University Security Control

Case study of Database Systems

a.a 2017/2018

Student: Francesco Giannico

Badge number: 678454

Computer Science

Index

Introduction

Realization of an university control system that aims to monitoring and allowing the entrance into a specific building.
A person that works or study for the University owns a pidcard, automatically issued after the recording his data into the database.
By swiping the own pidcard a person can try to access into a building he was authorized for.

Before the allowing of the access,besides the authorization for that building,there are others checks that the DBMS makes on which i will discuss in the next sections.

In order to allow the user to interfaces to the database, i have implemented a Web Application.

The Web application is composed by :

- Interfaces: Built by using the JSP
- Java Servlet: they are performed by using a servlet Container Apache Tomcat.

Further more, i have implemented all the web-app by planning the JSP and Servlet according to the Model-View-Controller(MVC) pattern.
The dbms used is the Oracle DBMS.

Specifications

	University security control
1.	A University has adopted a personal identity card (PID) system to improve security and to restrict access to certain
2.	groups of people (such as students, teachers, professors, secretaries, managers etc) and at certain times and dates. A
3.	person is issued a PID card as soon as they become part of the University community (either employed or on a
4.	course of study). Each person belongs to only one group which determines what buildings they can access and
5.	which are the time restrictions. Moreover, for each building, we know the persons authorisezed to access.
6.	To enter a building, a person (each having a unique personID) must have permission which is established when
7.	their PID card is swiped through a PID card reader outside the building they wish to enter. A PID card reader is
8.	located outside the door of a building users wish to access. Permission is granted only if their access credentials are
9.	successful. Every time the card is swiped, the captured data is logged, recording the date, personID (from the
10.	PIDcard) and the PIDreaderID.

Analyze the specifications, filtering ambiguities and grouping concepts in a homogeneous way. Represent the specifications in a E-R schema. Indicate the strategy used during the conceptual design. Add possible additional contraints and business rules. *(6 marks)*

Conceptual Design

Reorganize sentences by concepts

Note: extension of the specifications

After an accurate research on this domain i have integrated the input specifications. The specifics added are highlighted in red for each following group.

General character phrases:

A University has adopted a personal identity card(PID) system to improve security and to restrict access to certain groups of people(such as students,teachers,professors,secretaries,managers etc) and a certain times and dates.

Phrases related to the persons:

A person is issued a PID card as soon as they become part of the University community.

A person can be a employed or student.

Each person belongs to only one group of person which determines what buildings they can access and which are the time restrictions.

A person can access to the building by swiping its PIDCard on a Pidreader outside a building.

A person is characterized by a fiscalcode, name,surname and a birthdate. A person is identified by the fiscalcode.

An employee owns an end date of its actual contract, the profession and the contract number.

A student owns the badge number, the actual academic year and the status that can be active or not active (for example because the student career is ended ecc..)

If a person is deleted must be kept in the database since has to be possibile to check the owner of the pidcard and it pidcard became deactive.

The person must be adult.

<u>Phrases related to the pidcard:</u>

The pidcard must be released after the new person storing.

It own a status that can be Active or deactive.

It can be deactive for two reason:

1) The end work date of the employee's contract has expired.
2) The student's status is deactive.

it owns a release date (when the person is stored in the system).

It can be a special card and can be released only for the employee to allow the access at any hour in a building (for example the security man).

The pidcard allows by default the access to all the buildings associated to the department for which the person refers to. A person can be allowed to access in the other buildings besides the defaults.

Phrases related to the department:

It is identified by an unique name.

Each department is composed by or more buildings.

It has an opening hour and a closing hour.

It can have a extraordinary closing days.

Phrases related to the extraordinary closing days:

Each department can be closed in a date, from an hour to another one or for all day, consequently all the buildings must be closed.

It is characterized by the start and end hour (and minutes) and the date.

Phrases related to the building

Moreover,for each building, we know the persons authorized to access.

Outside each building there is a pidreader.

A pidreader is identified by an unique PIDreaderID.

A pidreader allows to access into the associated building after swiping a PIDCard.

Only for the authorized person and on a restrict time date and hour it allows the accessing.

Each building must refers to a specific department.

Each building has a opening hour and a closing hour.

The opening hour must be after or equal of the opening hour of the department.

The closing hour must be before or equal of the closing hour of the department.

Each building has an address, a name and a code.

It can have a extraordinary closing days, but differently from the departments, this extraordinary date are not propagated to the associated department and the other buildings.

Phrases related to validation

Every time the card is swiped, will be recorded the date, the hour and the minutes of that time, the PIDCard that has been swiped and the PIDreaderID.

An entrance is allowed when:

1) The pidcard is recognized by the system and is active.
2) The pidcard is authorized to enter in the building.
3) If the building isn't a special card,
    a. the building must be open at the time of the swiping.
    b. The building must be open on that day (it can be an extraordinary closing day).
        i. If it is an extraordinary day, must be checked if the actual hour is after the end hour of the extraordinary closing day.

Phrases related to entrance

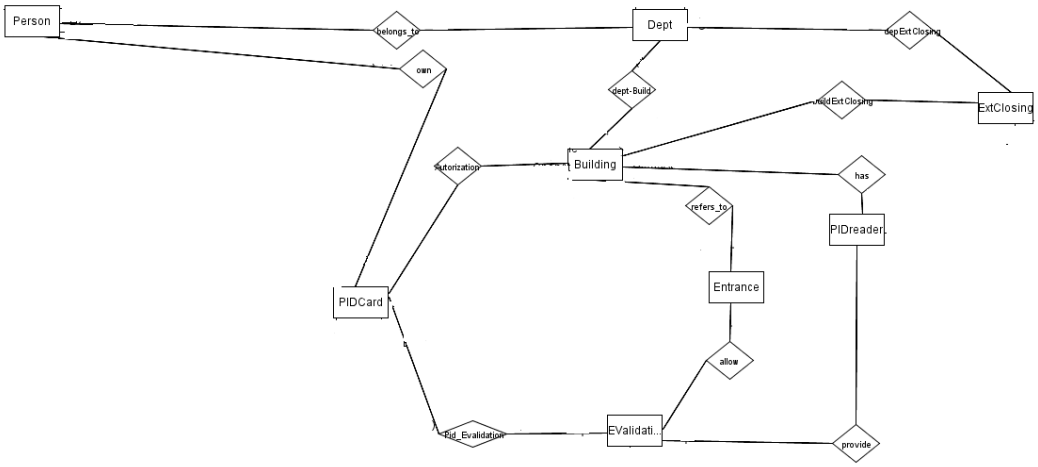
Every time the permission to access is granted will be recorded the date, the hour and the minutes of that time, the validation which it refers to.

Terms Glossary

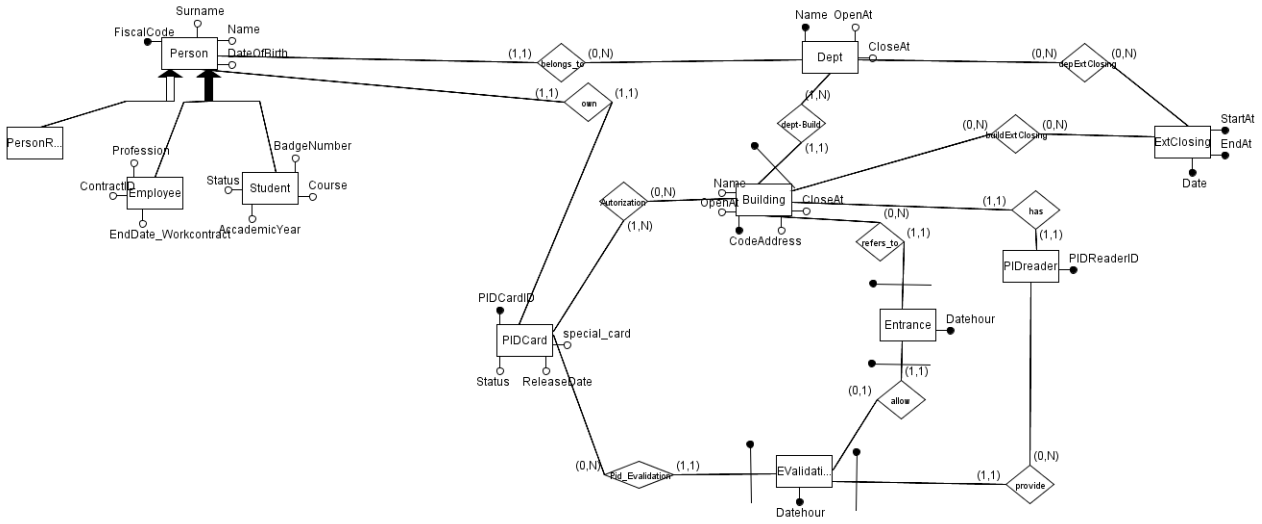
Term	Description	Link
PIDCard	Card issued for each Person which allows to access to a building.	Building,Evalidation,Person
PIDReader	Reader of the Cards, located outside the door of a building.	Building,Evalidation
Person	Persons in the University community: they can be Employees or Students.	Dept,pidcard
Building	Physical place in which the Persons works or study.	Dept,Entrance,pidreader,pidcard,Extclosing
Evalidation	Contains all the access attemps made by a pidcard into a pidreader	Entrance,pidreader,pidcard
Entrance	Contrains all the attemps of evalidation granted	Building,evalidation
Extclosing	The extraordinary closing of the building or the department	Dept,building
Dept	The departments of the university	Person,extclosing,building

Skeleton schema

After the analysis this i have built this skeleton schema:



Final conceptual schema



Unexpressible constraints:

- If a person is deleted must be kept in the database since has to be possibile to check the owner of the pidcard and it pidcard became deactive.
- The birthdate must be 18 years ago.
- The end date of the work contract must be after or equal of today.
- After inserting a person a pidcard must be generated and associated to.
- The relase date of the pidcard must be taken from the time on which the person is added to the database.
- The badge number must be unique.
- The building's opening hour must be after or equal of the opening hour of the department.
- The building's closing hour must be before or equal of the closing hour of the department.
- The department closing hour must be after the opening hour.
- The building closing hour must be after the opening hour.
- Each department can be closed in a date, from an hour to another one or for all day, consequently all the buildings must be closed.
- An entrance is allowed when:
- The pidcard is recognized by the system and is active.
- The pidcard is authorized to enter in the building.
- If the building isn't a special card,
    - the building must be open at the time of the swiping.
    - The building must be open on that day (it can be an extraordinary closing day).
        - If it is an extraordinary day, must be checked if the actual hour is after the end hour of the extraordinary closing day.

Logical Design

The logical model chosen for this project is object-relational.

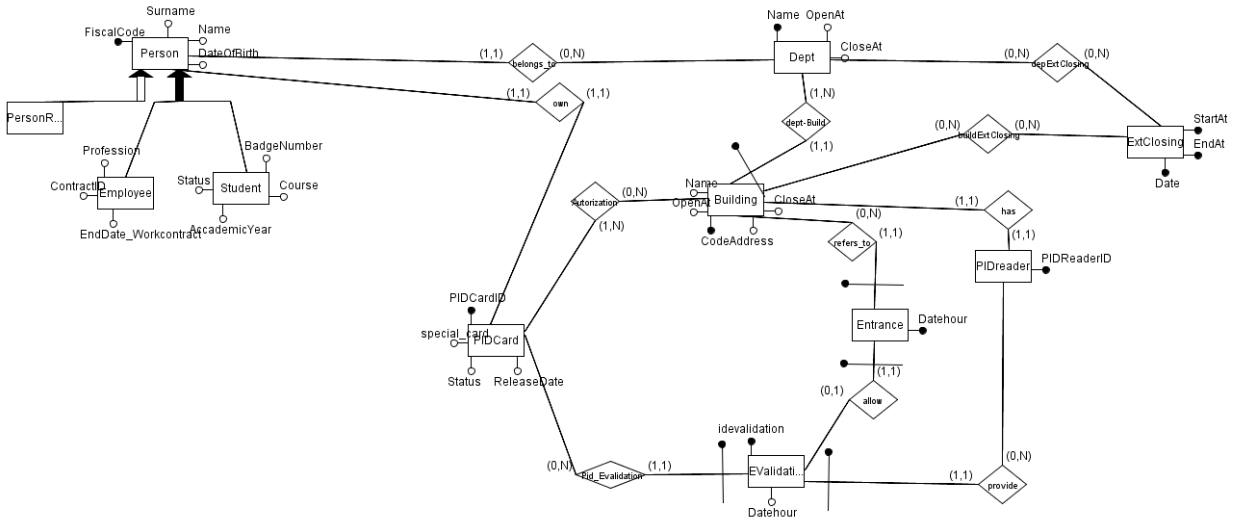
The logical design of this database was done by keeping in mind the following operations:

Operations Table

Number	Operation	Type	Frequency
1	Insert person	Interactive	50 times/week
2	Delete person	Interactive	50 times/week
3	Validate entrance	Interactive	300 times/minute (3 024 000 times/week)
4	Register entrance	interactive	50 times/minute (504 000 times/week)
5	Print the access statistics: per day and per building	batch	10 times/week
6	Print the access statistics: per hour and per building	batch	10 times/ week

Restructuring the conceptual model and choosing of the primary identifiers:

I have added the “idevalidation” since i twill be useful in the code.



Volumes Table

Concept	Type	Volume
Person	Entity	100 000
persondeleted	Entity	50*52(settimane)*5(anni)= 13 000
Employee	Entity	100 000*40%= 40 000
Student	Entity	100 000*60%= 60 000
PIDCard	Entity	100 000
ExtClosing	Entity	450/3= 150
Evalidation	Entity	5*52(weeks per year) * 3 024 000 = 786 240 000
Entrance	Entity	5*52(weeks per year) * 504 000= 131 040 000
PIDReader	Entity	60
Dept	Entity	30
Building	Entity	30*2= 60
Belongs_to	Relationship	100 000
Own	Relationship	100 000
deptExtClosing	Relationship	15(some dept)*6*5 = 450
buildExtClosing	Relationship	20 (some builings)*7*5 =700
Pid_EValidation	Relationship	786 240 000
Allow	Relationship	131 000 000
Refers_to	Relationship	131 000 000
DeptBuild	Relationship	60
Provide	Relationship	786 240 000
Has	Relationship	60
buildingallowed	Relationships	100 000 * 2 = 200 000

Assumptions made:

- 5 years of activity;
- Persons are for the 40% Employees while the remaining 60% are Students.
- Each department has 2 builings.
- Some departments have 6 extraordinary closing days in an year.
- Each extraordinary closing day refers to 3 departments.
- Some buildings have 7 extraordinary closing days in an year.
- Each extraordinary closing day refers to 4 buildings.
- Each PIDCard allows to access into 2 buildings.
- Each Building has 200 000 entrance.

Access Table

For each operation it has defined the access table.

The total cost access/week has been computed by considering the frequency of each operation and weighing writing operations as 2 accesses.

Operation 1(insert person)			
Concept	Concept Type	Number	Access Type
Person	E	1	R

Person	E	1	W

Total cost: ((1*2)+1) * 50 = 150 access/week

Operation 2(delete person)			
Concept	Concept Type	Number	Access Type
Person	E	1	R
Persondeleted	E	1	W

Total cost: ((1*2) +1)* 50 = 150 access/week

Operation 3(validate entrance)			
Concept	Concept Type	Number	Access Type
PIDCard	R	1	R
PidReader	E	1	R
EValidation	E	1	R
EValidation	E	1	W
buildingallowed	R	1	W
Provide	R	1	W
Building	E	3(2 times for authorization and 1 for PIDReader)	R
buildExtClosing	R	7	R

Total cost: ((3*2)+ 13) * 3024000 = 57 456 000 access/week

Operation 4(register entrance)			
Concept	Concept Type	Number	Access Type
EValidation	E	1	R
PIDReader	E	1	R
Building	E	1	R
Entrance	E	1	W
Allow	R	1	W
Refers_to	R	1	W

Total cost: ((3*2) + 3)* 504 000 = 4 536 000 times/week

The Operation 5 and 6 are batch, but i have also decided to check the operations'cost.

Operation 5			
Concept	Concept Type	Number	Access Type
Entrance	E	131 000 000	R
Building	E	200 000	R

Total cost: (131 000 000 + 200 000)*10= 1 312 000 000 access/week

Operation 6			
Concept	Concept Type	Number	Access Type
Entrance	E	131 000 000	R
Building	E	200 000	R

Total cost: (131 000 000 + 200 000)*10= 1 312 000 000 access/week

Redudancy Analysis

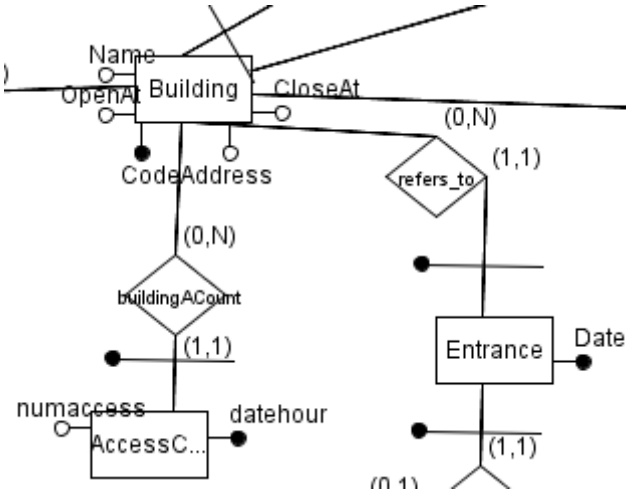
Considering the operations 5,6 and 4 can be convenient to have the table accessCount.

The idea was that in entrance can be more tuples with the same date but different hours and minuts.

For example we can have: '27/10/2015 10:50' and '27/10/2015 10:30' by the same pidcard and the same pidreader.

For this, i thought to create a new table named 'accessCount' in which there will be the tuples of entrance but without considering the minutes.

Since it is rare that a person access into a building more than two times in the same hour of the same day and since the operations 5 and 6 are required only 10 times at week, i haven't introduced this redundancy.



Volumes table for the new redundancies

Concept	Type	Volume
AccessCount	E	130 000 000 / 200 = 655 000
buildingACount	R	655 000

Asumptions made:

Each AccessCount refers to a 200 tuples in entrance.

Operation 4(register entrance, without redundancy)			
Concept	Concept Type	Number	Access Type
EValidation	E	1	R
PIDReader	E	1	R
Building	E	1	R
Entrance	E	1	W
Allow	R	1	W
Refers_to	R	1	W

Total cost: ((3*2) + 3)* 504 000 = 4 536 000 times/week

Operation 4(with redundancy)			
Concept	Concept Type	Number	Access Type
EValidation	E	1	R
PIDReader	E	1	R

Building	E	1	R
Entrance	E	1	W
Allow	R	1	W
Refers_to	R	1	W
AccessCount	E	1	R
AccessCount	E	1	W

Total cost: ((4*2) + 4)* 504 000 = 6 048 000 times/week

Operation 5(with redundancy)			
Concept	Concept Type	Number	Access Type
Building	E	60	R
AccessCount(contains 131000000/200 tuples)	E	655 000	R

Total cost: (60 + 655 000)*10= 655 060 access/week

Operation 5(without redundancy)			
Concept	Concept Type	Number	Access Type
Entrance	E	131 000 000	R
Building	E	200 000	R

Total cost: (131 000 000 + 200 000)*10= 1 312 000 000 access/week

Operation 6(without redundancy)			
Concept	Concept Type	Number	Access Type
Entrance	E	131 000 000	R
Building	E	200 000	R

Total cost: (131 000 000 + 200 000)*10= 1 312 000 000 access/week

Operation 6(with redundancy)			
Concept	Concept	Concept	Concept
Building	E	60	R
AccessCount(contains 131000000/200 tuples)	E	655 000	R

Total cost: (60 + 655 000)*10= 655 060 access/week

Accesses number without redundancy = (1 312 000*2)

Logical schema

Since it was develope by using Oracle syntax,see the section “Database implementation”.

Physical Design

The physical design has been made up by keeping in mind the operations.
In the following there are some decision that i have taken:

Operation 2:

Delete a person by finding her by name,surname,fiscal code or birthdate.
The optimization can be made by defining

B+Tree index on the attribute "name" of the table Person.
B+Tree index on the attribute "surname" of the table Person.
B+Tree index on the attribute "birthdate" of the table Person.
B+Tree index on the attribute "Fiscalcode" of the table Person.

Operation 3:

(register entrance)
The optimization can be made by defining a

B+Tree index on the attribute "datehour" of the table evalidation;

This can be useful for the searching of a validation in a specific date and hour.

Operation 4:

(register entrance)
The optimization can be made by defining a

B+Tree index on the attribute "datehour" of the table entrance;

This can be useful for the searching of a validation in a specific date and hour.

Database Implementation

Types

```
create or replace type hourminutesty as object(
            hours number,
            minutes number
);
create or replace type deptty as object(
          namedept varchar2(30),
          openat  ref hourminutesty,
          closeat ref hourminutesty
);
create or replace type buildingty as object(
         code integer,
         address varchar2(30),
         nameb varchar2(30),
         openat ref hourminutesty,
         closeat ref hourminutesty,
          dept ref deptty,
          constructor function buildingty(self in out nocopy buildingty,nameb varchar2,address varchar2,openat    ref
         hourminutesty,closeat ref hourminutesty,dept ref deptty) return self as result
);
create or replace type body buildingty is
 constructor function buildingty(self in out nocopy buildingty,nameb varchar2,address varchar2,openat ref
hourminutesty,closeat ref hourminutesty,dept ref deptty) return self as result is
         begin
                  self.nameb:=nameb;
                  self.address:=address;
                   self.openat:=openat;
                    self.closeat :=closeat;
                    self.dept:=dept;
                  return;
```

```
        end;
end;


create or replace type personty as object(
        namep varchar2(30),
        surname varchar2(30),
        fiscalcode char(16),
        dateofbirth date,
        dept ref deptty,
        constructor function personty(self in out nocopy personty, namep varchar2, surname varchar2, fiscalcode
        char,dateofbirth date,dept ref deptty) return self as result
) not final;
create or replace type body personty is
        constructor function personty(self in out nocopy personty, namep varchar2, surname varchar2, fiscalcode
        char,dateofbirth date,dept ref deptty) return self as result is
        begin
                self.namep := namep;
                self.surname := surname;
                self.fiscalcode := fiscalcode;
                  self.dateofbirth := dateofbirth;
                  self.dept:=dept;
                return;
        end;
end;
create or replace type employeety under personty(
        contractID varchar2(15),
        profession varchar2(30),
        enddate_workcontract date,
        constructor function employeety(self in out nocopy employeety,namep varchar2, surname varchar2,
        fiscalcode char,dateofbirth timestamp,dept ref deptty, contractID varchar2,profession varchar2,
        enddate_workcontract date) return self as result
);
create or replace type body employeety is
```

```
        constructor function employeety(self in out nocopy employeety,namep varchar2, surname varchar2,
        fiscalcode char,dateofbirth timestamp,dept ref deptty, contractID varchar2,profession varchar2,
        enddate_workcontract date) return self as result is

        begin

                self.namep := namep;

                self.surname := surname;

                self.fiscalcode := fiscalcode;

                self.dateofbirth := dateofbirth;

                  self.dept:=dept;

                self.contractID:=contractID;

                   self.profession:=profession;

                   self.enddate_workcontract:=enddate_workcontract;

                return;

        end;

end;

create or replace type studentty under personty(

          badgenumber char(6),

          status char(1),

          course varchar2(30),

          academicyear varchar2(10),

          constructor function studentty(self in out nocopy studentty,namep varchar2, surname varchar2, fiscalcode
        char,dateofbirth timestamp,dept ref deptty, badgenumber char, course varchar2, academicyear varchar2)
        return self as result

);

create or replace type body studentty is

        constructor function studentty(self in out nocopy studentty,namep varchar2, surname varchar2, fiscalcode
        char,dateofbirth timestamp,dept ref deptty,badgenumber char, course varchar2, academicyear varchar2)
        return self as result is

        begin

                self.namep := namep;

                self.surname := surname;

                self.fiscalcode := fiscalcode;

                 self.dateofbirth := dateofbirth;

                 self.dept:=dept;

                self.badgenumber := badgenumber;
```

```
                self.course := course;
                self.academicyear := academicyear;
                 --the following is setted by default
                self.status :='A';
                return;
        end;
end;
create or replace type buildingallowedty as object(
         building ref buildingty
);
create or replace type ref_buildingallowednt as table of buildingallowedty;
create or replace type pidcardty as object(
         pidcardID integer,
         status char(1),
         releasedate date,
         person char(16),
         buildingallowed ref_buildingallowednt,
         special_card char(1),
         constructor function pidcardty(self in out nocopy pidcardty,person char,buildingallowed
        ref_buildingallowednt) return self as result
);
create or replace type body pidcardty is
        constructor function pidcardty(self in out nocopy pidcardty,person char,buildingallowed
        ref_buildingallowednt) return self as result is
        begin
                self.person := person;
                self.releasedate := sysdate;
                self.status := 'A';
                self.special_card:='F';
                self.buildingallowed:=buildingallowed;
                return;
        end;
end;
```

```
create or replace type pidreaderty as object(
        pidreaderID integer,
        building ref buildingty,
        constructor function pidreaderty(self in out nocopy pidreaderty,building ref buildingty) return self as result
);
create or replace type body pidreaderty is
        constructor function pidreaderty(self in out nocopy pidreaderty,building ref buildingty) return self as result is
        begin
                self.building:=building;
                return;
        end;
end;
create or replace type evalidationty as object(
        datehour timestamp,
        pidcard ref pidcardty,
        pidreader ref pidreaderty,
        idevalidation integer,
        constructor function evalidationty(self in out nocopy evalidationty,timehour timestamp,pidcard ref
        pidcardty, pidreader ref pidreaderty) return self as result,
        constructor function evalidationty(self in out nocopy evalidationty,pidcard ref pidcardty, pidreader ref
        pidreaderty) return self as result
);
create or replace type body evalidationty is
    constructor function evalidationty(self in out nocopy evalidationty, timehour timestamp,pidcard ref pidcardty,
    pidreader ref pidreaderty) return self as result is
    begin
        self.pidcard := pidcard;
        self.pidreader :=pidreader;
        self.datehour:=timehour;
        return;
    end;
        constructor function evalidationty(self in out nocopy  evalidationty,pidcard ref pidcardty, pidreader ref
        pidreaderty) return self as result is
    begin
```

```
            self.pidcard := pidcard;
            self.pidreader :=pidreader;
            self.datehour := localtimestamp;
            return;
        end;
end;
create or replace type entrancety as object(
            evalidation integer,
            building ref buildingty,
            datehour timestamp
);
create or replace type extclosingty as object(
            startat ref hourminutesty,
            endat ref hourminutesty,
            datetime date
);
create or replace type buildextclosingty as object (
            building ref buildingty,
            extclosing ref extclosingty
);
create or replace type deptextclosingty as object (
            deptextclosing ref deptty,
            extclosing ref extclosingty
);
```

Tables

```
create table error_entrance (
            message varchar2(100)
);
create table hourminutes of hourminutesty(
  primary key(hours,minutes),
  CHECK (hours>=0 and hours<=23),
  CHECK(minutes>=0 and  minutes<=59)
```

```
);
create table person of personty(
        primary key(fiscalcode)
);
create table dept of deptty(
        primary key(namedept)
);
create table extclosing of extclosingty;
create table deptextclosing of deptextclosingty;
create table buildextclosing of buildextclosingty;
create table building of buildingty (
        primary key(code)
);
create table pidreader of pidreaderty(
        primary key(pidreaderID)
);
create table entrance of entrancety;
create table evalidation of evalidationty;
create table pidcard of pidcardty(
 primary key(pidcardID)
) nested table buildingallowed store as buildingallowedNT;
create table personremoved of personty;
```

Sequences

```
create sequence pidcard_seq start with 1;
create sequence pidreader_seq start with 1;
create sequence building_seq start with 1;
create sequence evalidation_seq start with 1;
```

Operations

Note: The underlined are inserted by the user.

Operation 1

Student

Insert into person values (studentty('name','surname','fiscalcode',

to_date('dateofbirth'),(select ref(d) from dept d where namedept='namedept'),

'badgenumber','course',academicyear);

Employee

Insert into person values (employeetty('name','surname','fiscalcode',

to_date('dateofbirth', 'yyyy-mm-dd'),(select ref(d) from dept d where namedept='namedept'),

'contractid','profession',to_date('enddateofworkcontract','yyyy-mm-dd');

Operation 2

Delete from person where namep='name';

Delete from person where namep='name' and surname='surname';

Delete from person where surname='surname';

Delete from person where fiscalcode='fiscalcode';

Operation 3

Evalidation

INSERT INTO evalidation values (evalidationty(

(SELECT ref(p) from pidcard p where pidcardid='pidcardid '),

(SELECT ref(d) from pidreader d where pidreaderid=' pidreaderid ')));

Operation 4

Register an entrance

insert into entrance values (entrancety('idevalidation',(select ref(b) from building b where building b.code='codebuild'),localtimestamp));

Operation 5

Building statistics per day

SELECT count(*) as num_access from entrance

where trunc(to_timestamp('yyyy-mm-dd, 'yyyy-mm-dd'),'j')=trunc(datehour)

and deref(building).code='code');

Operation 6

Building statistics per hour

```
SELECT count(*) as num_access from entrance

        where extract(hour from (to_timestamp('yyyy-mm-dd hh:mm', 'yyyy-mm-dd
hh24.mi.ss.?')))>=extract(hour from (datehour))

                and extract(hour from (to_timestamp('yyyy-mm-dd hh:mm', 'yyyy-mm-dd
        hh24.mi.ss.?')))>=extract(hour from (datehour))"

                and deref(building).code='code');
```

Procedures

```
create or replace procedure alter_all_triggers(status varchar2) is
  cursor c_tr is (select 'alter trigger ' || trigger_name as stmnt from user_triggers);
begin
  if status not in ('ENABLE', 'enable', 'DISABLE', 'disable') then
    dbms_output.put_line ('ONLY ''ENABLE DISABLE'' ACCEPTED AS PARAMETERS');
  end if;
  for tr in c_tr loop
    --disable all triggers
    execute immediate tr.stmnt || ' ' || status;
  end loop;
  if status in ('DISABLE', 'disable') then
  --enable some other triggers
    execute immediate 'alter trigger building_incr enable';
    execute immediate 'alter trigger pidreader_incr enable';
    execute immediate 'alter trigger pidcard_incr enable';
    execute immediate 'alter trigger evalidation_incr enable';
  end if;
end;
```

This procedure accept as values the following values:

- ‘ENABLE’ o ‘enable’ to activate all triggers;
- ‘DISABLE’ o ‘disable’ to deactivate all triggers, but not the trigger about sequences.

```
create or replace procedure populate_database as
begin
  alter_all_triggers('DISABLE');
  populate_hourminutes();
  populate_depts();
  populate_persons();
  populate_buildings();
```

```
  populate_pidreaders();
  populate_extclosings();
  populate_deptextclosings();
  populate_buildextclosings();
  populate_pidcards();
  populate_entrancesevalidations();
  alter_all_triggers('ENABLE');
end;
```

This procedure is used to call the all the others.

```
create or replace procedure populate_entrancesevalidations as
dt date;
extday integer;
rf_pidcard ref pidcardty;
rf_build ref buildingty;
rf_pidreader ref pidreaderty;
timehour timestamp;
var varchar2(50);
duplies number;
pid_readerexists number;
--counters
subiterations number;
iterations number;
num_building number;
evalid integer;
begin
  extday:=0;
  duplies:=0;
  iterations:=0;
  num_building:=0;
  pid_readerexists:=0;
  loop
  subiterations:=0;
```

```
--take a random pidcard
    select ref_pidcard into rf_pidcard
    from (select ref(p) as ref_pidcard from pidcard p order by dbms_random.value)
    where rownum=1;
     -- take a random building
       select count(t.building) into num_building --as ref_build,pidcardID
        from pidcard p,table(p.buildingallowed) t where p.pidcardID=deref(rf_pidcard).pidcardID;
        if num_building>0 then


         loop
            select ref_build into rf_build
            from (select t.building as ref_build from pidcard p,table(p.buildingallowed) t where
p.pidcardID=deref(rf_pidcard).pidcardID order by dbms_random.value)
            where rownum=1;
             --take the pidreader associated to the building
             select count(*) into pid_readerexists from pidreader p where p.building=rf_build;
              --dbms_output.put_line(pid_readerexists);
             if pid_readerexists>0 then
              select ref(p) into rf_pidreader from pidreader p where p.building=rf_build;
              -- dbms_output.put_line(pid_readerexists);
            --generate a valid date
              loop
                dt:=to_date(trunc(dbms_random.value(to_char(sysdate, 'J'), to_char(sysdate+
dbms_random.value(0,1830), 'J'))), 'J');
                select count(*) into extday from extclosing where datetime=dt;
              exit when extday=0;
              end loop;
              if extday=0 then
                  var:=dt || ' ' || round(dbms_random.value(9,19)) || ':00';
                  timehour:= TO_TIMESTAMP(var,'yyyy/mm/dd hh24:mi');
                 -- dbms_output.put_line(extract(hour from timehour));
             /*   select count(*) into duplies from evalidation where datehour=timehour and rf_pidcard=pidcard and
rf_pidreader=pidreader;
                  if duplies=0 then */
```

```
                    insert into evalidation values (evalidationty(timehour,rf_pidcard,rf_pidreader));
                  --  select ref(e) into rf_evalid from evalidation e where datehour=timehour and pidcard=rf_pidcard
and pidreader=rf_pidreader;
                    SELECT evalidation_seq.currval INTO evalid FROM DUAL;
                    insert into entrance values (entrancety(evalid,rf_build,timehour));
              /*  end if;*/
            end if;
          end if;
          subiterations:=subiterations+1;
           exit when subiterations=num_building;
           end loop;
        end if;
      iterations:=iterations+1;
     exit when iterations=10000;
    end loop;


end;
```

This procedure generates a valid entrace and then populate the validations.

```
create or replace procedure populate_pidcards as
cursor ps is select ref(p) from person p;
pers ref personty;
building ref buildingty;
duplies number;
fiscalcodet char(16);
begin
  open ps;
  loop
  duplies:=0;
  fetch ps into pers;
  select count(*) into duplies from pidcard p where deref(pers).fiscalcode=p.person;
  if duplies=0 then
     select fiscalcode into fiscalcodet from person p where deref(pers).fiscalcode=p.fiscalcode;
      insert into pidcard values (pidcardty(fiscalcodet,(cast(multiset(select ref(b) from building b where
b.dept=deref(pers).dept) as ref_buildingallowednt))));
```

```
  end if;
  duplies:=0;
  exit when ps%NOTFOUND;
  end loop;
  close ps;
end;
```

This procedure generate one pidcard for each person and sets the buildings in which they can access.

```
create or replace procedure populate_buildextclosings as
cursor bd is select ref(b) from building b;
bd_ref ref buildingty;
begin
 open bd;
 loop
 fetch bd into bd_ref;
    populate_buildextclosings_sub(bd_ref);
 exit when bd%NOTFOUND;
 end loop;
 close bd;
end;
```

this procedure populate the extraordinary day in which the building is closed.

```
create or replace procedure populate_hourminutes as
minutes number;
hours number;
begin
  hours:=-1;
  loop
  hours:=hours+1;
    minutes:=-1;
    loop
     minutes:= minutes+1;
     insert into hourminutes values (hourminutesty(hours,minutes));
     exit when minutes=59;
```

```
    end loop;
    exit when hours=23;
  end loop;
end;
```

this procedure populate a static table for utility

```
create or replace procedure populate_depts as
iterations number;
from9_0 ref hourminutesty;
to19_0 ref hourminutesty;
begin
  iterations := 0;
  select ref(r) into from9_0 from hourminutes r where hours=9 and minutes=0;
  select ref(r) into to19_0 from hourminutes r where hours=19 and minutes=0;
  loop
    insert into dept values (deptty(dbms_random.string('l',10),from9_0,to19_0));
    iterations := iterations + 1;
    exit when iterations = 30;
  end loop;
end;
```

this procedure allows to populate departments by setting random hour and minute values for closing and opening.

```
create or replace procedure populate_buildings as
iterations number;
dept ref deptty;
openat ref hourminutesty;
closeat ref hourminutesty;
begin
iterations:=0;
    loop
        select ref_dept into dept
        from (select ref(d) as ref_dept from dept d order by dbms_random.value)
        where rownum=1;
```

```
      select ref(h) into openat from hourminutes h where minutes=deref(deref(dept).openat).minutes and
hours=deref(deref(dept).openat).hours;

      select ref(h) into closeat from hourminutes h  where minutes=deref(deref(dept).closeat).minutes and
hours=deref(deref(dept).closeat).hours;


      insert into building values
(buildingty(dbms_random.string('x',10),dbms_random.string('x',10),openat,closeat,dept));

      iterations:=iterations+1;

      exit when iterations=60;

   end loop;

end;
```

This procedure generates some buildings to associate to each department.

```
create or replace procedure populate_pidreaders as

buildd ref buildingty;

duplies integer;

cursor cs is select ref(b) from building b;

begin

  duplies:=0;

  open cs;

  loop

     fetch cs into buildd;

     select count(*) into duplies from pidreader where buildd=building;

     if duplies=0 then

       insert into pidreader values (pidreaderty(buildd));

    end if;

  exit when cs%NOTFOUND;

  end loop;

  close cs;

end;
```

This procedure associate a pidreader to a building.

```
create or replace procedure populate_persons as

iteration number;
```

```
dept ref deptty;
person_added number;
begin
 iteration:=1;
 person_added:=0;
 loop
     select ref_dept into dept
     from (select ref(d) as ref_dept from dept d order by dbms_random.value)
     where rownum=1;
      person_added:=0;
         loop
         --students
           insert into person values
                (studentty(dbms_random.string('x',10),
                  dbms_random.string('x',10),dbms_random.string('x',10),
                  to_date(trunc(dbms_random.value(to_char(date '1960-01-01','J'), to_char(date '2000-01-01', 'J'))),
'J'),dept,dbms_random.string('x',6),
                  dbms_random.string('x',10),dbms_random.string('x',3)));
                  person_added:=person_added+1;
                 exit when person_added=100;
               end loop;
         --employee
          person_added:=0;
          loop
            insert into person values
                (employeety(dbms_random.string('x',10),
                  dbms_random.string('x',10),dbms_random.string('x',10),
                  to_date(trunc(dbms_random.value(to_char(date '1960-01-01','J'),to_char(date '1989-01-01', 'J'))),
'J'),dept,dbms_random.string('x',15),
                  dbms_random.string('l',10), to_date(trunc(dbms_random.value(to_char(sysdate, 'J'), to_char(sysdate+
dbms_random.value(0,1830), 'J'))), 'J')));
                  person_added:=person_added+1;
                 exit when person_added=100;
               end loop;
```

```
            iteration := iteration +1;
        exit when iteration = 100;
end loop;

end;
```

This procedure take a random department and associate to it some new persons that can be employee or student.

```
create or replace procedure populate_extclosings as
iterations number;
from9_0 ref hourminutesty;
from12_30 ref hourminutesty;
from15_45 ref hourminutesty;
to19_0 ref hourminutesty;
to12_30 ref hourminutesty;
to17_0 ref hourminutesty;
begin

  select ref(r) into from9_0 from hourminutes r where hours=9 and minutes=0;
  select ref(r) into from12_30 from hourminutes r where hours=12 and minutes=30;
  select ref(r) into from15_45 from hourminutes r where hours=15 and minutes=45;
  select ref(r) into to17_0 from hourminutes r where hours=15 and minutes=30;
  select ref(r) into to19_0 from hourminutes r where hours=19 and minutes=0;
  select ref(r) into to12_30 from hourminutes r where hours=12 and minutes=30;

  iterations:=0;
  loop
     insert into extclosing values (extclosingty(from9_0,to12_30,to_date(trunc(dbms_random.value(to_char(sysdate,
'J'), to_char(sysdate+ dbms_random.value(0,1830), 'J'))), 'J')));
    iterations:=iterations+1;
    exit when iterations=100;
  end loop;

  iterations:=0;
```

```
  loop
  insert into extclosing values (extclosingty(from12_30,to17_0,to_date(trunc(dbms_random.value(to_char(sysdate,
'J'), to_char(sysdate+ dbms_random.value(0,1830), 'J'))), 'J')));
   iterations:=iterations+1;
  exit when iterations=100;
  end loop;


  iterations:=0;
  loop
   insert into extclosing values (extclosingty(from15_45,to19_0,to_date(trunc(dbms_random.value(to_char(sysdate,
'J'), to_char(sysdate+ dbms_random.value(0,1830), 'J'))), 'J')));
   iterations:=iterations+1;
  exit when iterations=100;
  end loop;
end;
create or replace procedure populate_deptextclosings as
cursor dp is select ref(d) from dept d;
dep_ref ref deptty;
close_ref ref extclosingty;
iterations number;
begin
 open dp;
 loop
 fetch dp into dep_ref;
   iterations:=0;
   loop
     select ref_ext into close_ref
     from (select ref(ext) as ref_ext from extclosing ext order by dbms_random.value)
     where rownum=1;
     insert into deptextclosing values (deptextclosingty(dep_ref,close_ref));
    iterations:=iterations+1;
   exit when iterations=5;
   end loop;
 exit when dp%NOTFOUND;
```

```
 end loop;
 close dp;
end;
```

this procedure insert the extraordinary closing day of the department

---

```
create or replace procedure populate_buildextclosings_sub(bd_ref ref buildingty) as
cursor dpext is select ref(dpext) from deptextclosing dpext;
dept ref deptty;
close_ref ref deptextclosingty;
nomedept varchar2(50);
nomedept2 varchar2(50);
extclosing ref extclosingty;
begin
 open dpext;
 loop
   fetch dpext into close_ref;
     select d.namedept into nomedept from dept d where ref(d)=deref(close_ref).deptextclosing;
     select d.namedept into nomedept2 from dept d where ref(d)=deref(bd_ref).dept;
     select ref(e) into extclosing from extclosing e where ref(e)=deref(close_ref).extclosing;
     if nomedept=nomedept2 then
       insert into buildextclosing values (buildextclosingty(bd_ref,extclosing));
      end if;
    exit when dpext%NOTFOUND;
   end loop;
 close dpext;
end;
```

Active rules

Rules for the sequences

```
create or replace trigger evalidation_incr

before insert on evalidation

for each row

begin

 select evalidation_seq.nextval into :new.idevalidation

  from dual;

end;
```

The trigger insert a value the field “idevalidation” when a new tuple is added on the table “Evalidation”.

```
create or replace trigger building_incr

before insert on building

for each row

begin

 select building_seq.nextval into :new.code

  from dual;

end;
```

The trigger insert a value the field “code” when a new tuple is added on the table “Building”.

```
create or replace trigger pidreader_incr

before insert on pidreader

for each row

begin

 select pidreader_seq.nextval into :new.pidreaderID

  from dual;

end;
```

The trigger insert a value the field “pidreaderID” when a new tuple is added on the table “Pidreader”.

```
create or replace trigger pidcard_incr
before insert on pidcard
for each row
begin
 select pidcard_seq.nextval into :new.pidcardID
  from dual;
end;
```

The trigger insert a value the field “pidcardID” when a new tuple is added on the table “Pidcard”.

Trigger on Evalidation and Entrance

```
create or replace trigger checkduplies_evalidation
before insert on evalidation
for each row
declare
new_row evalidationty;
evalidationexist integer;
begin
 new_row := :new.sys_nc_rowinfo$;
  --check if there is a duplied evalidation
  select count(*) into evalidationexist from evalidation e where e.datehour=new_row.datehour and e.pidcard=new_row.pidcard and e.pidreader=new_row.pidreader;
  if evalidationexist>0 then
        raise_application_error(-20001, 'this evalidation already exists!');
  end if;
end;
```

This trigger checks if there are duplies in the evalidation table by checking the datehour,pidcard and pidreader matching.

```
create or replace trigger check_evalidation
after insert on evalidation
for each row
declare
pidcardactive number;
allowed number;
```

```
isopen number;
extclosingdates number;
untilhour number;
new_row evalidationty;
errore varchar2(100);
x integer;
rff_build ref buildingty;
evalidationexist integer;
enddate date;
is_special integer;
is_employee integer;
begin
    --check if the pidcard is active
    select count(*) into pidcardactive from pidcard p where :new.pidcard= ref(p) and status='A';
    if pidcardactive>0 then
        --   if it is an employee check if the contract is not ended.
          enddate:=null;
          select count(*) into is_employee
          from person f
           where (value(f) is of type (employeety)) and deref(:new.pidcard).person=fiscalcode;
           if is_employee>0 then
             select treat(value(f) as employeety).enddate_workcontract into enddate
             from person f
             where (value(f) is of type (employeety)) and deref(:new.pidcard).person=fiscalcode;
           end if;
            if enddate<>null and enddate>current_date then
                errore:='the work contract is ended so the card is now invalid!';
                --invalid the pidcard
                update pidcard
                set status='D'
                where deref(:new.pidcard).person=person;
               raise_application_error(-20001, errore);
             else
```

```
--check if the pidreader is of the one of the allowed structures
select count(*) into allowed from pidreader p where ref(p)=:new.pidreader
  and p.building in (select t.building as building from pidcard p,table(p.buildingallowed) t where
:new.pidcard= ref(p));
if allowed>0 then
  --check if it is a special card
  extclosingdates:=0;
  select count(*) into is_special from pidcard p where ref(p)=:new.pidcard and special_card='T';
  if is_special=0 then
    --check if the building is open at this hour
    select count(*) into isopen from building b where deref(:new.pidreader).building=ref(b)
     and to_number(extract(hour from :new.datehour))
       between (deref(b.openat).hours) and (deref(b.closeat).hours);

    if isopen>0 then
     --check if is open on this date and at this hour
      select count(b.extclosing) into extclosingdates
      from buildextclosing b
      where deref(:new.pidreader).building=b.building
       and  trunc(:new.datehour,'j')=deref(b.extclosing).datetime
       and to_number(extract(hour from :new.datehour)) <=
(deref(deref(b.extclosing).endat).hours);

    else
       errore:='this building is closed at this hour';
       raise_application_error(-20001,errore);
    end if;
  end if;
  --today isn't a special closing day or you have a special card!
  if extclosingdates=0 then
  --INSERT INTO ENTRANCE!, GREAT!
   select evalidation_seq.currval into x from dual;
   dbms_output.put_line('entrance recorded!');
   select ref(b) into rff_build from building b where ref(b)=deref(:new.pidreader).building;
```

```
                --non mettere mai deref() in un insert, non verrà aggiunta la tupla tacitamente!!
                insert into entrance values (entrancety(new_row.idevalidation,rff_build,:new.datehour));
              else
               errore:='today this building is closed';
                raise_application_error(-20001, errore);
              end if;


          else
            errore:='this pid card is not authorized to access in this building!';
            raise_application_error(-20001, errore);
          end if;
      end if;
    else
      errore:='the pid card is not active,you cannot access in this building!';
      raise_application_error(-20001, errore);
  end if;
 --capturing exception we can allow the evalidation storing attempts! but no entrance is stored!
 exception
 when NO_DATA_FOUND then
  dbms_output.put_line('no data');
   insert into error_entrance values ('no data');
  when others then
    dbms_output.put_line(errore);
     insert into error_entrance values (errore);
end;
```

This trigger grant the access to the person by checkgin that

- The pidcard is recognized by the system and is active.
- The pidcard is authorized to enter in the building
- If the building isn't a special card,
    - the building must be open at the time of the swiping.
    - The building must be open on that day (it can be an extraordinary closing day).
        - If it is an extraordinary day, must be checked if the actual hour is after the end hour of the extraordinary closing day.

Trigger on Person

```
create trigger delete_person

after delete on person

for each row

declare

--student

fiscalcode char(16);

badgenumber char(6);

academicyear varchar2(10);

course varchar2(30);

status char(1);

--employee

profession varchar2(30);

contractid varchar2(15);

enddate date;

old_row personty;

begin

  old_row := :old.sys_nc_rowinfo$;

  badgenumber:=null;

  select treat(old_row as studentty).badgenumber into badgenumber from dual;

  if badgenumber<>null then

  --recover other attributes

     select treat(old_row as studentty).course into course from dual;

     select treat(old_row as studentty).status into status from dual;

    select treat(old_row as studentty).academicyear into academicyear from dual;

     insert into personremoved values
(studentty(old_row.namep,old_row.surname,old_row.fiscalcode,old_row.dateofbirth,old_row.dept,badgenumber,c
ourse,academicyear));


  else

  --is an employee

   select treat(old_row as employeety).enddate_workcontract into enddate from dual;

    select treat(old_row as employeety).contractID into contractID from dual;

    select treat(old_row as employeety).profession into profession from dual;
```

```
   insert into personremoved values
(employeety(old_row.namep,old_row.surname,old_row.fiscalcode,old_row.dateofbirth,old_row.dept,contractID,pr
ofession,enddate));

  end if;

end;
```

If a person is deleted must be kept in the database since has to be possibile to check the owner of the pidcard.

```
create or replace trigger deactive_pidcard

after insert on personremoved

for each row

begin

 update pidcard

 set status='D'

 where :new.fiscalcode=person;

end;
```

After that the person is remoed his pidcard became deactive.

```
create or replace trigger check_studentstatus

after update on person

for each row

declare

status char(1);

new_row personty;

begin

  status:=null;

   new_row := :new.sys_nc_rowinfo$;

  select treat(new_row as studentty).status into status from dual;

  if status<>null  and status='D' then

      update pidcard

      set status='D'

      where :new.fiscalcode=person;

  end if;

end;
```

At the moment of the update of a student if the new state is “deactive” then deactive is card

---

```
create or replace trigger check_basicinformation
before insert on person
for each row
declare
enddate date;
new_row personty;
duplies integer;
badgenumber char(6);
pragma autonomous_transaction;
begin
  new_row := :new.sys_nc_rowinfo$;
  enddate:=null;
   --the birthdate became 19 years before today.
 if new_row.dateofbirth>add_months(trunc(current_date),-12*18) then
    raise_application_error(-20001, 'this person cannot be recorded into the system due to the age');
 end if;
 --if it is a student
 badgenumber:=null;
 select treat(new_row as studentty).badgenumber into badgenumber from dual;
 if badgenumber<>null then
 --check if there is a duplicate key of badgenumber
   select count(*) into duplies
   from person p
   where (value(p) is of type (studentty)) and treat(value(p) as studentty).badgenumber=badgenumber;
   if duplies>0 then
      raise_application_error(-20001, 'this student already exists');
   end if;
 else
   --if it is an employee
   select treat(new_row as employeety).enddate_workcontract into enddate from dual;
    --the contract is still valid
   if  enddate<current_date then
```

```
      raise_application_error(-20001, 'the end work date must be after today');
    else
      --check if it is a duplied contractID ( for example different fiscal code but equal contract id)
       select count(*) into duplies
       from person p
       where (value(p) is of type (employeety)) and treat(value(p) as employeety).contractid= treat(new_row as employeety).contractid;
       if duplies>0 then
         raise_application_error(-20001, 'this contractid already exists');
       end if;
   end if;
 end if;
end;
```

This trigger controls that the birthdate is of 18 years ago, that the end date of the work contract is after or equal of today, that the badge number is not duplicated.

```
create or replace trigger generate_pidcard
after insert on person
for each row
declare
new_row personty;
begin
  new_row := :new.sys_nc_rowinfo$;
  --set the building associated to the department to which the person belongs to.
  insert into pidcard values (pidcardty(new_row.fiscalcode,(cast(multiset(select ref(b) from building b where b.dept=new_row.dept) as ref_buildingallowednt))));
end;
```

This trigger ensures that after inserting a person a pidcard must be generated and associated to, taking the date from the database.

Triggers on building

```
create or replace trigger check_openclosebuild
after insert on building
for each row
declare
is_invalid integer;
begin
  select deref(:new.openat).hours into is_invalid
  from dual
  where deref(:new.openat).hours> deref(:new.closeat).hours;
  if is_invalid>0 then
    raise_application_error(-20001, 'the open hour is after the closing!');
  end if;


  --check if openhour is after or equal the open hour of the department and the closinghour is  before or equal the closinghour of the dept
  select count(*) into is_invalid from dept d where :new.dept=ref(d)
     and deref(:new.openat).hours<=deref(openat).hours and deref(:new.closeat).hours>=deref(closeat).hours;


   if is_invalid>0 then
    raise_application_error(-20001, 'the opening and closing hours are not synchronous with the department associated');
  end if;
end;
```

this trigger checks if the building closing hour is after the opening hour and

That the building's opening hour is after or equal of the opening hour of the department and

That the building's closing hour is before or equal of the closing hour of the department.

```
create or replace trigger generate_pidreader
after insert on building
for each row
follows check_openclosebuild
declare
bb ref buildingty;
```

```
pragma autonomous_transaction;
begin
  --recovering the ref of the new tuple
  select ref(b) into bb from building b where :new.code=code;
  insert into pidreader values  (pidreaderty(bb));
end;
```

this trigger creates a pidreader associated to the building after that the building is inserted.

```
--check if openat < closeat  for dept
create or replace trigger check_openclosedept
after insert on dept
for each row
declare
is_invalid integer;
begin
  select deref(:new.openat).hours into is_invalid
  from dual
  where deref(:new.openat).hours> deref(:new.closeat).hours;
  if is_invalid>0 then
     raise_application_error(-20001, 'the open hour is after the closing!');
  end if;
end;
```

This trigger checks if the department closing hour is after the opening hour.

```
--if there is a special date of closure for a department it will be the same for all the buildings associated to it
create or replace trigger extclosing
after insert on deptextclosing
for each row
declare
cursor cs is select ref(b) from building b where dept=:new.deptextclosing;
rf_build ref buildingty;
begin
```

```
  open cs;
  loop
  fetch cs into rf_build;
    insert into buildextclosing values (buildextclosingty(rf_build,:new.extclosing));
  end loop;
  close cs;
end;
```

This trigger checks that if a department is closed in a date, from an hour to another one or for all day, consequently all the buildings of this department must be closed.

Index Definition

On the base of the physical design, i hae defined these following indexes

Operation 2:the delete of a person after finding her by name, surname or fiscalcode.

create index idx_nameperson on person(namep);

create index idx_surnameperson on person(surname);

Cost without index

```
select  namep from person where namep='8D3J1K74DR';
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

ERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			136
TABLE ACCESS	PERSON	FULL	136
Filter Predicates			
NAMEP='8D3J1K74DR'			

With index on name

```
select  namep from person where namep='8D3J1K74DR';
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

PERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			1
INDEX	IDX_NAMEPERSON	RANGE SCAN	1
Access Predicates			
NAMEP='8D3J1K74DR'			

With index on Surname

```
select * from person where surname='I2A4RCDG18 ';
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

PERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			1
TABLE ACCESS	PERSON	BY INDEX ROWID	1
INDEX	IDX_SURNAMEPERSON	RANGE SCAN	1
Access Predicates			
SURNAME='I2A4RCDG18 '			

Without index on Surname

```
select * from person where surname='I2A4RCDG18 ';
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

PERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			136
TABLE ACCESS	PERSON	FULL	136
Filter Predicates			
SURNAME='I2A4RCDG18 '			

For this, both the index are mantained to optimize.

While for the **operation 3 and 4** i have then decided to don't implement the indexes since they are never used by the dbms since the query is made by using the identifiers and referements.

Web Application

System package

The applicative part of the system is divided into some parts:

- Controller: this package contains the Java class used to realize the Controller part of the pattern MVC.

  In this package there are all the servlet that handle the request coming from the JSP pages.

- Model: This package contains the class of the applicatie model.the package contains the business object, that models the applicative data and the DAO class that handle the database connection.

Control Components:

PersonController:

- insert a person: Employee or a Student.
- Find a person by name,surname or fiscal code or by combining them.
- Delete a person.

EntranceController:

- Try the access into a building
- View the statistics by Hour and Date and Building or by combinig them.

View Components:

The view component of the MVC patters has been implemented by JSP pages.

The interface has been realized using the Bootstrap framework and the JQuery library

There are the pages of the system:

Validation:

You have to choose your pidcard and the building in which you are trying to access and try to access.

insStudent

You can insert all the data about a student and then add it to the database

insEmployee

You can insert all the data about a employee and then add it to the database

findPerson

You can find a peerson by name,surname or fiscal code.

After the finding you can **delete a person**.

accessStatistics:

You can check what are the number of access into a specific building in a specific date or hour by selecting them.

Success: this page shows a success message if the operation has been executed correctly.

Failure: this page shows an error message if the operation was wrong.

Model components(only the used)

- Person: This classes encapsulates the data about a person.
- Employee: this classes extends the class Person adding more field according to the Employee type of the database.
- Student: this classes extends the class Person adding more field according to the Students type of the database.
- DAO: class that allows the connection to the database and the operation for writing and reading from.
- Building: This classes encapsulates the data about a Building.
- Department: This classes encapsulates the data about a department.
- Entrance: This classes encapsulates the data about an Entrance.
- PidCard: this classes encapsulates the data about a Pidcard.
- PidReader: this classes encapsulates the data about a Pidreader.