Coursework 1: Databases

Liban Abdulkadir

May 10, 2014

Question A

1. $\pi_{Title}(Module)$
2. $\pi_{Name}(\sigma_{Year=1}(Student))$
3. $\pi_{MID}(Enrolment)$

   There may be several *Enrolment* records with the same MID, however the default behaviour of $\pi_A(R)$ is to return unique A entries.

4. $\pi_{MID}(Module) - \pi_{MID}(Enrolment)$

   The right operand relation refers to the list of MIDs that have 1 or more students enrolled on the respective module. Subtracting this result from the list of all modules in existence using the difference operator gives us the opposite, which is modules with 0 students.

5. $R_1 = \pi_{MID}(Module) - \pi_{MID}(Enrolment)$
   $\pi_{Title}(Module \bowtie R_1)$

   R_1 is the relation giving MIDs of modules with no students from the previous question. The natural join operation allows us to get the values of all attributes of $Module$ for R_1. Projection is then used to obtain only the list of $Title$s.

6. $R_1 = (Student \times Module)$

   $R_2 = R_1 \bowtie Enrolment$

   $R_3 = \pi_{Name,Title}(R_2)$

   $\rho_{R_3(StudentName,ModuleTitle)}(R_3)$

   Relation R_1 describes all possible *Student-Module* combinations. The schema for this relation contains $Name$ and $Title$ (among others). After performing a natural join on R_1 and Enrolment, whose common attributes are SID and MID, only combinations such that $R_1.MID = Enrolment.MID \cap R_1.SID = Enrolment.SID$ remain. This gives us names of students and titles of modules they are enrolled on where $Name$ and $Title$ may appear more than once. Projection is then used to retain only student names ($Name$) and module titles ($Title$). Finally, we rename the attributes of the previous result to be more descriptive.

7. $R_1 = \sigma_{Year=3}(Student) \times \sigma_{Department='ComputerScience'}(Module)$

   $R_2 = R_1 \bowtie Enrolment$

   $R_3 = \pi_{Name}(R_2)$

   Cartesian product is used to obtain a relation representing all possible *Student-Module* combinations where every student is Year 3 and every module belongs to the Computer Science department. Similarly to the previous question, natural join is used to constrain all potential *Student-Module*

combinations to those that are valid (i.e. the student is enrolled on that module). Next, projection is used to obtain just the student names.

8. $R_1 = Student \times \sigma_{Title='Databases' \vee Title='Concurrency'}(Module)$

   $R_2 = R_1 \bowtie Enrolment$

   $R_3 = \pi_{SID}(R_2)$

   The condition in the selection operator ensures that only modules with titles 'Databases' or 'Concurrency' will be used later in the natural join, meaning *Enrolment* records with MIDs corresponding to other modules will not be selected.

9. $TS = \pi_{SID,Assignment+Exam}((\sigma_{Title='Databases'} Module) \bowtie Enrolment)$

   $TS_1 = \rho_{TS_1(SID,Total)}(TS)$

   $TS_2 = \rho_{TS_2(SID,Total)}(TS)$

   $TSS = \sigma_{TS_1.Total<TS_2.Total}(TS_1 \times TS_2)$

   $NO\_MAX = \rho_{NO\_MAX(SID,Total)}(\pi_{TS_1.SID,TS_1.Total}(TSS))$

   $ONLY\_MAX = TS_1 - NO\_MAX$

   $MAX\_NAMES = \pi_{Name}(ONLY\_MAX \bowtie Student)$

   TS represents SIDs and total *Assignment* and *Exam* mark for all students enrolled on the 'Databases' module. The renaming operator is used to correctly perform the product of TS with itself. All records for which $TS_1.Total < TS_2.Total$ are then excluded, which implies that $TS_1.Total$ will never be equal to the MAX of $Total$.
   $TS_1.SID$ and $TS_1.Total$ are then projected to become NO_MAX. At this point NO_MAX has an identical schema to TS_1 and TS_2 and does not contain the maximum (s) so the difference operator is used to obtain the record with just the maximum (s).
   Similarly to previous questions, the names corresponding to SIDs in $ONLY\_MAX$ are obtained by performing natural join with *Student* and then projection to keep just the names.

10. $AVGS = MID\mathcal{F}_{average(Assignment+Exam)\ as\ avgtotal}(Enrolment)$

    $E\_AVGS = Enrolment \bowtie AVGS$

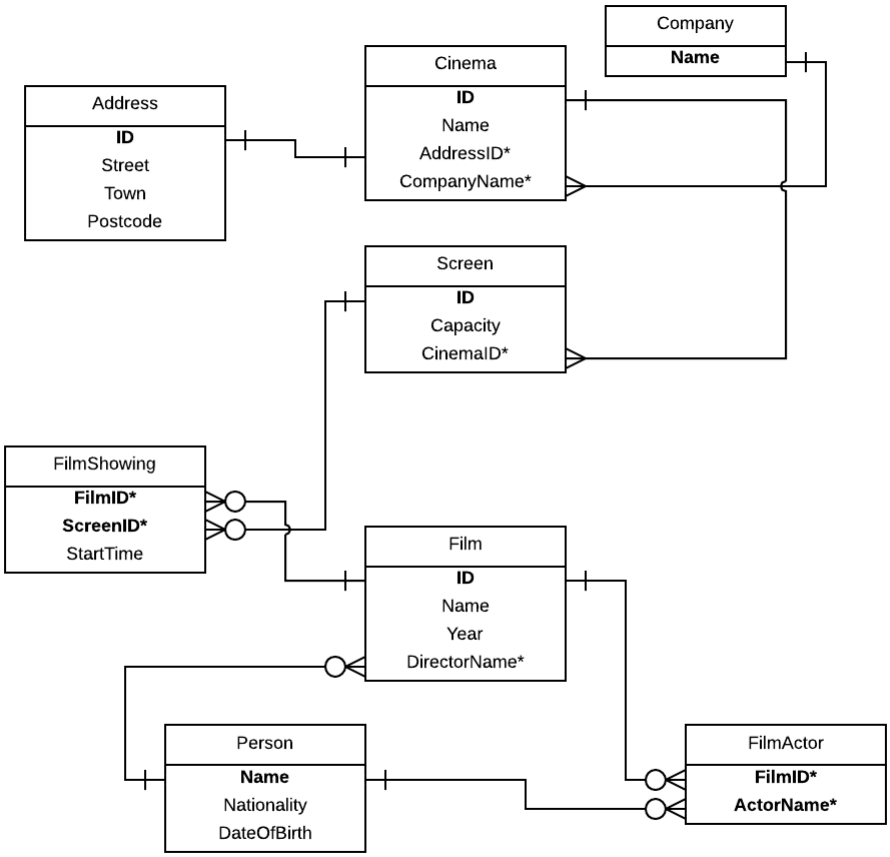
    $LOW\_SIDMIDS = \pi_{SID,MID}(\sigma_{Assignment+Exam<0.4*avgtotal}(E\_AVGS))$

    $NAME\_MIDS = \pi_{Name,MID}((\sigma_{Year=3} Student) \bowtie LOW\_SIDMIDS)$

    $NAME\_TITLES = \pi_{Name,Title}(NAME\_MIDS \bowtie Module)$

    First, the average total mark for every module is calculated. This is done using the *average* aggregate function with grouping by attribute MID. $AVGS$ has 2 attributes - MID and *avgtotal*. E_AVGS is *Enrolment* with the total mark average added corresponding to MIDs by joining $AVGS$. $LOW\_SIDMIDS$ gives us student IDs and corresponding module IDs from the enrolment table where the total $(Assignment + Exam)$ mark is less than 40% of the average mark for that module.
    Next we use natural join to restrict the result to Year 3 students and add $Name$ to the schema. Projection is then used to keep only $Name$ and MID. The result is then joined with $Module$ allowing us to obtain module titles given MIDs.

Question B



Primary keys are shown in **bold**. Foreign keys are marked with asterisks (*). Attributes that are both bold and marked with asterisks are compound keys (in entities *FilmShowing* and *FilmActor*).

Question C

```
-- Tested on PostgreSQL 9.1.10

-- DROP TABLE IF EXISTS Invoice CASCADE;
-- DROP TABLE IF EXISTS EmployedOn CASCADE;
-- DROP TABLE IF EXISTS Department CASCADE;
-- DROP TABLE IF EXISTS Staff;

CREATE TABLE Department (
    "DName" varchar(100) NOT NULL,
    "DNumber" integer NOT NULL PRIMARY KEY,
    "Address" varchar(255) NOT NULL,
    CONSTRAINT uq_dname UNIQUE ("DName")
);

CREATE TABLE Staff (
    "NINo" char(9) NOT NULL PRIMARY KEY,
    "First" varchar(100) NOT NULL,
    "Surname" varchar(100) NOT NULL,
    "DOB" date NOT NULL,
```

```
    "Address" varchar(255) NOT NULL,
    "Gender" char(1) NOT NULL,
    "Salary" numeric NOT NULL,
    "SuperNINo" char(9),
    "DNo" integer NOT NULL,

    CONSTRAINT ck_NINo CHECK ("NINo" ~* '^[a-zA-Z]{2}[0-9]{6}[a-zA-Z
       ]{1}$'),
    CONSTRAINT ck_Gender CHECK ("Gender" ~* '^[MF]$'),
    CONSTRAINT ck_SuperNINo CHECK ("SuperNINo" ~* '^[a-zA-Z
       ]{2}[0-9]{6}[a-zA-Z]{1}$'),
    CONSTRAINT fk_staff_dep FOREIGN KEY ("DNo")
        REFERENCES Department ("DNumber") ON DELETE RESTRICT ON UPDATE
           CASCADE
);

CREATE TABLE Invoice (
    "IName" varchar(100) NOT NULL,
    "INum" integer NOT NULL PRIMARY KEY,
    "DNo" integer NOT NULL,
    "InvStartDate" date NOT NULL,
    "InvEndDate" date,

    CONSTRAINT uq_iname UNIQUE ("IName"),
    CONSTRAINT fk_inv_dep FOREIGN KEY ("DNo")
        REFERENCES Department ("DNumber") ON DELETE RESTRICT ON UPDATE
           CASCADE,
    CONSTRAINT ck_inv_date CHECK ("InvStartDate" <= "InvEndDate")
);

CREATE TABLE EmployedOn (
    "ESSN" char(9) NOT NULL,
    "InNo" integer NOT NULL,
    "Hours" real NOT NULL,

    CONSTRAINT fk_eon_staff FOREIGN KEY ("ESSN")
        REFERENCES Staff ("NINo") ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_eon_inv FOREIGN KEY ("InNo")
        REFERENCES Invoice ("INum") ON DELETE RESTRICT ON UPDATE
           CASCADE
);
```