Practice lecture 1 – October 8th 2009

View-based data integration
(Relational data model)

We use view definition to build the global schema which will be used during query processing. The views are used both as a means to combine the data and to solve the conflicts.

1) Schema analysis (data analysis, schema normalization, ...)
2) Reverse engineering (from logical to conceptual schemas).
3) Identification and resolution of conflicts.
4) Conceptual schema integration (conceptual mapping).
5) Translation of the ER schema into the corresponding logical schema.
6) Definition of data views (SQL-based logical mappings).

Exercise:

3 relational data sources with different schemas:

ROOM-BOOKING-DB (DS1)
DS1.DEPARTMENT(dept-code, dept-name, address)
DS1.RESEARCH_STAFF (email, name, school, dept-code, position)
DS1.LECTURER(email, name, school)
DS1.LECTURE (lecturer, session);
DS1.SESSION(s-code, session-name, length, room-code)
DS1.ROOM(room-code, seats-number, conference-room)

NB: A research staff member is always bound to a department but, in general, a lecturer could be an external person.
A research staff must also be a lecturer.
Sessions are intended as both lectures and seminars.
The address has the following form (Chicago Av., Washington DC, USA)
People names are encoded as (name second_name$surname).

TAX-POSITION-DB (DS2)
DS2.STUDENT (s-code, name, surname, email, school-name, income-bracket)

COURSES-DB (DS3)

DS3.PERSON (first-name, last-name, birth-date, email)
DS3.BELONGS (first-name, last-name, division, position)
DS3.ENROLLED (first-name, last-name, course)
DS3.DIVISION (code, name, description, loc-id)
DS3.COURSE (course-name, edition, t-first-name, t-last-name)
DS3.LOCATION (loc-id, city, country)

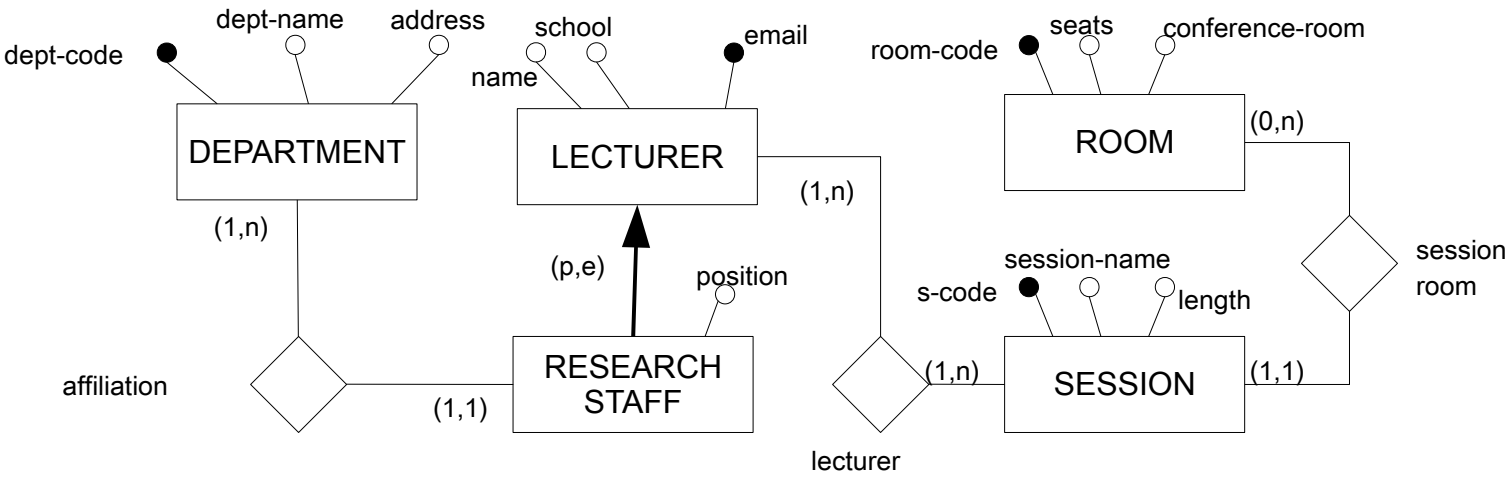
NB: There is not an explicit separation between students and professors; it can be inferred by the participations to certain relationships. A course has one and only one tenured professor. A student is enrolled in at least one course.

Step 1 - Schema analysis:

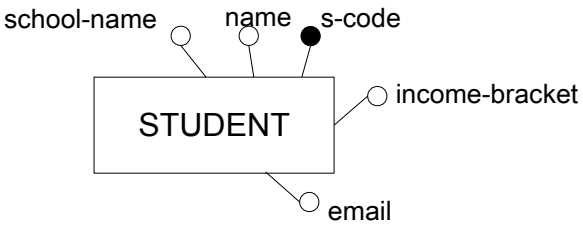
No normalization needed.
Main entities: Students, Professors, Research Staff Members, Rooms, Departments, Courses, Seminars, Talks, Locations

Step 2: Feature identification and reverse engineering.

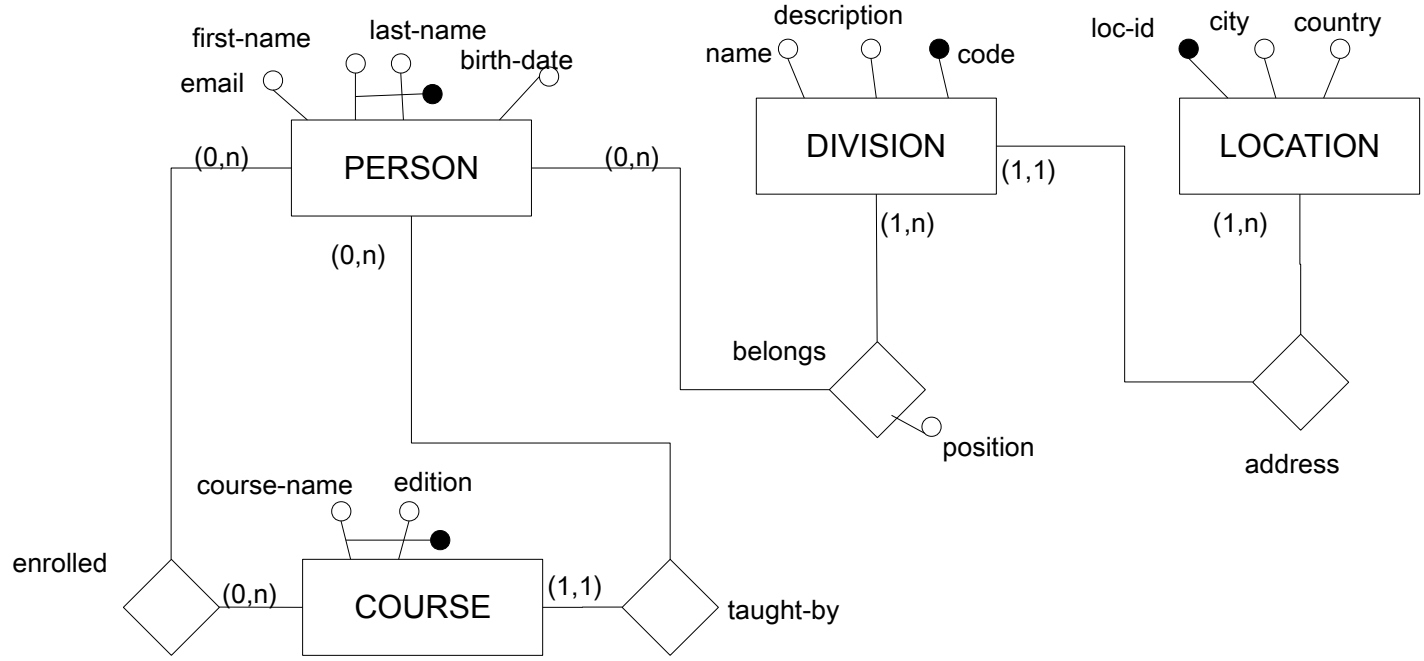
DS1 conceptual schema



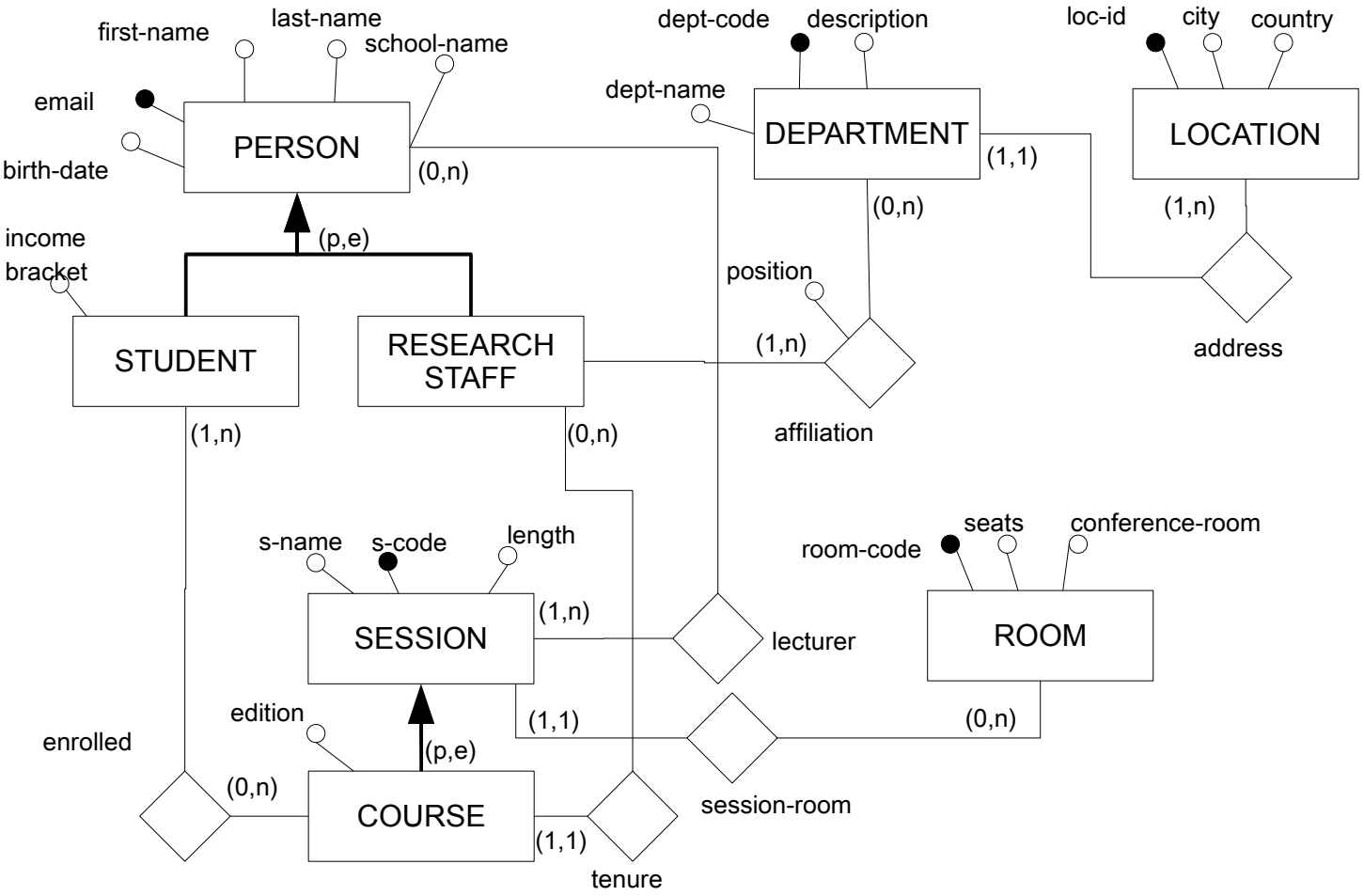
DS2 conceptual schema



DS3 conceptual schema



Steps 3 and 4: GS conceptual schema:



Conflicts analysis:

ENTITY NAME	DS1	DS2	DS3	CONFLICTS
Department	Department	not	Division	Address represented as entity in DS3 (more general)
Person	Lecturers	Student	Person	We need an ISA to differentiate Researchers, students and other lecturers. Key conflict.
Rooms	Room	not	not	
Courses	Session	not	Course	key can be emulated by course-name+year as course code.

Cardinality constraints:
Affiliation is the same in DS1 and DS3, but in DS1 we have **only one** dept. for a research-staff member.
In principle, courses in DS1 could be taught by more than one person.

Step 5: GS logical schema

GS.PERSON(email, first-name, last-name, birth-date school-name, income-bracket, role)
GS.SESSION (s-code, s-name, room-code, length)
GS.COURSE(s-code, s-name, tenure, room-code, length, edition)
GS.ENROLLED(email, course-code)
GS.LECTURER (email, s-code)
GS.AFFILIATION (email, dept-code, position)
GS.ROOM(room-code, seats, conference-room)
GS.DEPARTMENT(dept-code, dept-name, description, loc-id)
GS.LOCATION(loc-id, city, country)

Step 5: Logical Mappings (Some examples IBM DB2 9.5 SQL)

```
CREATE VIEW GS.PERSON (email, first-name, last-name, birth-date, school-name, income-bracket, role) AS

SELECT      P1.email, SUBSTR(P1.name, 0, LOCATE('$')), SUBSTR(P1.name, LOCATE('$')+1, LENGTH(P1.name)-1),
            NULL, P1.school, NULL, 'external lecturer'
FROM        DS1.LECTURER AS P1
WHERE       P1.email NOT IN (
                        SELECT email
                        FROM DS1.RESEARCH_STAFF)
UNION

SELECT      P1.email, SUBSTR(P1.name, 0, LOCATE('$')), SUBSTR(P1.name, LOCATE('$')+1, LENGTH(P1.name)-1),
            NULL, P1.school, NULL, 'research-staff'
FROM        DS1.LECTURER AS P1, DS1.RESEARCH_STAFF as RS
WHERE       P1.email = RS.email

UNION

SELECT      S2.email, S2.name, S2.surname, NULL, S2.school-name, S2.income-bracket, 'student'
FROM        DS2.STUDENT AS S2

UNION

SELECT      P3.email, P3.first-name, P3.last-name, P3.birth-date, null, 'student'
FROM        DS3.PERSON AS P3, DS3.ENROLLED AS E3
WHERE       P3.first-name = E3.first-name AND P3.last-name = E3.last-name

UNION

SELECT      P3.email, P3.first-name, P3.last-name, P3.birth-date, null, 'research-staff'
FROM        DS3.PERSON AS P3, DS3.COURSE AS C3
WHERE       P3.first-name = c3.t-first-name AND P3.last-name = cC3.t-last-name

CREATE VIEW GS.COURSE (s-code, s-name, tenure, room-code, length, edition) AS

SELECT      CONCAT(course-name, edition), C3.course-name, P3.email, S1.room-code, S1.length, C3.edition
FROM        DS3.COURSE as C3, DS1.SESSION as S1, DS3.PERSON as P3
WHERE       P3.first-name = C3.t-first-name AND P3.last-name = C3.t-last-name AND S1.session-name =
            C3.course-name
```

Further problems:

1) If a tuple of SESSION in DS1 represents a course which is not present also in DS3, we won't be able to find out that it is actually a course.
2) If we want to allow users of DS1 to query the whole database, we need to enforce a constraint which returns only one affiliation for each research staff member.
3) If a person gave different emails in DS1, DS2 and DS3 we won't be able to recognize the same person in different databases.