The University of Manchester

School of Computer Science

COMP23111: Fundamentals of Databases

Model Answers to Examples Class Tasks

Alvaro A A Fernandes

✉ a.fernandes@manchester.ac.uk

Mon 11 Dec 2017 12:02:59 GMT (V1.4)

Model Answers for Example Class 1: EC01

1. **Q1** How do you describe a *customer*?

   Q2 What do you need to know about them?

   Q3 What identifies them?

   Q4 Is it personal or rather something like the *table* they're sitting in?

   Q5 Do we need to store information about the customer (e.g., *name*, or all of it is in fact transient, i.e., it goes away after the customer leaves)?

   Q6 With regards to *kitchen*, is there a need to store information about *chefs* and their auxiliary staff?

   Q7 If so, what information is stored?

   Q8 What information is contained in an *order*?

   Q9 How do you relate `order` and *good* and *stockItem*?

   Q10 Is perhaps *good* the same as *ingredient* and it's *ingredient* that is seen as a *stockItem*?

   Q11 Or it *good* a *meal*? Perhaps *good* also encompasses products other than meals (e.g., in an Italian restaurant, they may sell olive oil, herbs and spices in tins, etc.)?

2. **A1** A customer, for us, is just the source of an order for a meal and of the payment for the meal.

   A2 We don't need to know anything about customers as such, but what meal they order and what payment information they provide.

   A3 For us, it is centred on the order, the order comes from a table, the table may have ordered as many meals as its capacity (say, four, if all seats are filled, though we sometimes join tables for large parties, so we don't enforce an upper limit per table).

   A4 As my previous answer makes clear, no, we don't take note of any identifying or descriptive information about customers as people. We do take their credit card but we don't store it anywhere. So, by customer, we really mean the table. It's how we see it.

   A5 All of it is in fact transient, i.e., it goes away after the customer leaves.

   A6 Yes, what we think of as "kitchen" here is actually the people who prepare the meals in an order, so yes, we do relate an order with the personnel (chefs and/or auxiliaries who were assigned the preparation of each meal in an order).

   A7 All we need to know is some identifying information about the member of staff, say, their internal unique initials.

   A8 An order has a unique number (which we generate internally). It comes from a table (also, has a unique number we give it) and contains many meals (these are also uniquely coded).

   A9 Hmm... It's complicated! Each meal is made of ingredients in certain weights or numbers. These ingredients are the stock items we store. What is meant by goods in the DFD is actually meals and sometimes products we sell (e.g., our branded ingredients).

   A10 No, good is not always the same as ingredient, because it could be meals, and indeed this is the normal case. But, yes, we sell own-brand ingredients and when we do they are goods, in this conceptual context.

   A11 Yes, a meal is also a good in the conceptual context of this DFD. Yes, good also encompasses products other than meals

3. **R1** There is no need to store data about a *customer*.

R2 We do store the `id` and `type` of a *paymentMethod*.

R3 There are two kinds of *paymentMethod*s: *cash* and *card*. If the payment is by *card*, we store `holderName` and `expiryDate` as well.

R4 We store data about *order* (its unique `number`).

R5 An *order* comes from *table* (which also has a unique `number`) and is for many *meal*s.

R6 We store data about *personnel*, of which there are two types, viz., *chef* and *auxiliary*. In all cases, we store their unique `initials`

R7 We need to know which *personnel* prepares which meal.

R8 A *meal* is madeOf *ingredient*s in certain `quantity`, which can be number, weight, etc.

R9 An *ingredient* is a stock item in the inventory, so it has a unique `code`, `price`, and `description`.

R10 Some *ingredient*s are sold as *good*s.

Model Answers for Example Class 2: EC02

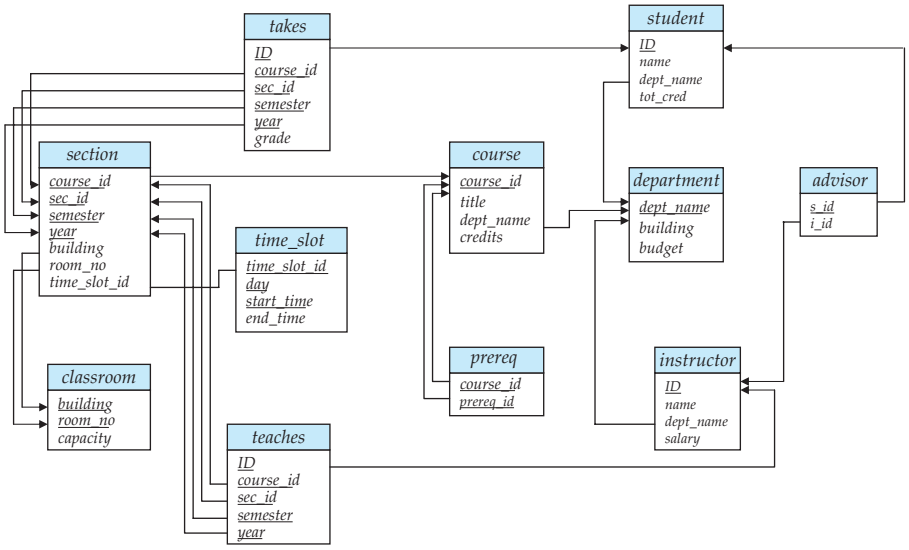

Figure 1: The University Database [SKS06] [again]

1. Retrieve the data about the instructors in the Physics department.

$$\sigma_{dept\_name='Physics'}(instructor)$$

2. Retrieve the ID, name and salary of the instructors in the university.

$$\pi_{ID,name,salary}(instructor)$$

3. Retrieve the course ID of all the courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both.

$$\pi_{course\_id}(\sigma_{semester='Fall' \wedge year=2009}(section)) \cup \pi_{course\_id}(\sigma_{semester='Spring' \wedge year=2010}(section))$$

4. Retrieve the course ID of all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

$$\pi_{course\_id}(\sigma_{semester='Fall' \wedge year=2009}(section)) - \pi_{course\_id}(\sigma_{semester='Spring' \wedge year=2010}(section))$$

5. Retrieve the course ID of all the courses taught in the Fall 2009 semester and in the Spring 2010 semester.

$$\pi_{course\_id}(\sigma_{semester='Fall' \wedge year=2009}(section)) \cap \pi_{course\_id}(\sigma_{semester='Spring' \wedge year=2010}(section))$$

where $R \cap S \equiv R - (R - S)$.

6. Retrieve the combination of the available data for each instructor and for each course taught.

   Hint: this is a Cartesian product, insofar as, here, it is not required that the combination is only between courses C taught by an instructor I, i.e., every instructor there is is combined with every taught course there is.

$$instructor \times teaches$$

7. Retrieve the salaries of instructors that are less than some other instructor salary.

   Hint: you will need to do a self-product (i.e., take the product of a relation with itself) and this requires you to use renaming (ρ) to be able to refer separately and independently to each copy that is an input to the self-product.

$$\pi_{i1.salary}(\sigma_{i1.salary<i2.salary}(\rho_{i1}(instructor) \times \rho_{i2}(instructor)))$$

8. Now, use the last expression to compute the largest salary (i.e., a salary such that is not less than any other salary in the university).

$$\pi_{i.salary}(instructor) - \pi_{i1.salary}(\sigma_{i1.salary<i2.salary}(\rho_{i1}(instructor) \times \rho_{i2}(instructor)))$$

9. Rewrite the previous expression using assignment to make clear the prior step in its construction.

$$I \leftarrow \pi_{i1.salary}(\sigma_{i1.salary<i2.salary}(\rho_{i1}(instructor) \times \rho_{i2}(instructor)))$$
$$F \leftarrow \pi_{i.salary}(instructor) - I$$

10. Retrieve the names of all instructors in the Physics department along with the course ID of all courses they have taught.

$$\pi_{i.ID,t.course\_id}(\sigma_{i.dept\_name='Physics'}(\sigma_{i.ID=t.ID}(instructor_i \times teaches_t)))$$

    or, equivalently

$$\pi_{i.ID,t.course\_id}(\sigma_{i.ID=t.ID}(\sigma_{i.dept\_name='Physics'}(instructor_i \times teaches_t)))$$

    Note that, here, we're abusing notation when we write R_r to mean that we use r to refer to elements in the extent of R.

11. Use the natural join to retrieve all the data about a student for every student who takes some course.

$$\pi_{s.ID,s.name,s.dept\_name,s.tot\_cred}(student_s \bowtie takes_t))$$

12. Now, rewrite the previous expression replacing the natural join with a Cartesian product followed by a selection.

$$\pi_{s.ID,s.name,s.dept\_name,s.tot\_cred}(\sigma_{s.ID=t.ID}(student_s \times takes_t))$$

13. Retrieve the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach.

$$\pi_{i.name,c.title}(\sigma_{i.dept\_name='Comp.Sci.'}(instructor_i \bowtie (teaches_t \bowtie course_c)))$$

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Figure 2: Tables `instructor` and `teaches` from the University Database [again]

14. Assume that at time t the state of the `instructor` and the `teaches` tables was as shown in Figure 2.

    Now let I' and T' be the result of evaluating the following expressions over Figure 2 at time t:

$$I' \leftarrow \sigma_{ID=10101 \vee ID=12121 \vee ID=15151}(\pi_{ID,name,dept\_name}(instructor))$$

$$T' \leftarrow \sigma_{course\_id='CS-101' \vee course\_id='FIN-201' \vee course\_id='BIO-101'}(\sigma_{ID=10101 \vee ID=12121 \vee ID=76766}(\pi_{ID,course\_id}(teaches)))$$

    (a) Write down the extent of I' and T'.

    $I' =$

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

    $T' =$

ID	course_id
10101	CS-101
12121	FIN-201
76766	BIO-101

    (b) Write down the table that results from evaluating the expression $I' \bowtie T'$ at time t.

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

    (c) Write down the table that results from evaluating the expression $I' ⟕ T'$ at time t.

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*null*

(d) Write down the table that results from evaluating the expression $I' \bowtie T'$ at time t.

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	*null*	*null*	BIO-101

(e) Write down the table that results from evaluating the expression $I' \rtimes T'$ at time t.

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*null*
76766	*null*	*null*	BIO-101

15. Assume the `salary` column in the `instructor` table is an annual salary. Retrieve the name and monthly salary of every instructor.

$$\pi_{name, salary/12 \rightarrow monthly\_salary}(instructor)$$

16. Retrieve the average annual salary per department.

$$_{dept\_name}\gamma_{avg(salary) \rightarrow average\_salary}(instructor)$$

Model Answers for Example Class 3: EC03

1. No nontrivial assumptions needed.

2. From the above description, we can presume that the following functional dependencies hold on the attributes:

```
FD1:
  {SSSN} →
    {SNAME,SNUM,SCADDR,SCPHONE,SPADDR,SPPHONE,BDATE,SEX,CLASS,MAJOR,MINOR,PROG}

FD2:
  {SNUM} →
    {SNAME,SSSN,SCADDR,SCPHONE,SPADDR,SPPHONE,BDATE,SEX,CLASS,MAJOR,MINOR,PROG}

FD3:
  {DEPTNAME} →
    {DEPTCODE,DEPTOFFICE,DEPTPHONE,DEPTSCHOOL}

FD4:
  {DEPTCODE} →
    {DEPTNAME,DEPTOFFICE,DEPTPHONE,DEPTSCHOOL}

FD5:
  {CNUM} →
    {CNAME,CDESC,CREDIT,LEVEL,CDEPT}

FD6:
  {SECCOURSE,SEMESTER,YEAR,SECNUM} →
    {INSTRUCTORNAME}

FD7:
  {SECCOURSE,SEMESTER,YEAR,SECNUM,SSSN} →
    {GRADE}
```

   These are the basic FDs that we can define from the given requirements; using inference rules IR1 to IR6, we can deduce many others.

3. FD1 and FD2 refer to student attributes; we can define a relation STUDENT and choose either SSSN or SNUM as its primary key.

   Similarly, FD3 and FD4 refer to department attributes, with either DEPTNAME or DEPTCODE as primary key.

   FD5 defines COURSE attributes, and FD6 defines SECTION attributes.

   Finally, FD7 defines GRADES attributes.

   We can create one relation for each of STUDENT, DEPARTMENT, COURSE, SECTION, and GRADES as shown below, where the primary keys are underlined.

   The COURSE, SECTION, and GRADES relations are in 3NF and BCNF if no other dependencies exist.

   The STUDENT and DEPARTMENT relations are in 3NF and BCNF under the definitions that allows us to ignore that both relations have secondary keys.

4. The primary keys are as follows:

```
STUDENT.
  SNUM      -- arbitrary choice over SSSN
DEPARTMENT.
  DEPTCODE  -- arbitrary choice over DEPTNAME
COURSE.
  CNUM
SECTION.
  (SECCOURSE,SEMESTER,YEAR,SECNUM)
GRADES.
  (SECCOURSE,SEMESTER,YEAR,SECNUM,SNUM)
```

We arbitrarily chose SNUM over SSSN for primary key of STUDENT, and DEPTCODE over DEPTNAME for primary key of DEPARTMENT.

The foreign keys are as follows:

```
STUDENT.MAJOR
  references DEPARTMENT.DEPTCODE
STUDENT.MINOR
  references DEPARTMENT.DEPTCODE
COURSE.CDEPT
  references DEPARTMENT.DEPTCODE
SECTION.SECCOURSE
  references COURSE.CNUM
GRADES.(SECCOURSE, SEMESTER, YEAR, SECNUM)
  references SECTION.(SECCOURSE, SEMESTER, YEAR, SECNUM)
GRADES.SNUM
  references STUDENT.SNUM
```

Model Answers for Example Class 4: EC04

1. A minimal set of attributes whose closure includes all the attributes in R is a key. Since the closure of $\{A, B\}^+$ is equal to R, one key of R is $\{A, B\}$ (and, in this case, it is the only key).

2. To normalize R intuitively into 2NF then 3NF, we take the following steps:

   (a) First, we identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key, $\{A\}$ or $\{B\}$, alone. We can calculate the closures $\{A\}^+$ and $\{B\}^+$ to determine partially dependent attributes:

   $$\{A\}^+ = \{A, D, E, I, J\}$$
   $$\{B\}^+ = \{B, F, G, H\}$$

   Hence

   $$\{A\} \rightarrow \{D, E, I, J\}$$

   where $\{A\} \rightarrow \{A\}$ is a trivial dependency, and

   $$\{B\} \rightarrow \{F, G, H\}$$

   where $\{B\} \rightarrow \{B\}$ is a trivial dependency.

   (b) To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in separate relations $R1$ and $R2$, along with the part of the key they depend on (A or B), which are copied into each of these relations but also remains in the original relation, which we call $R3$ below:

   $$R1 = \{[A], D, E, I, J\}$$
   $$R2 = \{[B], F, G, H\}$$
   $$R3 = \{[A, B], C\}$$

   The keys for $R1$, $R2$, $R3$ are bracketed.

   (c) Next, we look for transitive dependencies in $R1$, $R2$, $R3$. The relation $R1$ has the transitive dependency

   $$\{A\} \rightarrow \{D\} \rightarrow \{I, J\}$$

   so we remove the transitively dependent attributes $\{I, J\}$ from $R1$ into a relation $R11$ and copy the attribute D they are dependent on into $R11$. The remaining attributes are kept in a relation $R12$. Hence, $R1$ is decomposed into $R11$ and $R12$ as follows:

   $$R11 = \{[D], I, J\}$$
   $$R12 = \{[A], D, E\}$$

(d) The relation $R2$ is similarly decomposed into $R21$ and $R22$ based on the transitive dependency

$$\{B\} \rightarrow \{F\} \rightarrow \{G, H\}$$

as follows

$$R21 = \{[F], G, H\}$$
$$R22 = \{[B], F\}$$

(e) The final set of relations in 3NF is

$$\{R11, R12, R21, R22, R3\}$$

Model Answers for Example Class 5: EC05

1. (a) The number of possible schedules for two transactions T_1 and T_2 with, respectively, n_1 and n_2 operations is:

$$\frac{(n_1 + n_2)!}{(n_1!n_2!)}$$

   (b) In this case, $n_1 = 4$ and $n_2 = 2$, so the number of possible schedules is:

$$\frac{(4+2)!}{(4!2!)} = \frac{6 * 5 * 4 * 3 * 2 * 1}{4 * 3 * 2 * 1 * 2 * 1} = 15$$

   (c) Listed below are the possible schedules, and the type of each schedule:

   S1 $r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X);$ serial⇒ serializable
   S2 $r_1(X); w_1(X); r_1(Y); r_2(X); w_1(Y); w_2(X);$ (conflict) serializable
   S3 $r_1(X); w_1(X); r_1(Y); r_2(X); w_2(X); w_1(Y);$ (conflict) serializable
   S4 $r_1(X); w_1(X); r_2(X); r_1(Y); w_1(Y); w_2(X);$ (conflict) serializable
   S5 $r_1(X); w_1(X); r_2(X); r_1(Y); w_2(X); w_1(Y);$ (conflict) serializable
   S6 $r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); w_1(Y);$ (conflict) serializable
   S7 $r_1(X); r_2(X); w_1(X); r_1(Y); w_1(Y); w_2(X);$ ¬(conflict) serializable
   S8 $r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$ ¬(conflict) serializable
   S9 $r_1(X); r_2(X); w_1(X); w_2(X); r_1(Y); w_1(Y);$ ¬(conflict) serializable
   S10 $r_1(X); r_2(X); w_2(X); w_1(X); r_1(Y); w_1(Y);$ ¬(conflict) serializable
   S11 $r_2(X); r_1(X); w_1(X); r_1(Y); w_1(Y); w_2(X);$ ¬(conflict) serializable
   S12 $r_2(X); r_1(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$ ¬(conflict) serializable
   S13 $r_2(X); r_1(X); w_1(X); w_2(X); r_1(Y); w_1(Y);$ ¬(conflict) serializable
   S14 $r_2(X); r_1(X); w_2(X); w_1(X); r_1(Y); w_1(Y);$ ¬(conflict) serializable
   S15 $r_2(X); w_2(X); r_1(X); w_1(X); r_1(Y); w_1(Y);$ serial⇒ serializable

2. (a) In general, the number of *serial* schedules for a set of transactions $\{T_1, T_2, \ldots, T_N\}$ is $N!$

   (b) Since there are $N = 3$ transactions, there are $3! = 3 * 2 * 1 = 6$ serial schedules.

   (c) Listed below are the serial schedules:

```
1 T_1; T_2; T_3;
2 T_1; T_3; T_2;
3 T_2; T_1; T_3;
4 T_2; T_3; T_1;
5 T_3; T_1; T_2;
6 T_3; T_2; T_1;
```

References

[SKS06] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. 6th Edition. (No online copies available. For available printed copies, →See here.) McGraw-Hill, 2006.