

Advanced Database
Distributed Database
Exercise 6

1) We can identify two main components in this scenario. The first one can be designed as a distributed database covering institutions A and B. A global conceptual schema represents the users' view of the DDB, whose fragments are allocated to nodes in A and B. The second component can be modelled as a mediator system, or a federated database management system, depending on the level of heterogeneity between the two data models (i.e. the one in the first component and the other in the second one) and the type of applications running on top of the second component view. The fact that applications accessing data from A,B and C in a integrated fashion require read-only access may reduce the need for a full-fledged DBMS leading to a simpler mediator or multi-DBMS architecture.

Regarding the components of the DDB, one should consider at least two catalogs, one storing integration information regarding A and B, and the second managing information regarding (A,B) and C. The system integrating A and B will be implementing a functionality of a distributed database management system, whereas the one integrating (A,B) with C will have a single global query processor. Wrappers may be needed if the system deployed in C uses a different data model with respect to the one deployed in (A,B).

2) Data distribution – the system stores data distributed among sites A, B and C

Heterogeneity – the system integrating A with B is homogeneous, once its developed using a top-down approach with a complete agreement with respect to data model and database model. This is not the case of the integration of (A,B) with C, where the latter adopts a different database model and possibly a different data model as well.

Autonomy – the system deployed in A and B are not autonomous as they actively participate in distributed transactions. In respect to the view integrating (A,B) with C, we can consider that the components are autonomous.

3)

a) centralized database schema

Department(deptid, name)

Teacher(idteacher, name, deptid, PhDYear, PhDUniv, PhDAdvisor, level)

Course(idcourse, title, credit, deptid, idteacherResponsible)

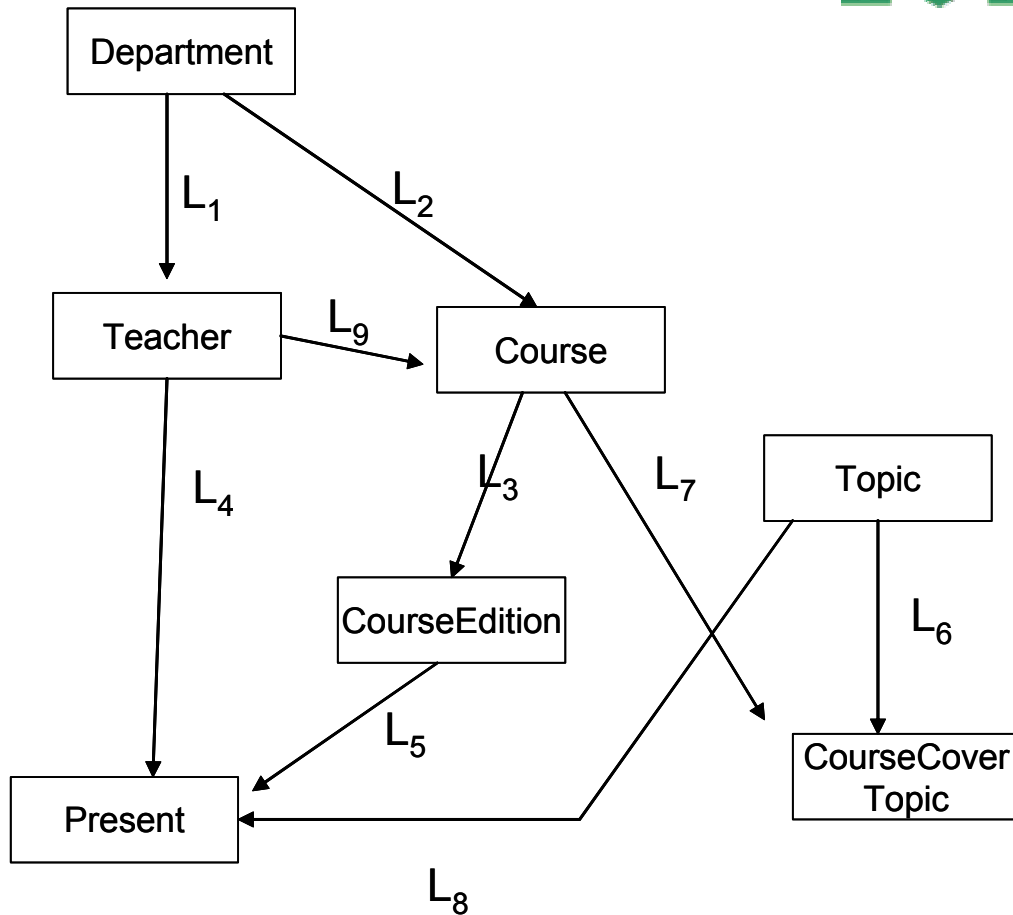
Course-Edition(idcourse, year, semester, idedition)

PresentCourseEdition(idteacher, idtopic, idedition)

Topic(idTopic, title, level, coveredtopic)

CourseCovers(idcourse, idtopic)

b) Dependency graph



c) Simple predicates

Academic Services:

Dept= 'd1' or 'd2' or 'd3'

Dept= 'd4' or 'd5'

Accounting Application

Credit ≤ 3

Credit > 3

Curriculum application

Level='fundamental'

Level='Intermediary'

Level='Advanced'

d) Fragments

Relation Department

D1= deptid in {'d1', 'd2', 'd3'}

D2= deptid in {'d4', 'd5'}

Relation Course

C1= credit ≤ 3

C2=credit > 3

Relation Topic

T1= $\sigma_{(level='fundamental')}$ Topic

T2 = $\sigma_{(\text{level} = \text{'intermediary'})}$ Topic

T3 = $\sigma_{(\text{level} = \text{'advanced'})}$ Topic


e)


Teacher


T1 = Teacher  (deptid=deptid) D1


T2 = Teacher  (deptid=deptid) D2

Course

C1 = Course  (deptid=deptid) D1 and credit ≤ 3

C2 = Course  (deptid=deptid) D1 and credit < 3

C3 = Course  (deptid=deptid) D2 and credit ≤ 3

C4 = Course  (deptid=deptid) D2 and credit < 3

CourseEdition

CE1 = CourseEdition  (idcourse=idcourse) C1

CE2 = CourseEdition  (idcourse=idcourse) C2

CE3 = CourseEdition  (idcourse=idcourse) C3

CE4 = CourseEdition  (idcourse=idcourse) C4

CourseCoverTopic (this fragmentation must be decided between Topic criterion and Course Criterion)


CCT1 = CourseCoverTopic  (idcourse=idcourse) C1


CCT2 = CourseCoverTopic  (idcourse=idcourse) C2

CCT3 = CourseCoverTopic  (idcourse=idcourse) C3


CCT4 = CourseCoverTopic  (idcourse=idcourse) C4

Present (this fragmentation must be decided among Topic, CourseEdition and Teacher criteria)

P1 = PresentCourseEdition  (idedition=idedition) CE1

P2 = PresentCourseEdition  (idedition=idedition) CE2

P3 = PresentCourseEdition  (idedition=idedition) CE3

P4 = PresentCourseEdition  (idedition=idedition) CE4

4) Vertical Fragmentation

Based on the access definition, we will restrict the vertical fragmentation analysis to the *Teacher* relation.

Consider the following attribute identifiers:

A₁=idteacher; A₂= {PhDYear, PhDUniv, PhDAdvisor}; A₃=deptid; A₄=name; A₅=level

a) The usage matrix associates attributes to queries, whenever the latter references the former:

	A1	A2	A3	A4	A5
Q ₁	1	1	1	0	0
Q ₂	1	0	0	1	1

The usage matrix does not provide a global cluster view of the relation. In order to have that, initially, one should identify the affinity between each two pair of attributes.

The affinity is computed by adding the access numbers for each query that references the attributes being considered.

As an example,

For computing $\text{Aff}(A1, A2) = 50$

I. identify queries referencing both attributes by looking at the usage matrix:

Q₁

II. sum the access patterns for all queries found in a) through all sites it runs:

a. $\text{sum} = 15 + 35 = 50$

b) The affinity matrix (symmetric) for Teacher is:

	A1	A2	A3	A4	A5
A1	120	50	50	70	70
A2	50	50	50	0	0
A3	50	50	50	0	0
A4	70	0	0	70	70
A5	70	0	0	70	70

III. It happens that the matrix in b) already proposes a clear vertical fragmentation criterion. For More complex scenarios, identifying the cluster of attributes may not be that evident. Under more complex scenarios, one would like to compute a maximizing function that would lead to a maximum affinity matrix:

$$AM = \sum_{(i=1..n)} \sum_{(j=1..n)} \text{Aff}(A_i, A_j) * [\text{Aff}(A_i, A_{(j-1)}) + \text{Aff}(A_i, A_{(j+1)})]$$

Thus, the BEA algorithm provides the matrix with the highest AM. The main intuition is to start with the two initial columns of the affinity matrix and insert the next one by analysing the place it gives the best *bond* with respect to the AM matrix. This is called the *contribution* of the new column in respect to the current AM matrix. Considering the matrix in b) we would initially consider columns A1 and A2 and analyze the effect of inserting A3 in terms of its contribution (*cont*):

For a composition of (A1, A3, A2), we would have:

$\text{Cont} = 2\text{bond}(A1, A3) + 2\text{bond}(A3, A2) - 2\text{bond}(A1, A2)$, where

$$\text{bond}(A_x, A_y) = \sum_{(j=1..n)} \text{Aff}(A_j, A_x) \text{Aff}(A_j, A_y)$$

$\text{Cont}(A0, A3, A1); \text{Cont}(A1, A3, A2)$ and $\text{Cont}(A1, A2, A3)$

	A0	A3	A1
A1	0	50	120



A2	0	50	50
A3	0	50	50
A4	0	0	70
A5	0	0	70

$$\text{Cont}(A0, A3, A1) = 2\text{bond}(A0, A3) + 2\text{bond}(A3, A1) - 2\text{bond}(A0, A1) = 2 * 11000 = 22000$$

$$\text{Bond}(A0, A3) = \text{Bond}(A0, A1) = 0$$

$$\text{Bond}(A3, A1) = 50 * 120 + 50 * 50 + 50 * 50 + 0 + 0 = 6000 + 2500 + 2500 = 11000$$

	A1	A3	A2
A1	120	50	50
A2	50	50	50
A3	50	50	50
A4	70	0	0
A5	70	0	0

$$\text{Cont}(A1, A3, A2) = 2\text{bond}(A1, A3) + 2\text{bond}(A3, A2) - 2\text{bond}(A1, A2) = 22000 + 15000 - 22000 = 15000$$

$$\text{Bond}(A1, A3) = 120 * 50 + 50 * 50 + 50 * 50 + 0 + 0 = 11000$$

$$\text{Bond}(A3, A2) = 50 * 50 + 50 * 50 + 50 * 50 + 0 + 0 = 7500$$

$$\text{Bond}(A1, A2) = 120 * 50 + 50 * 50 + 50 * 50 + 70 * 0 + 700 = 11000$$

	A2	A3	A4
A1	50	50	0
A2	50	50	0
A3	50	50	0
A4	0	0	0
A5	0	0	0

For the rightmost position, we have to consider the triple (A2, A3, A4). It occurs that A4 is not yet filled in. We use an empty column in its place.

$$\text{Cont}(A2, A3, A4) = 2\text{bond}(A2, A3) + 2\text{bond}(A3, A4) - 2\text{bond}(A2, A4) = 15000$$

$$\text{Bond}(A2, A3) = 50 * 50 + 50 * 50 + 50 * 50 + 0 + 0 = 7500$$

$$\text{Bond}(A3, A4) = 0$$

$$\text{Bond}(A2, A4) = 0$$

The order that maximizes the affinity is (A3, A1, A2).

The procedure would continue with the same process in order to insert columns A4 and A5.



Once the procedure is finished, we have a maximum Bonded Matrix. The next step is to define a cutting position that will fragment the relation into two parts. We consider the matrix diagonal. It forms two attribute groups: Top left (T) and bottom right (B). The question is to find the cutting point that maximizes the query access referencing only attributes in one of the two groups.

Comparing access references:

$T = \sum_{(q \text{ in } T)} \sum_{(S=1..n)} \text{acc}(q_i)$ /* queries that only reference attributes on the T group*/

$B = \sum_{(q \text{ in } B)} \sum_{(S=1..n)} \text{acc}(q_i)$ /* queries that only reference attributes on the B group*/

$\text{Both} = \sum_{(q \text{ in } \text{Both})} \sum_{(S=1..n)} \text{acc}(q_i)$ /* queries that reference attributes on both T and B groups */

These expressions determine the number of accesses exercised exclusively by applications accessing only *TOP*, only *Bottom* or attributes in *Both* groups.

We look for a point in the matrix diagonal that offers the highest value for:

$$Z = T*B - \text{Both}^2 \quad (\text{eq3})$$

Now, we use eq3 for all diagonal points and define the final fragmentation point.