ISYS2120 Assignment 4 Sample solutions (sem1, 2022)

Due: Sunday 30 October, 11:59pm Sydney time
Value: 5% [4% for the group as a whole, and 1% individual component]
Marking criteria

- Group assessment of each question with subparts (A, B.1, C): 1 – all subparts correct, 0.5 at least 3 subparts correct.
- Group assessment of the questions without subparts (B.2, B.3): 0.5 – correct, 0.25 shows understanding of some of the main ideas, with minor errors.
- Individual component: 0.5 for showing effort and progress in the week 12 lab (or file upload). 0.5 for taking the lead in at least one subpart from each of the questions: A, B.1, and C.

The questions:

A [relational algebra] (1 point) has subparts
Note: if you have difficulty producing relational algebra symbols, you may use text, with SEL_{C} to represent σ_C and PROJ_{D} to represent π_D and X JOIN Y to represent X ⋈ Y.

This question's parts all refer to a relational schema as follows
Patient(<u>AdmitNo</u>, PName, Insurance)
Nurse(N<u>StaffNo</u>, NName, NSalary)
Doctor(<u>DStaffNo</u>, DName, DSalary)
Locate(<u>AdmitNo</u>, Ward)
Supervision(<u>NStaffNo</u>, SupervisorNStaffNo)
Assign(<u>NStaffNo, Ward</u>)
Illness(<u>AdmitNo, Disease</u>)
Expertise(<u>DStaffNo, Disease</u>)
Treatment(<u>AdmitNo, DStaffNo, Disease</u>)

A1(i) Give a relational algebra expression to calculate the table containing the AdmitNo, PName for those patients who have the illness 'COVID'.

$$\pi_{AdmitNo,PName}\,(Patient\ \bowtie\ (\sigma_{Disease='COVID'}(Illness)))$$

The selection could instead be done on the result of the natural join of Patient with Illness.

A(ii) Give a relational algebra expression to calculate the table containing the DStaffNo, DName of those Doctors who treat a patient who has insurance 'NIB'.

$$\pi_{DStaffNo,DName}\,(\sigma_{Insurance='NIB'}(Doctor\ \bowtie\ Treatment\ \bowtie\ Patient\,))$$

The selection could also be moved inward, even appliedn just to the Patient table before the joins are done.

A(iii) Give a relational algebra expression to calculate the table containing the NStaffNo and DStaffNo of a pair of doctor and nurse where the doctor treats a patient who is located on the ward to which the nurse is assigned.

$$\pi_{NStaffNo,DStaffNo}\,(Locate \bowtie Treatment \bowtie Assigned)$$

Note, one could also join with Doctor and Nurse tables as well.

A(iv) Give a relational algebra expression to calculate the table containing theAdmitNo and DStaffNo of a pair of patient and doctor, where the patient has an illness in which the doctor has expertise.

$$\pi_{AdmitNo,DStaffNo}\,(Illness \bowtie Expertise)$$

Note, one could also join with Doctor and Patient tables as well.

A(v) Give a relational algebra expression to calculate the table containing the NStaffNo, NName and DStaffNo, DName for pairs of Doctor and Nurse where the Nurse is assigned to a ward where there is a Patient who is treated by the Doctor for the disease "Anthrax".

$$\pi_{NStaffNo,NName,DStaffNo,DName}\,(Doctor \bowtie (\sigma_{Disease='Anthrax'}(Treatment) \bowtie Locate \bowtie Assigned \bowtie Nurse))$$

The selection could also be moved outward, even to be applied to the whole multiway join.

B [relational design theory] (2 points)

Consider the following relational design, which collects data about the usage of drugs at several hospitals.

DrugUsage(HospitalName, HospitalNumber, DiseaseCode, DrugName, SizeofDose, Cost, ManufacturerName, ManufacturerAddress)

The following functional dependencies are valid in this schema:

- HospitalNumber $\rightarrow$ HospitalName
- HospitalName $\rightarrow$ HospitalNumber
- DrugName $\rightarrow$ DiseaseCode, ManufacturerName
- DrugName, SizeofDose $\rightarrow$ Cost
- ManufacturerName $\rightarrow$ ManufacturerAddress

B.1 (1 point) has subparts

B.1(i) Calculate the attribute closure HospitalNumber+. Show your working.

Start with HospitalNumber

Use HospitalNumber $\rightarrow$ HospitalName to increase the set to {HospitalNumber, HospitalName}

No fd increases the set, so HospitalNumber+ is {HospitalNumber, HospitalName}

B.1(ii) Calculate the attribute closure HospitalName+. Show your working.

Start with HospitalName

Use HospitalName $\rightarrow$ HospitalNumber to increase the set to {HospitalName, HospitalNumber}
No fd increases the set, so HospitalName+ is {HospitalName, HospitalNumber}

B.1(iii) Calculate the attribute closure DrugName+. Show your working
Start with DrugName
Use DrugName $\rightarrow$ DiseaseCode, ManufacturerName to increase the set to {DrugName, DiseaseCode, ManufacturerName}
Use ManufacturerName $\rightarrow$ ManufacturerAddress to increase the set to {DrugName, DiseaseCode, ManufacturerName, ManufacturerAddress}
No fd increases the set so DtugName+ is {DrugName, DiseaseCode, ManufacturerName, ManufacturerAddress}

B.1(iv) Calculate the attribute closure (DrugName, SizeOfDose)+. Show your working.
Start with {DrugName, SizeofDose}
Use DrugName $\rightarrow$ DiseaseCode, ManufacturerName to increase the set to {DrugName, SizeOfDose, DiseaseCode, ManufacturerName}
Use DrugName, SizeofDose $\rightarrow$ Cost to increase the set to {DrugName, SizeOfDose, DiseaseCode, ManufacturerName, Cost}
Use ManufacturerName $\rightarrow$ ManufacturerAddress to increase the set to {DrugName, SizeOfDose, DiseaseCode, ManufacturerName, Cost, ManufacturerAddress}
No fd increases the set so (DrugName, SizeOfDose)+ is {DrugName, SizeOfDose, DiseaseCode, ManufacturerName, Cost, ManufacturerAddress}

Note that fds could be applied in different order, giving a different sequence of intermediate sets.

B.1(v). Calculate the attribute closure ManufacturerName+. Show your working.
Start with ManufacturerName
Use ManufacturerName $\rightarrow$ ManufacturerAddress to increase the set to {ManufacturerName, ManufacturerAddress}
No fd increases the set, so ManufacturerName+ is {ManufacturerName, ManufacturerAddress}

B2. (0.5 point) State whether the relation DrugUsage is in BCNF. Show your working.
The relation DrugUsage with the given fds is not in BCNF, because there is a fd for which the attribute closure of the left hand side is not the set of all columns (and so, the lhs is not a superkey). In fact every fd in the list given, has this property. One example (enough to show that the relation is not in BCNF) is the fd DrugName, SizeofDose $\rightarrow$ Cost, where the attribute closure of the lhs , which was calculated in B.1(iv) above, is missing the columns HospitalNumber and HospitalName.

B3. (0.5 point) Give a decomposition of the relation DrugUsage, into two or more relations, with the properties that the decomposition is both lossless-join

and dependency preserving. Explain how you know that the decomposition has these properties. For each relation in the decomposition, state whether or not that relation is itself in BCNF, and explain your working.
We can decompose DrugUsage into DU1(DrugName, SizeOfDose, Cost) and DU2(HospitalName, HospitalNumber, DiseaseCode, DrugName, SizeofDose, ManufacturerName, ManufacturerAddress)
Of the original fds, the only one all of whose columns are found in DU1, and so is fd in DU1, is DrugName, SizeofDose $\rightarrow$ Cost. Each of the other original fds refers to columns which are all in DU2, and so these fds are all fds in DU2. Because every original fd is found as fd of one of the decomposed relations, the decomposition is dependency-preserving.
Because there is an fd (DrugName, SizeofDose $\rightarrow$ Cost) such that DU1 has all the columns of this fd, and DU2 has all the columns of DrugUsage except for what is on rhs of this fd, that proves that the decomposition is lossless-join (this is from a fact stated in lectures)

DU1 is in BCNF, since the only fd for DU1 has lhs which is a superkey (the fd shows this lhs determines the only other column of DU1).
DU2 is not in BCNF, since at least one fd has lhs which is not a superkey of DU2, since its attreibute closure misses some column. For example, the fd ManufacturerName $\rightarrow$ ManufacturerAddress has lhs ManufacturerName whose attribute closure is {ManufacturerName, ManufacturerAddress}.

Note there are many other decompositions which are also acceptable solutions for this question. Here is a decomposition which is lossless-join, dependency-preserving, and gives all the relations BCNF: Drug(DrugName, DiseaseCode, ManufacturerName), Manufacturer(ManufacturerName, ManufacturerAddress), Hospital(HospitalNumber, HospitalName), DrugDose(DrugName, SizeOfDose, Cost), DrugUsageMinimal(HospitalName, DrugName, SizeOfDose).

C. [index choices] (1 point) has subparts
Suppose a database has a table T(<u>a</u>,b,c,d) with primary key being the attribute a. For each query below, explain whether the query can be answered efficiently without creating any extra index; if the default structure does not allow efficient answering, you should write a CREATE INDEX statement that will improve performance for this query. You should also note whether or not the index you propose covers the query.

C(i)
SELECT *
FROM T
WHERE T.b = 53;
The default structure for the table (without any extra indices being created) has a primary clustered index on column a, because that is the primary key. Since the output of this query could have a variety of a values, and the query doesn't tell us what they are, the primary index does not help with this query, which would need an inefficient table scan to find the output. A statement that would allow efficient answering would be

CREATE INDEX IDX_T_b ON T (b);
This index does not cover the query, as the query returns columns a, c and d as well as b, and the index declared here only has values of column b.
(If you want a covering index, it could be on T(b,a,c,d)) or some other order of all the columns starting with b.)

C(ii)
SELECT T.c
FROM T
WHERE T.a = 'XYZ';
The default structure for the table (without any extra indices being created) has a primary clustered index on column a, because that is the primary key. As this query output have a specific value for column a, the index allows this query to be answered efficiently, going down the index to exactly the relevant record. Because the index is integrated, the whole record is part of the index and can be considered as covering the query, though the term is not typically used for an integrated index.

C(iii)
SELECT T.a, T.c
FROM T
WHERE T.c > 34 AND T.c < 47;
The default structure for the table (without any extra indices being created) has a primary clustered index on column a, because that is the primary key. Since the output of this query could have a variety of a values, and the query doesn't tell us what they are, the primary index does not help with this query, which would need an inefficient table scan to find the output. A statement that would allow efficient answering would be
CREATE INDEX IDX_T_c ON T (c);
This index does not cover the query, as the query returns column a as well as c, and the index declared here only has values of column c.
(If you want a covering index, it could be on T(c,a))

C(iv)
SELECT T.a
FROM T
WHERE T.b = 53 AND T.d = 81;
The default structure for the table (without any extra indices being created) has a primary clustered index on column a, because that is the primary key. Since the output of this query could have a variety of a values, and the query doesn't tell us what they are, the primary index does not help with this query, which would need an inefficient table scan to find the output. A statement that would allow efficient answering would be
CREATE INDEX IDX_T_b_d ON T (b,d);
This index does not cover the query, as the query returns column a and the index declared here only has values of columns b and d.

(Another useful index one could define instead is on T(d,b). If you want a covering index, it could be on T(b, d, a)) or some other order of columns starting with b,d or with d,b. and including a as well)

C(v)
SELECT *
FROM T
WHERE T.d = 81 AND T.b BETWEEN 49 AND 61;
The default structure for the table (without any extra indices being created) has a primary clustered index on column a, because that is the primary key. Since the output of this query could have a variety of a values, and the query doesn't tell us what they are, the primary index does not help with this query, which would need an inefficient table scan to find the output. A statement that would allow efficient answering would be
CREATE INDEX IDX_T_d _b ON T (d,b);
This index does not cover the query, as the query returns columns a and c as well as b and d, and the index declared here only has values of columns d and b.
(If you want a covering index, it could be on T(d, b, a, c)) or some other order of all columns starting with d,b)