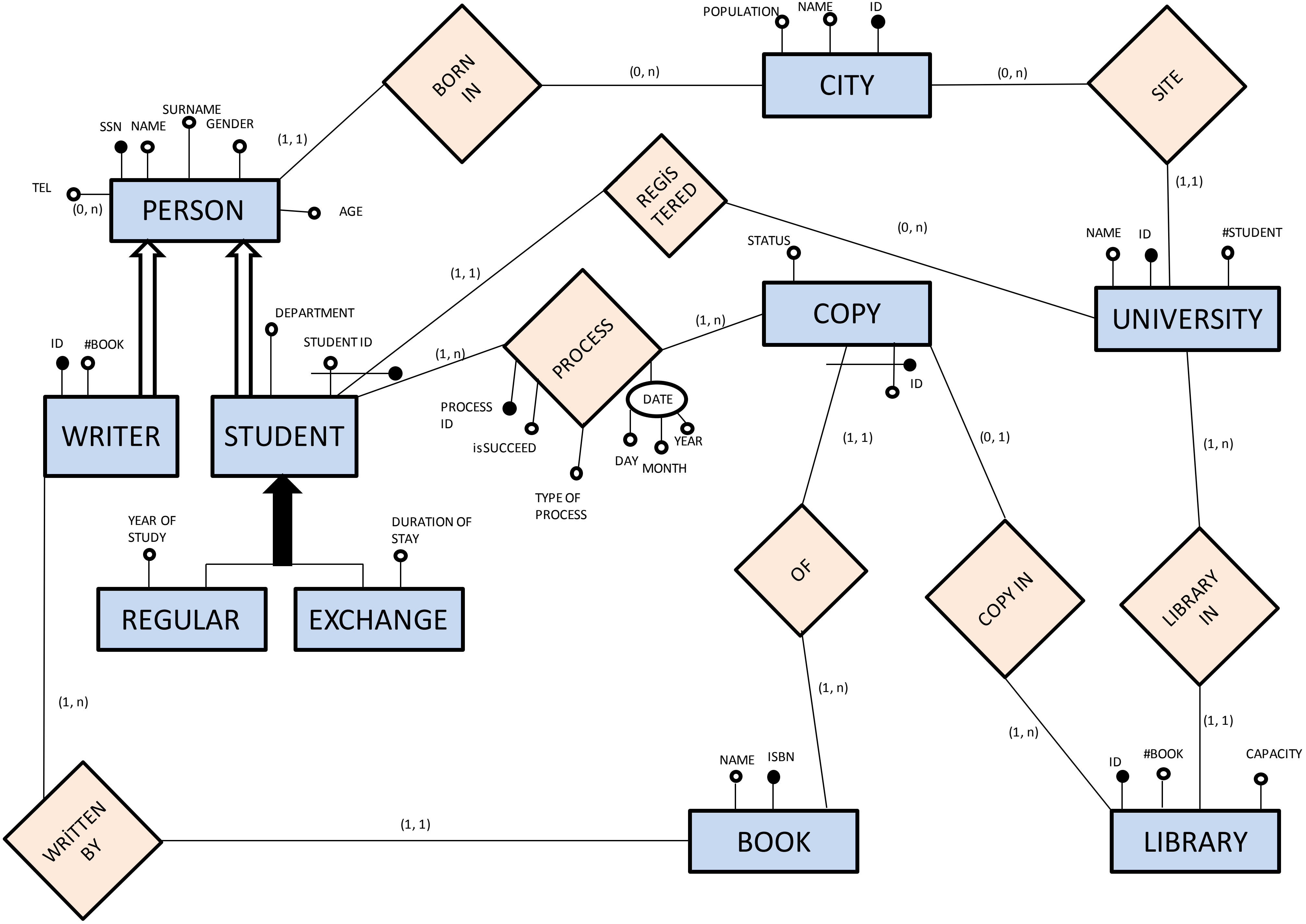
SIMPLE LIBRARY DATABASE

Simple library database project designed by Alp Furkan ÜRKMEZ (matr. Num. 10031115) with using:

- SQL, implemented on MySQL 8.0
- Java, implemented on Eclipse Java Neon , Java SE 1.8
- To establish a connection between MySQL and Eclipse, mysql java connector 5.1.47


POPULATION
NAME
ID
CITY
(0, n)
BORN IN
(0, n)
SITE
SURNAME
SSN
NAME
GENDER
(1, 1)
TEL
(0, n)
PERSON
AGE
REGİS TERED
(1,1)
(0, n)
STATUS
NAME
ID
#STUDENT
(1, 1)
COPY
UNIVERSITY
DEPARTMENT
ID
#BOOK
STUDENT ID
(1, n)
(1, n)
PROCESS
ID
WRITER
STUDENT
PROCESS ID
isSUCCEED
DATE
DAY
MONTH
YEAR
TYPE OF PROCESS
(1, 1)
(0, 1)
(1, n)
YEAR OF STUDY
DURATION OF STAY
REGULAR
EXCHANGE
OF
COPY IN
LIBRARY IN
(1, n)
(1, n)
(1, n)
(1, 1)
NAME
ISBN
ID
#BOOK
CAPACITY
WRİTTEN BY
(1, 1)
BOOK
LIBRARY

EXTERNAL CONSTRAINTS

- STUDENT can borrow any book if it registered the UNIVERSITY in which book located in its LIBRARY.
- If any REGULAR student starts an exchange program in any UNİVERSİTY, no longer it is REGULAR student.
- If a STUDENT is an EXCHANGE student, duration of stay cannot be longer than 10 months.

ENTITIES

CONCEPT	CONSTRUCT	VOLUME	DESCRIPTION
PERSON	ENTITY	100.000	KEEPS PERSON INFO.
STUDENT	ENTITY	20.000	KEEPS STUDENT INFO.
WRITER	ENTITY	100	KEEPS WRITER INFO.
REGULAR	ENTITY	15.000	KEEPS REGULAR STUDENT INFO.
EXCHANGE	ENTITY	5.000	KEEPS EXCHANGE STUDENT INFO.
CITY	ENTITY	81	KEEPS CITY INFO.
COPY	ENTITY	900.000	KEEPS COPY INFO.
BOOK	ENTITY	900	KEEPS BOOK INFO.
UNIVERSITY	ENTITY	10	KEEPS UNIVERSITYINFO.
LIBRARY	ENTITY	30	KEEPS LIBRARY INFO

RELATIONSHIPS

CONCEPT	CONSTRUCT	VOLUME	DESCRIPTION
BORN IN	RELATIONSHIP	100.000	PERSON BORN IN CITY
REGISTERED	RELATIONSHIP	20.000	STUDENT REGISTERED UNIVERSITY
SITE	RELATIONSHIP	10	UNIVERSITY SITE CITY
PROCESS	RELATIONSHIP	200.000	STUDENT MAKE PROCESS ON COPY
OF	RELATIONSHIP	900.000	COPY OF BOOK
COPY IN	RELATIONSHIP	900.000	COPY IN LIBRARY
LIBRARY IN	RELATIONSHIP	30	LIBRARY IN UNIVERSITY
WRITTEN BY	RELATIONSHIP	900	BOOK WRITTEN BY WRITER

ATTRIBUTES 1/3

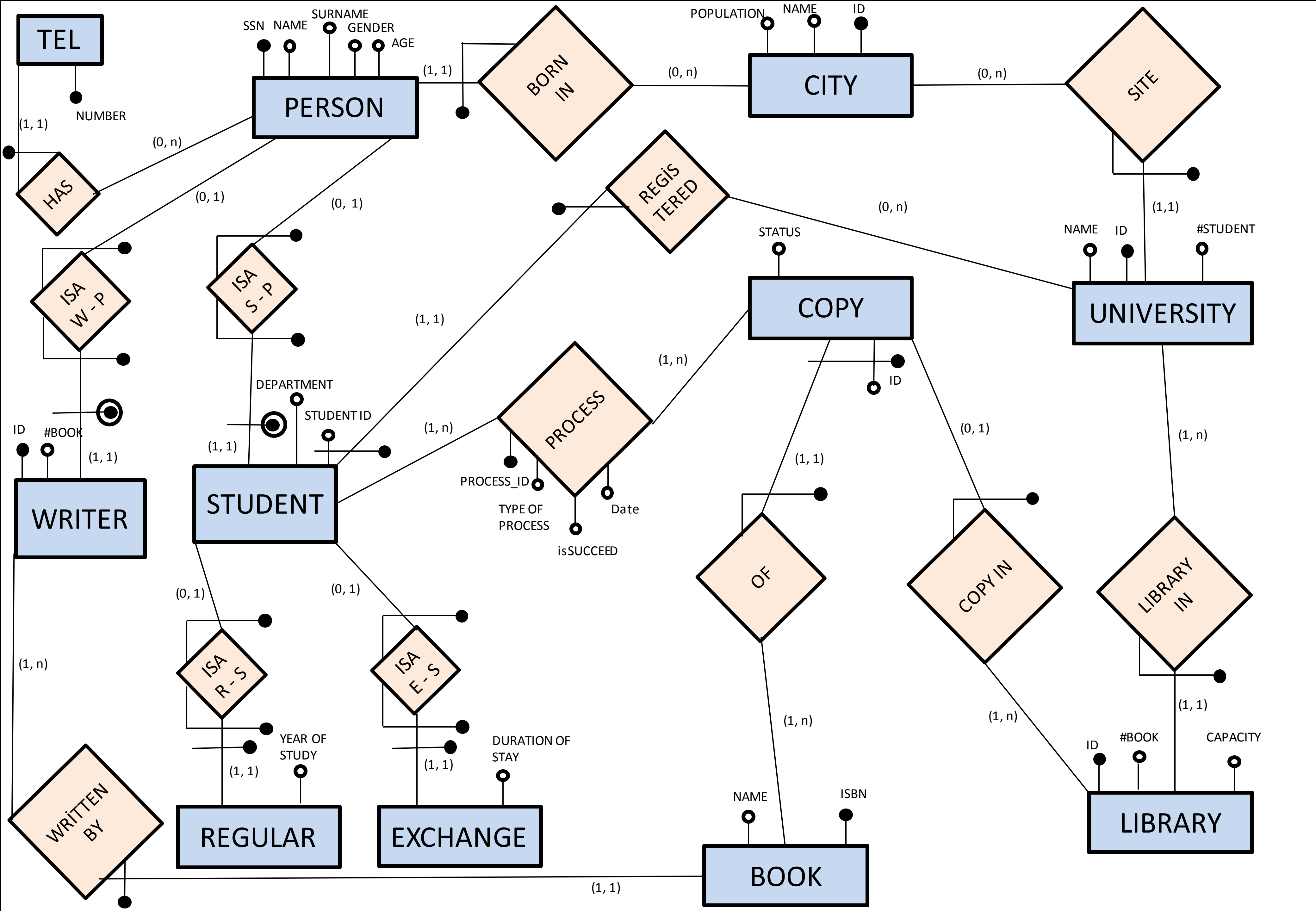
COLUMN	DATA TYPE	DESCRIPTION
SSN	INT	PRIMARY KEY OF A TABLE, PERSON’S ID NUMBER
NAME	VARCHAR(20)	PERSON’S NAME
SURNAME	VARCHAR(20)	PERSON’S SURNAME
GENDER	ENUM TYPE (‘M’, ‘F’)	M: MALE F: FEMALE
TEL	VARCHAR(20)	CELL PHONE NUMBER
AGE	INT	PERSON’S AGE
ISBN	INT	PRIMARY KEY, INTERNATIONAL SERIAL BANDROL NUMBER
NAME	VARCHAR(20)	NAME OF A BOOK
ID	INT	PRIMARY KEY, CITY ID
NAME	VARCHAR(20)	NAME OF A CITY
POPULATION	INT	POPULATION OF A CITY
ID	INT	PRIMARY KEY, WRITER ID
NUMBER OF BOOKS	INT	NUMBER OF BOOKS WRITTEN BY WRITER

ATTRIBUTES 2/3

COLUMN	DATA TYPE	DESCRIPTION
ID	INT	PRIMARY KEY WITH THE BOOK, COPY ID
STATUS	CHAR(1)	ACTUALLY ENUM BUT JDBC. (‘+’, ‘-’) AVAILABLE OR BORROWED
DURATION OF STAY	INT	D.O.S. OF EXCHANGE
ID	INT	LIBRARY ID
NUMBER OF BOOKS	INT	N.O.B IN LIBRARY
CAPACITY	INT	NUMBER OF SEATS IN LIBRARY
YEAR OF STUDY	INT	LEVEL OF REGULAR STUDENT
STUDENT ID	INT	PRIMARY KEY WITH UNIVERSITY
DEPARTMENT	VARCHAR(20)	DEPT. OF STUDENT
ID	INT	UNIVERSITY ID
NAME	VARCHAR(20)	UNIVERSITY NAME
NUMBER OF STUDENTS	INT	NUM. OF REGISTERED STUDENTS

ATTRIBUTES 3/3

COLUMN	DATA TYPE	DESCRIPTION
PROCESS ID	INT	PRIMARY KEY
ISSUCCEED	CHAR(1)	ACTUALLY ENUM (‘+’, ‘-’) BUT JDBC SUCCEED OR FAILED
TYPE OF PROCESS	ENUM (‘B’, ‘R’)	B: BORROW R: RETURN
DATE	COMPOSITE VARIABLE	DATE OF PROCESS


TEL
NUMBER
(1, 1)
SSN
NAME
SURNAME
GENDER
AGE
PERSON
(1, 1)
BORN IN
(0, n)
POPULATION
NAME
ID
CITY
(0, n)
SITE
(0, n)
HAS
(0, 1)
(0, 1)
REGİS TERED
(0, n)
(1,1)
NAME
ID
#STUDENT
UNIVERSITY
STATUS
COPY
ISA W - P
ISA S - P
(1, 1)
(1, n)
ID
DEPARTMENT
STUDENT ID
(1, n)
PROCESS
(0, 1)
(1, n)
ID
#BOOK
(1, 1)
(1, 1)
WRITER
STUDENT
PROCESS_ID
TYPE OF PROCESS
Date
isSUCCEED
(1, 1)
OF
COPY IN
LIBRARY IN
(0, 1)
(0, 1)
(1, n)
ISA R - S
ISA E - S
YEAR OF STUDY
DURATION OF STAY
(1, 1)
(1, 1)
(1, n)
(1, n)
(1, 1)
ID
#BOOK
CAPACITY
LIBRARY
NAME
ISBN
WRİTTEN BY
REGULAR
EXCHANGE
BOOK
(1, 1)

EXTERNAL CONSTRAINTS

- Each instance of STUDENT participates to ISA-R-S or ISA-E-S, but not to both.
- Each PERSON can participate either WRİTER or STUDENT or both or none of them.
- STUDENT can borrow any book if it registered the UNIVERSITY in which book located in its LIBRARY.
- If any REGULAR student starts an exchange program in any UNİVERSİTY, no longer it is REGULAR student.
- If a STUDENT is an EXCHANGE student, duration of stay cannot be longer than 10 months.

ADDITIONAL ENTITIES, RELATIONSHIPS & ATTRIBUTES AFTER RESTRUCTURING 1 / 2

CONCEPT	CONSTRUCT	VOLUME	DESCRIPTION
TEL	ENTITY	300.000	PERSON’S TELEPHONE NUMBERS

CONCEPT	CONSTRUCT	VOLUME	DESCRIPTION
ISA W-P	RELATIONSHIP	100	WRITER IS A PERSON
ISA S-P	RELATIONSHIP	20.000	STUDENT IS A PERSON
ISA R-S	RELATIONSHIP	15.000	REGULAR IS A STUDENT
ISA E-S	RELATIONSHIP	5.000	EXCHANGE IS A STUDENT
HAS	RELATIONSHIP	400.000	PERSON’S NUMBER

COLUMN	DATA TYPE	DESCRIPTION
NUMBER	VARCHAR(20)	PRIMARY KEY, TELEPHONE NUMBERS

ADDITIONAL ENTITIES, RELATIONSHIPS & ATTRIBUTES AFTER RESTRUCTURING 2 / 2

COLUMN	DATA TYPE	DESCRIPTION
PROCESS ID	INT	PRIMARY KEY
ISSUCCEED	CHAR(1)	ACTUALLY ENUM (‘+’, ‘-’) SUCCEED OR FAILED
TYPE OF PROCESS	ENUM (‘B’, ‘R’)	B: BORROW R: RETURN
DATE	DATE	DATE OF PROCESS

DIRECT TRANSLATION 1/6

University (ID, name, numberOfStudents)

foreign key: University[ID] $\subseteq$ Site[University]
inclusion: University[ID] $\subseteq$ LibraryIn[University]

Library (ID, capacity, numberOfBooks)

foreign key: Library[ID] $\subseteq$ LibraryIn[Library]
inclusion: Library[ID] $\subseteq$ CopyIn[Library]

Book (ISBN, name)

foreign key: Book[ISBN] $\subseteq$ WrittenBy[Book]
inclusion: Book[ISBN] $\subseteq$ of[Book]

City (ID, name, population)

Tel (number)

foreign key: Tel[number] $\subseteq$ Has[Tel]

DIRECT TRANSLATION 2/6

Copy (ID, Book, status)

foreign key: Copy[Book] ⊆ Book[ISBN]

inclusion: Copy[Book] ⊆ Process[ISBN]

Person (SSN, name, surname, gender, age)

foreign key: Person[SSN] ⊆ BornIn[Person]

Student (SSN, department, studentID)

foreign key: Student[SSN] ⊆ Person[SSN]

join constraints: studentID, University (in the join between a student and registered, equating the attributes student[SSN] and registered[Student] the pair of attributes studentID and University form a key.

inclusion: Student[SSN] ⊆ Process[SSN]

Writer (SSN, ID, numberOfBooks)

foreign key: Writer[SSN] ⊆ Person[SSN]

key: ID

inclusion: Writer[ID] ⊆ WrittenBy[Writer]

DIRECT TRANSLATION 3/6

Regular (SSN, studentID, University, yearOfStudy)

foreign key: Regular[SSN] $\subseteq$ Student[SSN]

foreign key: Regular[studentID, University] $\subseteq$ Student[studentID, University]

Exchange (SSN, studentID, University, durationOfStay)

foreign key: Exchange[SSN]$\subseteq$Student[SSN]

foreign key: Exchange[studentID, University] $\subseteq$ Student[studentID, University]

DIRECT TRANSLATION 4/6

Has (Tel, SSN)

foreign key: Has[Tel] ⊆ Tel[number]
foreign key: Has[SSN]⊆Person[SSN]

BornIn (SSN, City)

foreign key: BornIn[SSN]⊆Person[SSN]
foreign key: BornIn[City]⊆City[ID]

Site (University, City)

foreign key: Site[University]⊆University[ID]
foreign key: Site[City]⊆City[ID]

LibraryIn (Library, University)

foreign key: LibraryIn[LibraryIn]⊆Library[ID]
foreign key: LibraryIn[University]⊆University[ID]

DIRECT TRANSLATION 5/6

WrittenBy (ISBN, Writer)

foreign key: WrittenBy[ISBN]⊆Book[ISBN]

foreign key: WrittenBy[Writer]⊆ Writer[ID]

Process (processID, typeOfProcess, isSucceed, date, SSN, studentID, University, copyID, Book)

foreign key: Process[SSN]⊆Person[SSN]

foreign key: Process[studentID] ⊆ Student[studentID]

foreign key: Process[University] ⊆ University[ID]

foreign key: Process[CopyID] ⊆ Copy[ID]

foreign key: Process[Book] ⊆ Book[ISBN]

Registered(Student, University)

foreign key: Registered[Student] ⊆ Student[SSN]

foreign key: Registered[University] ⊆ University[ID]

DIRECT TRANSLATION 6/6

Complete generalization constraints:

$\text{Regular}[\text{SSN}] \cap \text{Exchange}[\text{SSN}] = \emptyset$

$\text{Student}[\text{SSN}] = \text{Regular}[\text{SSN}] \cup \text{Exchange}[\text{SSN}]$

RESTRUCTURING OF RELATIONAL SCHEMA

Horizontal Decomposition

Process (processID, typeOfProcess, isSucceed, date, SSN, studentID, University, copyID, Book)

Process+ (processID, typeOfProcess, isSucceed, date, SSN, studentID, University, copyID, Book)

foreign key: Process+[SSN]⊆Person[SSN]

foreign key: Process+[studentID] ⊆ Student[studentID]

foreign key: Process+[University] ⊆ University[ID]

foreign key: Process+[CopyID] ⊆ Copy[ID]

foreign key: Process+[Book] ⊆ Book[ISBN]

Process- (processID, typeOfProcess, isSucceed, date, SSN, studentID, University, copyID, Book)

foreign key: Process-[SSN]⊆Person[SSN]

foreign key: Process-[studentID] ⊆ Student[studentID]

foreign key: Process-[University] ⊆ University[ID]

foreign key: Process-[CopyID] ⊆ Copy[ID]

foreign key: Process-[Book] ⊆ Book[ISBN]

Process[processID] ⊆ Process+ [processID] ∪ Process- [processID]

Vertical Decomposition

Person (SSN, name, surname, gender, age)

PersonReg (SSN, name, surname)

foreign key: PersonReg[SSN] ⊆ PersonExt[SSN]

PersonExt (SSN, gender, age)

foreign key: PersonExt[SSN] ⊆ PersonReg[SSN]

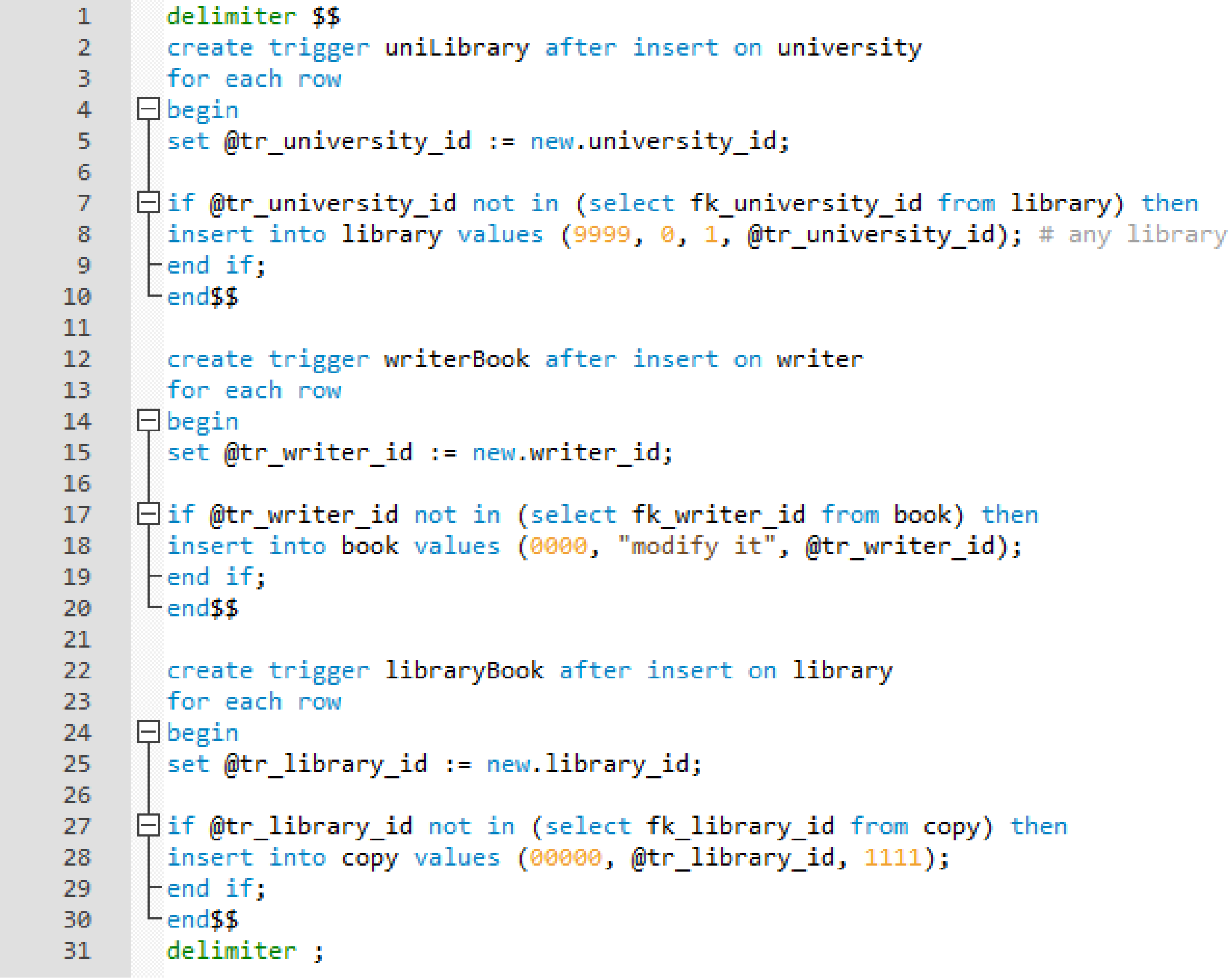
OPERATIONS

1. Show all copies inserted database with informations.
2. List processes .
3. List all students with their informations.
4. List universities with the average age of students.
5. Show people and their numbers who has more than one telephone number.

TABLE OF OPERATIONS

OPERATION	TYPE	FREQUENCY
1	Interactive	5 / day
2	Interactive	3 / day
3	Batch	1 / 5 months
4	Batch	1 / year
5	Interactive	1 / month

NOTES 1 / 2



```
delimiter $$
create trigger uniLibrary after insert on university
for each row
begin
set @tr_university_id := new.university_id;

if @tr_university_id not in (select fk_university_id from library) then
insert into library values (9999, 0, 1, @tr_university_id); # any library
end if;
end$$

create trigger writerBook after insert on writer
for each row
begin
set @tr_writer_id := new.writer_id;

if @tr_writer_id not in (select fk_writer_id from book) then
insert into book values (0000, "modify it", @tr_writer_id);
end if;
end$$

create trigger libraryBook after insert on library
for each row
begin
set @tr_library_id := new.library_id;

if @tr_library_id not in (select fk_library_id from copy) then
insert into copy values (00000, @tr_library_id, 1111);
end if;
end$$
delimiter ;
```

NOTES 2 / 2

- I also wanted to add those triggers in the previous slide because of the inclusions (1, n), but if I do this changes, instances are going to be complicated. So that, I left it without any triggers but the cardinality between University – LibraryIn, copyIn – Library and Writer – writtenBy is (1, n).