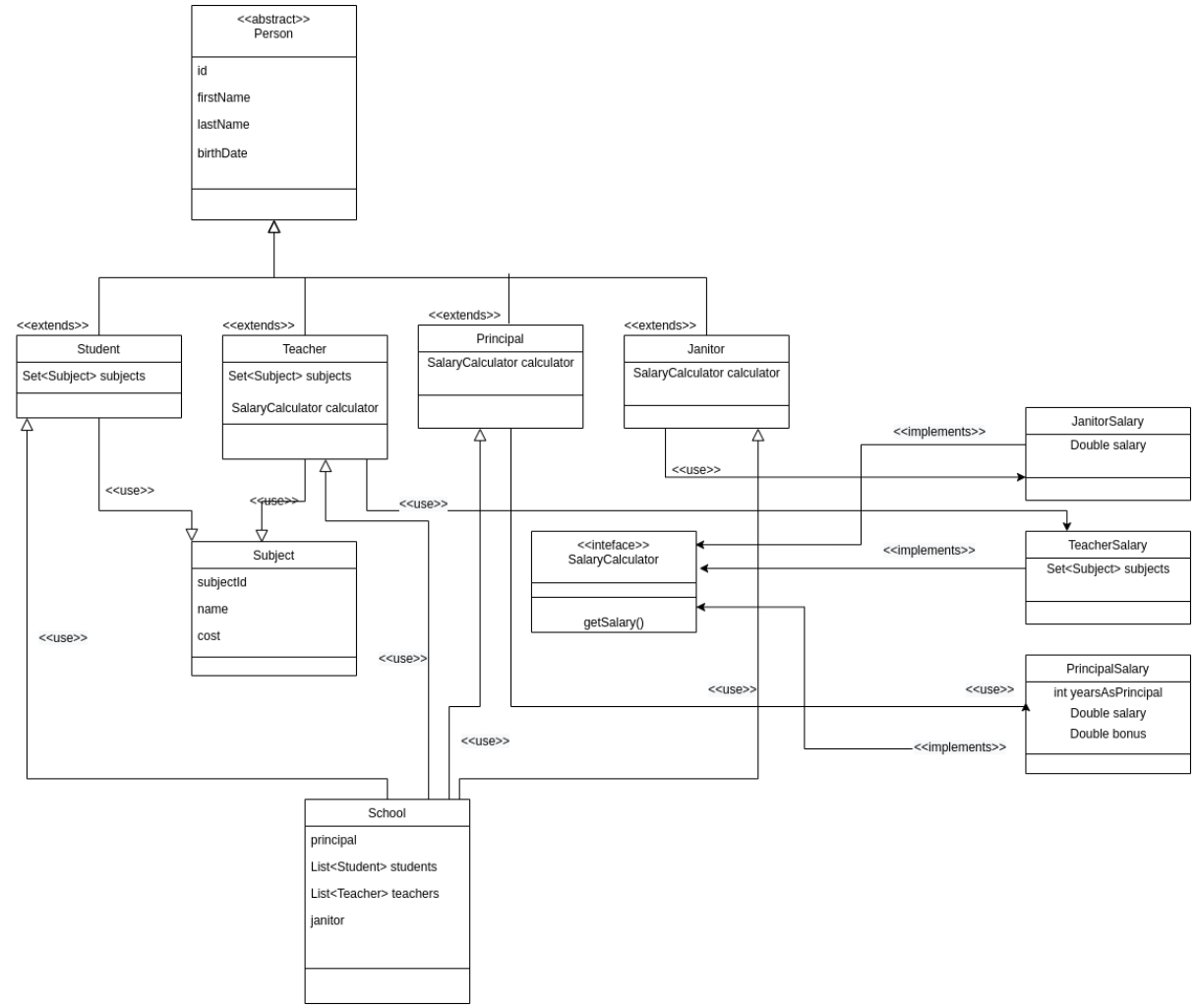
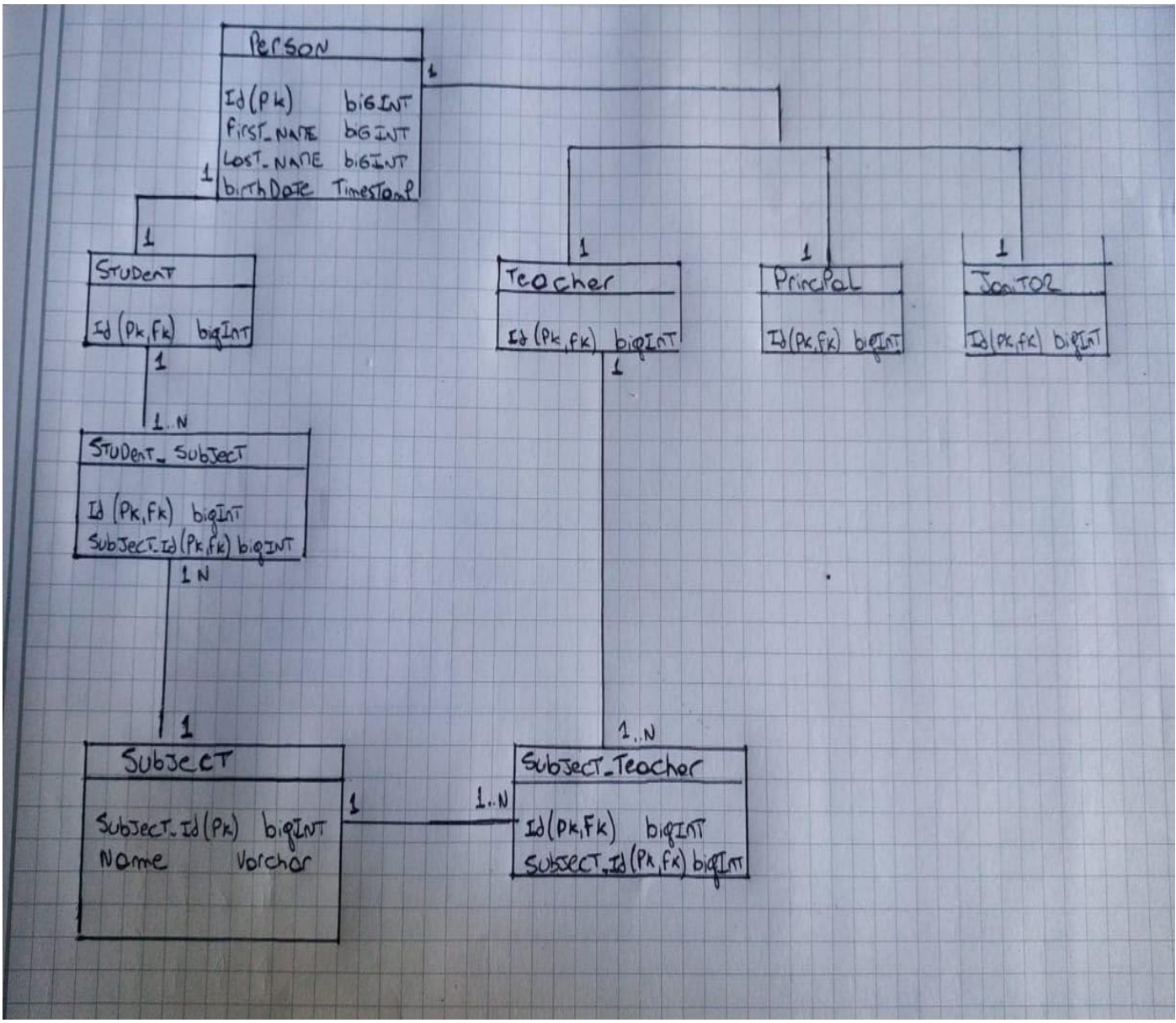
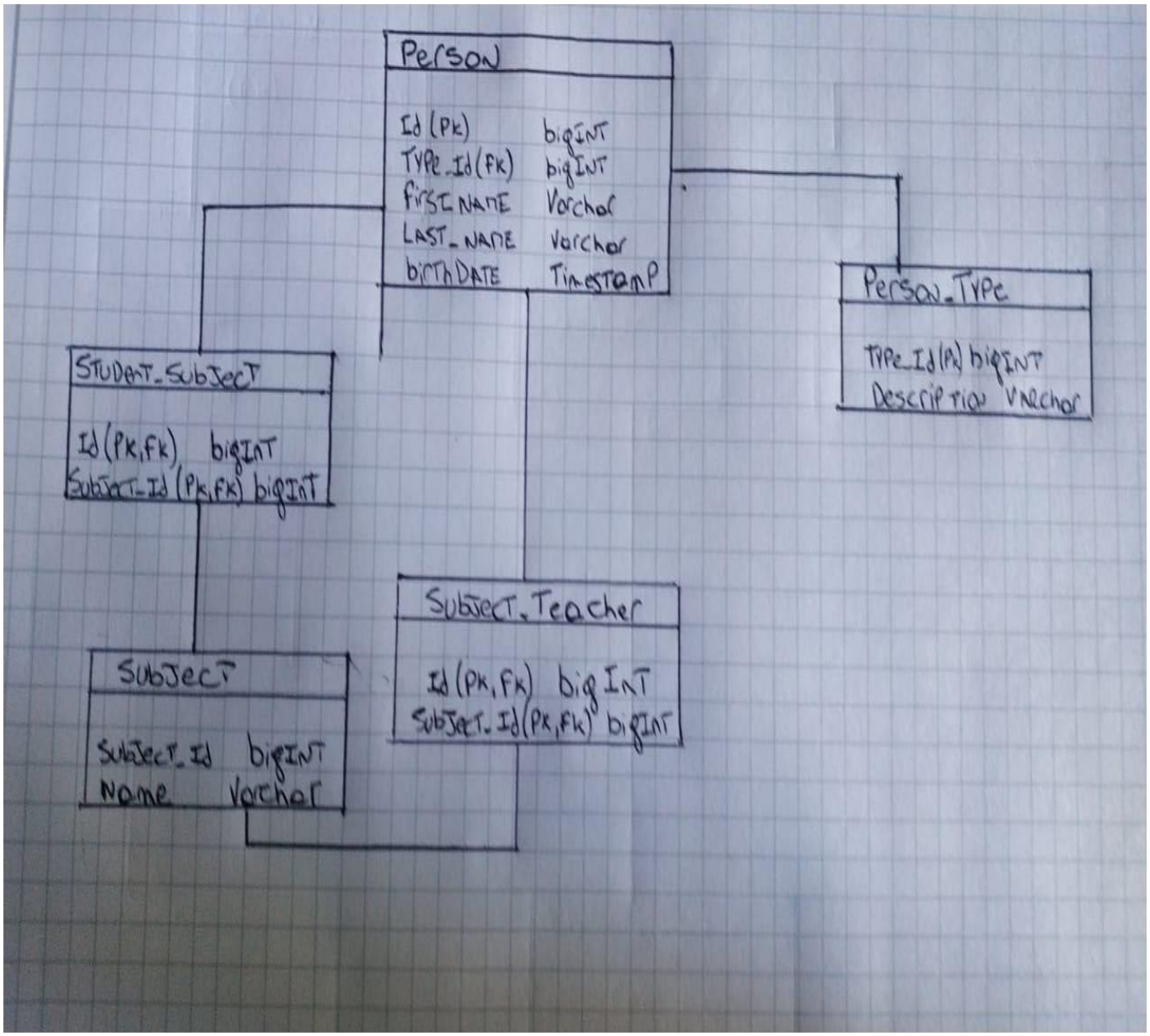
Class diagram



DB1


Person
Id (Pk) bigINT
first_name bigINT
last_name bigINT
birthDate TimeStamp
Student
Id (Pk,Fk) bigINT
Teacher
Id (Pk,fk) bigINT
Principal
Id (Pk,fk) bigINT
Janitor
Id (Pk,fk) bigINT
Student_Subject
Id (Pk,Fk) bigINT
Subject_Id (Pk,fk) bigINT
Subject
Subject_Id (Pk) bigINT
Name Varchar
Subject_Teacher
Id (Pk,Fk) bigINT
Subject_Id (Pk,fk) bigINT
1
1..N
1 N

DB2


Person
Id (Pk) bigINT
TYPE_Id (FK) bigINT
first_NAME Varchar
LAST_NAME Varchar
birthDATE TimeStamP
Person_TYPE
TYPE_Id (Pk) bigINT
Description VARchar
STUDENT_SUBJECT
Id (Pk,fk) bigINT
SUBJECT_Id (Pk,FK) bigINT
SUBJECT_Teacher
Id (Pk, FK) bigINT
SUBJECT_Id (PK,Fk) bigINT
SUBJECT
SUBJECT_Id bigINT
Name Varchar

D)

	PROS	CONS
DB1	It is more organized than DB2. Each table has all the necessary information(columns). And it doesn't have duplicated info.	It's bigger than DB2, it has more tables (more joins in queries).
DB2	The queries are more simple than in DB1. They have fewer joins.	The performance is worse than DB1 because this one has all the information in less tables so the queries to look for a teacher for example is through the same table than for a student.

E) First of all I would change the * for the columns that I really need (First and Last name). Also i could use a pagination, not to get all the results at one time. Another good optimization is to add an index to the "workingArea" column at the Janitor table. With that index we will have the rows of the table that match with "workingArea" faster than without the index. Another optimization could be adding index to the id used in the joins.

F) Considering that the values don't change very often, to optimize the query I should use a View. A view is a virtual table generated from a query. Creating a view I would have a new table created by the query so now the next query will be to this table making them fasters.

G)
```
SELECT s.Id
    FROM student s
    WHERE s.birthdate BETWEEN
    (CURRENT_DATE - interval '19 YEAR') AND
    (CURRENT_DATE - interval '21 YEAR')
```

I could optimize this query creating an index on the birthdate column. Also if the students table changes once a year (considering the new ones and the ones that left de school), I could create a view of this query and actualize it just once a year.

H) The way to have the logic in the database instead of it in the java code is using stored procedures. The way to persist a student is sending the students attributes as parameters in the stored procedure. The most important pro of using Stored procedures is it's performance. On the other hand, they are more difficult to test and to debug than java code.