POLITECNICO
MILANO 1863

Project DB2

Telco App

Luca Genoni - 10520445

May 30, 2022

Index

Specification

Luca Genoni - 10520445

POLITECNICO MILANO 1863

User

TELCO SERVICE APPLICATIONS

A telco company offers pre-paid online services to web users. Two client applications using the same database need to be developed.

CONSUMER APPLICATION

The consumer application has a public Landing page with a form for login and a form for registration. Registration requires a username (which can be assumed as the unique identification parameter), a password and an email.

Login leads to the Home page of the consumer application. Registration leads back to the landing page where the user can log in.

The user can log in before browsing the application or browse it without logging in. If the user has logged in, his/her username appears in the top right corner of all the application pages.

Packages

The Home page of the consumer application displays the service packages offered by the telco company.
A service package has an ID and a name (e.g., "Basic", "Family", "Business", "All Inclusive", etc). It comprises one or more services. Services are of four types: fixed phone, mobile phone, fixed internet, and mobile internet. The mobile phone service specifies the number of minutes and number of SMSs included in the package plus the fee for extra minutes and the fee for extra SMSs. The fixed phone service has no specific configuration parameters. The mobile and fixed internet services specify the number of Gigabytes included in the package and the fee for extra Gigabytes.

Packages relations

A service package must be associated with one validity period. A validity period specifies the number of months (12, 24, or 36). Each validity period has a different monthly fee (e.g., 20€/month for 12 months, 18€/month for 24 months, and 15€/month for 36 months).

A service package may be associated with one or more optional products (e.g., an SMS news feed, an internet TV channel, etc.). The validity period of an optional product is the same as the validity period that the user has chosen for the service package. An optional products has a name and a monthly fee independent of the validity period duration. The same optional product can be offered in different service packages.

Buy page

From the Home page, the user can access a Buy Service page for purchasing a service package and thus creating a service subscription. The Buy Service page contains a form for purchasing a service package. The form allows the user to select one service package from the list of available ones and choose the validity period duration and the optional products to buy together with the chosen service. The form also allows the user to select the start date of his/her subscription. After choosing the service packages, the validity period and (0 or more) optional products, the user can press a CONFIRM button.

Confirmation page

The application displays a CONFIRMATION page that summarizes the details of the chosen service package, the validity period, the optional products and the total price to be pre-paid: (monthly fee of service package * number of months) + (sum of monthly fees of options * number of months).
If the user has already logged in, the CONFIRMATION page displays a BUY button. If the user has not logged in, the CONFIRMATION page displays a link to the login page and a link to the REGISTRATION page. After either logging in or registering and immediately logging in, the CONFIRMATION page is redisplayed with all the confirmed details and the BUY button.

Order & successful payment

When the user presses the BUY button, an order is created. The order has an ID and a date and hour of creation. It is associated with the user and with the service package, its validity period and the chosen optional products. It also contains the total value (as in the CONFIRMATION page) and the start date of the subscription. After creating the order, the application bills the customer by calling an external service. If the external service accepts the billing, the order is marked as valid and a service activation schedule is created for the user. A service activation schedule is a record of the services and optional products to activate for the user with their date of activation and date of deactivation.

After-effect of failed payment

If the external service rejects the billing, the order is put in the rejected status and the user is flagged as insolvent. When an insolvent user logs in, the home page also contains the list of rejected orders.
The user can select one of such orders, access the CONFIRMATION page, press the BUY button and attempt the payment again. When the same user causes three failed payments, an alert is created in a dedicated auditing table, with the user ID, username, email, and the amount, date and time of the last rejection.
[...]

After-effect of failed payment

If the external service rejects the billing, the order is put in the rejected status and the user is flagged as insolvent. When an insolvent user logs in, the home page also contains the list of rejected orders.
The user can select one of such orders, access the CONFIRMATION page, press the BUY button and attempt the payment again. When the same user causes three failed payments, an alert is created in a dedicated auditing table, with the user ID, username, email, and the amount, date and time of the last rejection.
[...]

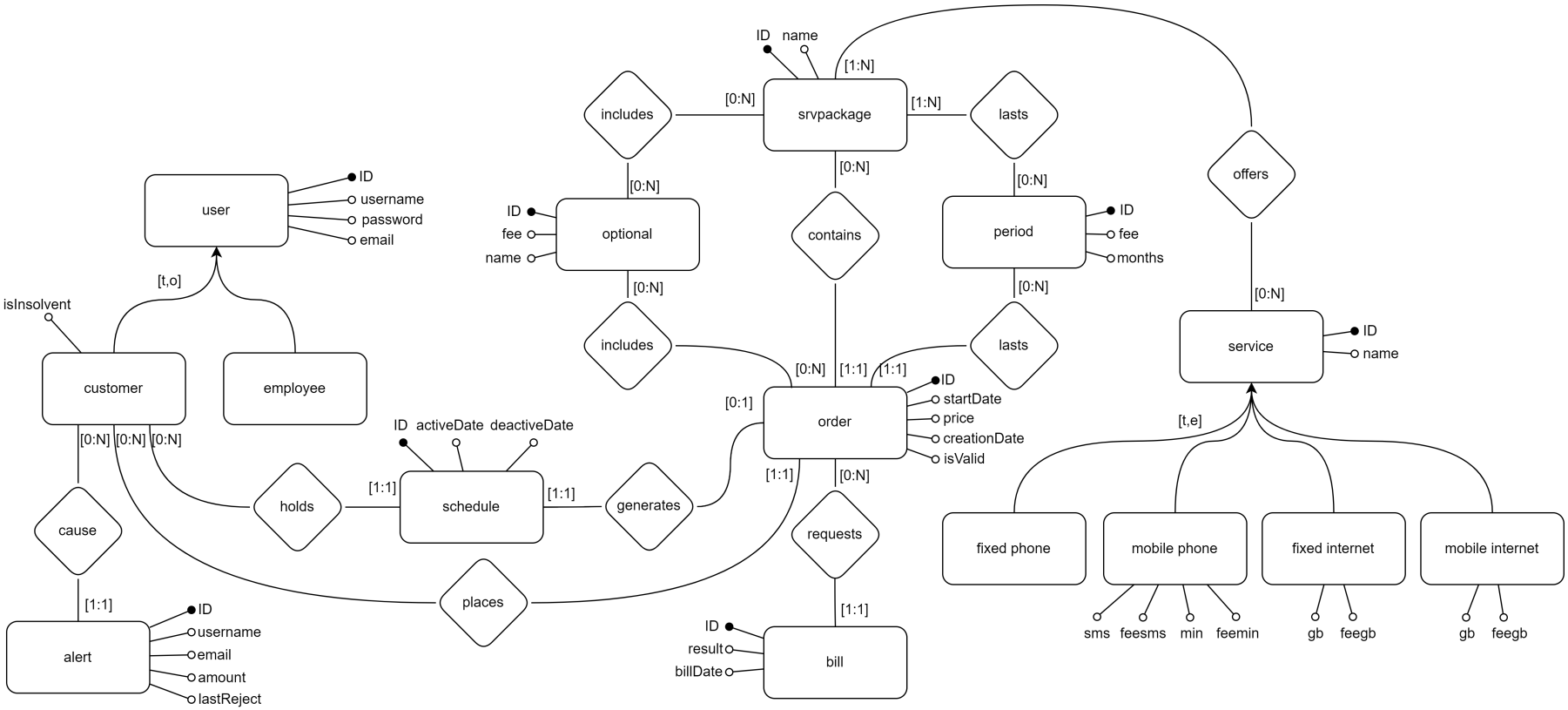
My assumptions

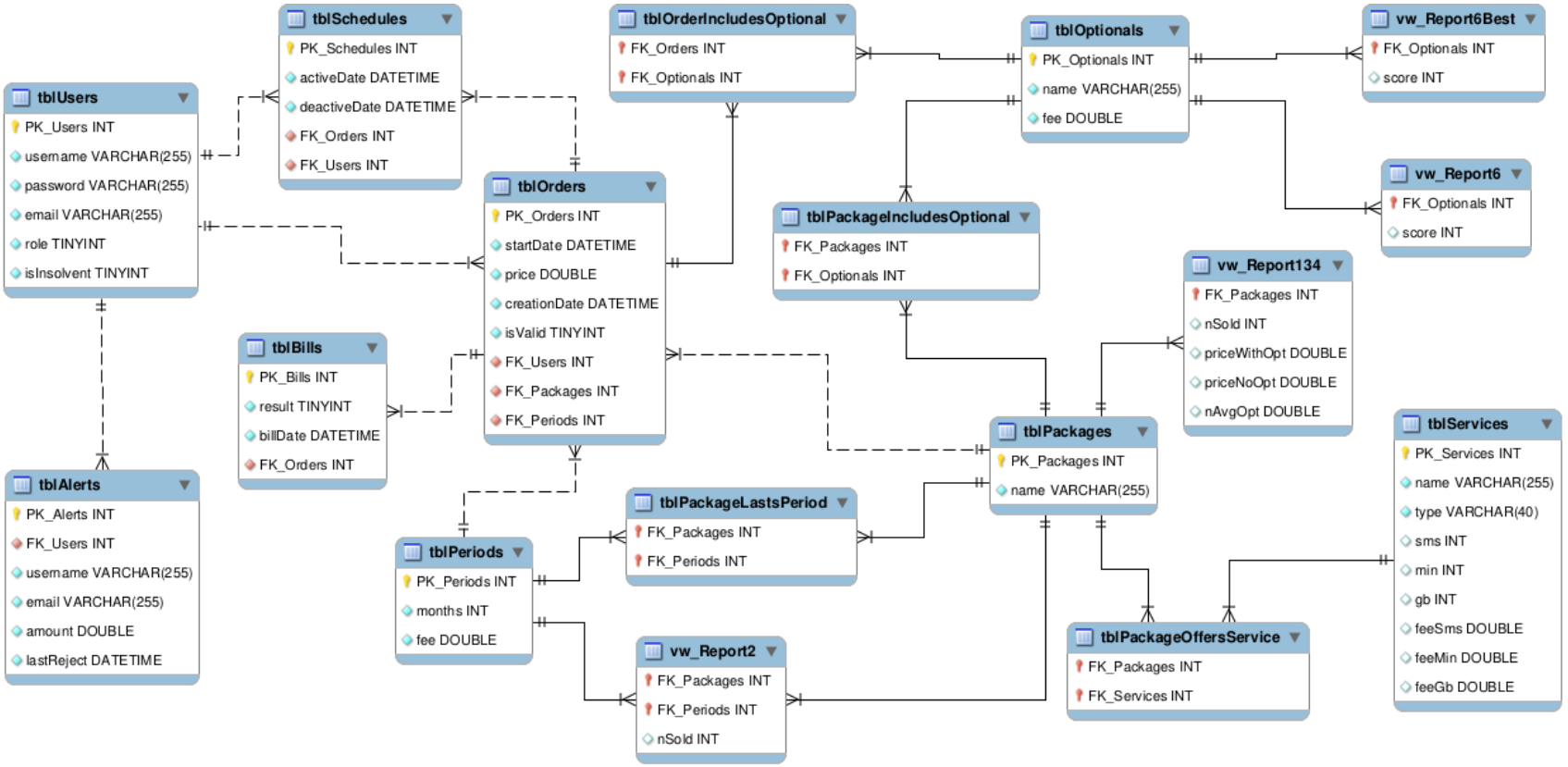
- Employee can also be customers
- Subscriptions are orders without a set user.
- A bill is a response from the payment gateway.
- An user is flagged insolvent with just one rejected bill of a non-valid order.
- An alert is generated with 3 or more reject bills of non-valid orders.
- An alert amount is the sum of prices of non-valid orders.
- The alert table is write-only for future inspection.

Conceptual and logical data models

Luca Genoni - 10520445

POLITECNICO MILANO 1863


ID
name
[1:N]
includes
[0:N]
srvpackage
[1:N]
lasts
offers
[0:N]
[0:N]
[0:N]
[0:N]
ID
fee
name
optional
contains
period
ID
fee
months
[0:N]
[0:N]
[0:N]
includes
lasts
[0:N]
[1:1]
[1:1]
ID
startDate
price
creationDate
isValid
order
[0:1]
[1:1]
[0:N]
service
ID
name
[t,e]
user
ID
username
password
email
[t,o]
isInsolvent
customer
employee
[0:N]
[0:N]
[0:N]
ID
activeDate
deactiveDate
[1:1]
schedule
[1:1]
holds
generates
cause
requests
places
[1:1]
[1:1]
[0:N]
fixed phone
mobile phone
fixed internet
mobile internet
sms
feesms
min
feemin
gb
feegb
gb
feegb
[1:1]
ID
username
email
amount
lastReject
alert
ID
result
billDate
bill


tblSchedules
PK_Schedules INT
activeDate DATETIME
deactiveDate DATETIME
FK_Orders INT
FK_Users INT
tblOrderIncludesOptional
FK_Orders INT
FK_Optionals INT
tblOptionals
PK_Optionals INT
name VARCHAR(255)
fee DOUBLE
vw_Report6Best
FK_Optionals INT
score INT
tblUsers
PK_Users INT
username VARCHAR(255)
password VARCHAR(255)
email VARCHAR(255)
role TINYINT
isInsolvent TINYINT
tblOrders
PK_Orders INT
startDate DATETIME
price DOUBLE
creationDate DATETIME
isValid TINYINT
FK_Users INT
FK_Packages INT
FK_Periods INT
tblPackageIncludesOptional
FK_Packages INT
FK_Optionals INT
vw_Report6
FK_Optionals INT
score INT
vw_Report134
FK_Packages INT
nSold INT
priceWithOpt DOUBLE
priceNoOpt DOUBLE
nAvgOpt DOUBLE
tblBills
PK_Bills INT
result TINYINT
billDate DATETIME
FK_Orders INT
tblPackages
PK_Packages INT
name VARCHAR(255)
tblServices
PK_Services INT
name VARCHAR(255)
type VARCHAR(40)
sms INT
min INT
gb INT
feeSms DOUBLE
feeMin DOUBLE
feeGb DOUBLE
tblAlerts
PK_Alerts INT
FK_Users INT
username VARCHAR(255)
email VARCHAR(255)
amount DOUBLE
lastReject DATETIME
tblPackageLastsPeriod
FK_Packages INT
FK_Periods INT
tblPeriods
PK_Periods INT
months INT
fee DOUBLE
vw_Report2
FK_Packages INT
FK_Periods INT
nSold INT
tblPackageOffersService
FK_Packages INT
FK_Services INT

Trigger design and code

Luca Genoni - 10520445

POLITECNICO MILANO 1863

Report134

TR_tblPackages_Insertvw_Report134

- E: insert packages
- A: insert vw_Report134
  - After for each row

```
1  create trigger TR_tblPackages_Insertvw_Report134
2  after insert on tblPackages for each row
3  begin
4      insert into vw_Report134 (FK_Packages) values (new.
           PK_Packages);
5  end;
```

TR_tblOrders_Updatevw_Report134

- E: update order
- C: is valid
- A: update vw_Report134
  - After for each row

```
1  create trigger TR_tblOrders_Updatevw_Report134
2  after update on tblOrders for each row
3  begin
4      declare var_nSold int;
5      declare var_nOpt int;
6      declare var_months int;
7      declare var_fee double;
8
9      if new.isValid = 1 then
10         select count(*) into var_nSold from tblOrders where FK_Packages = new.
             FK_Packages and isValid = 1;
11         select count(*) into var_nOpt from tblOptionals opt,
             tblOrderIncludesOptional oio where oio.FK_Orders = new.PK_Orders and
             opt.PK_Optionals = oio.FK_Optionals;
12         select months, fee into var_months, var_fee from tblPeriods where
             PK_Periods = new.FK_Periods;
13         update vw_Report134 set nSold = var_nSold
14             ,priceWithOpt = priceWithOpt + new.price
15             ,priceNoOpt = priceNoOpt + (var_fee * var_months)
16             ,nAvgOpt = ((nAvgOpt*(var_nSold-1)) + var_nOpt)/(var_nSold)
17             where FK_Packages = new.FK_Packages;
18     end if;
19 end;
```

Report2

TR_tblPackageLastsPeriod_Insertvw_Report2

- E: insert period of a package
- A: insert vw_Report2
  - After for each row

```
1 create trigger TR_tblPackageLastsPeriod_Insertvw_Report2
2 after insert on tblPackageLastsPeriod for each row
3 begin
4     insert into vw_Report2 (FK_Packages, FK_Periods) values
          (new.FK_Packages,new.FK_Periods);
5 end;
```

TR_tblOrders_Updatevw_Report2

- E: update order
- C: is valid
- A: update vw_Report2
  - After for each row

```
1 create trigger TR_tblOrders_Updatevw_Report2
2 after update on tblOrders for each row
3 begin
4     if new.isValid = 1 then
5         update vw_Report2 set nSold = nSold + 1
6             where FK_Packages = new.FK_Packages and
          FK_Periods = new.FK_Periods;
7     end if;
8 end;
```

Report6

TR_tblOptionals_Insertvw_Report6

- E: insert optional
- A: insert vw_Report6
  - After for each row

```
1  create trigger TR_tblOptionals_Insertvw_Report6
2  after insert on tblOptionals for each row
3  begin
4      insert into vw_Report6 (FK_Optionals) values (new.
           PK_Optionals);
5  end;
```

TR_tblOrders_Updatevw_Report6

- E: update order
- C: is valid
- A: update vw_Report6, delete/insert vw_Report6Best
  - After for each row

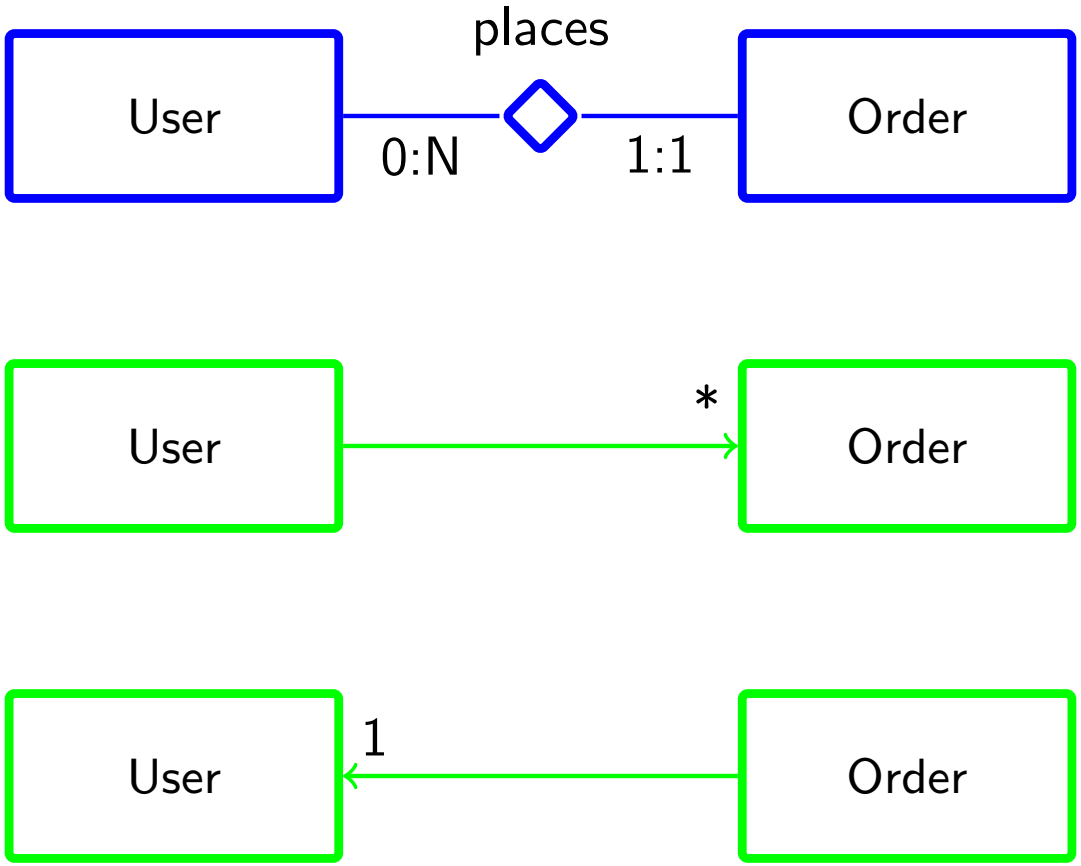
```
1  create trigger TR_tblOrders_Updatevw_Report6
2  after update on tblOrders for each row
3  begin
4      declare var_months int;
5      declare var_fee double;
6
7      if new.isValid = 1 then
8          select months into var_months from tblPeriods where PK_Periods = new.
               FK_Periods;
9          delete from vw_Report6Best where score = (select Max(r.score) from
               vw_Report6 r);
10         update vw_Report6 r
11                 inner join tblOptionals o on r.FK_Optionals = o.PK_Optionals
12                 inner join tblOrderIncludesOptional oio on r.FK_Optionals = oio.
               FK_Optionals
13             set r.score = r.score + (o.fee*var_months)
14             where oio.FK_Orders = new.PK_Orders;
15         insert into vw_Report6Best (FK_Optionals, score)
16             select *
17             from vw_Report6
18             where score = (select Max(r.score) from vw_Report6 r);
19     end if;
20 end;
```

ORM relationship design with explanations

Luca Genoni - 10520445

POLITECNICO MILANO 1863

User-Order



User → Order

@OneToMany

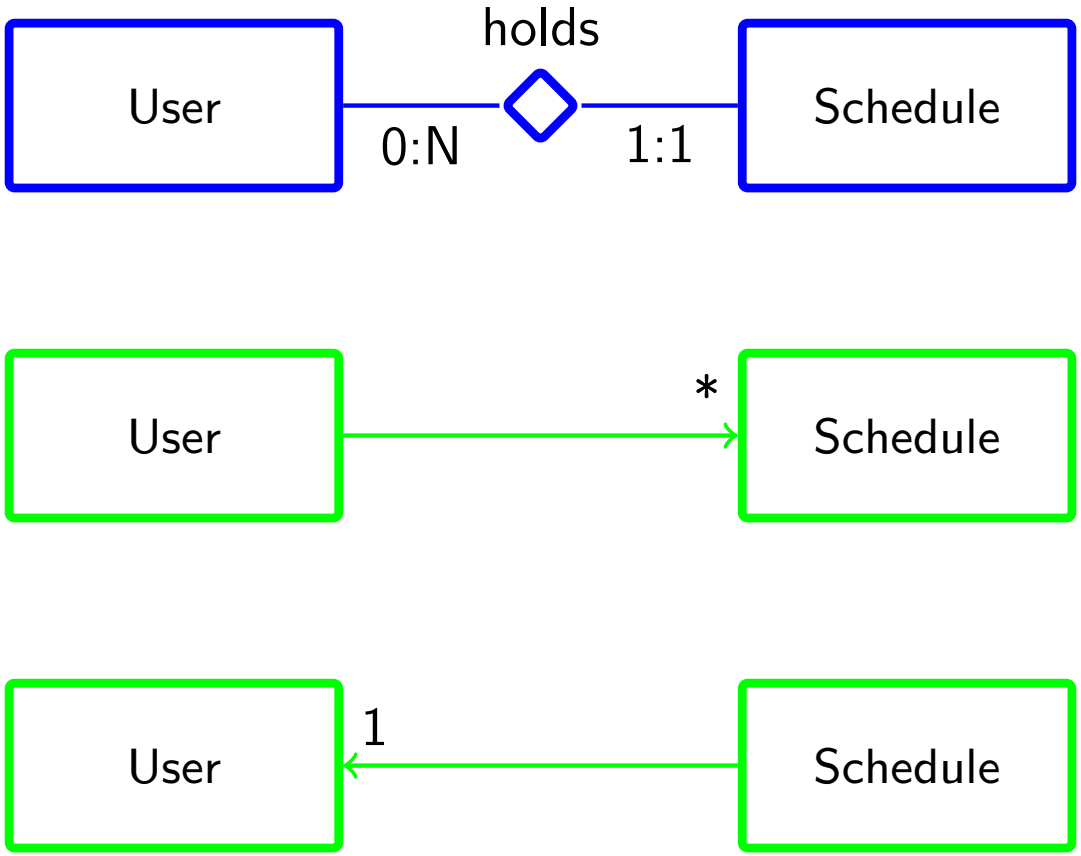
- default fetch LAZY
- cascade ALL
- orphanRemoval

Order → User

@ManyToOne

- fetch LAZY

User-Schedule



User → Schedule

@OneToMany

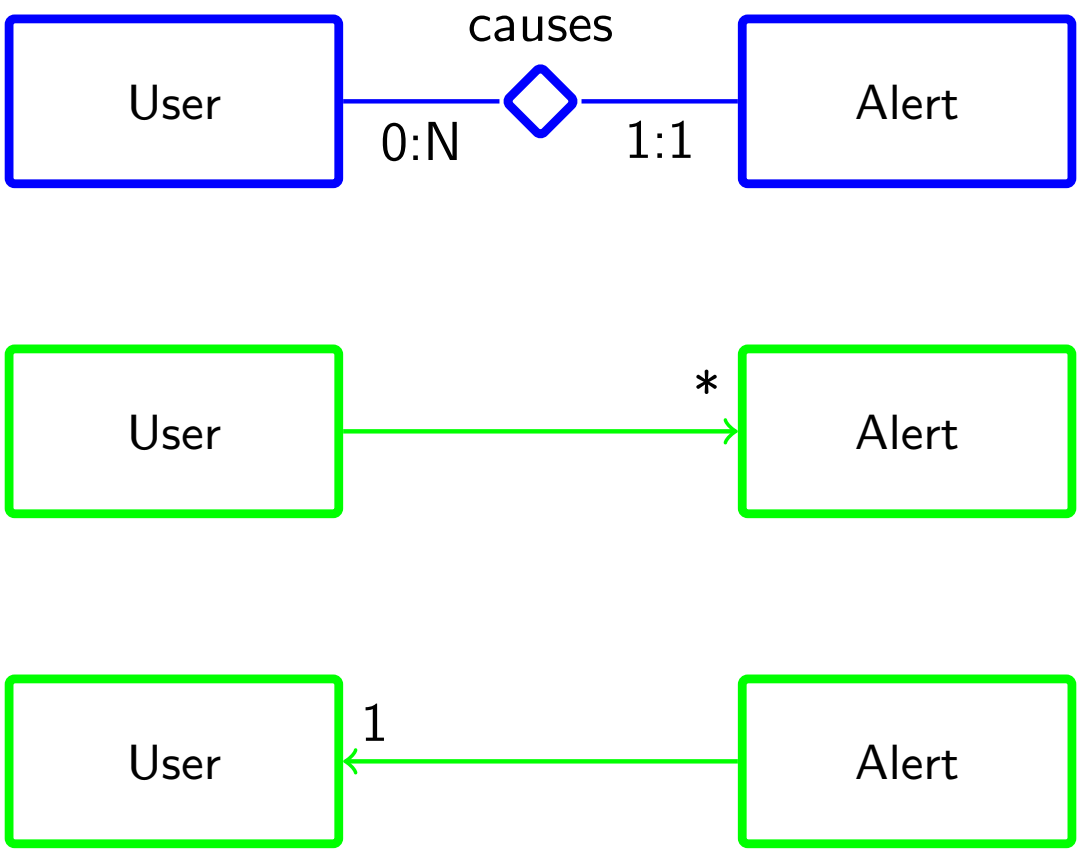
- default fetch LAZY
- cascade ALL
- orphanRemoval

Schedule → User

@ManyToOne

- fetch LAZY

User-Alert



User → Alert

@OneToMany

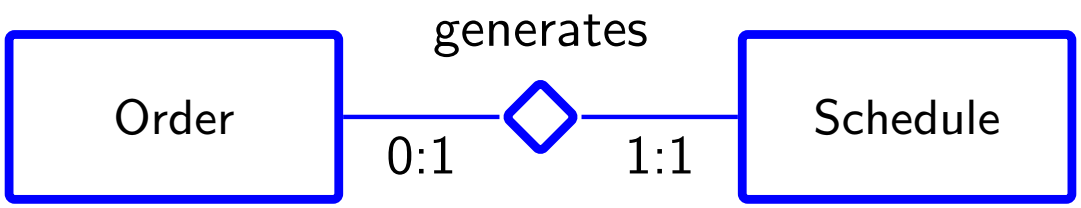
- default fetch LAZY
- cascade PERSIST, MERGE, REFRESH, DETACH

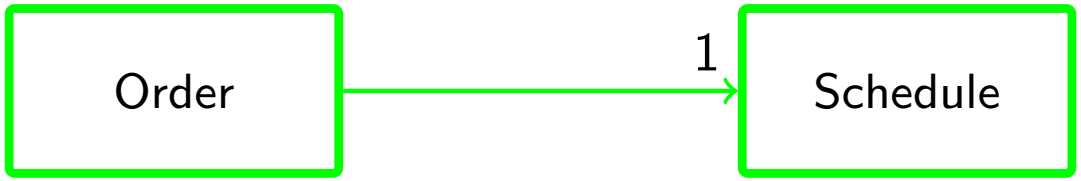
Alert → User

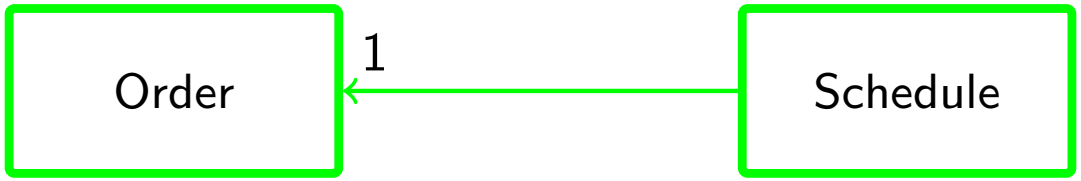
@ManyToOne

- fetch LAZY

Order-Schedule







Order $\rightarrow$ Schedule

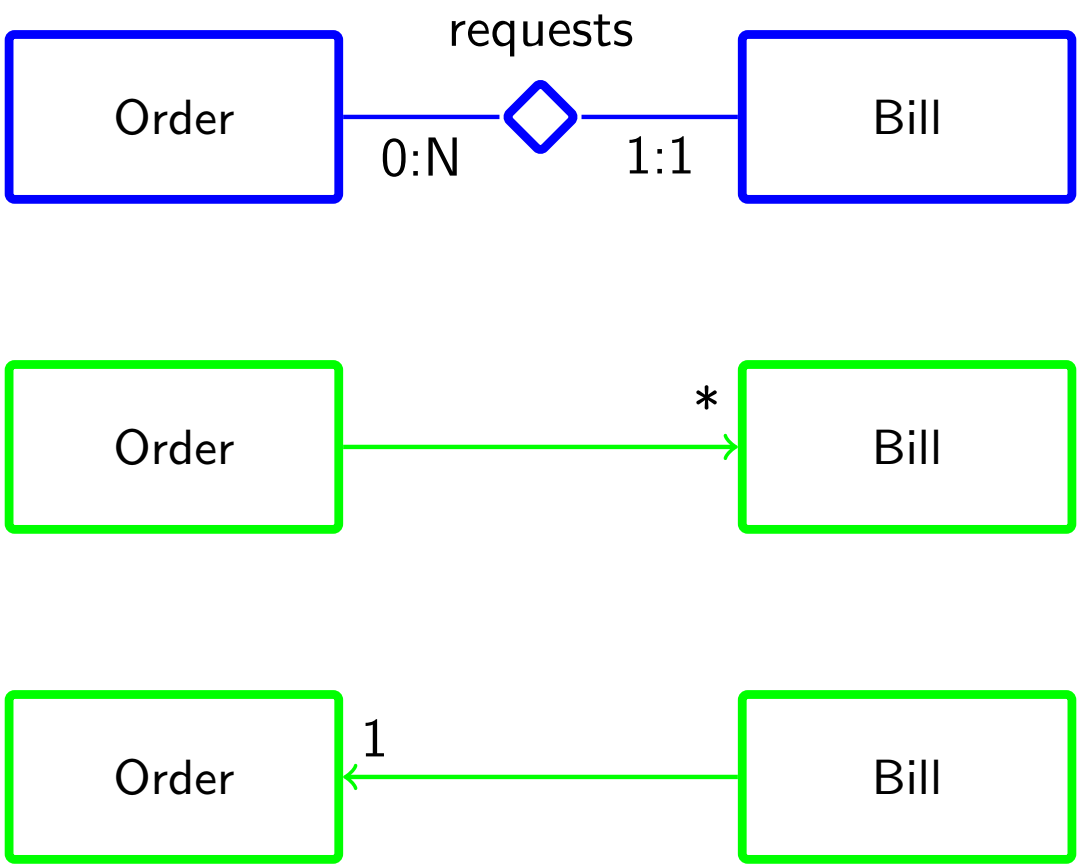
/

Schedule $\rightarrow$ Order

@OneToOne

- fetch LAZY

Order-Bill



Order → Bill

@OneToMany

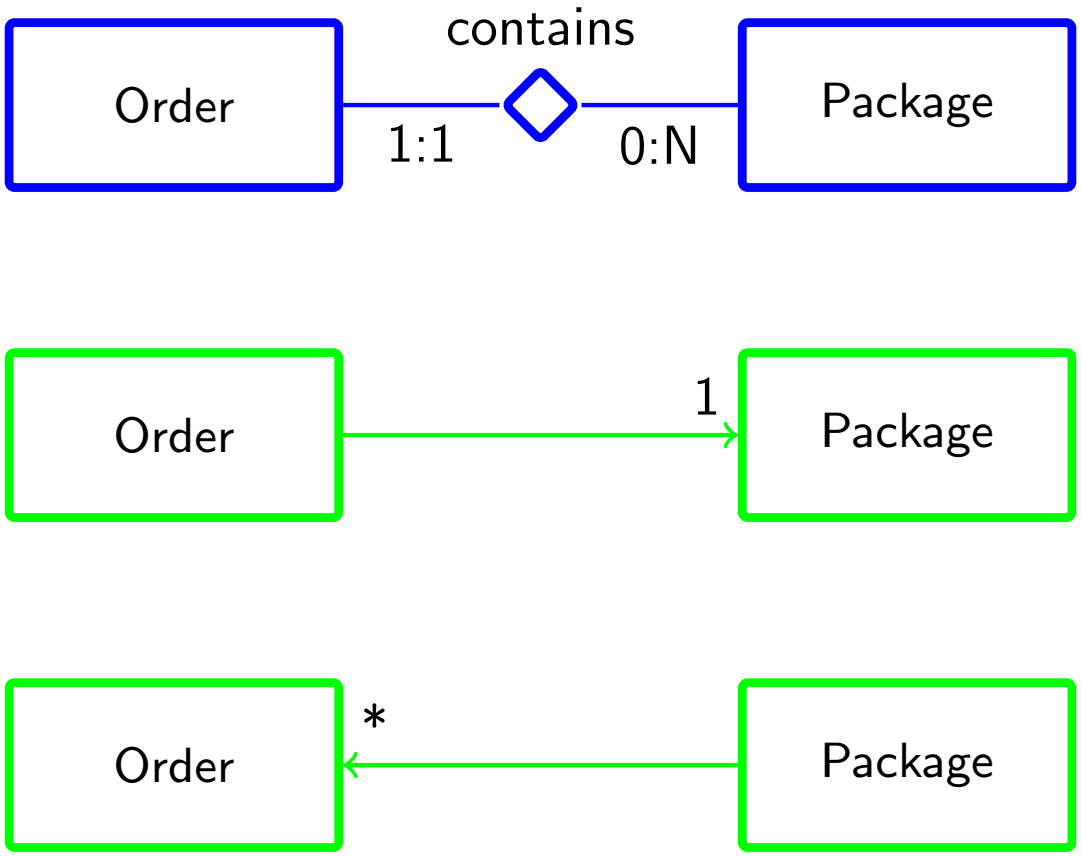
- default fetch LAZY
- cascade ALL
- orphanRemoval

Bill → Order

@ManyToOne

- fetch LAZY

Order-Package



Order $\rightarrow$ Package

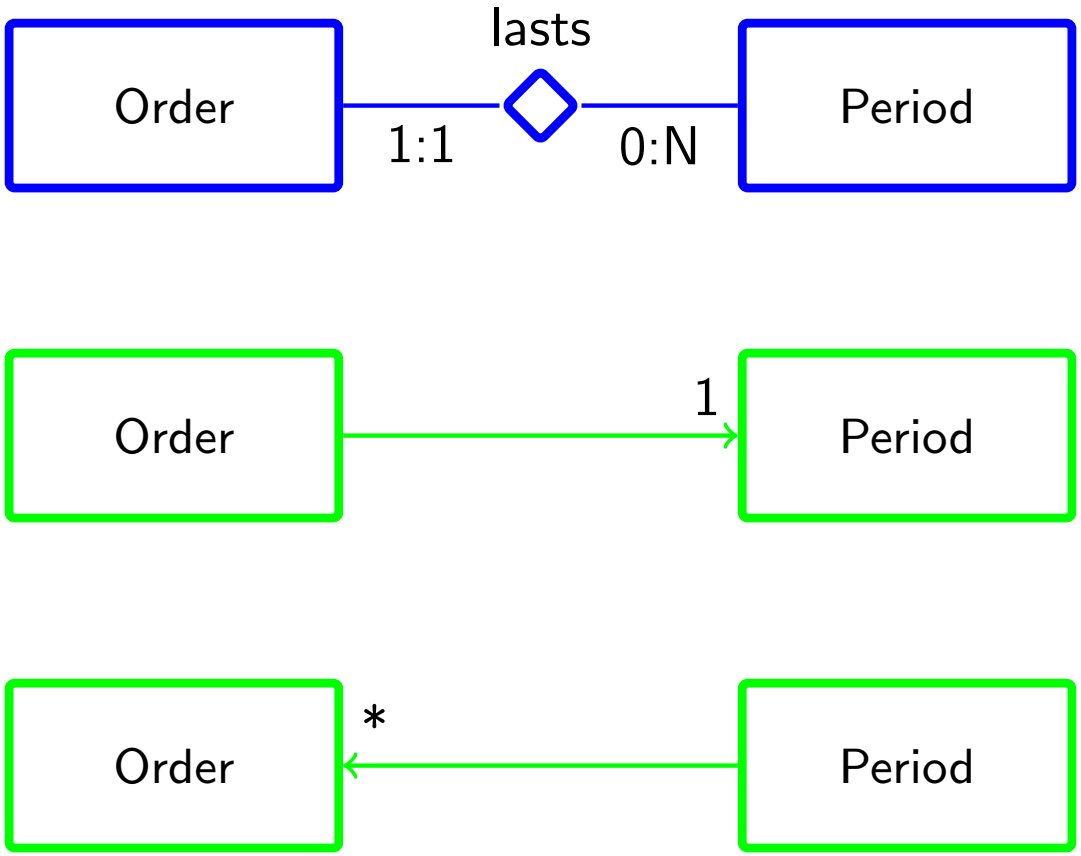
@ManyToOne

- fetch LAZY

Package $\rightarrow$ Order

/

Order-Period



Order → Period

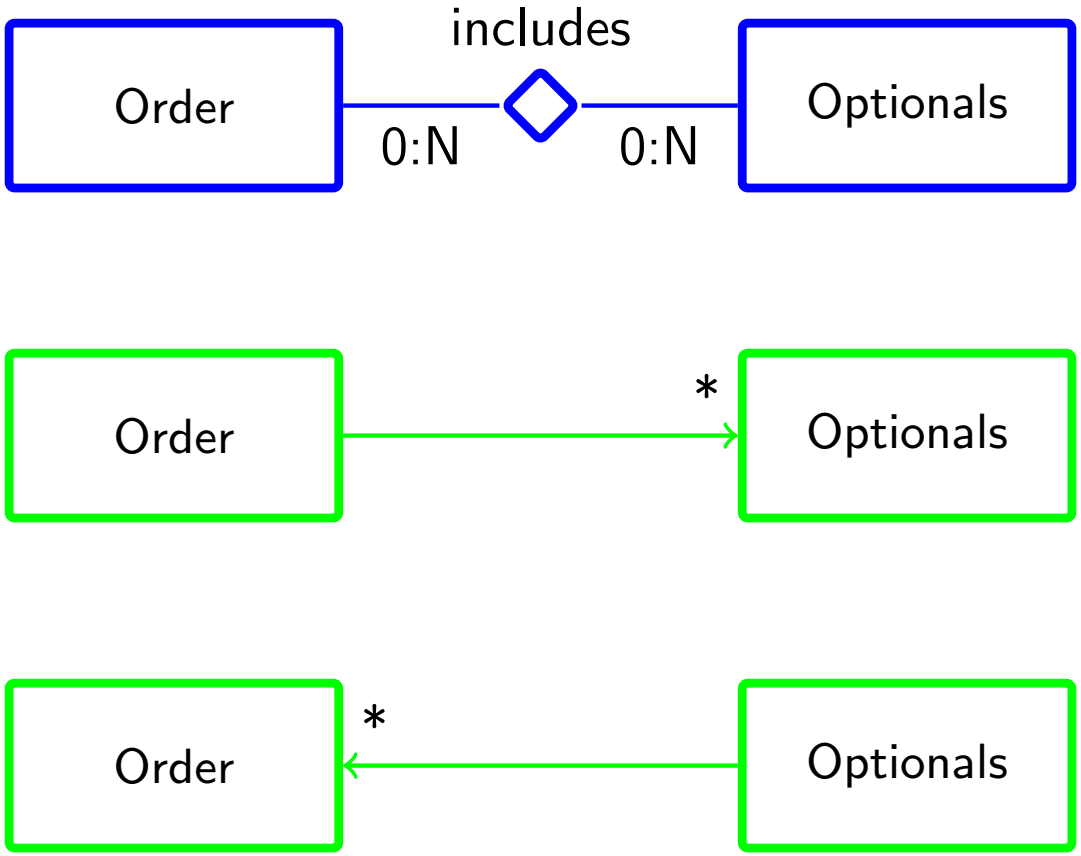
@ManyToOne

- fetch LAZY

Period → Order

/

Order-Optionals



Order → Optionals

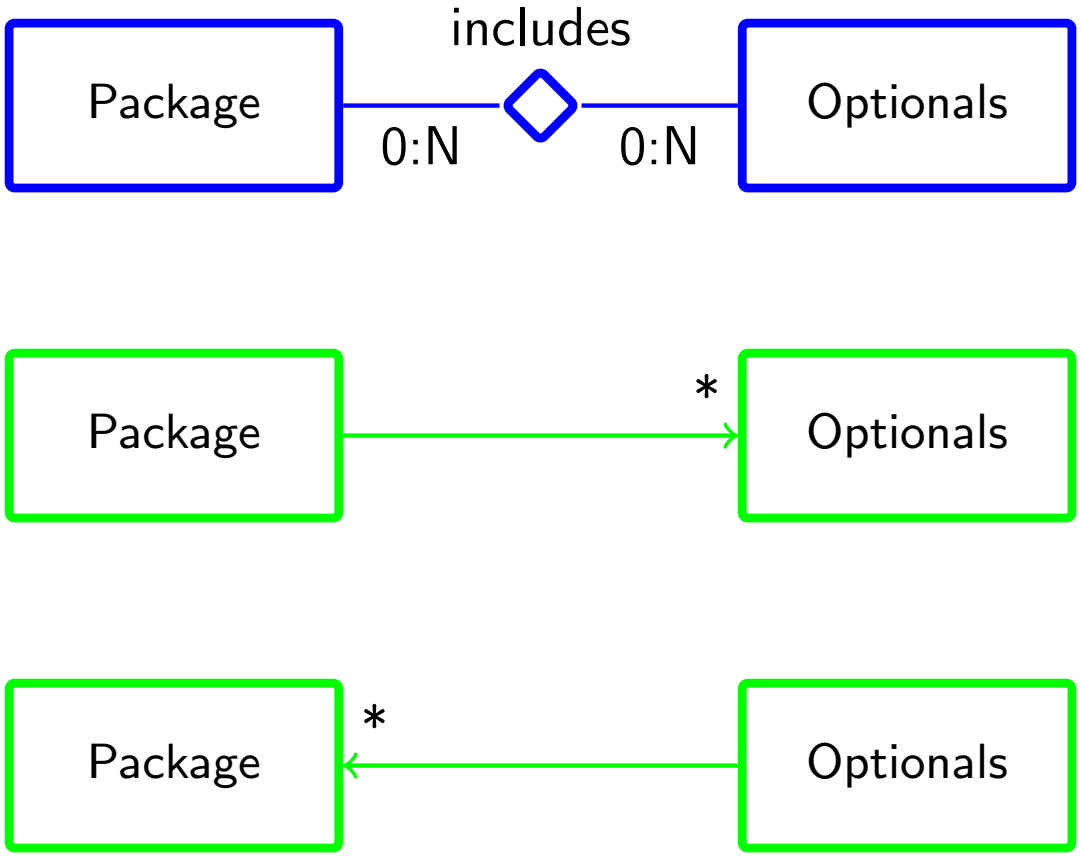
@ManyToMany

- default fetch LAZY
- cascade PERSIST, MERGE

Optionals → Order

/

Package-Optionals



Package → Optionals

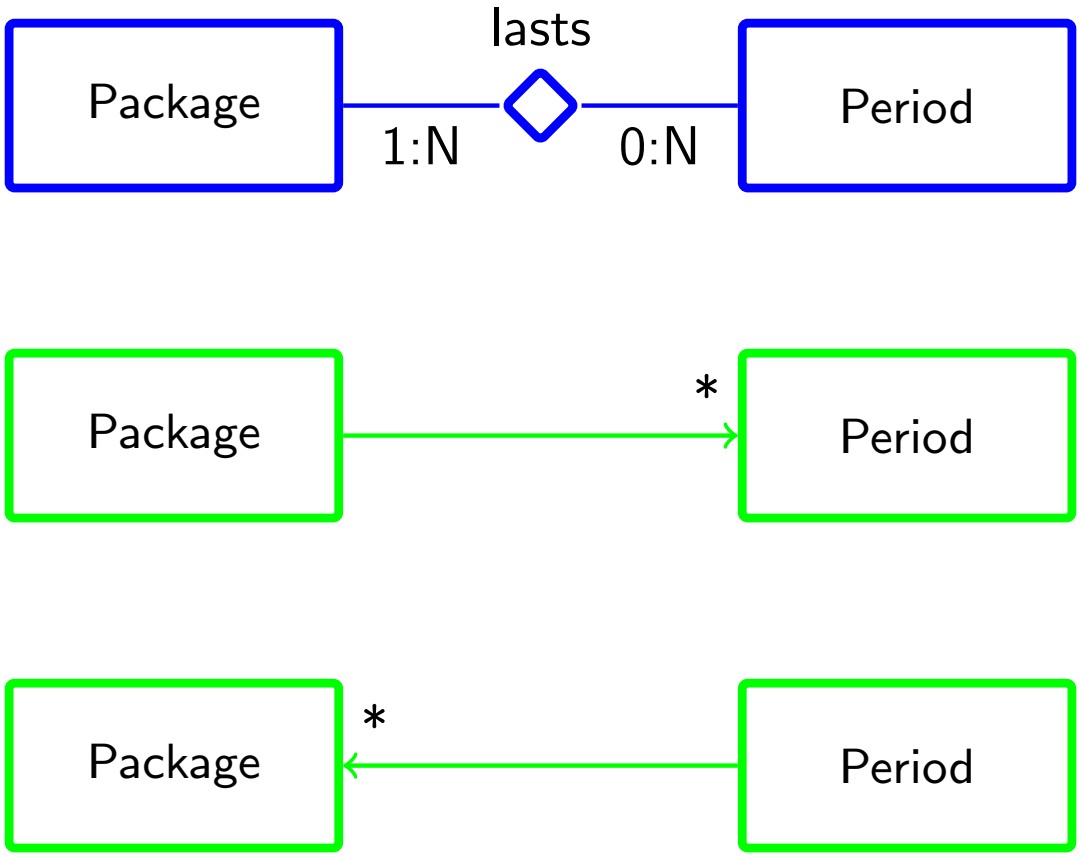
@ManyToMany

- default fetch LAZY
- cascade PERSIST, MERGE

Optionals → Package

/

Package-Period



Package → Period

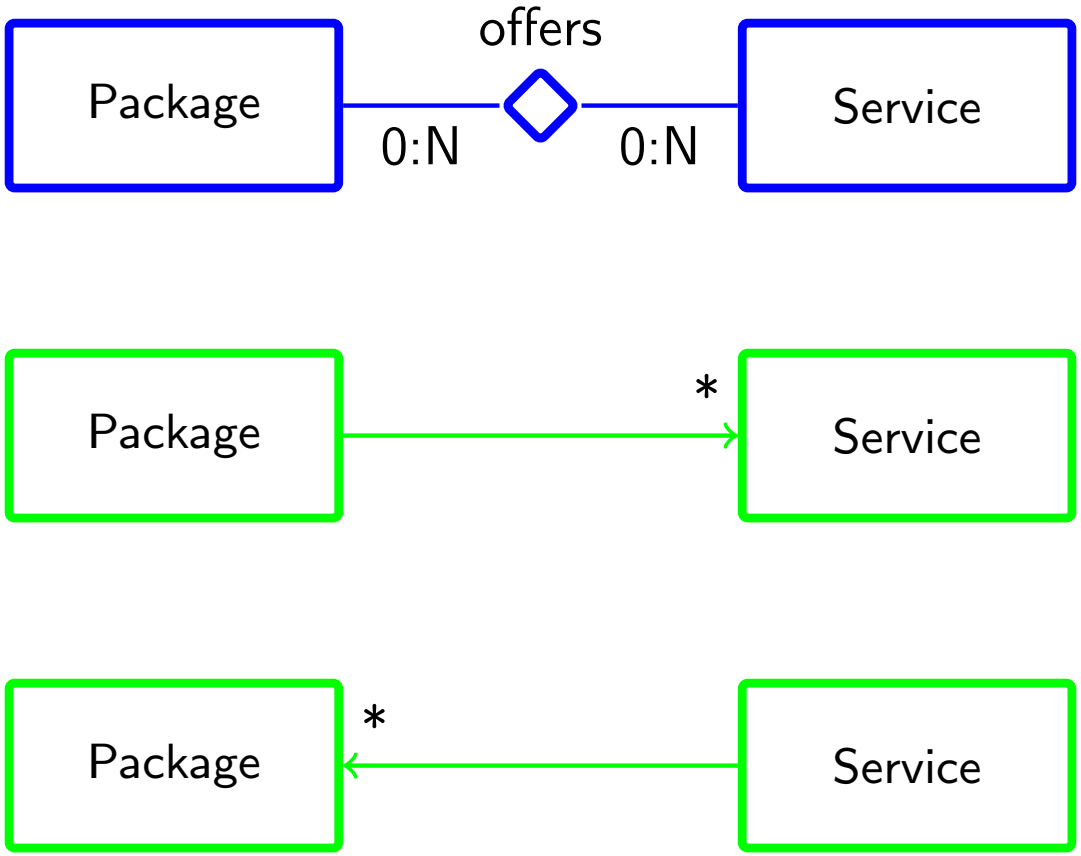
@ManyToMany

- default fetch LAZY
- cascade PERSIST, MERGE

Period → Package

/

Package-Service



Package → Service

@ManyToMany

- default fetch LAZY
- cascade PERSIST, MERGE

Service → Package

/

Entities code

Luca Genoni - 10520445

TblPackage


```java
@Entity
@Table(name = "tblPackages")
@NamedQuery(name = "TblPackage.findAll", query = "SELECT t FROM TblPackage t")
@NamedQuery(name = "TblPackage.findByName", query = "SELECT t FROM TblPackage t WHERE t.name = ?1")
public class TblPackage implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Packages;

 @Column(nullable = false, length = 255)
 private String name;

 @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
 @JoinTable(name = "tblPackageIncludesOptional", joinColumns = {
   @JoinColumn(name = "FK_Packages", nullable = false) }, inverseJoinColumns = {
     @JoinColumn(name = "FK_Optionals", nullable = false) })
 private List<TblOptional> tblOptionals;

 @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
 @JoinTable(name = "tblPackageLastsPeriod", joinColumns = {
   @JoinColumn(name = "FK_Packages", nullable = false) }, inverseJoinColumns = {
     @JoinColumn(name = "FK_Periods", nullable = false) })
 private List<TblPeriod> tblPeriods;

 @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
 @JoinTable(name = "tblPackageOffersService", joinColumns = {
   @JoinColumn(name = "FK_Packages", nullable = false) }, inverseJoinColumns = {
     @JoinColumn(name = "FK_Services", nullable = false) })
 private List<TblService> tblServices;
```

TblService


```java
@Entity
@Table(name = "tblServices")
@NamedQuery(name = "TblService.findAll", query = "SELECT t FROM TblService t")
public class TblService implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Services;

 private double feeGb;

 private double feeMin;

 private double feeSms;

 private int gb;

 private int min;

 @Column(nullable = false, length = 255)
 private String name;

 private int sms;

 @Column(nullable = false, length = 40)
 private String type;
```

TblPeriod


```java
@Entity
@Table(name = "tblPeriods")
@NamedQuery(name = "TblPeriod.findAll", query = "SELECT t FROM TblPeriod t")
public class TblPeriod implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Periods;

 @Column(nullable = false)
 private double fee;

 @Column(nullable = false)
 private int months;
```

TblOptional


```java
@Entity
@Table(name = "tblOptionals")
@NamedQuery(name = "TblOptional.findAll", query = "SELECT t FROM TblOptional t")
@NamedQuery(name = "TblOptional.findByName", query = "SELECT t FROM TblOptional t WHERE t.name = ?1")
public class TblOptional implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Optionals;

 @Column(nullable = false)
 private double fee;

 @Column(nullable = false, length = 255)
 private String name;
```

TblBill


```java
@Entity
@Table(name = "tblBills")
@NamedQuery(name = "TblBill.findAll", query = "SELECT t FROM TblBill t")
@NamedQuery(name = "TblBill.findAllRejectedOfRejected", query = "SELECT t FROM TblBill t WHERE t.result=0 and t.tblOrder.isValid
        =0")
public class TblBill implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Bills;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date billDate;

 @Column(nullable = false)
 private byte result;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Orders", nullable = false)
 private TblOrder tblOrder;
```

TblUser


```java
@Entity
@Table(name = "tblUsers")
@NamedQuery(name = "TblUser.findAll", query = "SELECT t FROM TblUser t")
@NamedQuery(name = "TblUser.checkSignIn", query = "SELECT t FROM TblUser t WHERE t.username = ?1 and t.password = ?2")
@NamedQuery(name = "TblUser.findAllInsolvent", query = "SELECT t FROM TblUser t WHERE t.isInsolvent = 1")
@NamedQuery(name = "TblUser.findByUsername", query = "SELECT t FROM TblUser t WHERE t.username = ?1")
public class TblUser implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Users;

 @Column(nullable = false, length = 255)
 private String email;

 @Column(nullable = false)
 private byte isInsolvent;

 @Column(nullable = false, length = 255)
 private String password;

 @Column(nullable = false)
 private byte role;

 @Column(nullable = false, length = 255)
 private String username;
```

TblSchedule


```java
@Entity
@Table(name = "tblSchedules")
@NamedQuery(name = "TblSchedule.findAll", query = "SELECT t FROM TblSchedule t")
public class TblSchedule implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Schedules;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date activeDate;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date deactiveDate;

 @OneToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Orders", nullable = false)
 private TblOrder tblOrder;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Users", nullable = false)
 private TblUser tblUser;
```

TblAlert


```java
@Entity
@Table(name = "tblAlerts")
@NamedQuery(name = "TblAlert.findAll", query = "SELECT t FROM TblAlert t")
public class TblAlert implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Alerts;

 @Column(nullable = false)
 private double amount;

 @Column(nullable = false, length = 255)
 private String email;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date lastReject;

 @Column(nullable = false, length = 255)
 private String username;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Users", nullable = false)
 private TblUser tblUser;
```

TblOrder

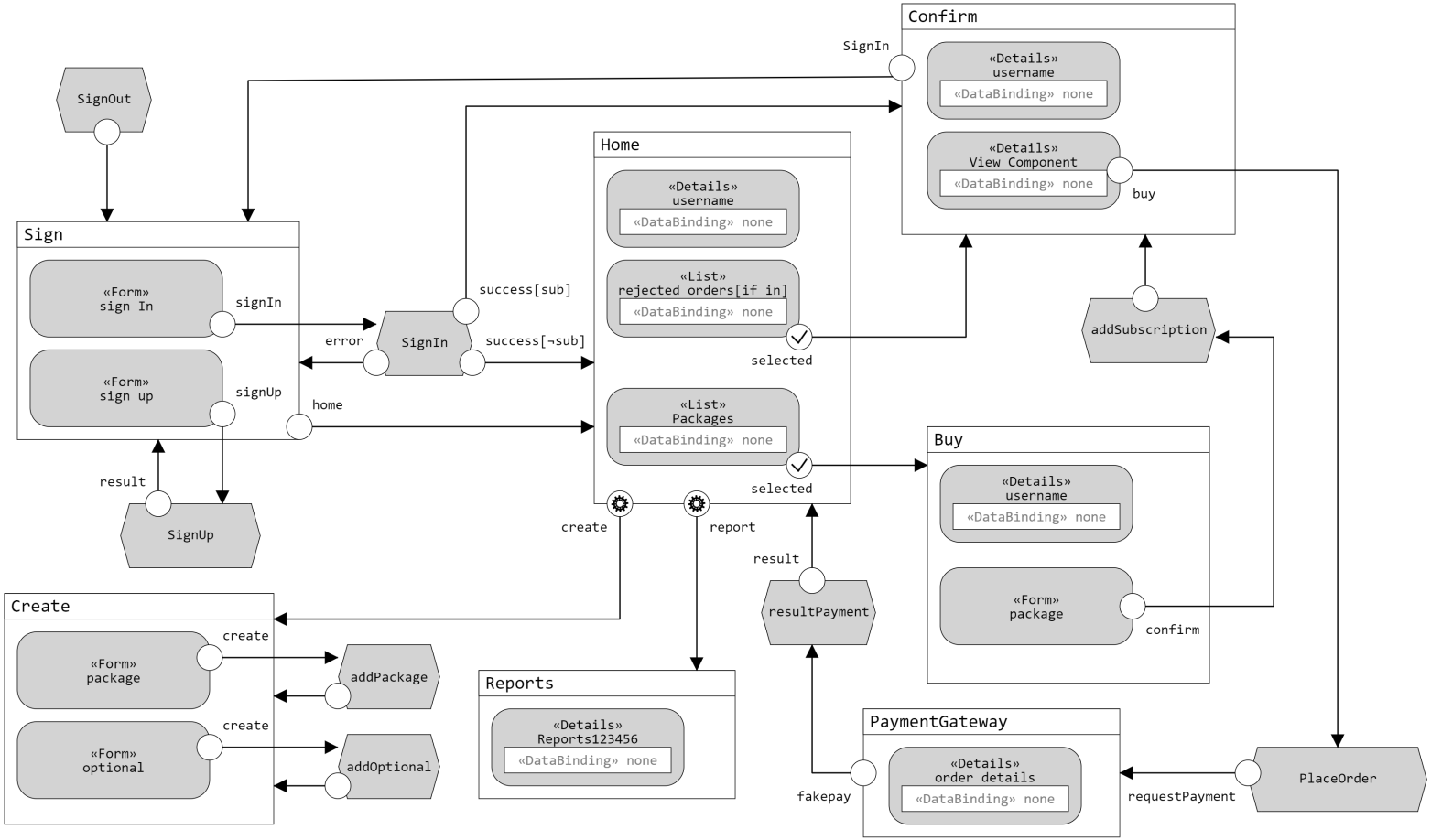

```java
@Entity
@Table(name = "tblOrders")
@NamedQuery(name = "TblOrder.findAll", query = "SELECT t
        FROM TblOrder t")
@NamedQuery(name = "TblOrder.findAllRejected", query = "
        SELECT t FROM TblOrder t WHERE t.isValid=0")
@NamedQuery(name = "TblOrder.findAllRejectedOfUser", query =
         "SELECT t FROM TblOrder t WHERE t.isValid=0 and t.
        tblUser = ?1")
public class TblOrder implements Serializable {
 private static final long serialVersionUID = 1L;

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(unique = true, nullable = false)
 private int PK_Orders;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date creationDate;

 private byte isValid;

 @Column(nullable = false)
 private double price;

 @Temporal(TemporalType.TIMESTAMP)
 @Column(nullable = false)
 private Date startDate;
```

```java
 @OneToMany(mappedBy = "tblOrder", fetch = FetchType.LAZY,
        cascade = CascadeType.ALL, orphanRemoval = true)
 private List<TblBill> tblBills;

 @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.
        MERGE })
 @JoinTable(name = "tblOrderIncludesOptional", joinColumns =
         {
   @JoinColumn(name = "FK_Orders", nullable = false) },
        inverseJoinColumns = {
     @JoinColumn(name = "FK_Optionals", nullable = false) })
 private List<TblOptional> tblOptionals;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Packages", nullable = false)
 private TblPackage tblPackage;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Periods", nullable = false)
 private TblPeriod tblPeriod;

 @ManyToOne(fetch = FetchType.LAZY)
 @JoinColumn(name = "FK_Users", nullable = false)
 private TblUser tblUser;
```

Interaction diagrams or functional analysis of the specifications

Luca Genoni - 10520445

POLITECNICO MILANO 1863


Confirm
SignIn
«Details»
username
«DataBinding» none
«Details»
View Component
«DataBinding» none
buy
SignOut
Home
«Details»
username
«DataBinding» none
«List»
rejected orders[if in]
«DataBinding» none
selected
addSubscription
Sign
«Form»
sign In
signIn
success[sub]
error
SignIn
success[¬sub]
«Form»
sign up
signUp
home
«List»
Packages
«DataBinding» none
selected
result
SignUp
create
report
Buy
«Details»
username
«DataBinding» none
result
resultPayment
«Form»
package
confirm
Create
«Form»
package
create
addPackage
«Form»
optional
create
addOptional
Reports
«Details»
Reports123456
«DataBinding» none
PaymentGateway
«Details»
order details
«DataBinding» none
fakepay
requestPayment
PlaceOrder

List of components

Model objects

- Model objects (Entities - Beans)
  - TblAlert
  - TblBill
  - TblOptional
  - TblOrder
  - TblPackage
  - TblPeriod
  - TblSchedule
  - TblService
  - TblUser
  - Vw_Report134
  - Vw_Report2
  - Vw_Report2PK
  - Vw_Report6Best

Data access objects

Data access objects (Business Components - EJBs)

- UserService
  - createUser(String username, String email, String password, int role)
  - checkCredentials(String usrn, String pwd)
  - checkIfEmployee(int idUser)
  - findUser(int idUser)
  - findAllRejectedOrdersOfUser(int idUser)
- OrderService
  - createSubscription(Date startDate, int idperiod, int idsrvpkg, List<Integer> idopts)
  - createOrder(TblUser u, TblOrder o)
  - findRejectOrder(int ido,int idUser)
  - createBill(int ido, byte result)
- ProductService
  - createSrvpackage(String name, List<Integer> idservices, List<Integer> idperiods, List<Integer> idoptionals, int idUser)
  - createOptional(String name, double fee, int idUser)
  - findSrvpackage(int id)
  - findAllPackages()
  - findAllServices()
  - findAllPeriods()
  - findAllOptionals()
- ReportService
  - findAllVw_Report134()
  - findAllVw_Report2()
  - findAllInsolventTblUser()
  - findAllRejectedTblOrder()
  - findAllTblAlert()
  - findAllVw_Report6Best()

Controllers & Templates

Controller navigation (servlet)

- GoToSignPage
- GoToHomePage
- GoToBuyPage
- GoToConfirmPage
- GoToPaymentPage
- GoToEmpCreate
- GoToEmpReports

Controller action (servlet)

- ActionSignIn
- ActionSignUp
- ActionSignOut
- ActionAddEmpPackage
- ActionAddEmpOptional
- ActionAddSubscription
- ActionPlaceOrder
- ActionResultPayment

Templates (Views)

- Sign
- Home
- Buy
- Confirm
- Payment
- EmpCreate
- EmpReports