Introduction to Relational Databases

- Bachelor Computer Science, Lille 1 University
- Lecture 4/12
- Topic: Introduction to SQL as a query language
  - basic queries (i.e. without sub-queries)
  - inserting data into tables

Basic Queries in SQL

SQL as a query language

- SQL queries are declarative
  - The user specifies which information he wants, but not how to extract it from the data
- The DBMS’s query optimizer translates the queries into an internal, procedural representation
- The programmer concentrates on legibility, not on efficiency
- This is a key point in relational databases !

SQL queries

Syntax:

l `select` *AttrExpr* {, *AttrExpr*}
▯ `from` *Table* {, *Table*}
▯ [`where` *Condition*]

Its three parts are called clauses.

Meaning:

- FROM: Make the Cartesian product of the tables in the `from` clause,
- WHERE: Only consider those lines satisfying the `where` clause
- SELECT: For each line, evaluate the expression in the select clause, and return

Algebraic interpretation of SQL queries

- Generic query :

```
select Table1.Attribute1 , … , TableN.AttributeN
 from Table1, …, TableM
 where Condition
```

- Corresponds to the relational algebra query:

$$\pi_{Table1.Attribute1,\ldots,TableN.AttributeN}\Big(\\ \quad \sigma_{Condition}(Table1 \times \ldots \times TableM) \\ \Big)$$

Example:
managing university exams

SID	NAME	CITY	MAJOR
123	Pierre	Lyon	Inf
415	Celine	Lille	Inf
702	Estelle	Paris	Log

SID	CLASS	DATE	GRADE
123	1	7-9-13	10
123	2	8-1-13	8
702	2	7-9-13	5

CID	TITLE	TEACHER
1	maths	Leguichet
2	CS	Duchat

Basic queries

```
select  *
from Student
```

SID	NAME	CITY	MAJOR
123	Pierre	Lyon	Inf
415	Celine	Lille	Inf
702	Estelle	Paris	Log

Basic queries

Student

Sid	Name	City	Major

```
select Name
from Student
where Major = ’Log’
```

Algebraic interpretation

(without duplicates)

$$\Pi_{\text{Name}}\ \sigma_{\text{Major='Log'}}\ \text{Student}$$

Syntax of select clause

```
select  *

select  Name, City

select  distinct  City

select  City as HomeTown

select  Grade * 0.05 as Bonus

select  sum(Income)
```

Syntax of from clause

```
from  Student

from Student as X

from  Student, Exam

from  Student natural join Exam


from  Student join Exam
     on Student.Sid=Exam.Sid
```

Syntax of where clause

- Boolean expressions with simple predicates (as in algebra's σ clause : and, or, not)
-
- Some extra predicates:
  -
  - **between: value within range**
    - **Date between '1-1-15' and '31-12-15'**
  - **like: pattern matching on strings**

```
Major like 'Lo%'
 Matr like 'MI_777_8%'
```

Conjunction of predicates

- Extract computer science students from Lyon:

```
select *
from Student
where Major = 'Inf' and
      City = 'Lyon'
```

- Result:

Sid	Name	City	Major
123	Pierre	Lyon	Inf

Disjunction of predicates

- Extract students from Lyon or from Lille:

```
select *
from Student
where City = ’Lyon’ or
      City = ’Lille’
```

- Result:

Sid	Name	City	Major
123	Pierre	Lyon	Inf
415	Celine	Lille	Inf

Boolean expressions

- Extract students from Paris, that study computer science or logistics:

```
Select *
from Student
where City = ’Paris’ and
      ( Major = ’Inf’ or Major = ’Log’)
```

▯

- Result:

Sid	Name	City	Major
702	Estelle	Paris	Log

Like operator

- Extract students with a name having an ‘i’ at its second position, and as last two positions ‘ot’:

```
Select *
from Student
where Name like ‘_i%ot’
```

- Result:

Sid	Name	City	Major
123	Pierrot	Lyon	Inf

Duplicates

- In relational algebra, the result of queries do not contain any duplicates
- In SQL however, the tables returned by queries may contain identical lines
- Duplicates can be eliminated by the keyword `distinct`
-

Duplicates

```
select
distinct Major
from Student
```

Major
Inf
Log

```
select Major
from Student
```

Major
Inf
Inf
Log

Dealing with null values

- Null values represent three distinct situations:
  - a value does not apply
  - a value applies, but is unknown
  - Unknown whether value applies or not

- SQL-89 uses a two-valued logic
  - A comparison with *null* returns FALSE

- SQL-2 uses a three-valued logic: True, False, Unknown
  - A comparison with *null* returns UNKNOWN
  - Predicate to test null values:
    - *Attribute* **`is`** [**`not`**] **`null`**

Queries with NULL values

```
select *
from Student
where City  is   [not]   null
```

if `City` has the value *null* then
`(City = ’Milano’)` has value Unknown

Predicates and NULL values

- 3 valued logics (T,F,U)

T and U = U
T or U = T

F and U = F
F or U = U

U and U = U
U or U = U
not U = U

- P =

```
(City is not null) and
 (Major like ’CS%’)
```

City	Major
Lyon	CS
Lyon	NULL
NULL	CS
Lyon	Log

P	TUPLE SELECTED
T	yes
U	no
F	no
F	no

Queries and NULL values

```
select *
from Student
where Major = ’CS’ or
      Major <> ’CS’
```

Is equivalent to:

```
select *
from Student
where Major is not null
```

Basic queries with two tables

```
select Name
from Student, Exam
where Student.Sid = Exam.Sid
  and Major like ’Lo%’ and Grade >= 15
```

Name
Pierre

Simple query with 3 tables

- Names of students in “Maths” class with at least one 18

|

```
select Name
from Student, Exam, Class
where  Student.Sid = Exam.Sid
  and Class.Cid = Exam.Cid
  and Title like ’Mat%' and Grade = 18
```

- Algebra :

$\pi_{\text{Name}}\ \sigma_{\text{(Title like ’Mat\%') and (Grade =18)}}$ (Student * Exam * Class)

Equivalent queries, different syntax

```
select Name
from Student, Exam
where
     Student.Sid = Exam.Sid
     and Major like 'Mat%' and Grade= 18
```

```
select Name
from Student join Exam
     on Student.Sid = Exam.Sid
where
     Major like 'Mat%'and Grade= 18
```

Syntax for joins in SQL-2

- SQL-2 introduced the following Syntax for joins, in the `from clause`:

```
select AttrExpr {, AttrExpr}
from
          Table { [ JoinType] join Table on Conditions}

[ where OtherConditions ]
```

- JoinType can be ***inner*, *right*, *left* or , *full***

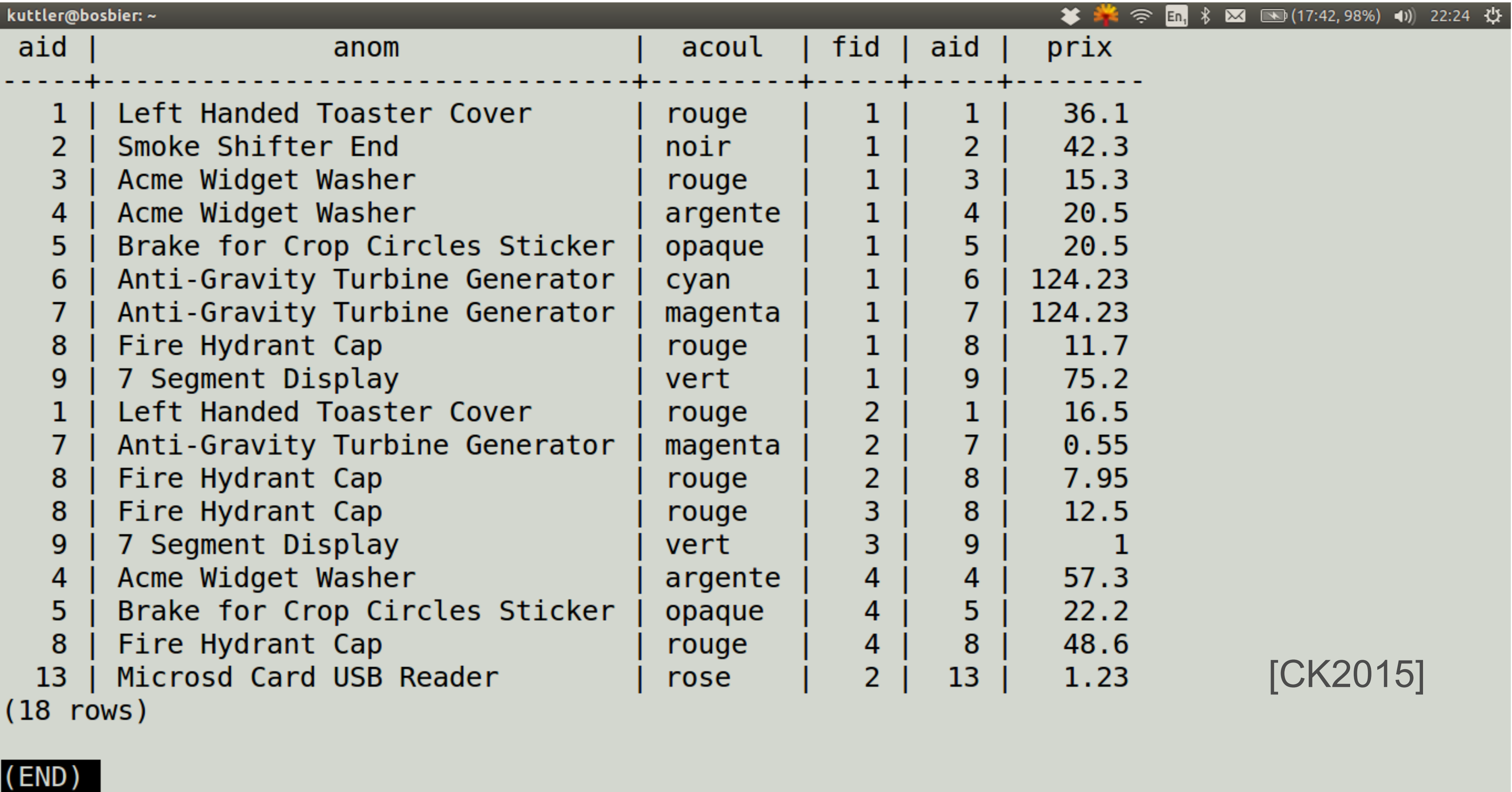
Join types

- `inner`: the usual join
  - by default, the keyword inner is **omitted**
  - the condition is an equality test between attribute values, and their partner.
  - The result only contains tuples *with* partners

- `right, left, full`:
  - **three different outer joins**
  - the keyword **must** appear
  - the external join finds everything the inner finds, and in addition, ALSO finds tuples without partner
  - Missing information is filled with NULL

Inner join

```
select * from Articles join  Catalogue
              on(Articles.aid=Catalogue.aid)
```



```
 aid |               anom               |  acoul   | fid | aid |  prix
-----+----------------------------------+----------+-----+-----+--------
   1 | Left Handed Toaster Cover        | rouge    |   1 |   1 |   36.1
   2 | Smoke Shifter End                | noir     |   1 |   2 |   42.3
   3 | Acme Widget Washer               | rouge    |   1 |   3 |   15.3
   4 | Acme Widget Washer               | argente  |   1 |   4 |   20.5
   5 | Brake for Crop Circles Sticker   | opaque   |   1 |   5 |   20.5
   6 | Anti-Gravity Turbine Generator   | cyan     |   1 |   6 | 124.23
   7 | Anti-Gravity Turbine Generator   | magenta  |   1 |   7 | 124.23
   8 | Fire Hydrant Cap                 | rouge    |   1 |   8 |   11.7
   9 | 7 Segment Display                | vert     |   1 |   9 |   75.2
   1 | Left Handed Toaster Cover        | rouge    |   2 |   1 |   16.5
   7 | Anti-Gravity Turbine Generator   | magenta  |   2 |   7 |   0.55
   8 | Fire Hydrant Cap                 | rouge    |   2 |   8 |   7.95
   8 | Fire Hydrant Cap                 | rouge    |   3 |   8 |   12.5
   9 | 7 Segment Display                | vert     |   3 |   9 |      1
   4 | Acme Widget Washer               | argente  |   4 |   4 |   57.3
   5 | Brake for Crop Circles Sticker   | opaque   |   4 |   5 |   22.2
   8 | Fire Hydrant Cap                 | rouge    |   4 |   8 |   48.6
  13 | Microsd Card USB Reader          | rose     |   2 |  13 |   1.23
(18 rows)

(END)
```

[CK2015]

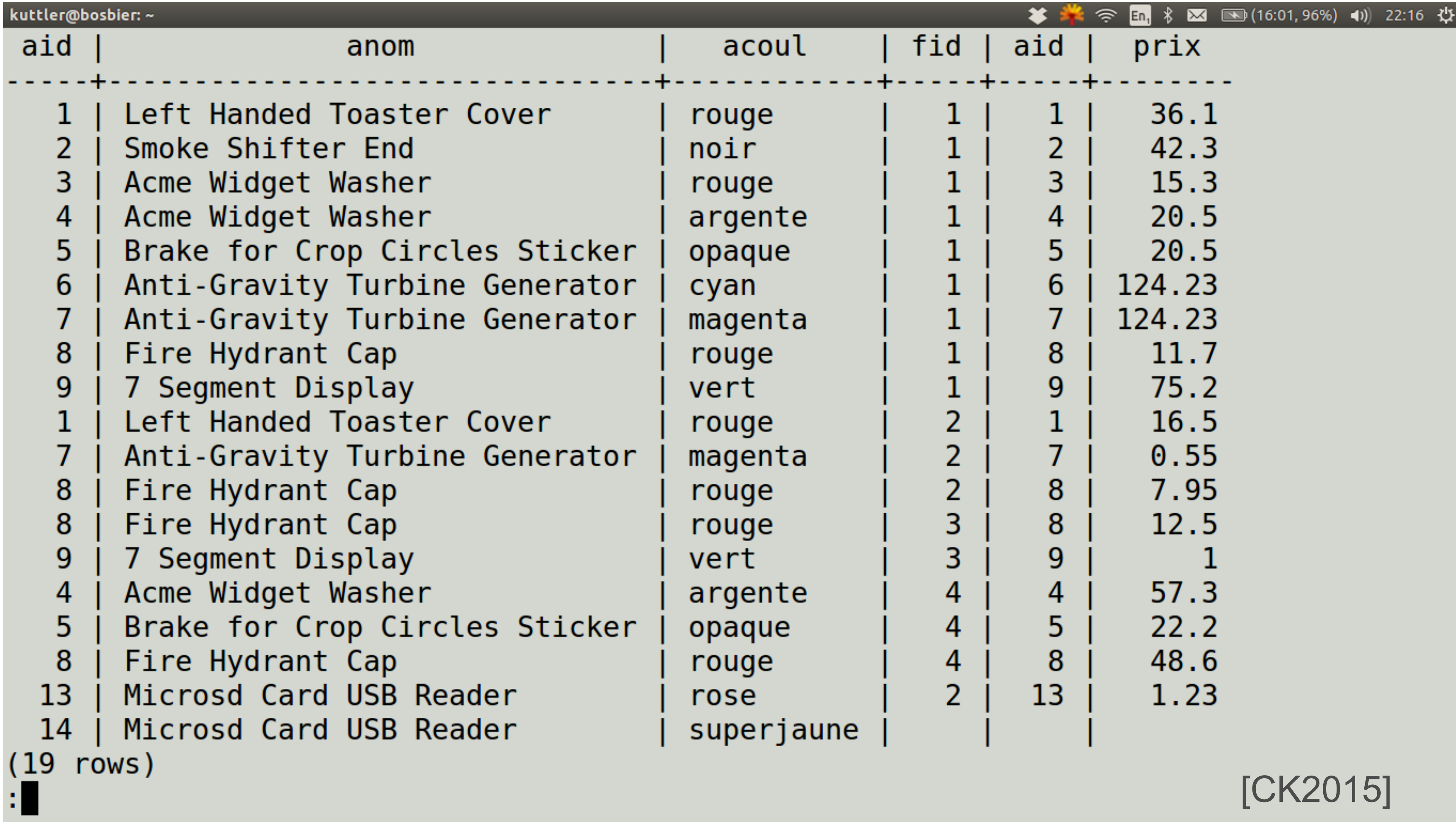
Left join

- Trouver les articles avec leurs prix et fournisseurs, incluant les articles non fournissables:

```
select *
from Articles left join  Catalogue
       on(Articles.aid=Catalogue.aid)
```

[CK2015]

```
select * from Articles left join
Catalogue on(Articles.aid=Catalogue.aid)
```



```
 aid |              anom              |   acoul    | fid | aid |  prix
-----+--------------------------------+------------+-----+-----+--------
   1 | Left Handed Toaster Cover      | rouge      |   1 |   1 |   36.1
   2 | Smoke Shifter End              | noir       |   1 |   2 |   42.3
   3 | Acme Widget Washer             | rouge      |   1 |   3 |   15.3
   4 | Acme Widget Washer             | argente    |   1 |   4 |   20.5
   5 | Brake for Crop Circles Sticker | opaque     |   1 |   5 |   20.5
   6 | Anti-Gravity Turbine Generator | cyan       |   1 |   6 | 124.23
   7 | Anti-Gravity Turbine Generator | magenta    |   1 |   7 | 124.23
   8 | Fire Hydrant Cap               | rouge      |   1 |   8 |   11.7
   9 | 7 Segment Display              | vert       |   1 |   9 |   75.2
   1 | Left Handed Toaster Cover      | rouge      |   2 |   1 |   16.5
   7 | Anti-Gravity Turbine Generator | magenta    |   2 |   7 |   0.55
   8 | Fire Hydrant Cap               | rouge      |   2 |   8 |   7.95
   8 | Fire Hydrant Cap               | rouge      |   3 |   8 |   12.5
   9 | 7 Segment Display              | vert       |   3 |   9 |      1
   4 | Acme Widget Washer             | argente    |   4 |   4 |   57.3
   5 | Brake for Crop Circles Sticker | opaque     |   4 |   5 |   22.2
   8 | Fire Hydrant Cap               | rouge      |   4 |   8 |   48.6
  13 | Microsd Card USB Reader        | rose       |   2 |  13 |   1.23
  14 | Microsd Card USB Reader        | superjaune |     |     |
(19 rows)

:
```

[CK2015]

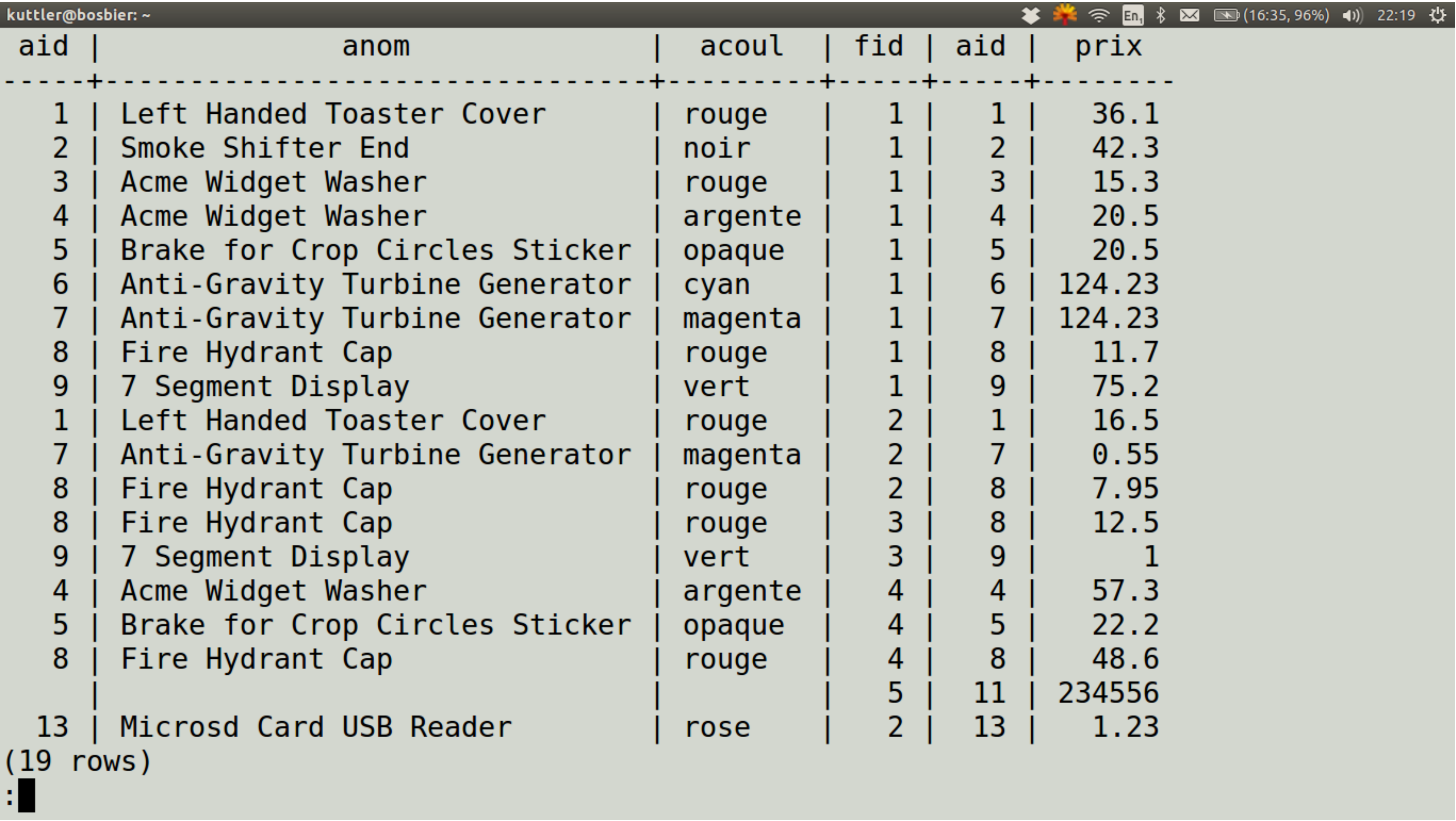
Right join

- Trouver les articles fournissables avec leurs prix et fournisseurs, incluant les données érronnées du catalogue (aid orphelin):

```
select *

 from Articles right join  Catalogue
                on
              (Articles.aid=Catalogue.aid)
```

[CK2015]



```
 aid |              anom              |  acoul  | fid | aid |  prix
-----+--------------------------------+---------+-----+-----+--------
   1 | Left Handed Toaster Cover      | rouge   |   1 |   1 |   36.1
   2 | Smoke Shifter End              | noir    |   1 |   2 |   42.3
   3 | Acme Widget Washer             | rouge   |   1 |   3 |   15.3
   4 | Acme Widget Washer             | argente |   1 |   4 |   20.5
   5 | Brake for Crop Circles Sticker | opaque  |   1 |   5 |   20.5
   6 | Anti-Gravity Turbine Generator | cyan    |   1 |   6 | 124.23
   7 | Anti-Gravity Turbine Generator | magenta |   1 |   7 | 124.23
   8 | Fire Hydrant Cap               | rouge   |   1 |   8 |   11.7
   9 | 7 Segment Display              | vert    |   1 |   9 |   75.2
   1 | Left Handed Toaster Cover      | rouge   |   2 |   1 |   16.5
   7 | Anti-Gravity Turbine Generator | magenta |   2 |   7 |   0.55
   8 | Fire Hydrant Cap               | rouge   |   2 |   8 |   7.95
   8 | Fire Hydrant Cap               | rouge   |   3 |   8 |   12.5
   9 | 7 Segment Display              | vert    |   3 |   9 |      1
   4 | Acme Widget Washer             | argente |   4 |   4 |   57.3
   5 | Brake for Crop Circles Sticker | opaque  |   4 |   5 |   22.2
   8 | Fire Hydrant Cap               | rouge   |   4 |   8 |   48.6
     |                                |         |   5 |  11 | 234556
  13 | Microsd Card USB Reader        | rose    |   2 |  13 |   1.23
(19 rows)

:
```

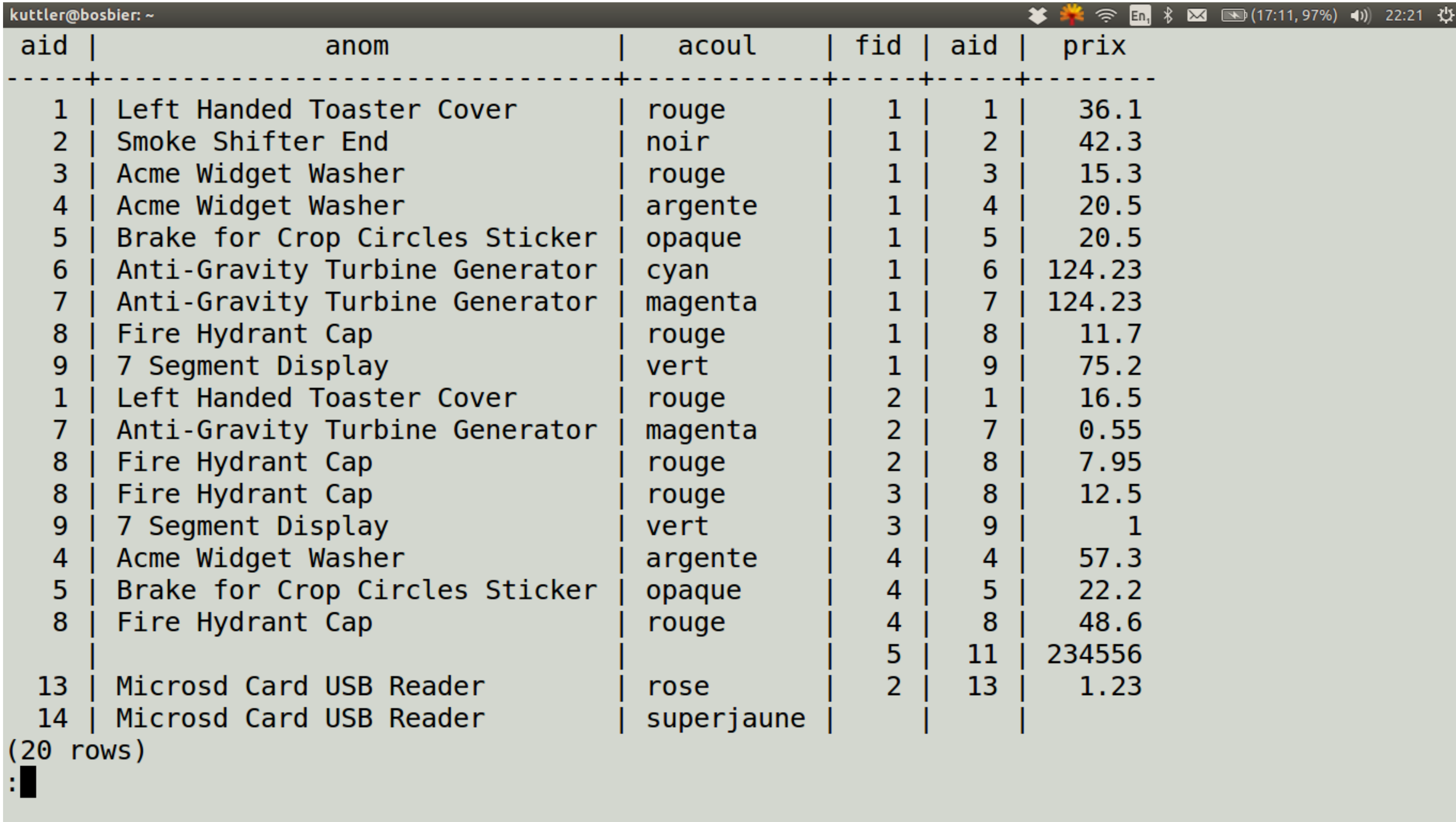
[CK2015]

Full join

- Trouver les articles avec leurs prix et vendeurs, incluant les articles non fournissables, et les aids orphelins du catalogue:

```
select *
 from Articles full join  Catalogue
      on
      Articles.aid=Catalogue.aid
```

[CK2015]



```
 aid |              anom              |   acoul    | fid | aid |  prix
-----+--------------------------------+------------+-----+-----+--------
   1 | Left Handed Toaster Cover      | rouge      |   1 |   1 |   36.1
   2 | Smoke Shifter End              | noir       |   1 |   2 |   42.3
   3 | Acme Widget Washer             | rouge      |   1 |   3 |   15.3
   4 | Acme Widget Washer             | argente    |   1 |   4 |   20.5
   5 | Brake for Crop Circles Sticker | opaque     |   1 |   5 |   20.5
   6 | Anti-Gravity Turbine Generator | cyan       |   1 |   6 | 124.23
   7 | Anti-Gravity Turbine Generator | magenta    |   1 |   7 | 124.23
   8 | Fire Hydrant Cap               | rouge      |   1 |   8 |   11.7
   9 | 7 Segment Display              | vert       |   1 |   9 |   75.2
   1 | Left Handed Toaster Cover      | rouge      |   2 |   1 |   16.5
   7 | Anti-Gravity Turbine Generator | magenta    |   2 |   7 |   0.55
   8 | Fire Hydrant Cap               | rouge      |   2 |   8 |   7.95
   8 | Fire Hydrant Cap               | rouge      |   3 |   8 |   12.5
   9 | 7 Segment Display              | vert       |   3 |   9 |      1
   4 | Acme Widget Washer             | argente    |   4 |   4 |   57.3
   5 | Brake for Crop Circles Sticker | opaque     |   4 |   5 |   22.2
   8 | Fire Hydrant Cap               | rouge      |   4 |   8 |   48.6
     |                                |            |   5 |  11 | 234556
  13 | Microsd Card USB Reader        | rose       |   2 |  13 |   1.23
  14 | Microsd Card USB Reader        | superjaune |     |     |
(20 rows)
:
```

[CK2015]

Variables in SQL

Full syntax for variables

```
select
    AttrExpr [[ as ] Alias ] {, AttrExpr [[ as ] Alias ]
from
    Table [[ as ] Alias ] {, Table [[ as ] Alias ] }
[ where Condition ]
```

Two purposes:

- renaming the result in the select clause
- variables for relation names in the from clause

Basic queries with variables for relations names

Who are Giorgio’s employees?

Eid	Name	HiringDate	Income	MgrID
1	Piero	1-1-05	3 M	2
2	Giorgio	1-1-02	2,5 M	null
[illegible]	Giovanni	1-7-06	2 M	2

Who are Giorgio’s employees?

```
select  E.Name, E.MgrID, B.Sid, B.Name
from
      Employee as E, Employee as B
where
      E.MgrID = B.Eid
      and B.Name = ’Giorgio’
```

E.Name	E.MgrID	B.Eid	B.Name
Piero	2	2	Giorgio
Giovanni	2	2	Giorgio

Modification commands

Modification commands in SQL

- Operations
  - **`insert`** `add new lines`
  - **`delete`** `remove existing lines`
  - **`update`** modify values of attributes
- set-oriented: all operations can apply to a set of tuples
- The commands can contain a condition, that may access other tables

Insertion

- Syntax:
  insert into *TableName* [(*AttributeList*)]
  < **values** (*ValueList*) | *SelectSQL*>

- Using **values**:

```
insert into Student
values ('456878', 'Giorgio Rossi','Lyon','Log')
```

- Using a query:

```
insert into Chtis
   (select *
    from Student
     where City = 'Lille')
```

Insertion

- The order of attributes and values matters: positional notation – the first value is affected to the first attribute, etc
- If the *AttributeList* is omitted, then all attributes of the relation are considered, in the order in which they appear in the table's definition.
- If the *ValueList* does not contain all attributes of the relation, the remaining attributes are assigned the default value (if specified, otherwise NULL)

Insertion

- Using **`values with`** *AttributeList*:

```
insert into Student(Sid,Name,City,Major)
values ('456878', 'Antoine Bailleul',
        'Bergues', 'Log')
```

- Using a query with *AttributeList*:

```
    insert into Chtis(Sid,Name,City,Major)
    (select Sid, Name, City, Major
     from Student
     where City = 'Lille')
```

Deletion

- Syntax:

```
delete from TableName [ where Condition ]
```

- Delete the student with identifier 678678:

```
  delete from Student where Sid = '678678'
```

- Delete all students that haven’t taken any exam:

```
delete from Student
        where Sid not in
               (select Sid from Exam)
```

Deletion

- The **delete** command deletes all tuples satisfying the condition from the table.
- The command can lead to deletions in other tables, if there is a referential integrity constraint with **cascade** .
- When the **where** clause is ommited, the **delete** command deletes all tuples
  - Example: deleting all tuples from STUDENT (maintaining the table's schema):

    delete from Student
- The complete STUDENT table can be deleted (content and schema):
  - **drop table Student cascade**

Modifications

Syntax:

update *TableName*

set *Attribute* **=** < *Expression* | *SelectSQL* | **null** | **default** >

{ **,** *Attribute* **=** < *Expression* | *SelectSQL* | **null** | **default** >}

[**where** *Condition*]

Examples:

```
update Exam
set Grade = 20
where Date = ‘1-4-15’
```

```
update Exam
set Grade = Grade + 1
where Sid = '787989'
```

Modifications

- Although the language is set-oriented, the order of command is very important

```
update Employee
   set Income = Income * 1.1
   where Income <= 30

update Employee
   set Income = Income * 1.15
   where Income > 30
```

- When the commands are written in this order, some employees can benefit from two increases! With the opposite order, this can can't happen.