View-based data integration (Relational data model)

Technologies for Information Systems
November, 7th 2011

Exercise

There are 3 relational data sources with different schemas
- ROOM-BOOKING-DB (DS1)
- TAX-POSITION-DB (DS2)
- COURSES-DB (DS3)

We use view definition to build the global schema which will be used during query processing

The views are used both to combine the available data and to solve conflicts

Exercise - main steps

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)

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, room-code)

DS1.ROOM(room-code, seats-number, conference-room)

N.B.:
- A research staff member is always bound to a department but, in general, a lecturer could be an external person
- A research staff member must also be a lecturer
- Sessions are intended as both courses 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, edition)

DS3.DIVISION (code, name, description, loc-id)

DS3.COURSE (course-name, edition, t-first-name, t-last-name)

DS3.LOCATION (loc-id, city, country)

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

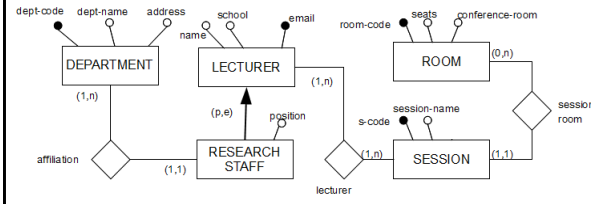
Step 1 - Schema analysis

No normalization needed

Main entities:
- Students
- Professors
- Research Staff Members
- Rooms
- Departments
- Courses
- Seminars
- Locations

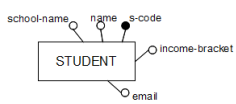
Step 2 - Reverse engineering

DS1 – conceptual schema (ER model)



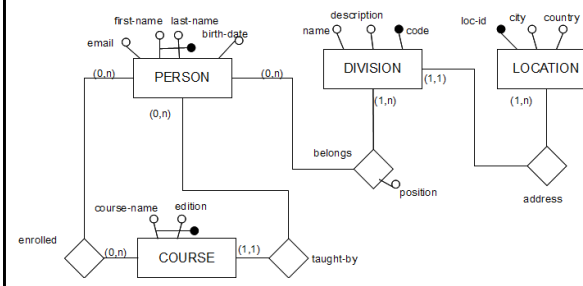
Step 2 - Reverse engineering

DS2 – conceptual schema (ER model)



Step 2 - Reverse engineering

DS3 – conceptual schema (ER model)

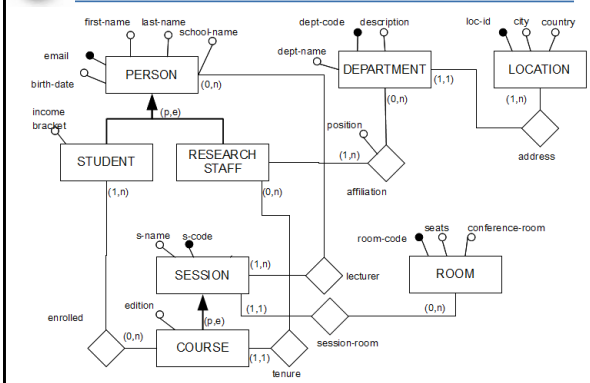


Step 3 - Conflict analysis

Entity name	DS1	DS2	DS3	Conflicts
Department	Department	Not	Division	•Address represented as entity in DS3 (more general)
Person	Lecturer	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+edition as course code

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

Step 4 - GS conceptual schema



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)
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 6 – Logical mappings (SQL-based)

```
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, 'research-staff'
FROM      DS1.LECTURER AS P1, DS1.RESEARCH_STAFF as RS
WHERE     P1.email = RS.email

UNION

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)
```


Step 6 – Logical mappings (SQL-based)

```
UNION

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

UNION

SELECT    DISTINCT 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    DISTINCT 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 = C3.t-last-name;
```


Step 6 – Logical mappings (SQL-based)

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

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

Also the other "tables" of the global schema GS must be defined as views

- GS.SESSION
- GS.LOCATION
- GS.DEPARTMENT

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