Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

4) Databases

Emmanuel Benoist

Spring Term 2021

Computer Science Division

Databases

Introduction

Data Bases and Node-RED

MySQL syntax

- Create a new Data Base
- Set the rights for a DB
- Create tables
- Fill information into tables
- Select information (can sometime be very tricky)
- Update information

MySQL in Node-RED

- A node exists for MySQL
- Reads the sql-query from the flow
- Executes the query
- Sends an object (containing the result) into the flow

Basics of MySQL

Basics of MySQL commands

Creation functions (often done within PHP-MyAdmin)

- Create a new table
- Set the properties of fields (auto-increment, default value,...)

Routine functions (will be used in your programs)

- Insert an element in a table
- Select elements out of a table
- Select elements out of many tables
- Change the content of a record
- Delete some records

Create a Table

Creation of a table

Syntax

- `CREATE TABLE` *table name* (*definition of the fields*)

Create a small table

```
CREATE TABLE ‘category‘ (
 ‘name‘ VARCHAR( 100 ) NOT NULL ,
 ‘categoryID‘ TINYINT NOT NULL AUTO_INCREMENT ,
 PRIMARY KEY ( ‘categoryID‘ )
);
```

- Create a table with two fields
- a string which length can not exceed 100
- A primary key that is a counter

Create a new table

The table can have fields of the following types:

- `TINYINT SMALLINT MEDIUMINT INT BIGINT` that are integers (more or less long)
- `VARCHAR` for short strings (smaller than 256 chars)
- `TEXT` for texts with a fixed length (max 64 kB)
- `DATE` date in format `YYYY-MM-DD`
- `TIMESTAMP` contains a unix timestamp
- `TIME` format `hh:mm:ss`
- `DECIMAL` number with a point.
- `FLOAT`
- `DOUBLE` real numbers
- `BLOB` Any Binary data (image, sound, long text, ...)
- ...

Create a new table (Cont.)

Other attributes or features

- `NULL` or `NOT NULL`
- `AUTO_INCREMENT` for counters

The table has also properties

- `PRIMARY KEY`
- `COMMENT` description of the table

Create other tables

The `article` and `vat` tables

```
CREATE TABLE ‘article‘ (
 ‘articleID‘ INT NOT NULL AUTO_INCREMENT ,
 ‘name‘ VARCHAR( 100 ) NOT NULL ,
 ‘vatID‘ TINYINT NOT NULL ,
 ‘categoryID‘ INT NOT NULL ,
 ‘Price‘ DECIMAL NOT NULL ,
 PRIMARY KEY ( ‘articleID‘ )
);

CREATE TABLE ‘vat‘ (
 ‘vatID‘ TINYINT NOT NULL AUTO_INCREMENT ,
 ‘rate‘ DECIMAL NOT NULL ,
 PRIMARY KEY ( ‘vatID‘ )
) COMMENT = ’The table containing VAT rates’;
```


See the content of a DB

See the content of a data base

See all tables

```
mysql> show tables;
+------------------+
| Tables_in_example |
+------------------+
| article          |
| category         |
| vat              |
+------------------+
3 rows in set (0.00 sec)
```


See the content of a data base (Cont.)

See all columns of a table

```
mysql> show columns from vat;
+-------+--------------+------+-----+---------+----------------
| Field | Type         | Null | Key | Default | Extra
+-------+--------------+------+-----+---------+----------------
| vatID | tinyint(4)   |      | PRI | NULL    | auto_increment
| rate  | decimal(10,2) |      |     | 0.00    |
+-------+--------------+------+-----+---------+----------------
2 rows in set (0.00 sec)
```


Tables: Change rows and Insert

Change a Table - ALTER

Remove columns

```
ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
```

Add a new column

```
ALTER TABLE ‘article‘ ADD ‘description‘ BLOB
  NOT NULL ;
```

Change an existing column

```
ALTER TABLE ‘article‘ CHANGE ‘Price‘ ‘price‘
    DECIMAL( 10, 2 ) DEFAULT ’0’ NOT NULL;
```

Fill data into a table - INSERT

Syntax

- `INSERT INTO` *tablename* [(*list of fields*)] `VALUES` (*list of values*);
- all `not null` fields must be set, other can be just two commas.

Insert a row in a table

```
INSERT INTO ‘article‘ ( ‘articleID‘ , ‘name‘ , ‘vatID‘ ,
   ‘categoryID‘ , ‘price‘ , ‘description‘ )
 VALUES (’’, ’Pencil’, ’0’, ’0’, ’1.50’, ’’);
```

Other possibility

```
INSERT INTO article values
  (’’,’Mercedes Class E’,’0’,’0’,’100000’,
   ’The same Mercedes Lady Diana has used’
  );
```


Change the content of one or many rows

`UPDATE` a table

```
UPDATE ‘article‘ SET ‘description‘ =
  ’A very nice black pencil with white stripes’
  WHERE ‘articleID‘ = ’1’ LIMIT 1 ;
```


Select Information

Select information

Syntax

- `SELECT` *Field list* `FROM` *list of tables* [`WHERE` *conditions*] [`LIMIT` *limits*]
- Field list can also be a joker (*)
- Conditions can be combined with boolean connectors (`AND`, `OR`, `NOT`)
- If we only want to see a part of a list, we can limit it.

Select all the rows and columns of a table

```
mysql> select * from vat;
+-------+------+
| vatID | rate |
+-------+------+
|     1 | 7.00 |
|     2 | 7.65 |
+-------+------+
```

Select information(Cont.)

Select only some columns

```
mysql> select name, price from article;
+------------------+-----------+
| name             | price     |
+------------------+-----------+
| Pencil           |      1.70 |
| Mercedes Class E | 100000.00 |
+------------------+-----------+
2 rows in set (0.00 sec)
```


Select data

Select only some rows

```
mysql> select name, price from article
    -> where articleID=1;
+--------+-------+
| name   | price |
+--------+-------+
| Pencil |  1.70 |
+--------+-------+
1 row in set (0.01 sec)
```


Merge data from different tables

Merge two tables

- Fields must know from which table they come (the same field can be in the two tables).
- We can rename a requested field with the `AS` keyword.

```
mysql> select article.name, vat.rate, article.price
    -> from article, vat where article.vatID= vat.vatID;
+------------------+------+-----------+
| name             | rate | price     |
+------------------+------+-----------+
| Pencil           | 7.00 |      1.70 |
| Mercedes Class E | 7.00 | 100000.00 |
+------------------+------+-----------+
```


Merge ...(Cont.)

Merge and compute

```
mysql> select article.name, vat.rate, article.price,
    -> article.price*(1+vat.rate/100) as priceWithVAT
    -> from article, vat where article.vatID= vat.vatID;
+------------------+------+-----------+--------------+
| name             | rate | price     | priceWithVAT |
+------------------+------+-----------+--------------+
| Pencil           | 7.00 |      1.70 |       1.8190 |
| Mercedes Class E | 7.00 | 100000.00 |  107000.0000 |
+------------------+------+-----------+--------------+
2 rows in set (0.00 sec)
```

Join

INNER JOIN If there is no match, the row is not shown

```
select article.name, vat.rate, article.price
       from article inner join vat
       on article.vatID= vat.vatID;
```

LEFT JOIN If there is no match, the second table is replaced by an empty record.

```
select article.name from article left join vat
       on article.vatID= vat.vatID
       where vat.rate is null;
```

(gives the list of articles with undefined VAT)

More on SELECT

Result of a select can be put into a temporary table

```
create temporary table valueVAT
  (select vat.rate, article.name
   from vat,article
   where vat.vatID=article.vatID
  )
;
```

You can access to the content and then delete this table

```
select * from valueVAT;

drop table IF EXISTS valueVAT;
```


Select and more options

Order result (`DESC` or `ASC`)

```
select name, price from article order by price desc;
```

Group rows

```
mysql> select vatID, count(vatID)
     > from article GROUP BY vatID;
+-------+--------------+
| vatID | count(vatID) |
+-------+--------------+
|     1 |            2 |
+-------+--------------+
1 row in set (0.00 sec)
```

`SELECT` **can have a lot of functions an combine all of them**

Delete fields

Delete the content of a table respectively to a where clause

```
delete from article where articleID=3;
```

Node-RED and Database

The MySQL node

- **Must install MySQL for Node-RED**
- **New node is created**
- **Configuration:**
  - Server (usually `localhost`)
  - Username
  - Password
- **Connect to one Database**
  - select the Database to be linked to

The MySQL Node

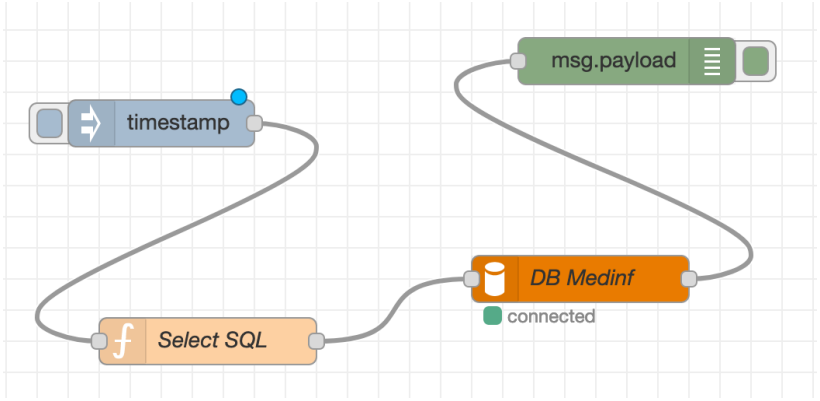
- **Input**
  - The MySQL query must be inside the property `msg.topic`
- **Output**
  - The result is written inside `msg.payload`
  - It type is a JSON object (attention! not a JSON string).

First Flow

- **We have four nodes**
  - Inject : to start the action
  - Function : to define the value of the query
  - Mysql : to define the connection and where the query is executed
  - Debug : for the output
- **Function:**

```
msg.topic="select␣*␣from␣staff";
return msg;
```

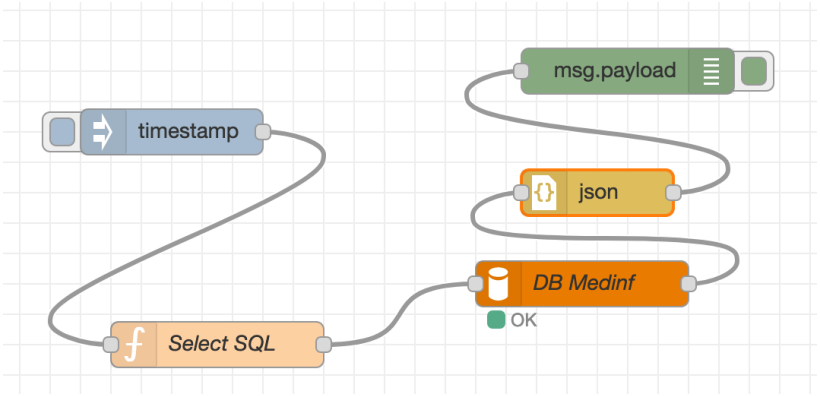
First Flow



Output in JSON String

- **Need to transform the output into a JSON String**
- **Add a JSON node**
  - Transform Object into a JSON String
- **JSON String is displayed in the Debug Panel**

Output in JSON String



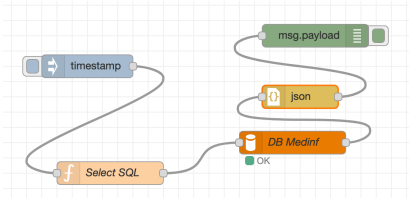
Another Select

- **A more complex Select query:**

  select ∗ from patient, vital_sign where patient.patientID = ↘
  →vital_sign.patientID;

- **Included in the Function node**

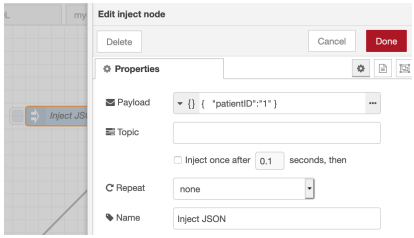
```
msg.topic="select␣*␣from␣patient,␣vital_sign␣where␣↘
→patient.patientID␣=␣vital_sign.patientID";
return msg;
```



JSON as input to the query

- **We change the `Inject` node**
  - We inject a json object

```
{
    "patientID":"1"
}
```



JSON as input to the query (Cont.)

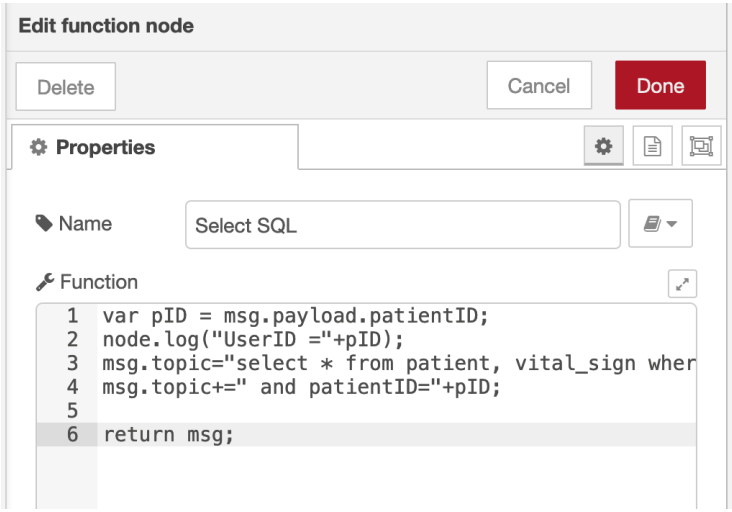
- **We change the function to read data from JSON**
- **We print the read value in the log**
  - Can be read inside the console (in the virtual machine).
- **We insert the data inside the query.**

```
var pID = msg.payload.patientID;
node.log("UserID␣=␣"+pID);
msg.topic="select␣*␣from␣patient,␣vital_sign␣where␣↘
→patient.patientID␣=␣vital_sign.patientID";
msg.topic+="␣and␣patient.patientID="+pID;
return msg;
```

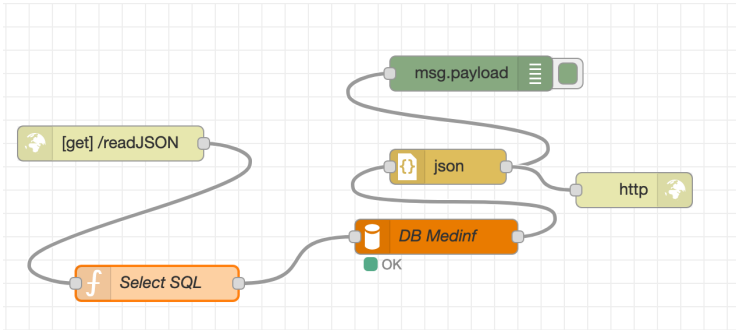

JSON as input to the query

- **We insert HTTP request node**
  - For reading the query
  - We type the URL : `/readJSON`
- **We insert an output HTTP Response node**
  - Outputs the JSON string
- **We need to update the function**
  - Not to read the input anymore (we have no input now)

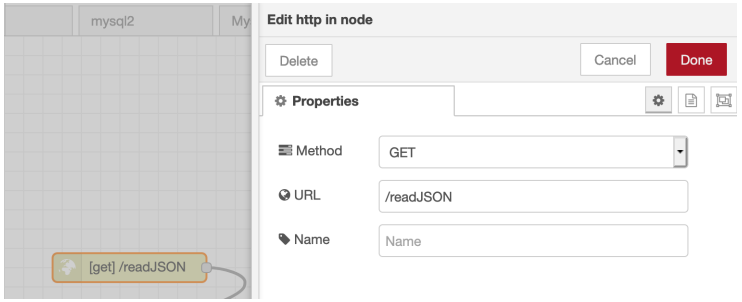
JSON as input to the query (Cont.)



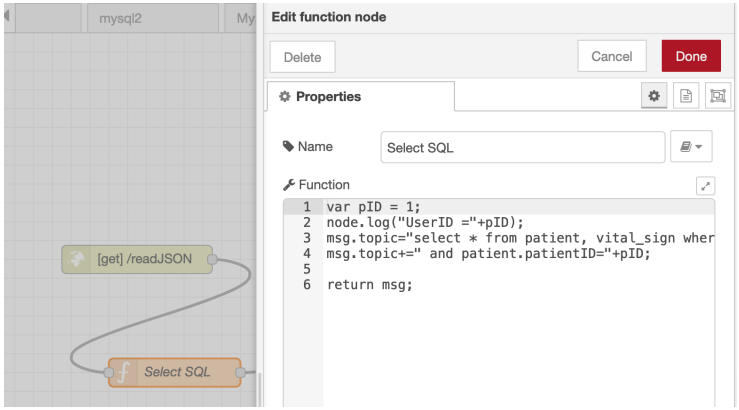
JSON on an HTTP server (Cont.)



JSON on an HTTP server (Cont.)



JSON on an HTTP server (Cont.)



Read the number of the patient from the query string

- **If URL is :** `/readJSONinput?patient=1`
- **We can read the parameters in the msg object.**
  - The variable `msg.req.query.patient` contains 1
  - the object `msg.req.query` contains all the parameters
- **New Function node:**

```
if(typeof(msg.req.query.patient)!== "undefined"){
    var pID = msg.req.query.patient;
    node.log("UserID␣="+pID);
    msg.topic="select␣*␣from␣patient,␣vital_sign␣where␣↘
    →patient.patientID␣=␣vital_sign.patientID";
    msg.topic+="␣and␣patient.patientID="+pID;
}
else{
     msg.topic="select␣*␣from␣patient";
}
```

Read the number of the patient from the query string (Cont.)

- **URL:** `http://localhost:1880/readJSONinput?patient=1`

Conclusion

Conclusion

DB are the center of our work

- Do not require a programmer to write HTML
- they are used to access DB's
- forms and db's are the two pillars of web programming
- a lot of other finesses to be discovered
- SQL : a semester course of 2 hours a week

Integration in Node-RED

- Node MySQL
- Makes a connection to the DB (need configuration)
- Input in `msg.topic`
- Output is an object `msg.payload`
- Output can be serialized into a JSON string and linked to an HTTP server.