The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2273-8

Second Workshop on Open Source and the Internet for Building Global Scientific Communities with Emphasis on Environmental Monitoring and Distributed Instrumentation

28 November - 16 December, 2011

Databases

C. Kavka
ESTECO Srl. Trieste
Italy

Databases

Carlos Kavka
Esteco SRL
Trieste, Italy

Second Workshop on Open Source and the Internet for Building Global Scientific Communities
The Abdus Salam ICTP, Trieste, Italy
28 November - 16 December 2011

www.esteco.com

Presentation outline

- Databases basic concepts
- Database design
- The SQL language
- Examples in MySQL and PHP
- MySQL tools
- References

Why databases?

- Why it is not convenient just to use a text file to store data?

name, **model**, **type**, **maxresolution**, **price**
sensor1, DS1822, temperature, 12, 1.61
sensor2, DS2450, adc, 16, 2.62
sensor3, DS1822, temperature, 12, 1.61
sensor4, DS1821, temperature, 8, 2.24
…

using for example comma separated fields

- Because it is very difficult to answer queries and almost impossible to administrate!

Databases

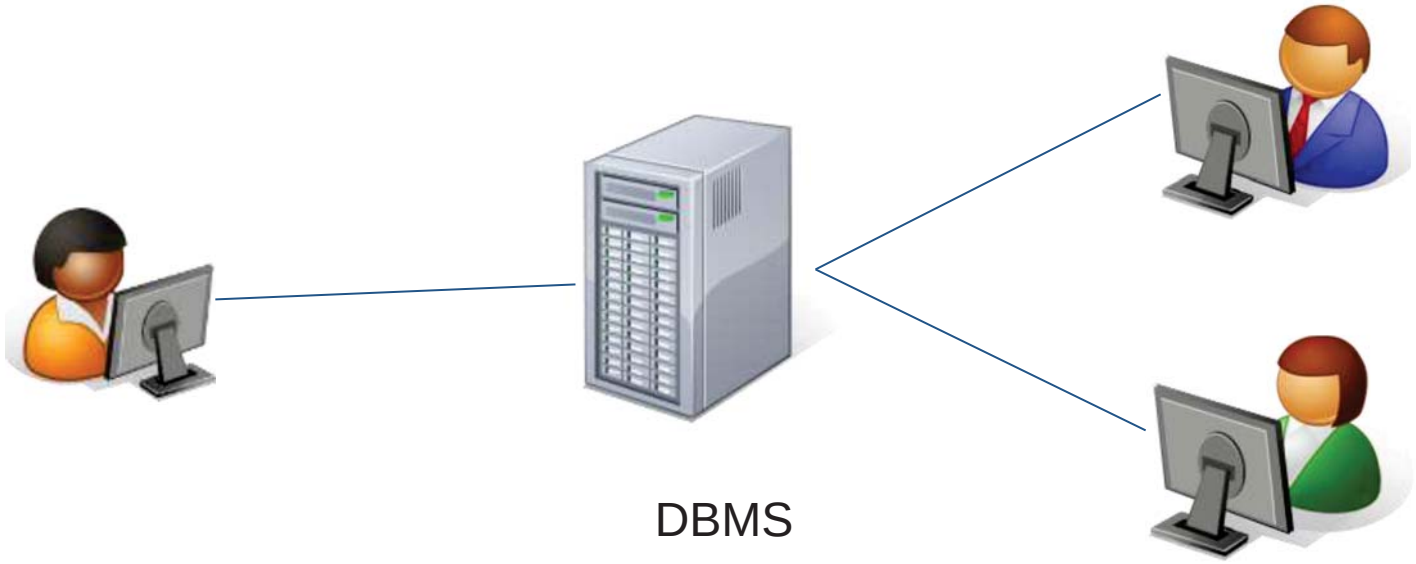
- A database is a collection of information that exists over a long period of time (1)

- The information is organized in such a way that it can easily be accessed, managed and updated

Data Base Management Systems

- A database management system (DBMS) is a piece of software that organizes the storage of data

- It controls the creation, maintenance and the use of the data by the end users.



DBMS

Database advantages

- Data can be handled with independence from the storage details
- Efficient access
- Reduced application development time
- Uniform data administration
- Security and access control
- Multiuser access (with concurrency control)
- Safety from system failures
- High performance and high availability

A bit of history

- The first commercial databases were developed in the 60's:
  - were evolutions of file systems
  - could not guarantee that data cannot be lost
  - did not support efficient access

- The first database models (hierarchical and network):
  - failed to provide a high level query language
  - force programmers to be concerned about the data storage structure

Relational model

- Developed after the famous Codd's paper:
  - Codd, E.F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM 13 (6): pages 377–387.
- The relational model provides significant changes to previous models:
  - queries are defined in a high level language
  - there is no need for the programmers to know the underlying data structure
- It is the most successful model used even today

Relational model

- Data is stored in tables (relations)
  - each table has a list of attributes (or columns)
  - each attribute has a domain (or type)
  - each relation contains a set of tuples (or rows)

each row is a relation

each column has a type

each column is an attribute

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25

Relational model

- Provides a query language (SQL), where the user specifies what answers a query should return, but not how the query is to be executed

select * from device where price < 5.00;

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
9	DS1821	temperature	8	2,24

Main relational databases in the market

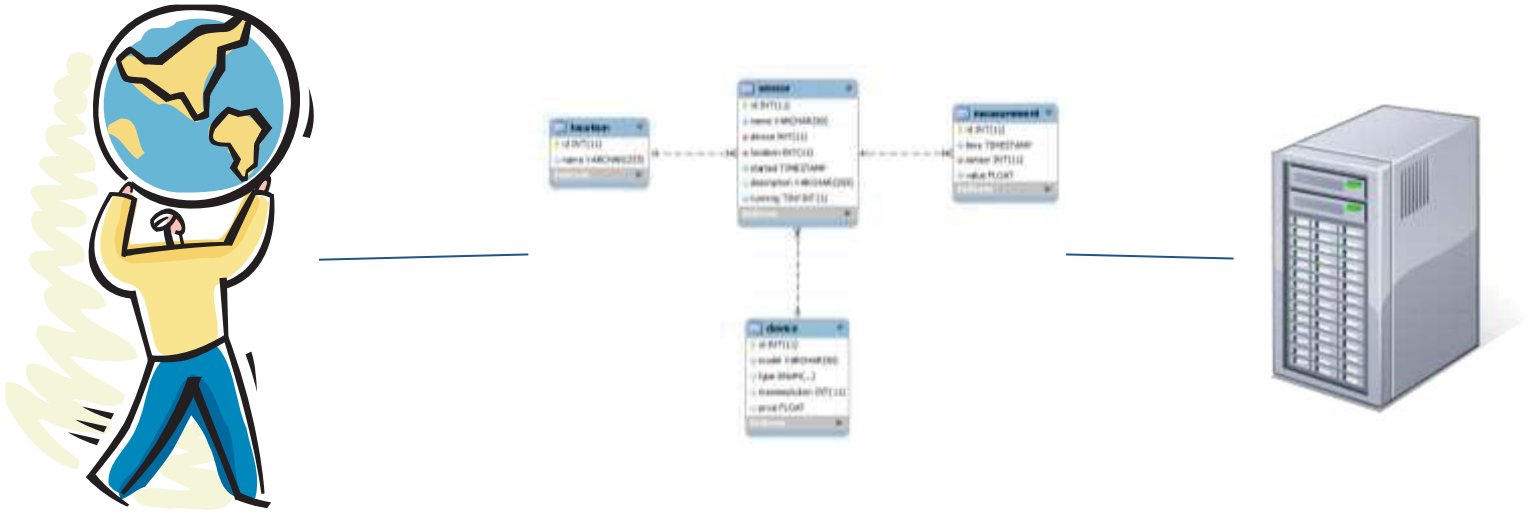
- MySQL (open source)
- Ingres (open source)
- PostgreSQL (open source)
- Oracle
- Microsoft SQL Server
- IBM DB2
- Sybase
- and many others

Why MySQL?

- open source
- Lightweight
- widely used
- easy to install and administrate
- community edition is free of cost
- largest open source database community
- thousands of enterprises use it for their mission-critical deployments

Database design

- The first step is to understand the real-world domain being modeled:
  - it should be specified by using a formal database design model (like for example the entity/relationship model)



real world domain

model

DBMS

Entity/Relationship model

- It helps to create the database schema (specification of how data is to be structured logically)

- The main components of the model are:
  - entities
  - relationships
  - attributes

- We will build an entity relationship model in steps with many examples.

Keys

- A set of attributes is a key for a relation if it is not allowed that two different rows agree on all values for the attributes of the key

name	type	maxresolution	price
DS1822	temperature	12	1,61
DS2450	adc	16	2,62
DS1821	temperature	8	2,24

the attribute (name) can be a key

clearly, the attribute (type) cannot be a key

Keys - examples

name	model	location
sensor1	DS1822	Adriatico
sensor2	DS2450	Galileo
sensor3	DS1822	Galileo
sensor4	DS1821	Leonardo

If name identifies univocally a sensor, it is a key

name	model	location
sensor1	DS1822	Adriatico
sensor2	DS2450	Galileo
sensor1	DS1822	Galileo
sensor4	DS1821	Leonardo

However, if we can have sensors with the same name in different locations, the set of attributes (name, location) can be the key

Database design

- We should carefully
  - avoid anomalies when performing database operations:
    - insertion
    - update
    - removal
  - follow normal forms in order to guarantee:
    - coherence
    - no redundancy
  - try to use only relationships “one to many”

Update anomaly

- Consider the following table with information about sensors:
  - What happens if we update the price of sensor1 only in the first row?

We will get an inconsistency.

name	model	type	location	maxresolution	price
sensor1	DS1822	temperature	Adriatico	12	1,61
sensor2	DS2450	adc	Galileo	16	2,62
sensor3	DS1822	temperature	Galileo	12	1,61
sensor4	DS1821	temperature	Leonardo	8	2,24

Update anomaly

- The problem can be solved by splitting the table

name	model	location
sensor1	1	Adriatico
sensor2	2	Galileo
sensor3	1	Galileo
sensor4	3	Leonardo

here we store information only about sensors

here we store information only about devices

id	name	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS2450	adc	16	2,62
3	DS1821	temperature	8	2,24

- Details on devices are not duplicated

Cancellation anomaly

- What happens if we remove sensor2 (based on a device used once)?

name	model	location	type	maxresolution	price
sensor1	DS1822	Adriatico	temperature	12	1,61
sensor2	DS2450	Galileo	adc	16	2,62
sensor3	DS1822	Galileo	temperature	12	1,61
sensor4	DS1821	Leonardo	temperature	8	2,24

we lose information about device DS2450!

Cancellation anomaly

- Again, the problem can be solved by splitting the table

name	model	location
sensor1	1	Adriatico
sensor2	2	Galileo
sensor3	1	Galileo
sensor4	3	Leonardo

even if we remove the last entry in first table, we can keep information on devices

id	name	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS2450	adc	16	2,62
3	DS1821	temperature	8	2,24

Insertion anomaly

- What happens if we add a new sensor with a previously used device?

name	model	location	type	maxresolution	price
sensor1	DS1822	Adriatico	temperature	12	1,61
sensor2	DS2450	Galileo	adc	16	2,62
sensor3	DS1822	Galileo	temperature	12	1,61
sensor4	DS1821	Leonardo	temperature	8	2,24
sensor5	DS1822	Adriatico	temperature	24	1,61

we can introduce inconsistencies!

Insertion anomaly

- Again, the problem can be solved by splitting the table

name	model	location
sensor1	1	Adriatico
sensor2	2	Galileo
sensor3	1	Galileo
sensor4	3	Leonardo
sensor5	1	Adriatico

no information is duplicated now

id	name	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS2450	adc	16	2,62
3	DS1821	temperature	8	2,24

First normal form

- At each row-column intersection, only one atomic value is allowed:

id	name	type	maxresolution	price	locations
1	DS1822	temperature	12	1,61	Adriatico
2	DS2450	adc	16	2,62	Adriatico, Galileo, Leonardo
3	DS1821	temperature	8	2,24	Leonardo, Galileo

wrong

- No repeating group is permitted in a table:

id	name	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS2450	adc	16	2,62
3	DS1821	temperature	8	2,24
4	DS1821	temperature	8	2,24

wrong

Second normal form

- It concerns only tables where the primary key consists of many columns.
  - Every non-key column must depend on the entire primary key

name	model	location	address
sensor1	1	Adriatico	Via Grignano, 9
sensor2	2	Galileo	via Beirut, 7
sensor1	1	Galileo	via Beirut, 7
sensor4	3	Leonardo	Strada Costiera, 11

the problem is that address depends only on location, not on the whole key (name, location)

Third normal form

- No non-key column depends on another non-key column

id	name	type	maxresolution	price	discount	total
1	DS1822	temperature	12	1,61	0,20	1,41
2	DS2450	adc	16	2,62	0,10	2,52
3	DS1821	temperature	8	2,24	0,25	1,99

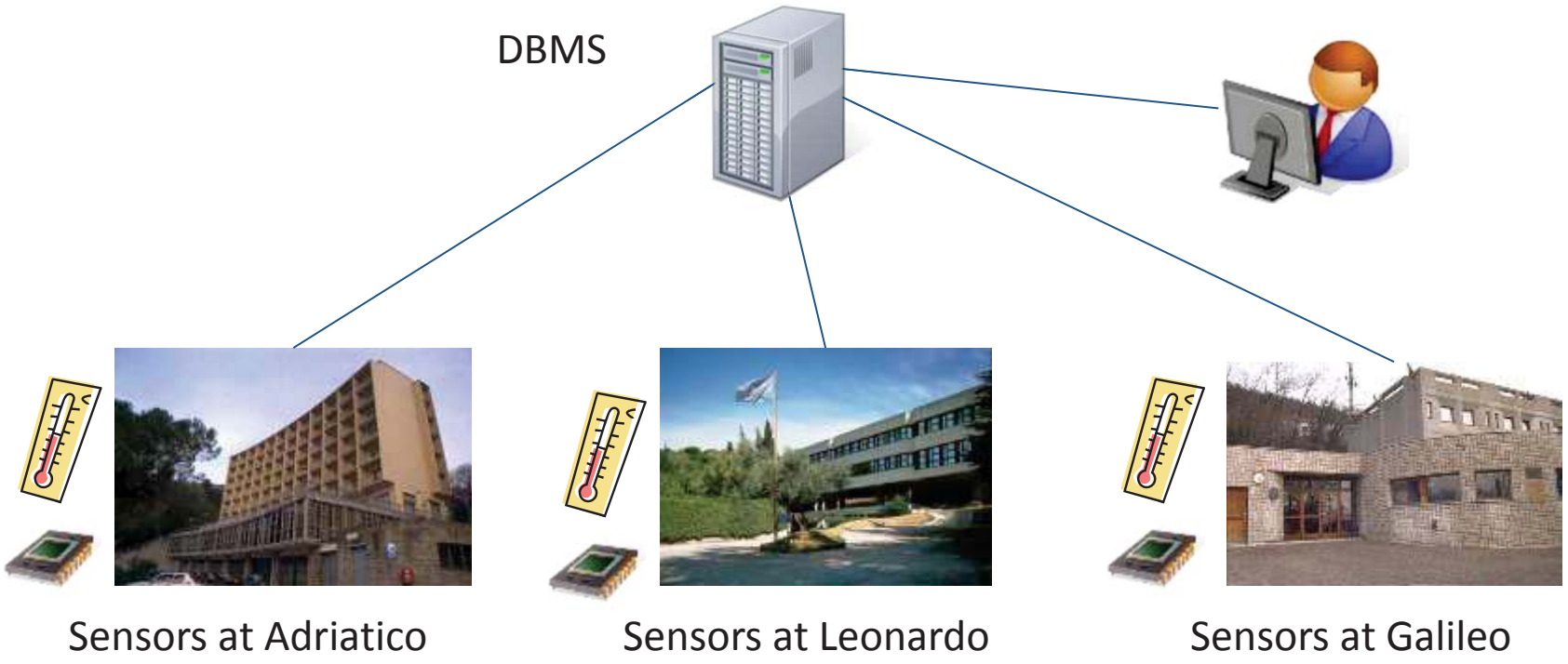
the problem is that the total price can be computed based on price and discount (non key columns)

Relationships

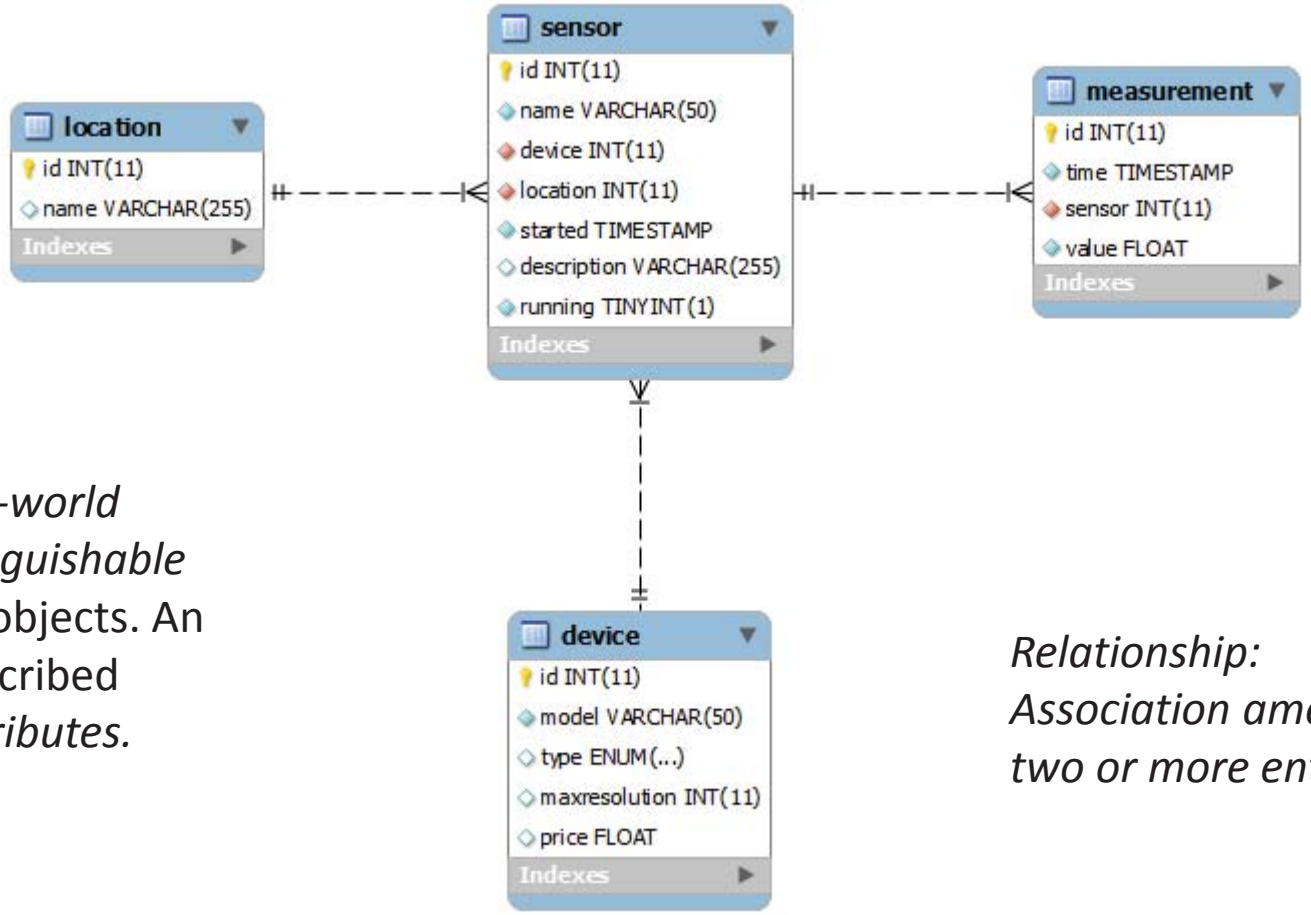
- There exist three types:
  - **one to one**: each row on a table relates to one row in the other table (both tables can be joined in one)
  - **many to many**: many rows in one table relates to many rows in other table (usually convenient to add a new table to split this relationship)
  - **one to many**: one row in one table relates to many rows in the other table (preferred)

Example

- We will define an example to be used in these lectures and also in the laboratory.
  - 4 tables: devices, locations, sensors, measurements



Entity/Relationship diagram for our example



Entity: Real-world object distinguishable from other objects. An entity is described through *attributes.*

Relationship: Association among two or more entities.

Table DEVICE

- This table keeps information on hardware devices

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

Table LOCATION

- This table keeps information on locations

LOCATION	
id	**name**
1	Adriatico
2	Galileo
3	Leonardo

Table SENSOR

- This table keeps information on the sensors used in experiments, which are based on a specific device and are installed on a particular location

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
5	large-exp1	9	2	2009-06-12 23:00:31	accelerator	0
6	large-exp2	3	1	2009-06-12 23:00:22	accelerator	0
7	small-exp1	6	3	2009-04-28 11:50:49	tiny analysis	1

Table MEASUREMENT

- This table keeps information on measurements performed with a specific sensor at a specific time

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
7	2009-09-04 16:05:50	1	23,5
8	2009-09-04 16:25:59	1	23,5
9	2009-09-04 16:36:10	1	23,6
10	2009-09-04 16:41:54	4	125,1
11	2009-09-04 16:43:55	7	12,8
12	2009-09-04 16:45:23	1	24,2
13	2009-09-04 16:46:35	1	24,3
14	2009-09-04 16:47:16	4	109,2

Relations

- Relations between SENSOR, DEVICE and LOCATION:

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

Relations

- Relations between SENSOR and MEASUREMENT

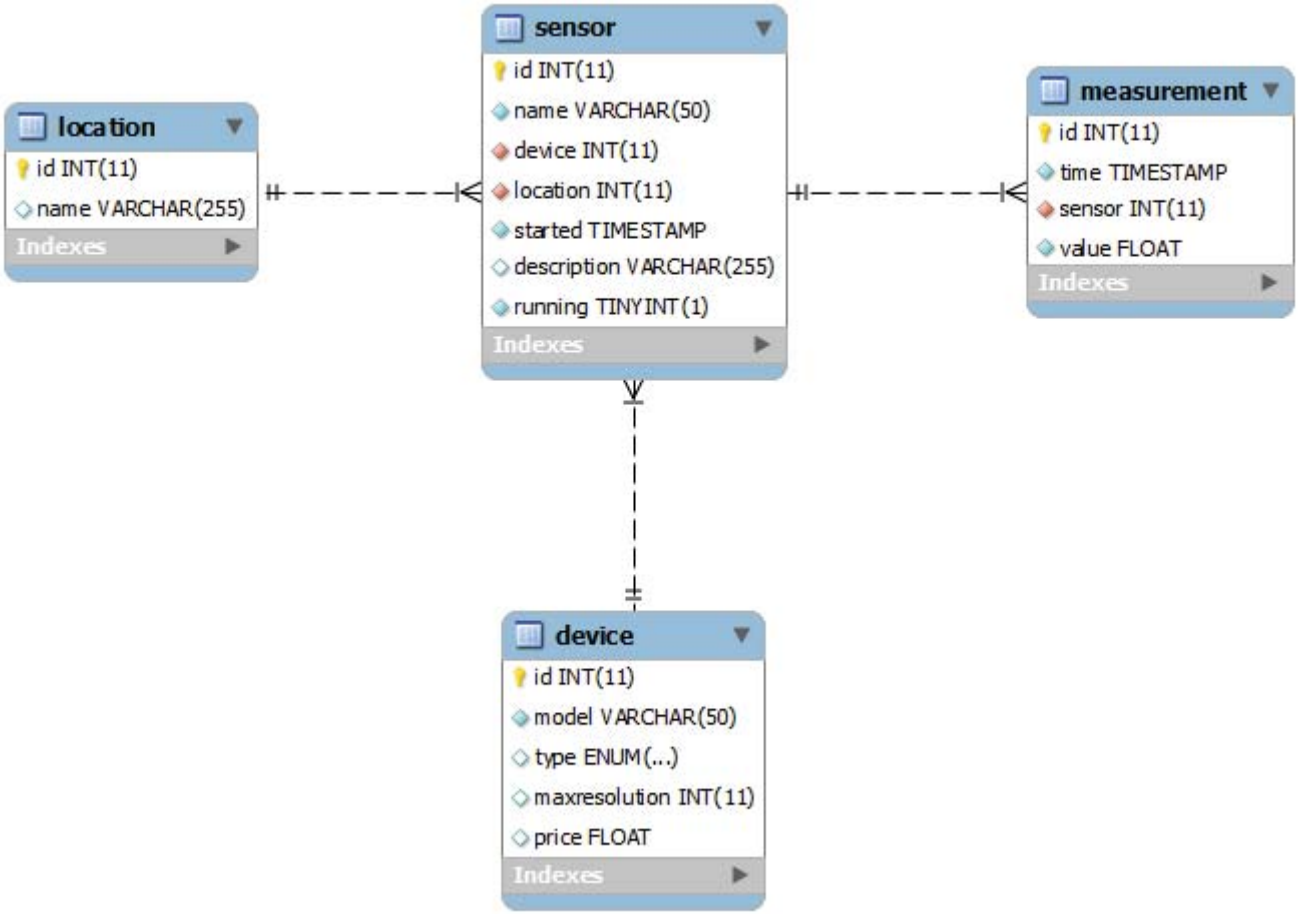
SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
...	...	...	...	...	...	...

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
...	...	...	...

Entity/Relationship diagram (revisited)

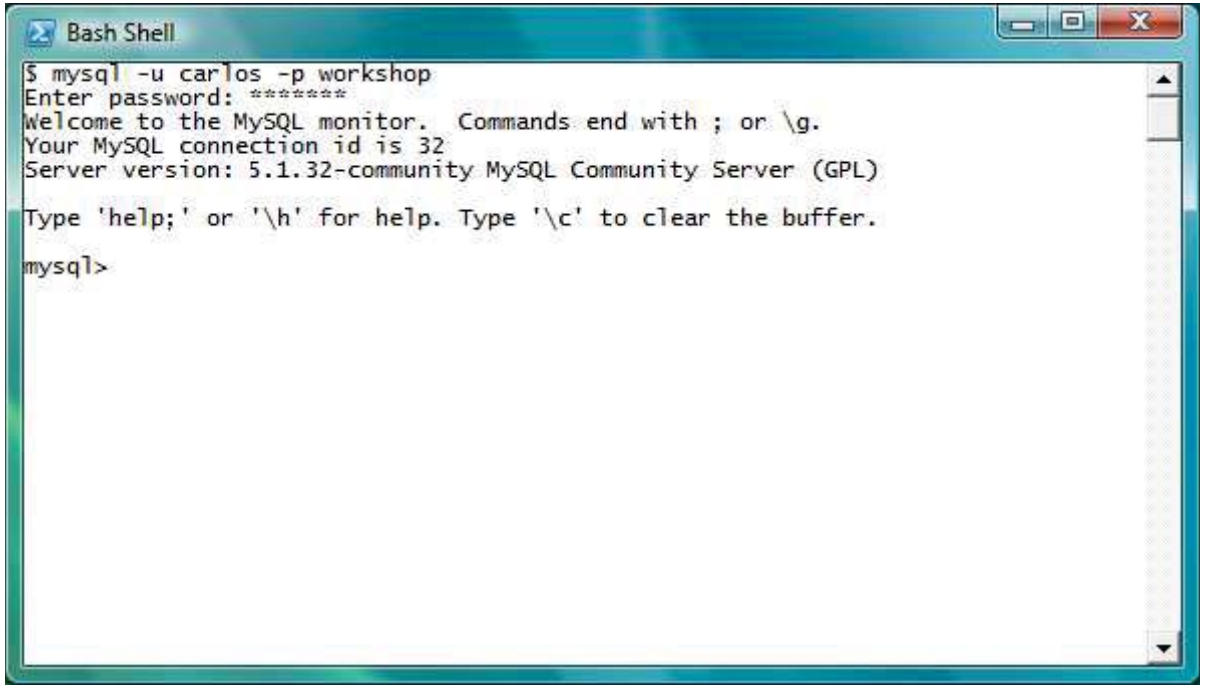


MySQL – database connection

- mysql client:

ask for password

mysql [-u <user>] [-p] [-h host] [-P port] [database]



MySQL – database creation and use

- Create database

CREATE DATABASE <database name>

- Use database:

USE <database name>

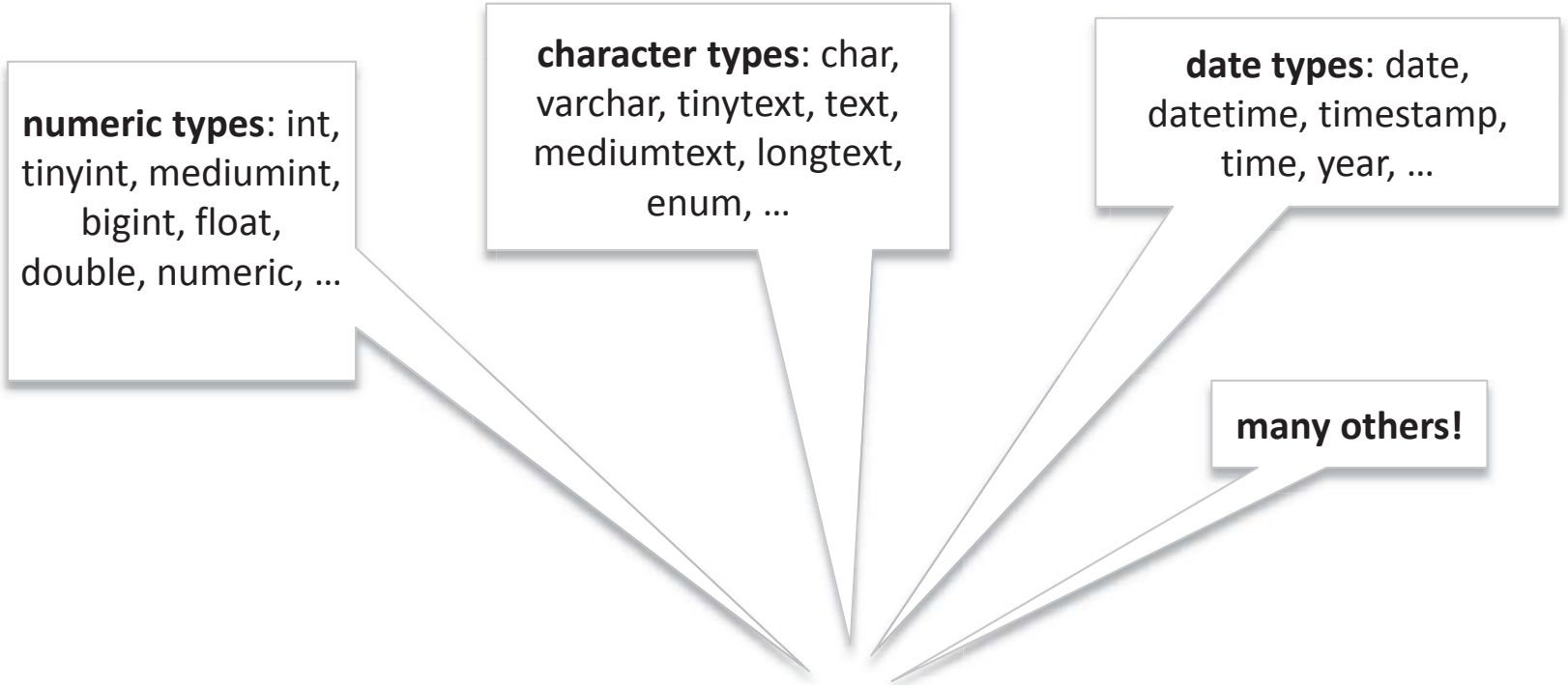
```
mysql> CREATE DATABASE workshop;
Query OK, 1 row affected (0.00 sec)

mysql> USE workshop;
Database changed
```

MySQL – table creation command

- A (very) simplified version of the create command looks like:

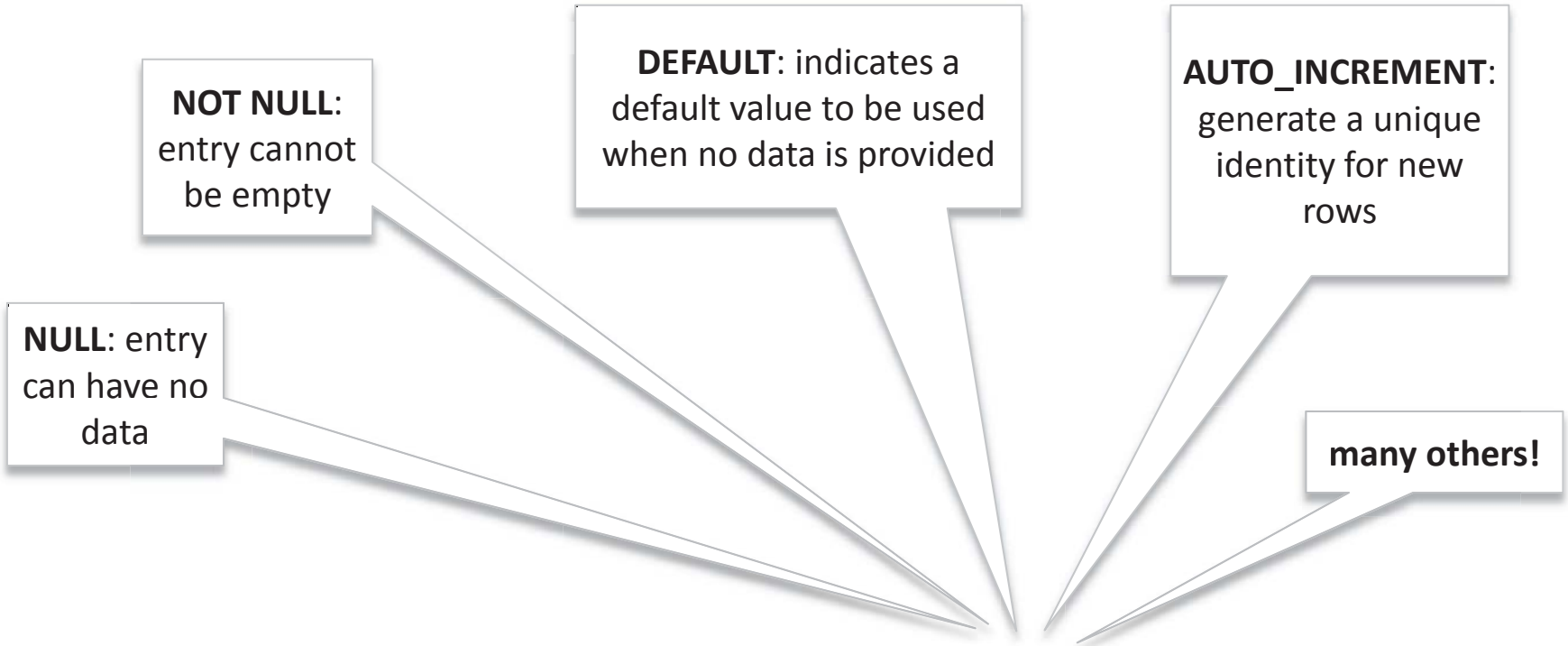
```
CREATE TABLE table ( column_name type [modifiers] , …
                    [ INDEX name (column_name, …), …],
                    [ PRIMARY KEY name (column_name, …), …]
                    [ FOREIGN KEY (index) [reference definition], …]);
```


MySQL – basic types




```
CREATE TABLE table ( column_name type [modifiers] , …
                    [ INDEX name (column_name, …), …],
                    [ PRIMARY KEY name (column_name, …), …]
                    [ FOREIGN KEY (index) [reference definition], …]);
```

MySQL – attribute modifiers




```
CREATE TABLE table ( column_name type [modifiers] , …
                    [ INDEX name (column_name, …), …],
                    [ PRIMARY KEY name (column_name, …), …]
                    [ FOREIGN KEY (index) [reference definition], …]);
```

MySQL – attribute modifiers

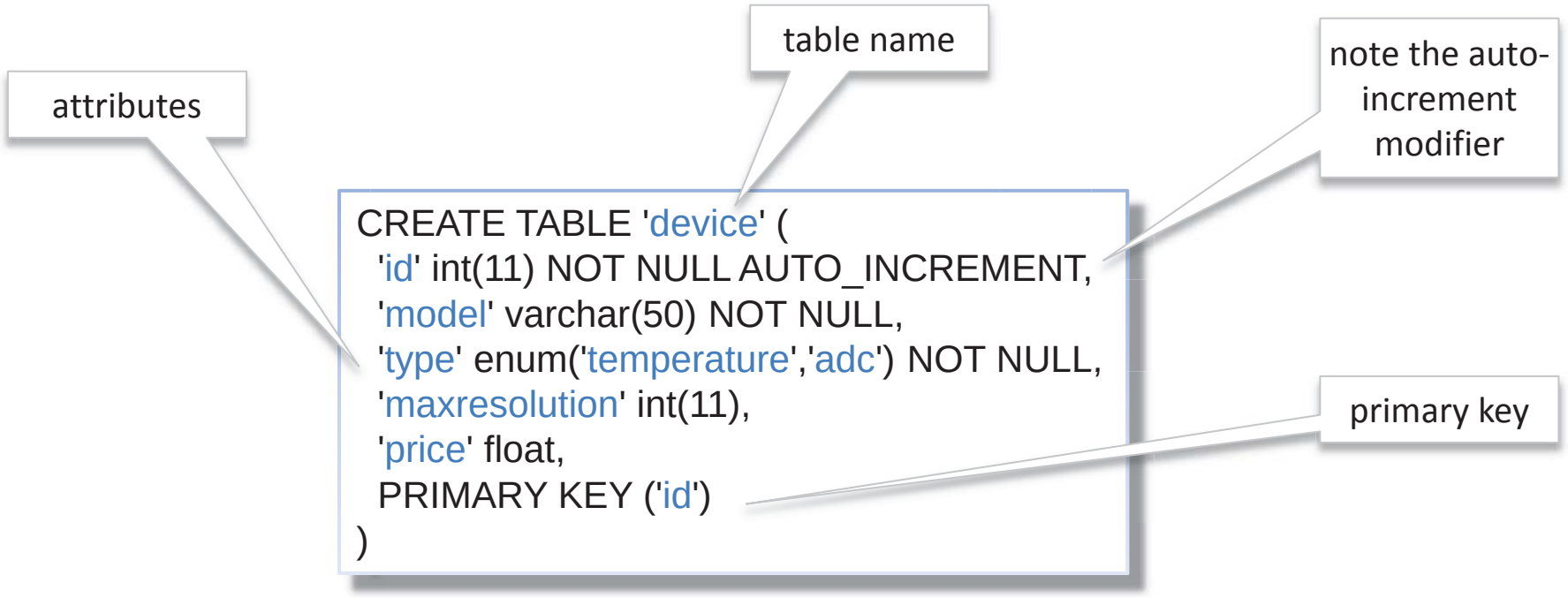
PRIMARY KEY: define a column (or set of columns) from which all other column depend

```
CREATE TABLE table ( column_name type [modifiers] , …
                   [ INDEX name (column_name, …), …],
                   [ PRIMARY KEY name (column_name, …), …]
                   [ FOREIGN KEY (index) [reference definition], …]);
```

INDEX: create an index for a column (or set of columns) to allow efficient queries on it

FOREIGN KEY: indicates that a local index corresponds to a key on a different table, and what to do when a row is removed or updated

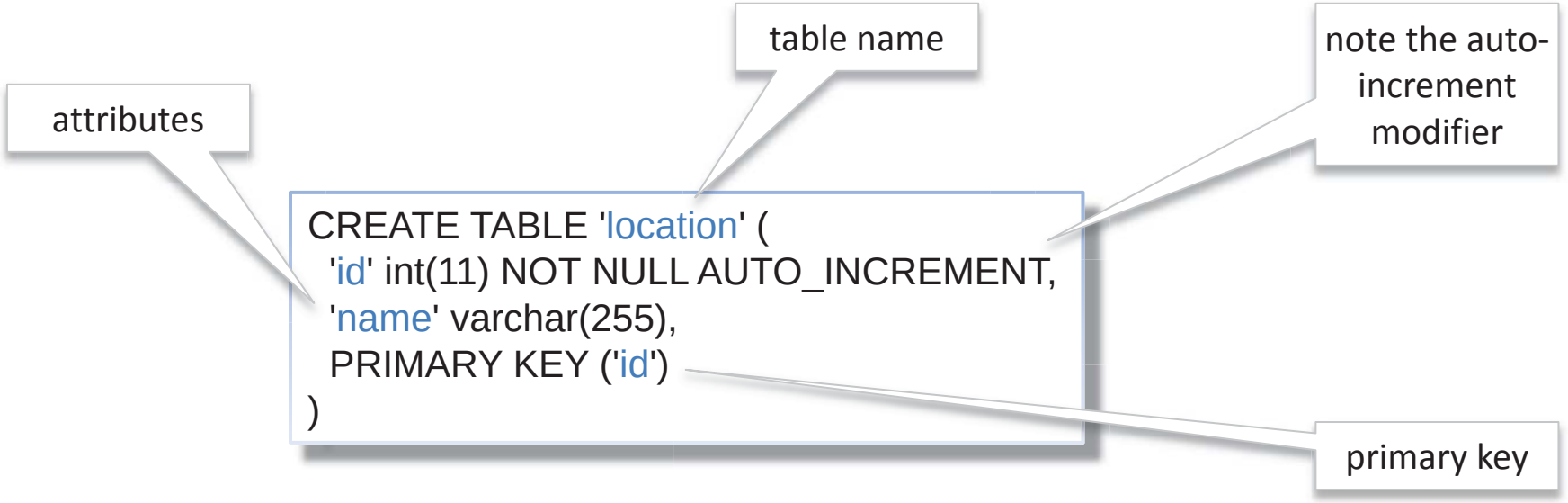
Example – table device



DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

Example – table location



LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

Example – table sensor

- Sensor has foreign keys for device and location:

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

Example – table sensor

```
CREATE TABLE 'sensor' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'name' varchar(50) NOT NULL,
  'device' int(11) NOT NULL,
  'location' int(11) NOT NULL,
  'started' timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  'description' varchar(255),
  'running' tinyint(1) NOT NULL DEFAULT '1',
  PRIMARY KEY ('id'),
  KEY 'device' ('device'),
  KEY 'location' ('location'),
  CONSTRAINT 'device'
    FOREIGN KEY ('device') REFERENCES 'device' ('id'),
  CONSTRAINT 'location'
    FOREIGN KEY ('location') REFERENCES 'location' ('id')
)
```

primary key

keys

two foreign keys definition

Example – table measurements

- Measurement has a foreign key to sensors

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
…	…	…	…

Example – table measurement

```
CREATE TABLE 'measurement' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'time' timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  'sensor' int(11) NOT NULL,
  'value' float NOT NULL,
  PRIMARY KEY ('id'),
  KEY 'sensor' ('sensor'),
  CONSTRAINT 'sensor'
    FOREIGN KEY ('sensor') REFERENCES 'sensor' ('id')
)
```

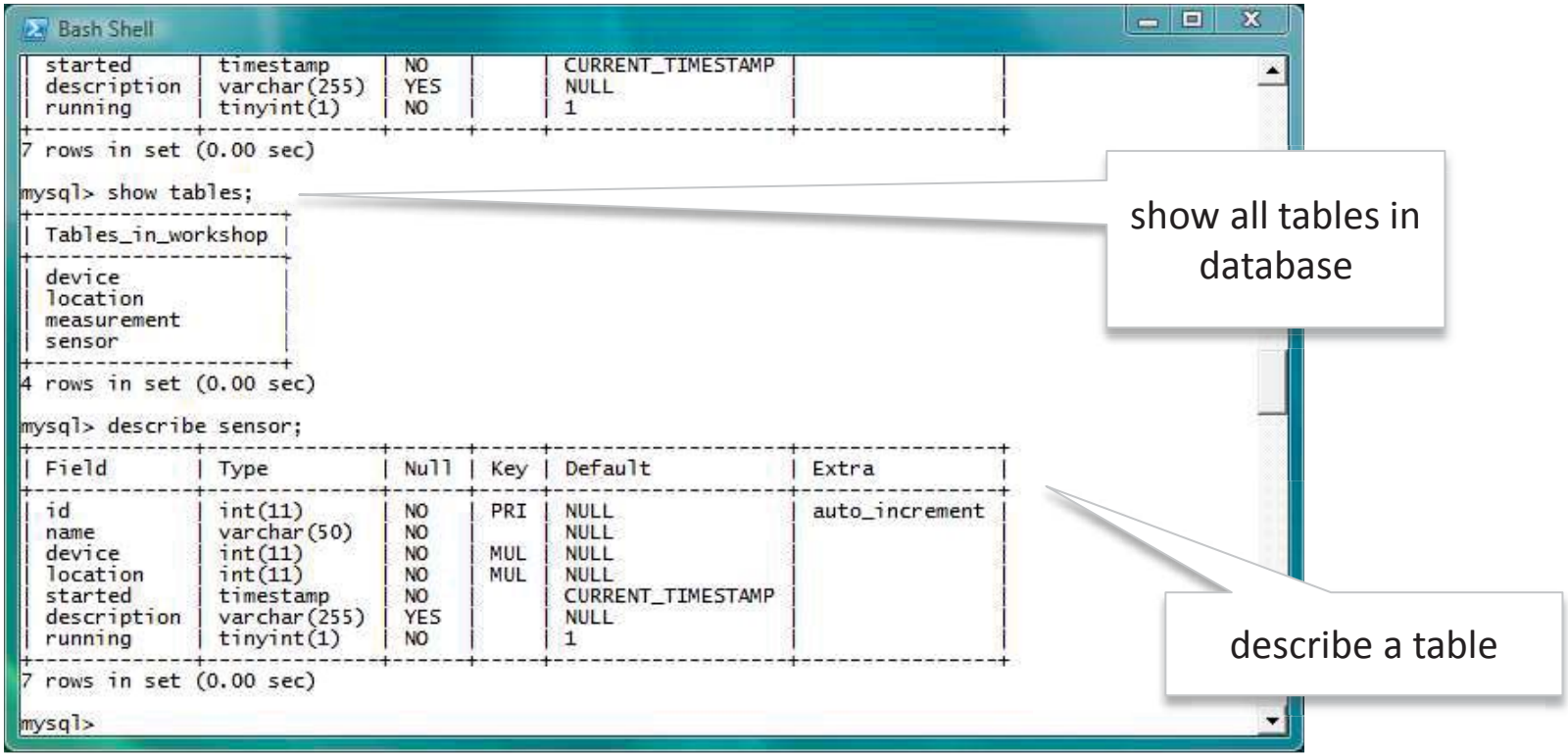
primary key

key

one foreign key definition

MySQL – explore tables

- Tables can be inspected with the commands show tables and describe:



MySQL - insertion

- insertion into table location

note that the id is not specified

```
INSERT INTO location (name) VALUES ('Adriatico'), ('Galileo'), ('Leonardo');
```

many rows can be inserted at the same time

LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

MySQL - insertion

- insertion into table device

the id is not specified

```
INSERT INTO device (model, type, maxresolution, price)
      VALUES ('DS1822','temperature',12,1.61);
INSERT INTO device (model, type, maxresolution, price)
      VALUES ('DS1825','temperature',12,1.72);
…
```

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

MySQL - insertion

- insertion into table measurement:

```
INSERT INTO measurement (sensor, value)
      VALUES (1,28.1);
INSERT INTO measurement (sensor, time, value)
      VALUES (4,' 2009-09-04 15:00:00', 99.4);
```

in the first command, the time is not specified

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
…	…	…	…
<auto>	<insertion time>	1	28,1
<auto>	2009-09-04 15:00:00	4	99,4

MySQL – remove

- remove rows:

```
DELETE FROM device WHERE id='4';
```

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

MySQL – remove

- remove rows:

```
DELETE FROM device WHERE type='adc';
```

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

MySQL – remove

- remove rows:

```
DELETE FROM device WHERE type='adc' AND price > 10;
```

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

MySQL - select

- The select command allows to retrieve data from the database.
- It is extremely powerful!
- A reduced syntax of the command is:

SELECT col1, col2, ... [INTO OUTFILE 'filename'] delimiters] FROM table1, table2, … [WHERE clause] [GROUP clause] [HAVING clause][ORDER BY clause]

MySQL – select (where clause)

select * from sensor;

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
5	large-exp1	9	2	2009-06-12 23:00:31	accelerator	0
6	large-exp2	3	1	2009-06-12 23:00:22	accelerator	0
7	small-exp1	6	3	2009-04-28 11:50:49	tiny analysis	1

SELECT name, started FROM sensor WHERE running=TRUE;

name	started
sensor1	2009-09-02 19:54:48
sensorhq	2009-09-02 19:54:48
data51	2009-09-01 13:00:21
room24	2009-08-21 14:44:09
small-exp1	2009-04-28 11:50:49

only two columns are returned

MySQL – select (where clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT model, price FROM device WHERE type='temperature' AND
maxresolution >= 12;
```

model	price
DS1822	1,61
DS1825	1,72
DS28EA00	2,25

note that the attributes maxresolution and type used for filtering were not selected to be returned

MySQL – select (distinct clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

SELECT DISTINCT type FROM device;

type
temperature
adc

without **distinct** we would have got 9 rows instead of 2

MySQL – select (between clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT * FROM device WHERE price BETWEEN 10 AND 20;
```

id	model	type	maxresolution	price
4	MAX1220	adc	12	13,79
7	MAX11040	adc	24	13,45

only devices with price between 10 and 20 are returned

MySQL – select (in clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT * FROM device WHERE maxresolution IN (12, 24);
```

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45

note that by using **between** we would have got also row 3

MySQL – select (like clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT * FROM device WHERE model LIKE 'DS1%';
```

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

like performs pattern matching

MySQL – select (order by clause)

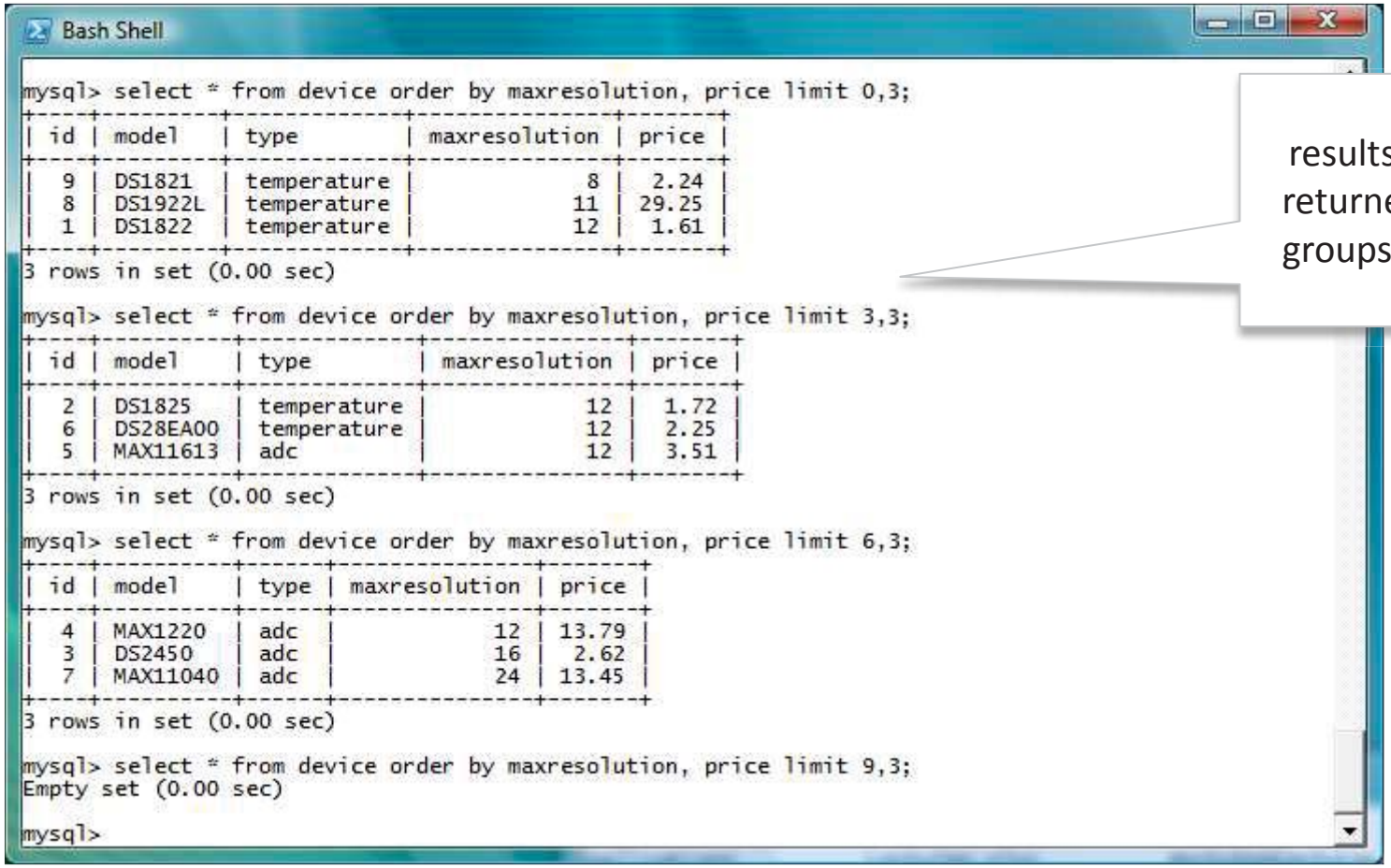
SELECT * FROM device ORDER BY maxresolution, price;

id	model	type	maxresolution	price
9	DS1821	temperature	8	2,24
8	DS1922L	temperature	11	29,25
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
6	DS28EA00	temperature	12	2,25
5	MAX11613	adc	12	3,51
4	MAX1220	adc	12	13,79
3	DS2450	adc	16	2,62
7	MAX11040	adc	24	13,45

note that results are ordered first by maxresolution and then by price

MySQL – select (limit clause)

- limiting the number of results



MySQL – select (group clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

SELECT maxresolution, COUNT(*) FROM device GROUP BY maxresolution;

maxresolution	count(*)
8	1
11	1
12	5
16	1
24	1

In the column count(*) we get the number of devices with this particular maxresolution

MySQL – select (group clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

SELECT type, COUNT(*) AS 'total' FROM device GROUP BY type;

type	total
temperature	5
adc	4

alias defined with "as" provides a better name for columns and can also avoid the need for qualification (see later)

MySQL – select (group clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT type, SUM(price) FROM device GROUP BY type;
```

type	sum(price)
temperature	37,07
adc	33,37

also other operations like min(), max() and avg() can be used combined with “group by”

MySQL – select (having clause)

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
4	MAX1220	adc	12	13,79
5	MAX11613	adc	12	3,51
6	DS28EA00	temperature	12	2,25
7	MAX11040	adc	24	13,45
8	DS1922L	temperature	11	29,25
9	DS1821	temperature	8	2,24

```
SELECT type, SUM(price) FROM device GROUP BY type HAVING
MAX(price)<20;
```

type	sum(price)
adc	33,37

"having" can be used to filter the results considered when performing operations under a "group by" option

MySQL – select (join)

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

```
SELECT name, model FROM
    sensor, device WHERE
    sensor.device=device.id;
```

note the need for qualification for the attribute id.

results from the join

name	model
sensor1	DS2450
sensorhq	DS28EA00
data51	MAX11040
room24	DS2450
large-exp1	DS1821
large-exp2	DS2450
small-exp1	DS28EA00

MySQL – select (join)

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	1
3	data51	7	3	2009-09-01 13:00:21	lab experiment	1
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

```
SELECT sensor.name AS 'sensor',
    location.name AS 'location‘
    FROM sensor, location
    WHERE sensor.location=location.id;
```

results from the join

sensor	location
sensor1	Adriatico
large-exp2	Adriatico
sensorhq	Galileo
large-exp1	Galileo
data51	Leonardo
room24	Leonardo
small-exp1	Leonardo

MySQL – select (join)

SENSOR

id	name	device	location	started	description	running
1	sensor1	3	1	2009-09-02 19:54:48	my first sensor	1
2	sensorhq	6	2	2009-09-02 19:54:48	my best sensor	
3	data51	7	3	2009-09-01 13:00:21	lab experiment	
4	room24	3	3	2009-08-21 14:44:09	internal	1
…	…	…	…	…	…	…

DEVICE

id	model	type	maxresolution	price
1	DS1822	temperature	12	1,61
2	DS1825	temperature	12	1,72
3	DS2450	adc	16	2,62
…	…	…	…	…

LOCATION

id	name
1	Adriatico
2	Galileo
3	Leonardo

results from the join

```
SELECT sensor.name AS 'sensor',
    model, location.name AS 'location'
    FROM sensor, device, location
    WHERE sensor.device=device.id
    AND sensor.location=location.id;
```

sensor	model	location
sensor1	DS2450	Adriatico
sensorhq	DS28EA00	Galileo
data51	MAX11040	Leonardo
room24	DS2450	Leonardo
large-exp1	DS1821	Galileo
large-exp2	DS2450	Adriatico
small-exp1	DS28EA00	Leonardo

MySQL – select (join)

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
…	…	…	…

```
SELECT value FROM
    measurement, sensor WHERE
    measurement.sensor=sensor.id
    AND sensor.name="sensor1";
```

only the temperature measurements for **sensor1** are obtained

results from the join

value
22,3
22,3
23,4
23,5
23,5
23,5
23,6
24,2
24,3

MySQL – select (join)

MEASUREMENT

id	time	sensor	value
1	2009-09-04 10:03:14	1	22,3
2	2009-09-04 12:03:24	1	22,3
3	2009-09-04 12:34:13	4	100,8
4	2009-09-04 13:54:30	1	23,4
5	2009-09-04 14:05:11	1	23,5
6	2009-09-04 14:55:35	7	13,5
…	…	…	…

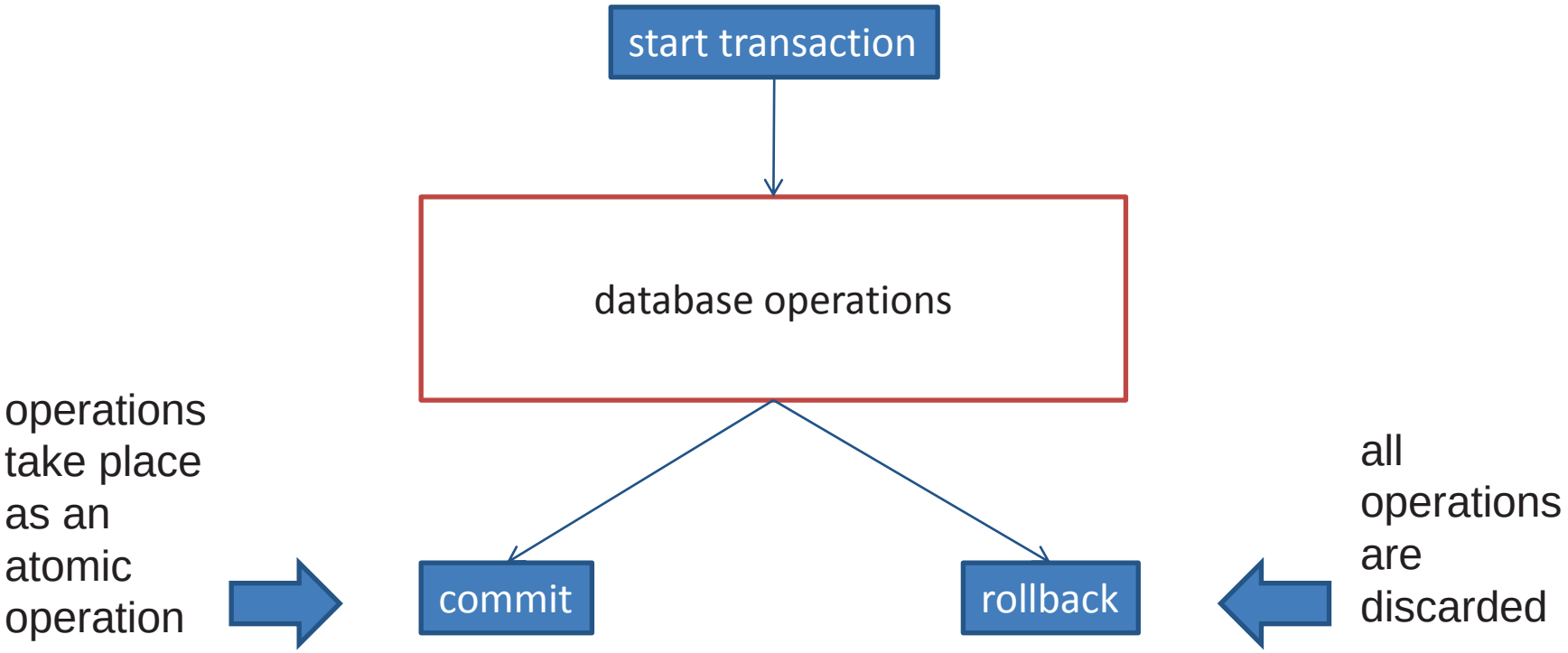
minimum, maximum and average temperatures for sensor1 are obtained

```
SELECT MIN(value), MAX(value), AVG(value)
    FROM measurement, sensor WHERE
    measurement.sensor=sensor.id AND
    sensor.name="sensor1";
```

min(value)	max(value)	avg(value)
22,299999	24,2999992	23,4

Transactions

- Delimits a set of operations that are processed as a whole (unique or atomic operation)

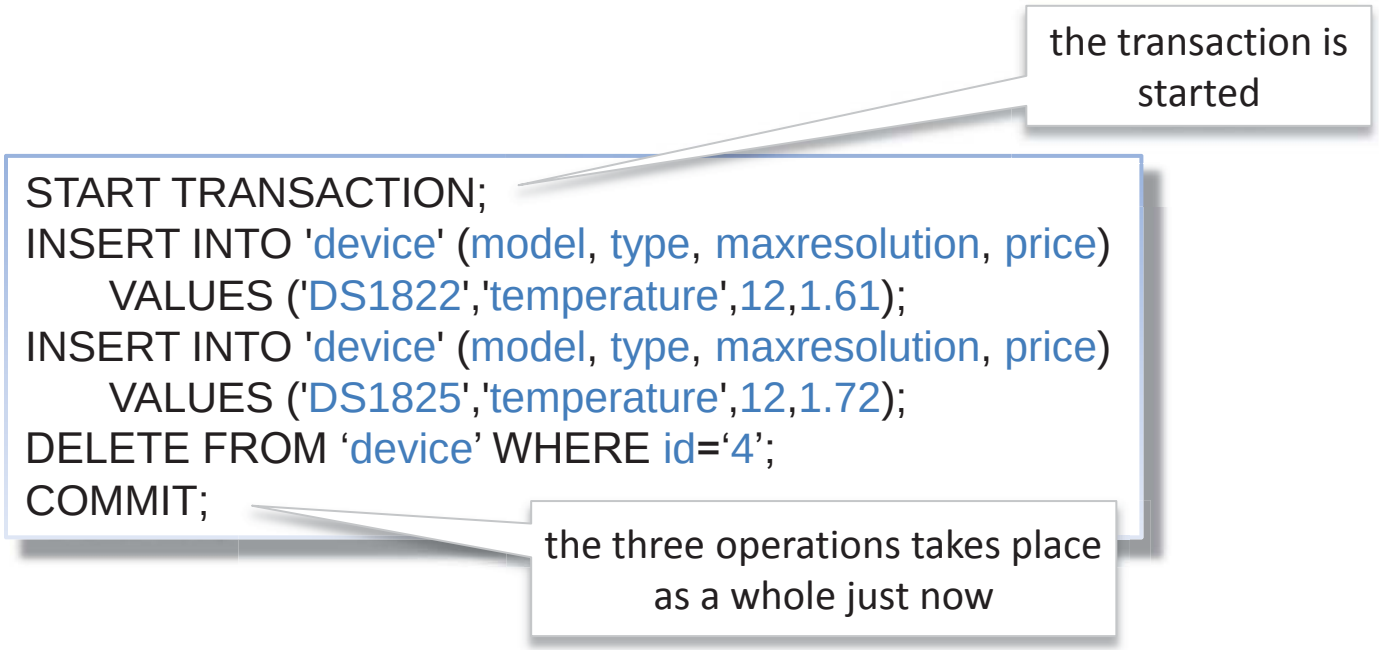


Transactions

- Used to enforce the ACID rules [(1)]:
  - **Atomicity**: Either all the tasks in a transaction must be done, or none of them (all or nothing!)
  - **Consistency**: Every transaction must preserve the integrity constraints
  - **Isolation**: Two simultaneous transactions cannot interfere with one another
  - **Durability**: The effect of completed transactions cannot be lost

Transactions

- The two insertions and the removal operation take place as an atomic operation.



- Care should be taken to synchronize database access with other process (which may be need to wait).

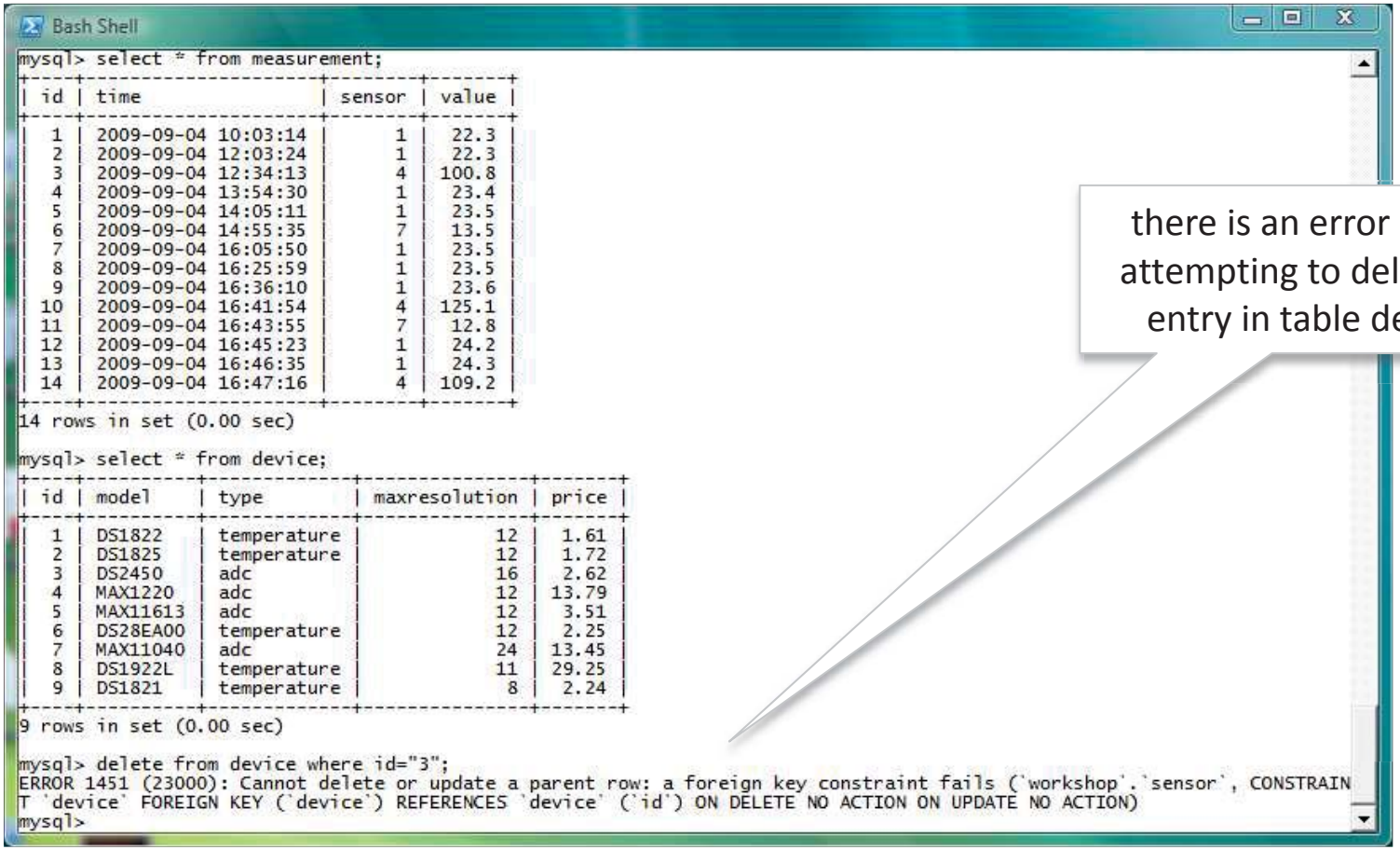
Referential integrity

- Referential integrity is a MUST in database applications.
- It can be controlled by specifying the actions to perform in case of update or removal of rows involved with foreign keys:

```
[CONSTRAINT [name]]
FOREIGN KEY [index_name] (index_col_name, ...)
REFERENCES tbl_name (index_col_name,...)
[ON DELETE reference_option]
[ON UPDATE reference_option]
reference_option: RESTRICT | CASCADE | SET NULL | NO ACTION
```

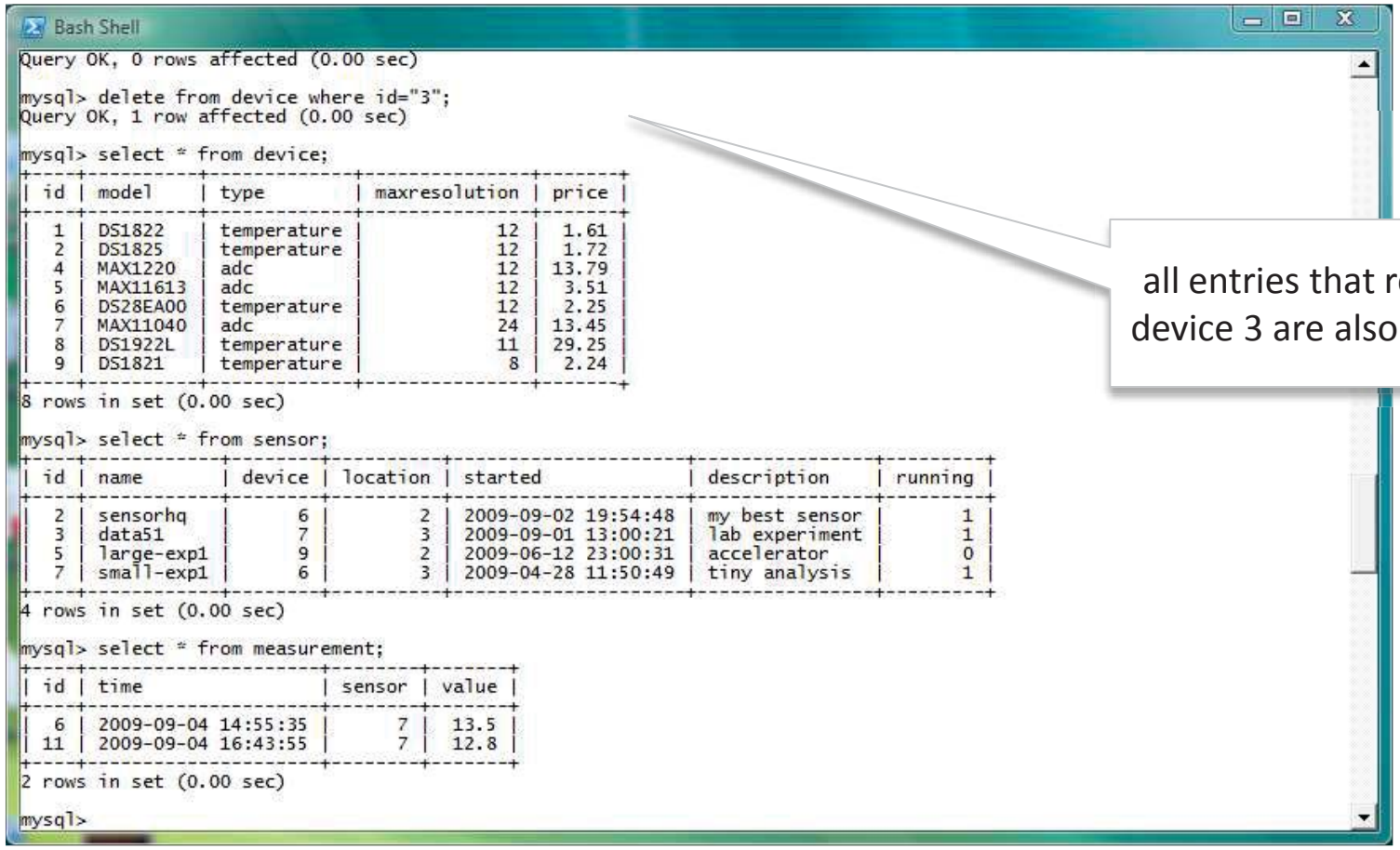
Referential integrity

- If foreign keys are defined with NO ACTION:

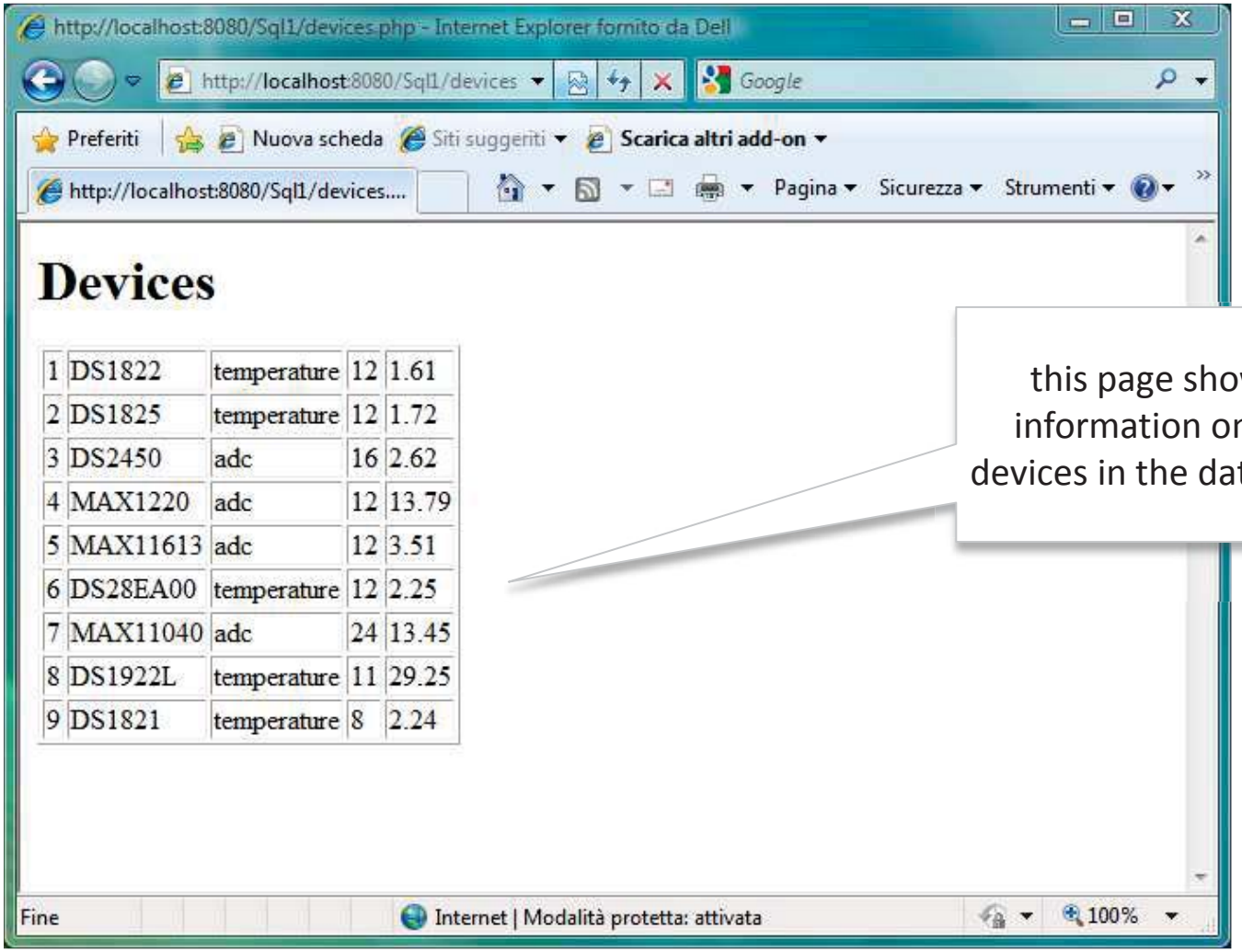


Referential integrity

- If foreign keys are defined with ON DELETE CASCADE.



PHP example – display devices



PHP example – display devices

```
<html>
  <body>
    <h1>Devices</h1>
<?php
$conn = mysql_connect('localhost', 'carlos', 'password') or die('Connection failed');
$db = mysql_select_db('workshop') or die('Database access failed');
$result= mysql_query('select * from device') or die('Database query failed');
echo '<table border="1">';
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
  echo '<tr>';
  while (list($key, $value) = each($row)) {
    echo '<td>' . $value . '</td>';
  }
  echo '</tr>';
}
echo '</table>';
?>
</body>
</html>
```

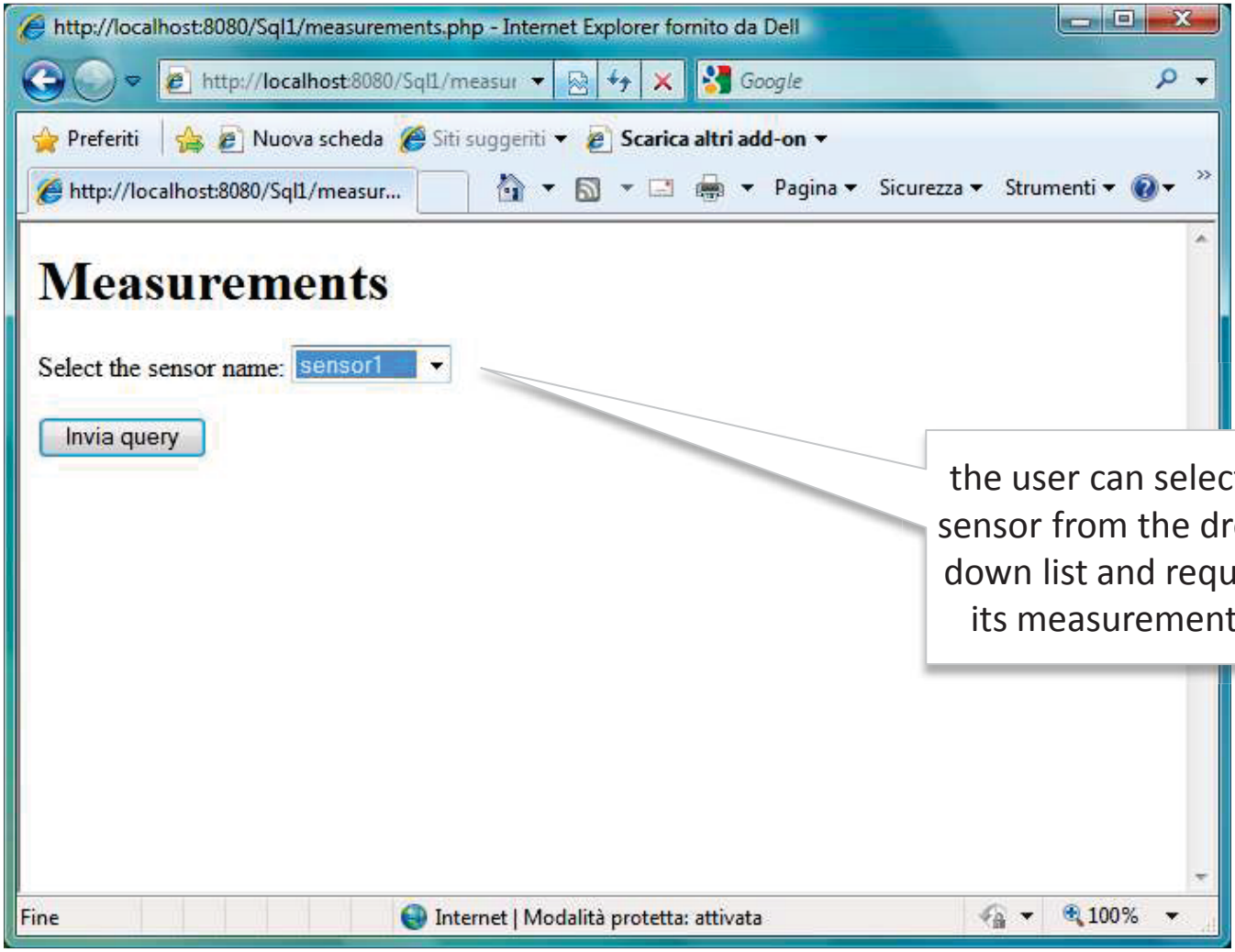
database connection

database selection

query

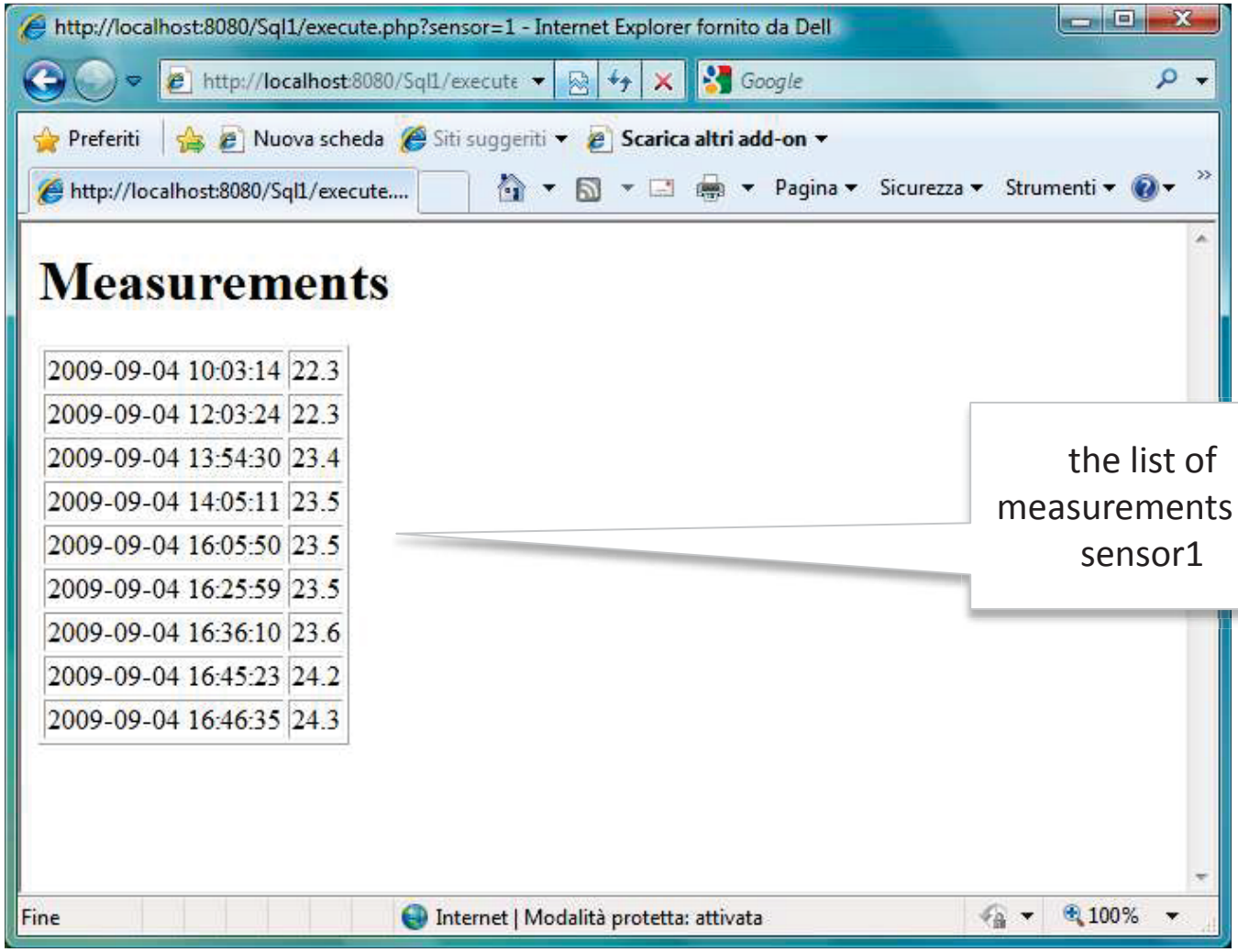
an HTML table is built to display the results returned in an associative array

PHP example – display measurements



the user can select a sensor from the drop-down list and require its measurements

PHP example – display measurements



PHP example – measurements client

```
<html>
  <body> <h1>Measurements</h1>
<?php
$conn = mysql_connect('localhost', 'carlos', ‘password')
                or die('Database connection failed');
$db = mysql_select_db('workshop') or die('Database access failed');
$result= mysql_query('select id,name from sensor order by name')
                 or die('Database query failed');
echo '<form action="execute.php" method="get">';
echo 'Select the sensor name: <select name="sensor">';
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo '<option value="' . $row["id"] . '">' . $row["name"] . "</option>";
}
echo '</select>';
echo '<br><br><input type="submit"></input>';
echo '</form>';
?>
</body>
</html>
```

database connection

database selection

query

a form with a selection field is build in HTML by using the information returned by the query

PHP example – measurements display

```
<html>
  <body> <h1>Measurements</h1>
<?php
$id = $_GET["sensor"];
$conn = mysql_connect('localhost', 'carlos', 'password') or die('Database connection failed');
$db = mysql_select_db('workshop') or die('Database access failed');
$result= mysql_query('select time,value from measurement where sensor="' . $id . '"')
                    or die('Database query failed');
echo '<table border="1">';
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo '<tr>';
    while (list($key, $value) = each($row)) {
        echo '<td>' . $value . '</td>';
    }
    echo '</tr>';
}
echo '</table>';
?>
</body></html>
```

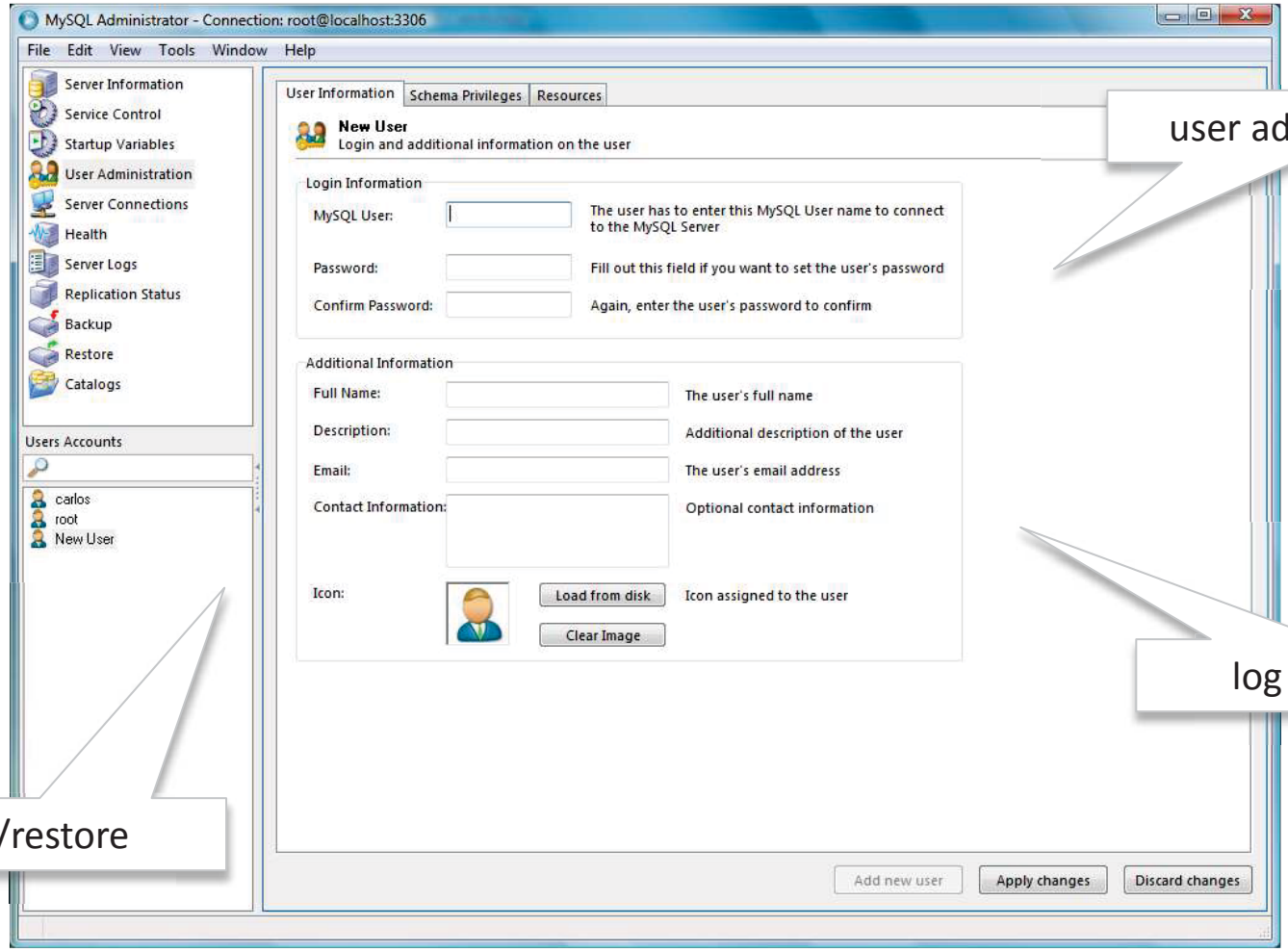
database connection

database selection

query

an HTML table is built to display the results returned in an associative array

MySQL tools: Administrator

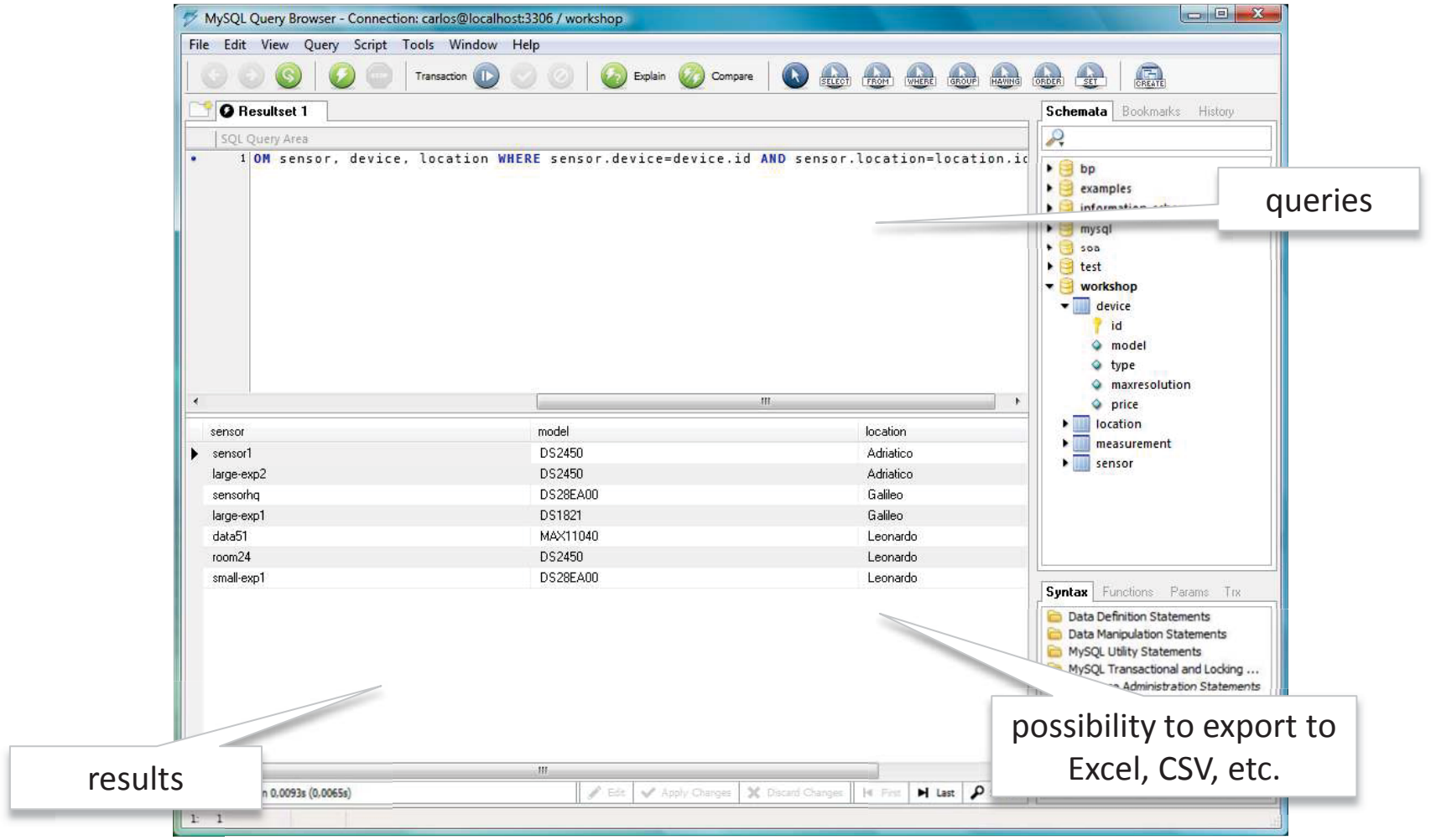


user administration

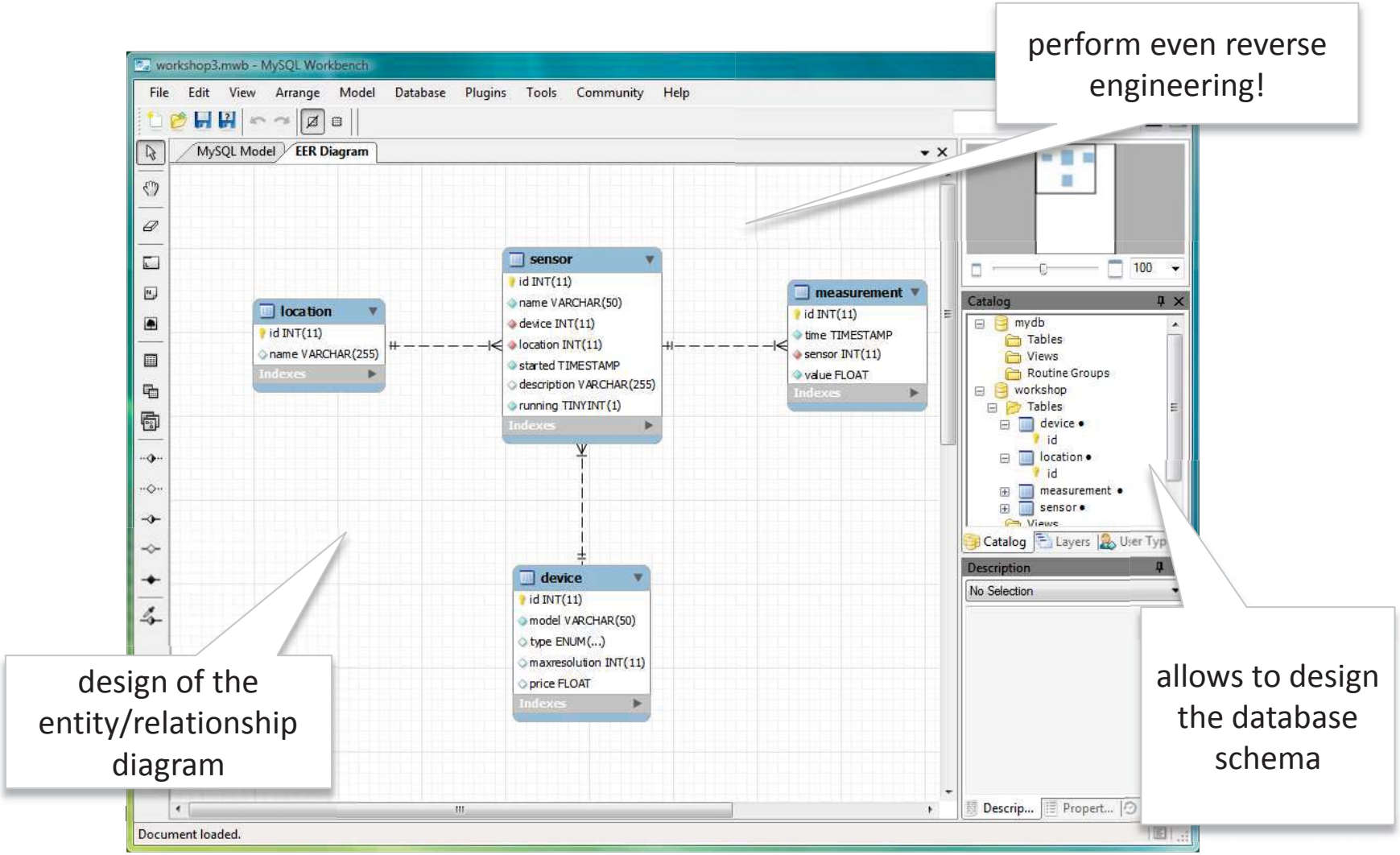
log information

backup/restore

MySQL tools: Query browser



MySQL tools: Workbench



References

- **(1) Database Systems: The Complete Book**, *Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom.* (2008). Prentice Hall

- **(2) MySQL documentation**:
  - http://www.mysql.com (documentation section)

- **(3) Lecture notes on databases**. *Paul Bartholdi*. ICTP Workshops on Distributed Laboratory Instrumentation Systems (2006). ICTP.