

Task #1

Create a relational database on computers in computer classroom 308, using MySQL server and any client.

Save tables using Export database (in *.sql format)

Prepare the SELECT, INSERT and UPDATE commands for each typical operation of the intended application – record the task and your solution, at least seven tasks (prepare them in a text file).

(for example – book library: add new user, add new book, record a book reservation, list of free books, list titles of books loaned by user specified by name, change user's mail address, erase book from library, return a book by user, count, how many books are free for loan).

Task #2

Create the same database, using MS Access 2013.

Task #3

Draw the E-R diagram for your database

SQL server

RDBMS separates the database from an application.

Application communicates with RDBMS using SQL.

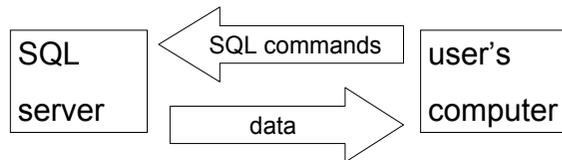
SQL = Structured Query Language

(originally not a programming language – basic standard is missing commands for cycles and variables; many extensions exist with this kind of commands)

Client - Server

Thick client

Client program assembles SQL commands and generates user interface (on the user's computer)

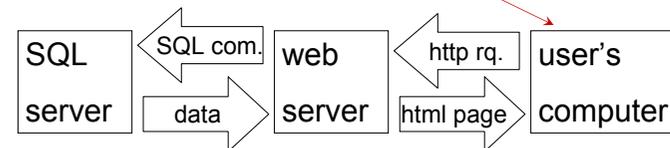


Client - Server

Thin client

SQL commands are assembled on the web server (it usually is, but don't have to be the same server – SQL server on another computer can make its maintenance easier).

As a client, we can use any web browser – in real, this is much more complicated program than thick client, but universal.



DDL, DML

Data Definition Language
Data Manipulation Language
Data Control Language
All of them are part of SQL.

Micka

-One of many SQL Clients
-Diploma thesis by Ing. Zbyněk Munzar, FEE CTU
-Can be retrieved from www.senon.cz/micka/

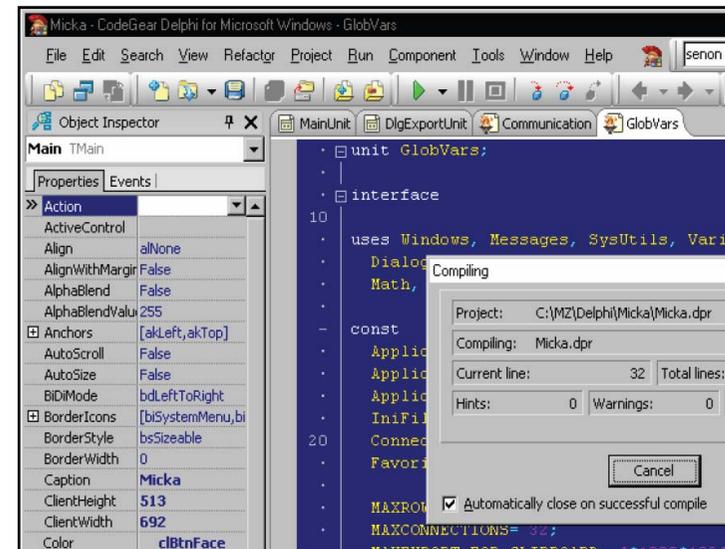


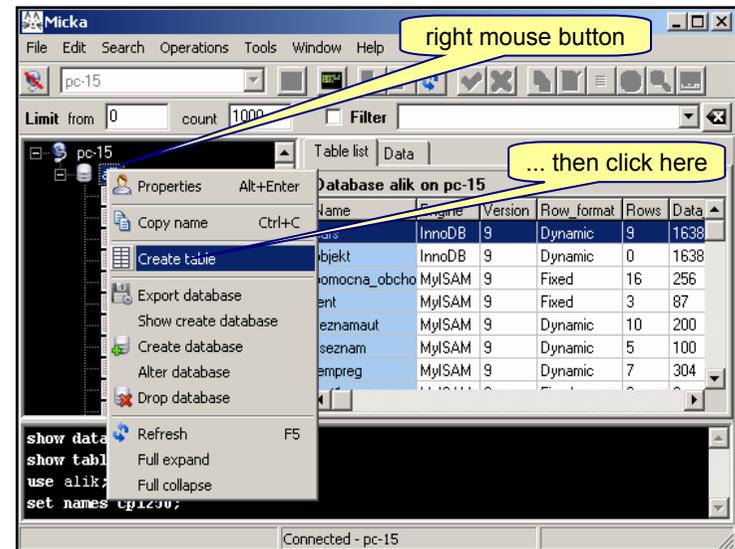
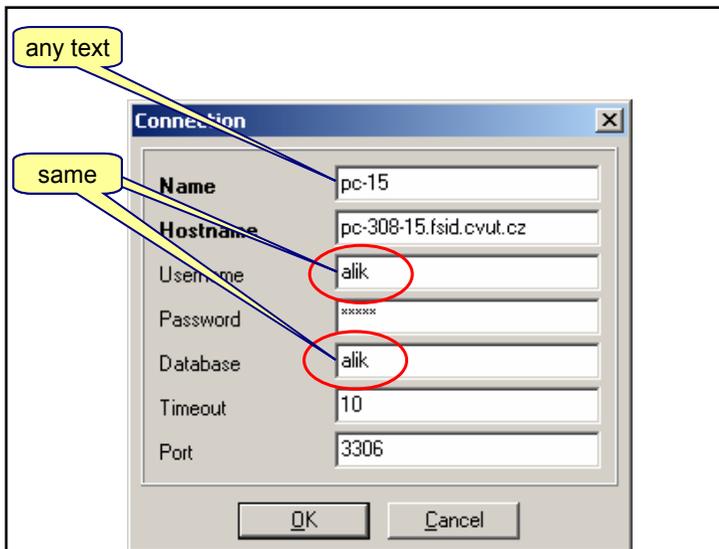
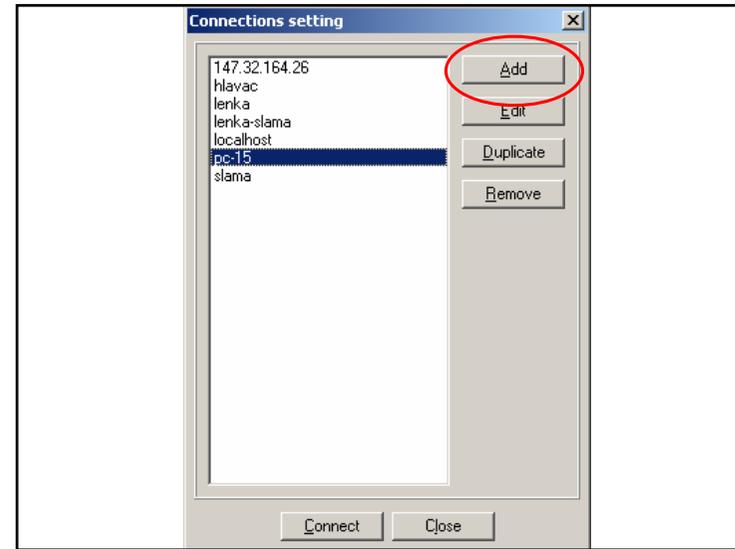
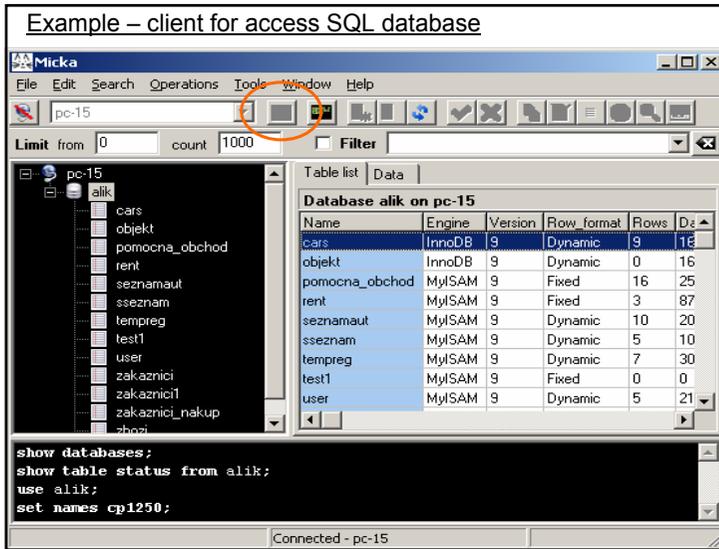
More common clients:

phpMyAdmin – part of MySQL installation
phpWebAdmin – can be installed on SQL server
HeidiSQL (generally recommended)
Emma
MySQL GUI Tools

Charakteristika aplikace

- Micka
- Desktopová aplikace
- Vyvinuta v Object Pascalu v prostředí Delphi
- V tuto chvíli pouze pro 32 bitové MS Windows
- Univerzální klient pro správu MySQL databází
- Standardní GUI
- Snadná rozšiřitelnost
- Potřeba neustálého vývoje





A,Á,Ä,a,á,ä CP 1250 cs sorting O,Ó,Ö,o,ó,ö
 B,b P,p
 C,c Q,q
 Č,č R,r,Ř,ř
 D,d Š,š
 ě,ě Š,š
 E,É,Ě,ě,é,ě T,t
 F,f ě,ě T,t
 G,g Ť,ť
 H,h U,Ú,Ů,u,ú,ů V,v
 CH,Ch,ch W,w
 I,Í,Ī,í,ĭ X,x
 J,j Y,Ý,y,ý
 K,k Z,z
 L,Ľ,Ĺ,l,ĺ Ž,ž
 M,m
 N,n
 Ń,ń

http://en.wikipedia.org/wiki/Diacritical_marks

Create table alik.employee

Field	Type	Collation	Null	Def
id	int(11)	<NULL>	NO	<N

right mouse button:

- + Add field Ins
- Change field Enter
- Copy field Ctrl+C
- Drop field Ctrl+Del
- Copy field name
- Refresh F5

Field

Position: After id

Name: Name

Type: VarChar

Length: 64

Default: <NULL>

Charset: cp1250 Collation: cp1250_bin

Comment:

NULL Allowed

OK Cancel

alík.employee - Míčka

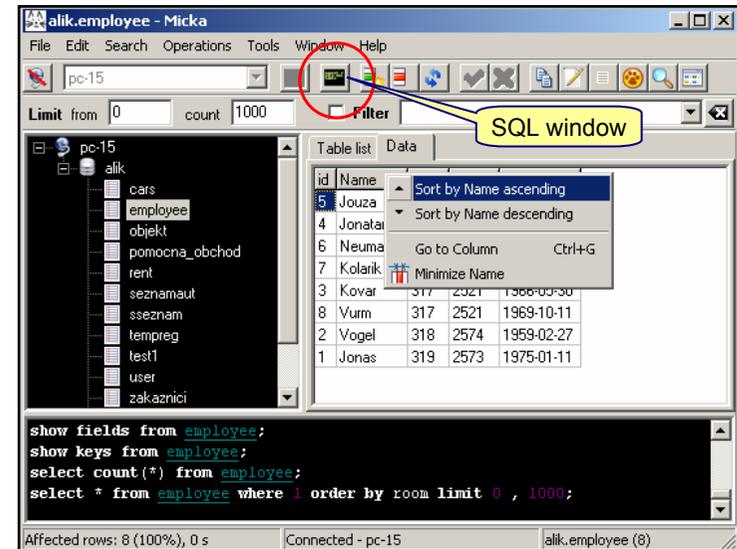
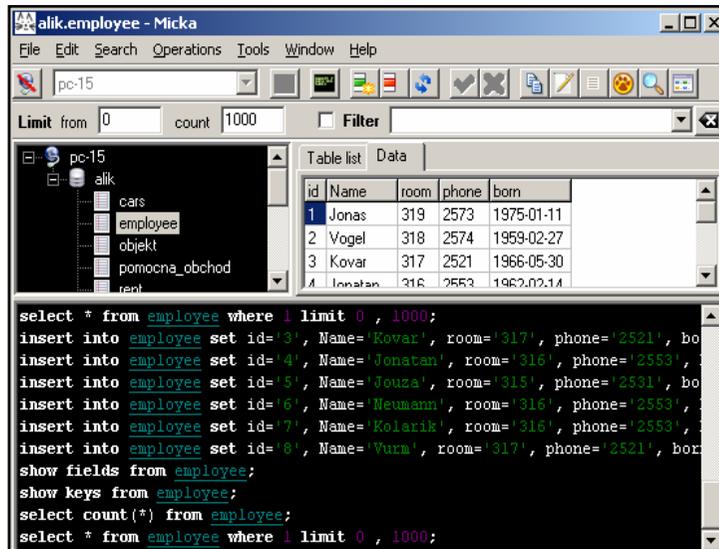
Limit from 0 count 1000 Filter

id	Name	room	phone	born
3	Kovar	317	2521	1966-05-30
1	Jonas	319	2573	1975-01-11
4	Jonatai	316	2553	1962-02-14
5	Jouza	315	2531	1953-02-28
6	Neuma	316	2553	1971-03-25
7	Kolarik	316	2553	1973-11-03
8	Vurm	317	2521	1969-10-11
2	Vogel	318	2574	1959-02-27

```

insert into employee set id='5', Name='Jouza', room='315', phone='2531', bo
insert into employee set id='6', Name='Neumann', room='316', phone='2553',
insert into employee set id='7', Name='Kolarik', room='316', phone='2553',
insert into employee set id='8', Name='Vurm', room='317', phone='2521', bor

```



select

SELECT * FROM tablename ;

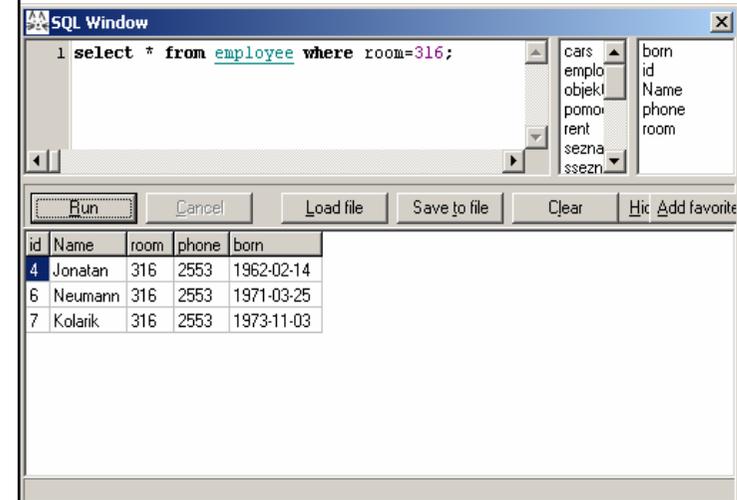
SELECT Name,room FROM employee ;

SELECT Name,room FROM employee
WHERE room = '316';

SELECT Name,room FROM employee
ORDER BY room;

SELECT Name,room FROM employee
WHERE room = '316' ORDER BY Name;

Result example:



important notice for this ppt

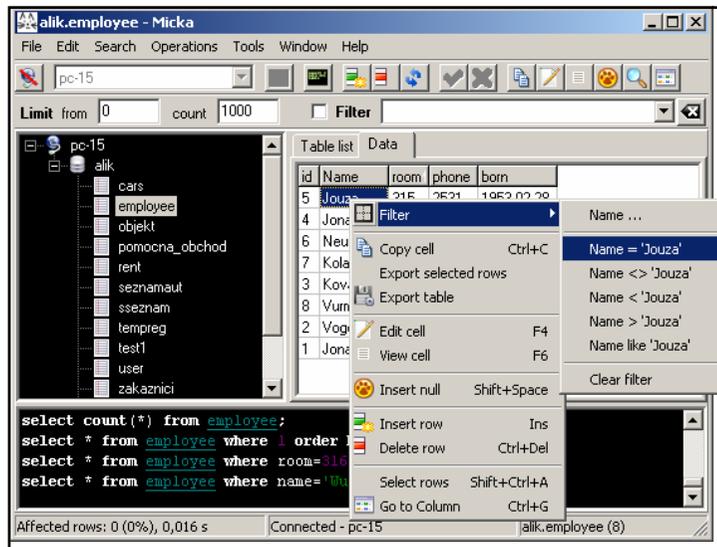
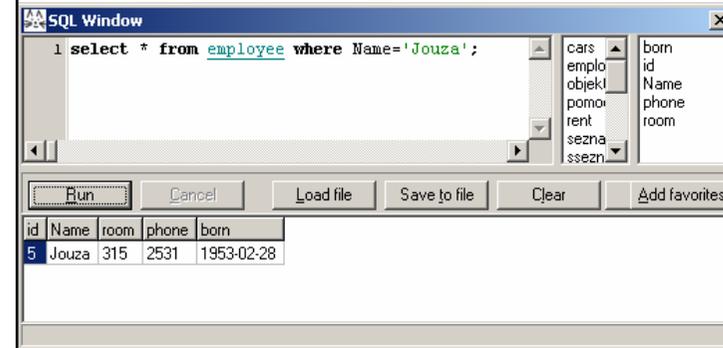
This text was written in MS Powerpoint.

In this software, all apostrophes is changed to special characters (left and right quotation). It could be repaired or this feature switched off, but both is unreliable.

So, please, after copy to your “SQL command window”, substitute them to the correct one. Thanks.

select

```
SELECT Name,room FROM employee  
WHERE Name = 'Jouza';
```



select

```
select * from employee order by id where  
CONVERT( 'Name' USING utf8 ) = 'Jouza'
```

Typical character set:

USA – ASCII, 7 bit, no conversion function, DOS 437

WestEu. – ISO 8859-1, DOS 850, CP1252

CE – ISO 8859-2, DOS 852, CP1250

Greek – ISO 8859-7, CP1253

Cyrillic – ISO 8859-5, DOS 856, CP1251

UTF16

UTF8 – see <http://en.wikipedia.org/wiki/UTF-8>

select where

http://www.w3schools.com/sql/sql_between.asp

```
SELECT * FROM employee
WHERE room >= '312' AND room <= '316';
```

OR, XOR

```
SELECT * FROM employee
WHERE room BETWEEN '312' AND '316';
```

Result is dependent on the database engine; on the MySQL, both examples are the same, but on other systems it can contain only one limit value, or none of them ("between"). It is better to use the first method.

```
SELECT * FROM employee
WHERE room IS NULL OR room < '300';
```

```
SELECT * FROM employee
WHERE room IS NOT NULL ;
```

select count

MIN(...

```
SELECT MAX(room) FROM employee;
```

```
SELECT COUNT(*) FROM employee
WHERE room >= '312' AND room <= '317';
```

```
SELECT COUNT(DISTINCT(room))
FROM employee;
```

see GROUP BY

```
SELECT AVG(price) FROM stock;
```

```
SELECT SUM(price * amount) FROM stock;
```

```
SELECT MAX(price) FROM stock;
```

SQL can work with arithmetic operators (+,-,...)

The screenshot shows an SQL Window with the following query:

```
1 select ((orders.km2-orders.km1)*cars.priceperkm)
2 + ((orders.o_to-orders.o_from)*cars.priceperday)
3 as price_per_order
4 from cars,orders
5 where cars.id=orders.id_c and orders.o_to is not null;
```

Below the query, there are buttons: Run, Cancel, Load file, Save to file, and Clear. The result of the query is displayed in a table:

price_per_order
420
50

An orange circle highlights the text "as ... alias, explained latter" with an arrow pointing to the "as price_per_order" line in the query.

The screenshot shows an SQL Window with the following query:

```
1 -- overall turnover of our firm
2 select sum((orders.km2-orders.km1)*cars.priceperkm)
3 + sum((orders.o_to-orders.o_from)*cars.priceperday)
4 as turnover
5 from cars,orders
6 where cars.id=orders.id_c and orders.o_to is not null;
```

Below the query, there are buttons: Run, Cancel, Load file, Save to file, and Clear. The result of the query is displayed in a table:

turnover
470

select like: wildcards

```
SELECT * FROM employee  
WHERE Name LIKE 'J%';
```

Typical wildcard:

% ... any combination of letters

_ ... just one letter (exactly one, not 'none')

The screenshot shows an SQL Window with the following query:

```
1 SELECT * FROM employee  
2 WHERE Name LIKE 'J%';
```

The results table is:

id	Name	login	room	born
1	Jonas	<NULL>	319	1975-01-11
5	Jouza	<NULL>	315	1953-02-28

```
SELECT * FROM table1, table2 WHERE ...;
```

```
SELECT Name, room, block, phone  
FROM employee, rooms  
WHERE employee.room = rooms.room ;
```

The screenshot shows two SQL result windows. The left window shows the results of the query:

id	Name	room	born
1	Jonas	319	1975-01-11
2	Vogel	318	1959-02-27
3	Kovar	317	1966-05-30
4	Jonatan	316	1962-02-14
5	Jouza	315	1953-02-28
6	Neumann	316	1971-03-25
7	Kolarik	316	1973-11-03
8	Vurm	317	1969-10-11

The right window shows the results of the join query:

room	phone	block	address
314	2881	A1	Charles square
315	2882	A1	Charles square
316	2883	A1	Charles square
317	2884	A1	Charles square
318	2885	A1	Charles square
319	2886	A1	Charles square
307	2758	A3	Charles square
306	2857	A3	Charles square

Orange arrows point from the 'room' column in the left table to the corresponding 'room' column in the right table, illustrating the join condition.

select join

must be rooms.room

```
SELECT Name, room, block, phone  
FROM employee, rooms  
WHERE employee.room = rooms.room ;
```

is equivalent to

```
SELECT Name, room, block, phone  
FROM employee JOIN rooms  
ON employee.room = rooms.room ;
```

<http://www.sql-tutorial.net/SQL-JOIN.asp>

The screenshot shows an SQL Window with the following query:

```
1 SELECT Name, rooms.room, block, phone  
2 FROM employee, rooms  
3 WHERE employee.room = rooms.room ;
```

The results table is:

Name	room	block	phone
Jouza	315	A1	2882
Jonatan	316	A1	2883
Neumann	316	A1	2883
Kolarik	316	A1	2883
Kovar	317	A1	2884
Vurm	317	A1	2884
Vogel	318	A1	2885
Jonas	319	A1	2886

SQL Window

```

1 SELECT Name,rooms.room,block,phone
2 FROM employee JOIN rooms
3 ON employee.room = rooms.room ;
4

```

Name	room	block	phone
Jouza	315	A1	2882
Jonatan	316	A1	2883
Neumann	316	A1	2883
Kolarik	316	A1	2883
Kovar	317	A1	2884
Vurm	317	A1	2884
Vogel	318	A1	2885
Jonas	319	A1	2886

select outer join

but we have no empl. w/o room

Modified JOIN can include empty lines:

```

SELECT Name,rooms.room,block,phone
FROM employee LEFT JOIN rooms
ON employee.room = rooms.room ;

```

is the same as

```

SELECT Name,rooms.room,block,phone
FROM employee LEFT OUTER JOIN rooms
ON employee.room = rooms.room ;

```

select right outer join

In or case, it could be shown on the „RIGHT“:

```

SELECT Name,rooms.room,block,phone
FROM employee RIGHT JOIN rooms
ON employee.room = rooms.room ;

```

is the same as

note: FULL JOIN exists

```

SELECT Name,rooms.room,block,phone
FROM employee RIGHT OUTER JOIN rooms
ON employee.room = rooms.room ;

```

SQL Window

```

1 SELECT Name,rooms.room,block,phone
2 FROM employee RIGHT JOIN rooms
3 ON employee.room = rooms.room ;

```

Name	room	block	phone
<NULL>	314	A1	2881
Jouza	315	A1	2882
Jonatan	316	A1	2883
Neumann	316	A1	2883
Kolarik	316	A1	2883
Kovar	317	A1	2884
Vurm	317	A1	2884
Vogel	318	A1	2885
Jonas	319	A1	2886
<NULL>	307	A3	2758
<NULL>	306	A3	2857

List of cars, including not in the connection table:

```

1 select cars.type, orders.id_u, users.login
2 from cars
3 left join orders on cars.id=orders.id_c
4 left join users on orders.id_u=users.id;
5

```

type	id_u	login
Audi A100	4	babka
Citroen 2CV	4	babka
Renault 5	3	smith
Renault 8	5	berta
Renault 12	1	vlk
Peugeot 304	4	babka
Ford Escort	<NULL>	<NULL>
Fiat 125	<NULL>	<NULL>
Peel P50	<NULL>	<NULL>

alias in select

Used for renaming table columns (the result):

```

SELECT Name AS name, phone AS tel
FROM employee, rooms
WHERE employee.room = rooms.room ;

```

```

1 SELECT Name AS name, phone AS tel
2 FROM employee, rooms
3 WHERE employee.room = rooms.room ;

```

name	tel
Jouza	2882
Jonatan	2883

select ... group by

Useful for sum, count or avg (average) function:

```

SELECT COUNT(*) FROM employee
GROUP BY room;

```

... will return only counts w/o room n., so ...

```

SELECT room, COUNT(*) FROM employee
GROUP BY room;

```

... will return a reasonable table:

In the rooms are so many people:

```

1 SELECT room, COUNT(*) FROM employee
2 GROUP BY room;
3

```

room	COUNT(*)
315	1
316	3
317	2
318	1
319	1

in this case, using AS would be useful

inserting data

Two form of the **INSERT** command:

```
INSERT INTO table1  
VALUES (value1, value2, value3...)
```

... or ...

```
INSERT INTO table1 (column1, column2,...)  
VALUES (value1, value2,...)
```

... the second one is safe against changing the table structure.

auto increment???

insert into

```
INSERT INTO employee  
VALUES('9','Gregor','307','1954-01-15');
```

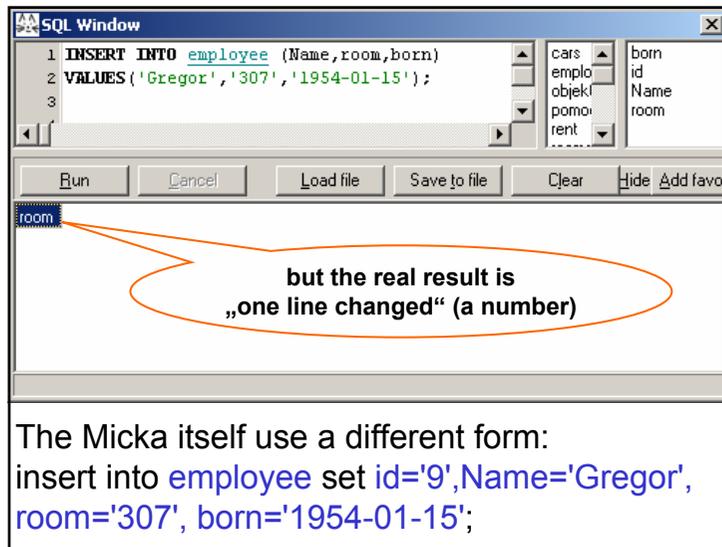
... or we can use ...

```
INSERT INTO employee (Name,room,born)  
VALUES('Gregor','307','1954-01-15');
```

... we can skip the ID column, because it was defined as auto increment; if we skip any other column with value, the „default“ value will be inserted; if is not defined, then the Null value.

apostrophes

no apostrophes



SQL Window

```
1 INSERT INTO employee (Name,room,born)
2 VALUES ('Gregor','307','1954-01-15');
3
```

cars emplo objek pomor rent born id Name room

Run Cancel Load file Save to file Clear Hide Add favor

room

but the real result is „one line changed“ (a number)

The Micka itself use a different form:
insert into employee set id='9',Name='Gregor',
room='307', born='1954-01-15';

delete

it will delete only one line
but random one (in SQL is no order)

-Deletes one or more lines

-defined by the „where“ clause:

```
DELETE FROM employee  
WHERE room = '316';
```

... it is useful to specify the **primary key** to prevent to delete more lines, but we can use:

```
DELETE FROM employee  
WHERE room = '316' LIMIT 1;
```

update

-Update rewrites one or more lines, again defined by the „where“ clause:

```
UPDATE employee  
SET room = '316'  
WHERE Name = 'Jouza';
```

... if WHERE clause is true for more lines, then the specified value (see SET) will be changed for every of this lines (be careful!).

LIMIT can be used, but there is no order in SQL.

DDL: alter table

(We will never need this, we will use Micka):

```
ALTER TABLE employee  
ADD login VarChar(33) character set cp1250  
collate cp1250_bin null default null  
AFTER Name;
```

(all rows will be filled by the default value = null)

```
ALTER TABLE employee  
DROP COLUMN login;
```

(again, only as example – not in application)

alter table

To rename table itself:

```
ALTER TABLE employee  
RENAME TO people;
```

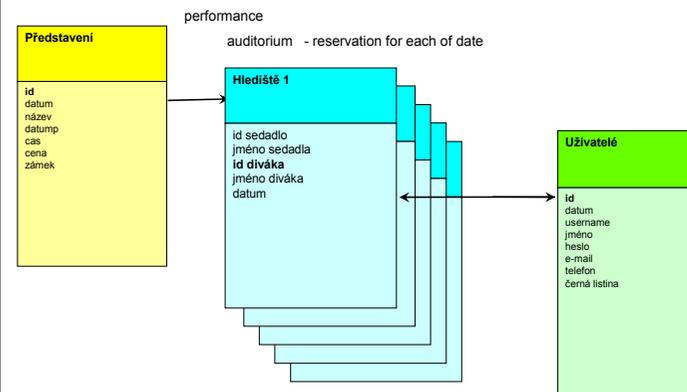
Change property of column, i.e. from 11 to 33:

```
ALTER TABLE employee  
MODIFY room VarChar(33)  
character set cp1250 collate cp1250_bin ;
```

(any use of this command is risky, could be fatal for our application)

DDL: create table

There could be demand for create new table:



DDL: create table

```
CREATE TABLE action57  
( seat VarChar(8) NOT null PRIMARY KEY,  
  user Int );
```

(on the prev. slide, new “action” table is created and immediately filled by default seat numbers – rows were marked by letters)

The application can delete table using:

```
DROP TABLE action17;
```

More complete on: <http://www.1keydata.com/sql/sqldrop.html>

create table

```
CREATE TABLE <tablename>  
( <col. name><col. type><col. properties>,  
  <col. name><col. type><col. properties>,  
  ... ..  
  <col. name><col. type><col. properties>);  
<col. properties> (separated by space):  
  NULL | NOT NULL  
  DEFAULT <value>  
  AUTO INCREMENT  
  PRIMARY KEY | <foreign key definition>
```

foreign key definition

```
ALTER TABLE boats  
ADD FOREIGN KEY (id_u)  
REFERENCES loaners(id)
```

```
ON DELETE SET NULL  
ON UPDATE CASCADE ;
```

CASCADE |
SET NULL |
RESTRICT |
NO ACTION
(last two are the same)

DEFAULT

OPTIONAL

(on the prev. slide, the part from FOREIGN can be added directly to line)

The application can delete foreign key using:

```
ALTER TABLE boats DROP FOREIGN KEY...
```

foreign key definition

More complete on: www.mysqltutorial.org/mysql-foreign-key/

Example from: www.w3schools.com/sql/sql_foreignkey.asp/:

```
CREATE TABLE orders  
(  
  o_id int NOT NULL PRIMARY KEY,  
  order_no int NOT NULL,  
  p_id int FOREIGN KEY REFERENCES persons(id)  
)
```

create pre-filled table

```
SELECT * FROM employee
WHERE room >= '316' AND room <= '316'
INTO employee316;
```

(will create table employee316;
if there are no WHERE clause, complete
table will be copied)

Don't forget to delete table using:

```
DROP TABLE employee316;
```

More examples on: www.w3schools.com/sql/sql_select_into.asp

CREATE VIEW

VIEW can speed up the SELECT commands:

```
CREATE VIEW test_v
AS SELECT Name,room FROM employee ;
```

```
DROP VIEW test_v ;
```

VIEW can be used same way as a table:

```
SELECT * FROM test_v ;
```

<http://www.mysqltutorial.org/create-sql-updatable-views.aspx>

<http://dev.mysql.com/doc/refman/5.0/en/create-index.html>

For developers...

If you like to create client like Micka, you need some other
commands. USE will select the database, SHOW can show
the list of available databases, tables etc.:

```
USE database_name;
```

```
SHOW TABLES;
```

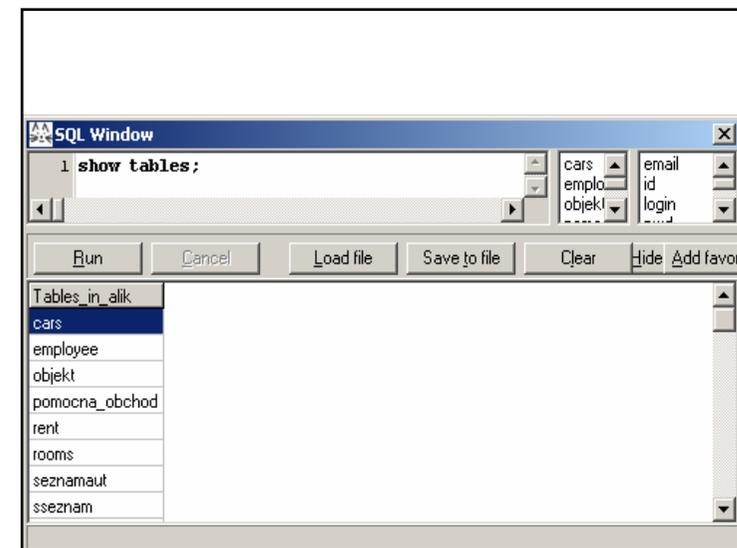
If you need to know the table structure, use:

```
DESCRIBE table_name;
```

More info on the address:

<http://dev.mysql.com/doc/refman/5.0/en/show-tables.html>

<http://dev.mysql.com/doc/refman/5.0/en/describe.html>

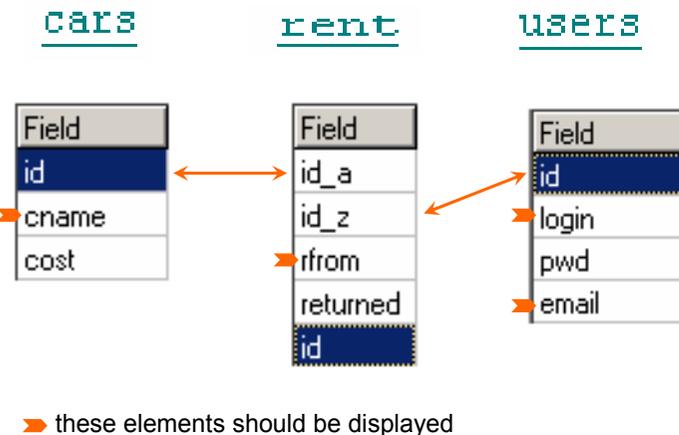


SQL Window

```
1 describe tempreg;
```

Field	Type	Null	Key	Default	Extra
login	varchar(33)				
pwd	varchar(33)				
email	varchar(133)				
rfrom	datetime			0000-00-00 00:00:00	
tmpkey	int(11)	YES		111	
id	int(11)		PRI	<NULL>	auto_increment

Connection table example



Connection table – SELECT and its result:

SQL Window

```
1 select cars.cname,rent.rfrom,
2 users.login,users.email
3 from cars,rent,users
4 where cars.id=rent.id_a
5 and rent.id_z=users.id
6 order by users.id ;
```

cname	rfrom	login	email
Renault 12	2011-04-21 00:00:00	vlk	vlk@email.cz
Renault 5	2011-04-21 00:00:00	novak	alois.novak@login.cz
Renault 8	2011-05-03 00:00:00	novak	alois.novak@login.cz
Audi A100	2011-04-25 12:33:00	novak	alois.novak@login.cz
Audi A100	2011-04-15 00:00:00	smith	john.smith@gggmail.com

Connection table – SELECT - note:

If you like to list only lines, where login is “novak”, you can add condition like

... and users.login = “novak” ...

complete SQL command:

```
select cars.cname,rent.rfrom,users.login,users.email
from cars,rent,users
where cars.id=rent.id_a and rent.id_z=users.id
and users.login="novak";
```

subselect

<http://www.tutorialspoint.com/sql/sql-sub-queries.htm>

- SELECT can return table or value
- so can be used instead of table or value

Long example:

```
select cust_l from
(select customers.login as cust_l,
sum(orders.pcs * stock.ppp) as howmany
from customers,orders,stock
where customers.id=orders.customer
and stock.id=orders.article
group by orders.customer order by howmany desc)
as ctable;
```

task was:
not list this column

alias for subselect is required by MySQL

copy from table to table

- First table with columns id, name, email
- Second table can contain more columns :-)

insert into students

```
(select
id+(select max(id) from students),
name, email
from anotherstudents
where someanothercondition);
```

max id as subselect

end of subselect

duplicity and alias for table

- The next simple example from somewhere on the internet will erase eventual duplicity names (for example, if you have two students with name Smith, this looks like a good idea to kick one off)

```
delete x
from students1 x
join students1 z on x.name = z.name
where x.id > z.id ;
```

(this keeps the MIN(ID); if you want MAX(ID), then just reverse the direction to x.id < z.id on the end of subselect)

<http://stackoverflow.com/questions/6205640/sql-select-all-records-not-selected-by-another-query>

select ... where ... in ...

The "in" condition can be used to test, if the value of property (SQL tests table row-by-row) is presented in the list of values (set, but typically column=property of another table). The another table can be got as another select, i.e. subselect.

In the next example, select will list rows from the first table only in the case, that the property name is not in the second table. Possibly accidentally - not any of the properties is a primary key, or at least unique property of a real object (as e-mail).

```
select * from students1
where name not in
(select name from students);
```

DCL: transaction

START TRANSACTION;

there can be any set of commands,
if we require that all of them are either successful,
or no one from them will be done.

then either:

COMMIT;

if all of the set of commands were
successful, it confirms them

or:

ROLLBACK;

if any of the commands fails,
it will return previous state

More complete on: <http://dev.mysql.com/doc/refman/5.1/en/commit.html>

Savepoint

SAVEPOINT *name-of-savepoint*;

... then we can play with the database; if we don't like
new database appearance, we can go back to any
savepoint, we have prepared, using command:

ROLLBACK TO *name-of-savepoint*;

... all the savepoints can be deleted,
for example by using command:

COMMIT;

or:

ROLLBACK;

More complete on: <http://dev.mysql.com/doc/refman/5.1/en/savepoint.html>

Literature ...

Many sources have been mentioned in text.

For the list of the commands, see

http://www.w3schools.com/sql/sql_quickref.asp

(incomplete, but nice table)

next: MS Access

then: <http://www1.fs.cvut.cz/cz/u12110/prt/dks/dks5.pdf>