POLITECNICO
DI TORINO

Recap Database (02LSZOA)

Jacopo Nasi
Computer Engenieer
Politecnico di Torino

II Periodo - 2016/2017

March 9, 2017

Contents

1 Introduction **4**
1.1 Introduction to database 4

2 Relational data model and algebra **6**
2.1 Relational Data Model 6
2.2 Relational Algebra 8

Acknowledgments

Questo breve riepilogo non ha alcuno scopo se non quello di agevolare lo studio di me stesso, se vi fosse di aiuto siete liberi di usarlo.
Le fonti su cui mi sono basato sono quelle relative al corso offerto (**Database (02LSZOA)**) dal Politecnico di Torino durante l'anno accademico 2016/2017.
Non mi assumo nessuna responsabilità in merito ad errori o qualsiasi altra cosa.
Fatene buon uso!

1 Introduction

Everydays in our life we need to use data and the database are the structure to it in a computer-way.

1.1 Introduction to database

Information management Informations are recorded and exchanged in different forms.
The data are rough symbols which have to be interpreted and correlated to provide information.
The data are far more stable over time than the process that manage them, the bank structure hasn't change for decades instead the procedures that manage the data are changeing often.

Database There are two definitions of *database*:

- **General**: collection of data that represents information interesting for a computer system.
- **Technical**: collection of data managed by a DBMS.

The **DBMS** (*DataBase Management System*) is a software system able to manage collections of data (large, shared and persistent) ensuring the reliability and privacy. The main part of data is stored in the secondary memory.
The DB is an integrated resource, shared by several company sectors and the objective are:

- Reduction of data redundancy.
- reduction of data inconsistency.
- competing acces control mechanism.

Other importart characteristics are:

- **Data Persistance**: the lifetime is not limited to execution of programmes that use them.
- **Reliability**: backup and recovery functionality.
- **Privacy**
- **Efficency**: Capacity to carry out operations using a set of acceptable rescources for user.
- **Efficacy**: Capacity to render user activities productive.

The DBMS is an "extension" of the File System with more integrated services.

Data model The data model is a set of concepts utilized for organizing and describing a structure of data in a way understandable by a computer. This model build the relations to organize the data into sets of homogenous records (table).

Code	**Name**	**TeacherID**
M2170	Information Systems	D101
M4880	Computer Networks	D102
F0410	Databases	D321

ID	**Name**	**Deparment**	**Phone#**
D101	Green	Computer Engeneering	123456
D102	White	Telecomunications	978965
D321	Black	Computer Engeneering	456545

Before the relational model other model, near to the physical, were used like hierarchical or network. Now we use Object o XML.
The *schema* is what describe the structure of the data, is not changing over time and is represented by the heading of each table. THe schema of the previous table are:

Code	**Name**	**TeacherID**

ID	**Name**	**Deparment**	**Phone#**

The *instance* is the content of each table and is rapidly variable and represented by the row in the table.
The are tow models:

- **Conceptual**: Can rappresente data independently from the logical and it describe real world concepts (used in the desing phase).
- **Logical**: Describe the data structure in the DBMS and is used by programmes to accessing the data and it's indipendent from the physical structures.

Data independence Is an important characteristics, it guarantees that users and programeers which utilize a database can ignore the designing details used n the constructions of the database.
The **physical independence** enables interaction with the DBMS independently from the physical structure of the data and for the external level too and allows to change the way the data is physically stored without affecting the software that utilizing it.
The **logical independence** enables interaction with the external level independently from the logical, allow the modifications on the logial level maintaining the external structures unaltered and is possibile to add new views without changing the logial schema.

Data access The language used for the data access is user-friendly, interactive (SQL) and the commands are able to interact with other languages with, the programmer, are able to develop specific functionalities.
There are 2 types of languages:

- **DDL** (*Data Definition Languages*): Used to define the logical, external and physical schemas and access authorizations.
- **DML** (*Data Manipulation Languages*): Used for querying and updating database instances.

There are multiple roles in the system organizations:

- **Administrator**: Control and management of the database (performance, reliability, auth, ecc...).
- **Programmer**: Define and realize the structure of the db.
- **User**: Use software with predefined activities (end) or formulate queries not predefined (casual).

DBMS pro & cons The main advantages are:

- Data as a common resource.
- Unified and precise data model.
- Centralizable.
- Data independence.

the disadvantages are:

- Expensive with product purchase.
- Provide a set of service in an integrated form and is not possibile to separated the unused ones.

2 Relational data model and algebra

2.1 Relational Data Model

Introduction It was proposed around 1970 to support higer abstract levels compared to the previous model. Is now the main model exploited in commercial DBMSs.

Definition The are some important word:

- **Attribute**: Column of a table.
- **Domain**: Value set that can be assumed by an attribute.
- **Tuple** (or record): A row in a table.
- **Cardinality**: Number of tuples in a relation.
- **Degree**: Numbers of attributes in a relation.

We need to remember that attributes and tuples are not ordered and are distinct.

Reference between relations The relational model is value-based and the references between data in different relations are represented by means of values of the domains. This allows the independence od physical structures, only relevant information are stored, it's easy to be transferred and the link is not oriented.

Null values Some information could be not available for ani tuples in the relation, for a student table, for example, the DegreeYear it could be not yet defined, or others like this...
Is a good practise to used a special vaue belonging to the domain to represente the lack of information. The *null value* is not a good choice because is not a value of the domain and in means the absence of both value and domain.

Integrity Constraints Are property that must be satisfied by all database instances, they can be **Intra-relation** constraints defined on the attributes of a single relation (ex: unique constraints, domain constraints, tuple constraints) or **Inter-realation** constraints defined in many relations at the same time (ex: referential constraint).

Primary Key Is an attribute set that uniquely identifies tuples in a relation in the students case for example there is no pair of students with the same value for StudentID. Are unique and minimal, there exists no other key that are contained in a subset of the key. The primary key can be made by multiple NON UNIQUE keys. It can't assume the *null value.*

Tuples, domain and referential constraint These are condition over data, a domain constraint express the conditions on the value assumed by an attribute of a tuple, the tuple constraint express conditions on the values of each tuple indipendently of other tuples. The referential constraint are between more table, an attribute (external) of the A table refers to a key (internal) in the B table that must be exist. The realational constraint are imposed in order to guarantee that the values refer to actual values in the referenced relation.

2.2 Relational Algebra

Introduction This algebra extend algebra of sets for the relational model and defines a set of operators that operate on relations and whole result is a relation.
There are 2 kind of division for the operator:

- **Unary Operator**
  - Selection (σ)
  - Projection (π)
- **Binary Operator**
  - Cartesian Product (x)
  - Join ($\bowtie$)
  - Union ($\cup$)
  - Intersection ($\cap$)
  - Difference ($-$)
  - Division (/)

or in this mode:

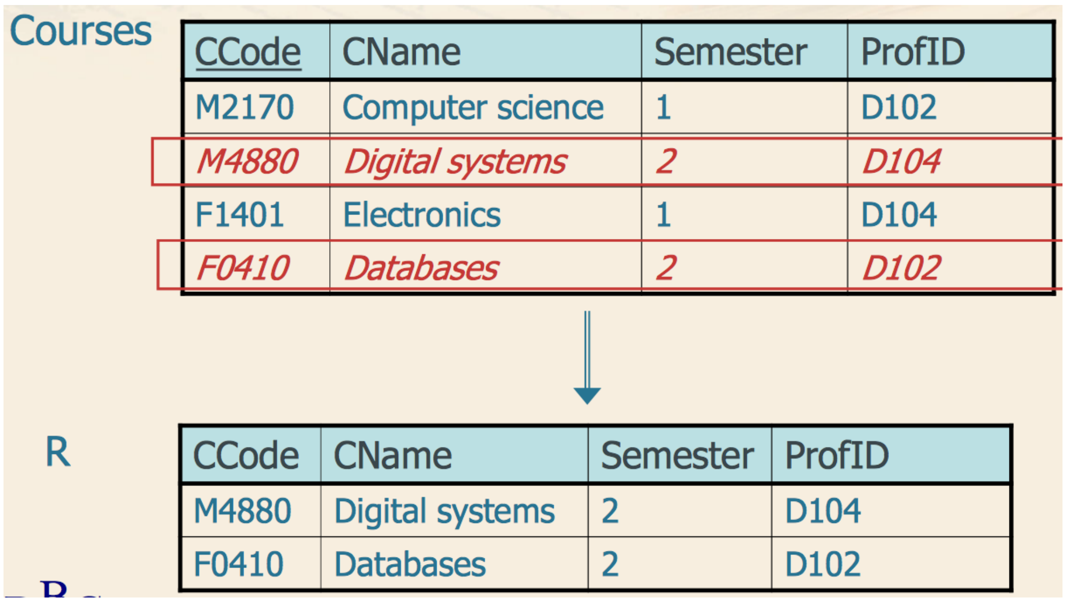
- **Set Operator**
  - Union ($\cup$)
  - Intersection ($\cap$)
  - Difference ($-$)
  - Cartesian Product (x)
- **Binary Operator**
  - Join ($\bowtie$)
  - Selection (σ)
  - Projection (π)
  - Division (/)

Selection extract a “horizontal” subset from the relation. For the query: *”Find the courses held in the second semester”* the result will be like the figure 1.
The selection generate a relation R:

- With the same schema as A.

- Containing all the tuples of relation A because of which predicate p is true.

The literal relation is $R = \sigma_p A$ in this case A is the table Courses and the predicate p is semester=2.



Courses

CCode	CName	Semester	ProfID
M2170	Computer science	1	D102
M4880	*Digital systems*	*2*	*D104*
F1401	Electronics	1	D104
F0410	*Databases*	*2*	*D102*

R

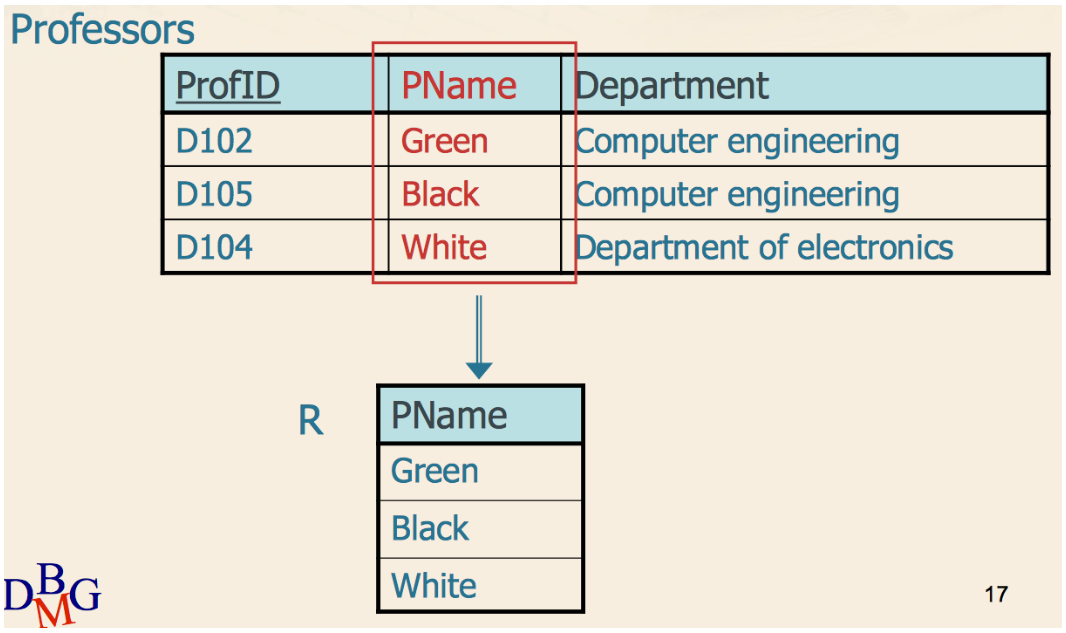
CCode	CName	Semester	ProfID
M4880	Digital systems	2	D104
F0410	Databases	2	D102

Figure 1: Selection

Projection extracts a “vertical” subset of attributes from the relation. For the query: *”Find the names of professors”* the result will be like figure 2. The projection generate a relation R:

- Whose schema is the list of attributes L.
- Containing all the tuples present in A.

The literal relation is $R = \pi_L A$ in this case A is the table Professors and the predicate is L is PName. Is also possible to use *Selection* and *Projection* together.


Professors

ProfID	PName	Department
D102	Green	Computer engineering
D105	Black	Computer engineering
D104	White	Department of electronics

R

PName
Green
Black
White

17

Figure 2: Projection

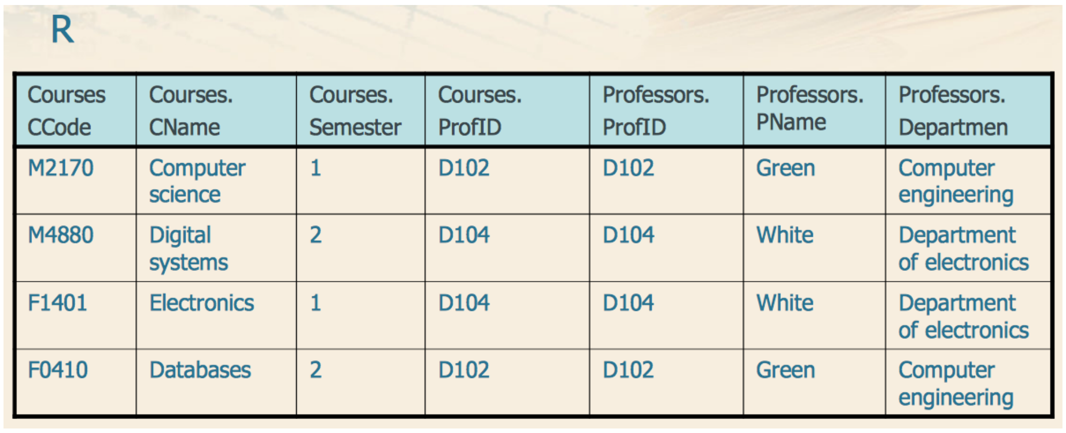
Cartesian Product it generates all the pairs formed by all tuple of A and all tuple of B. The result for the query: *"Find the cartesian product of courses and professor"* will generated a big relation with, for each tuple of A all the tuple of B. A relation with 3 tuples in A and 4 tuples in B, will give you a table of 12 tuples. The literal expression is $R = AxB$ and is commutative and associative.

Join generate a relation with all pairs formed by a tuple of A and a tuple of B that are *semantically linked*. The query: *"Find information about courses and the professor that hold them"* will give you back a relation like figure 3. The join can be expressed like a mix of *cartesian product, selection and projection.*
There are 3 different types of join:

- **Natural**: $R = A \bowtie B$ (contains all the pairs with the same value of a given attribues)
  - The attributes which are present in A's schema and not in B's.
  - The attributes which are present in B's schema and not in A's.
  - A single copy of common attributes.
- **Theta**: $R = A \bowtie_p B$ (contains all the pairs with the same value of a given attribues that meet a predicate p)
  - Whose schme ais the unione of the schemes of A and B.

  - Containing all the pairs made up of a tuple of A and a tuple of B for which predicate $(X\theta Y)$ is true.

- **Equi**: $R = A \bowtie_p B$ (special case with θ is $(=)$)
- **Semi**: $R = A \ltimes_p B$ (similar to theta but only with the attributes of A)



R

Courses CCode	Courses. CName	Courses. Semester	Courses. ProfID	Professors. ProfID	Professors. PName	Professors. Departmen
M2170	Computer science	1	D102	D102	Green	Computer engineering
M4880	Digital systems	2	D104	D104	White	Department of electronics
F1401	Electronics	1	D104	D104	White	Department of electronics
F0410	Databases	2	D102	D102	Green	Computer engineering

Figure 3: Join

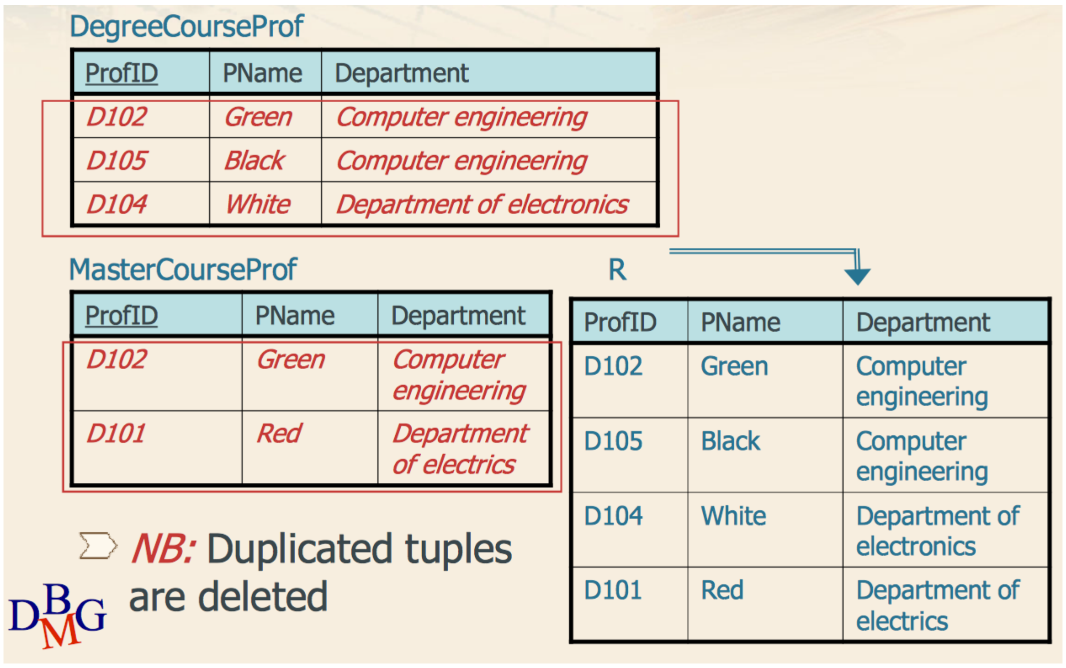
Outer-join is a version of join where you can conserve the information relative to tuples that are not semantically linked by the join predicate, the tuples will be completed with *null value*. There are 3 types of outer-join:

- **Left**: $R = A \ltimes\!\!\!\bowtie_p B$ Are conserved the tuples of the left table, with null value for the attributes of the right part.
- **Right**: $R = A \bowtie\!\!\!\rtimes_p B$ Are conserved the tuples of the right table, with null value for the attributes of the left part.
- **Full**: $R = A \ltimes\!\!\!\bowtie\!\!\!\rtimes_p B$ Are conserved the tuples of both table.

Union selects all the tuples present in at least one of the two (A and B) relations. If there are duplicated tuple this command will provide only one of this. The main characteristics are:

- R has the same schema of A and B. (This means that both A and B must have the same schema)
- Containing all the tuples belonging to A and all the tuples belonging to B.

THe literal expression is $R = A \cup B$. For example the result of the query: *"Find infomration relative to the profesors of degree or master courses"* will show to you the result of the figure 4



DegreeCourseProf

ProfID	PName	Department
D102	Green	Computer engineering
D105	Black	Computer engineering
D104	White	Department of electronics

MasterCourseProf

ProfID	PName	Department
D102	Green	Computer engineering
D101	Red	Department of electrics

R

ProfID	PName	Department
D102	Green	Computer engineering
D105	Black	Computer engineering
D104	White	Department of electronics
D101	Red	Department of electrics

NB: Duplicated tuples are deleted

Figure 4: Union

Intersection select only the tuples that are present on both table A and B. The main characteristics are:

- R has the same schema of A and B. (This means that both A and B must have the same schema)
- Containing all the tuples belonging to both A and B.

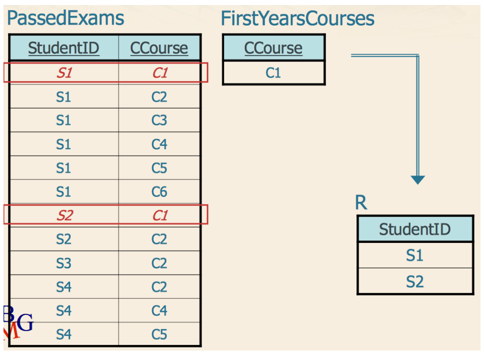
THe literal expression is $R = A \cap B$.

Difference this operator will extract only the tuples that are in the A but that are not in the B. A possibile query is: *"Find the professors teaching degree courses but not master's"*. It has the same specs of the intersection or union, but is not commutative and associative.
Like the previous operator if the schemas are different you need to perform a projection before using it.

Anti-join selects all tuples of A that are *not semantically linked* to tuples of B. The information of the second table will not appear in the result. R will have the same schema of A and containing all the tuples of A for which there is no tuple of B for which the predicate p is true.

Division the division select all tuples of A that have a corrisponding value in B, is necessary that the A tuples have ALL the corresponding value of B.

If our query is: *"Find all the student that have passed the exams of ALL courses of the first year"* the result will be like figures 5 and 6.



PassedExams

StudentID	CCourse
S1	C1
S1	C2
S1	C3
S1	C4
S1	C5
S1	C6
S2	C1
S2	C2
S3	C2
S4	C2
S4	C4
S4	C5

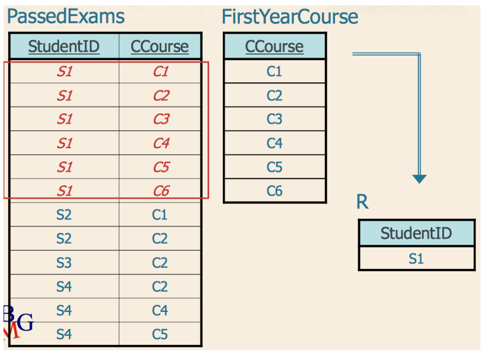
FirstYearsCourses

CCourse
C1

R

StudentID
S1
S2

Figure 5: Division 1st example



PassedExams

StudentID	CCourse
S1	C1
S1	C2
S1	C3
S1	C4
S1	C5
S1	C6
S2	C1
S2	C2
S3	C2
S4	C2
S4	C4
S4	C5

FirstYearCourse

CCourse
C1
C2
C3
C4
C5
C6

R

StudentID
S1

Figure 6: Division 2nd example