner_longitude, partner_latitude) <-> point(%s,%s)) AS distance FROM res_partner
    WHERE partner_longitude is not null
    AND partner_latitude is not null
    AND partner_weight > 0) AS d
    ORDER BY distance LIMIT 1""", (longitude, latitude))
```

Future Work: Evaluierung Squeryl ORM

- Sieht sich selbst als alternativen ORM-Ansatz
- Kein JPA

```
// Equivalent JPA query
Query q = entityManager.createQuery(
  //We'll get an SQLException if there's a typo here :
  "SELECT AVG(g.scoreInPercentage) FROM Grades g where g.subjectId = :subjectId");

// a runtime exception if mathId is of the wrong type
q.setParameter(1, mathId); // or if 1 is not the right index

// ClassCastExeption possible
Number avg = (Number) q.getSingleResult();

// NullPointerException if the query returns null
//(ex.: if there are no math Grades in the table)
avg.floatValue();
```

JPA

Squeryl

```
//All is validated at compile time here :

var avg: Option[Float] = // The compiler 'knows' that this query returns an
Option[Float]
  from(grades)(g =>
    // mathId has to by type compatible with g.subjectId to compile
    where(g.subjectId === mathId)
    compute(avg(g.scoreInPercentage))
  )
```

Future Work: Evaluierung Squeryl ORM

- Sieht sich selbst als alternativen ORM-Ansatz
- Kein JPA

- Wie technisch ist der Code wirklich?
  - Werden die „Versprechungen“ eingehalten? Auf welche Kosten?
  - Lassen sich die Konzepte auf „große“ (Enterprise-)ORM *(Hibernate, ...)* anwenden?
    - Warum? Warum nicht? Mit welchen Einschränkungen?