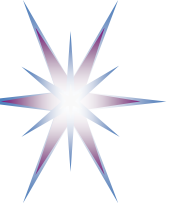


Big Data Infrastructure and Technologies

Practice A0 – Getting familiar with SQL

SQL Basics and Tools, Data Modeling

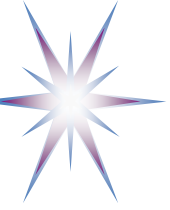
Dr. Yuri Demchenko
University of Amsterdam



Outline


- Part 1. Data modelling: 1NF, 2NF, 3NF
- Part 2. Basic SQL commands and scripting
- Part 3. Installing SQL database for practice: mySQL/LAMP on VirtualBox, Amazon Aurora

- Skip to slide #34 Part 2 for SQL exercises
- Use <http://w3schools.com/sql/> online tutorial for exercises



Practice and Assignments - Recommendations

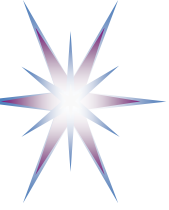
- Importance of following instructions
 - To achieve expected learning outcome
 - It is important to go beyond minimum required and reflect on experience
- Correctly naming files and providing ID (meta)data
 - Data management literacy is important
 - Not all are doing it correct/literate
- Correctly labeling you instances, buckets, snapshots
 - Few students did this
 - All other instance names are of wide variety...
- Self-learning is a key approach in IT and Computer Science
 - Find necessary information, tutorial, template



Reporting (if required) – Best practice

Report content

- Task, objectives
- Artefact
- Answer
- Analysis
- Conclusion

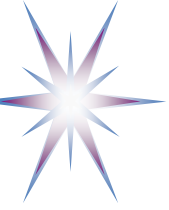


Part 1. Data Modelling

Why do you need a Data model?

The Datamodel is fundamental for consistency of the data used with the business applications and websites

- To structure information and processes
- Maintain compatibility between systems and applications
- Guarantee consistent integration of all information
- Maintain overall consistency of website
- Efficiency
- Scalability



Layered Data model

1. Conceptual design

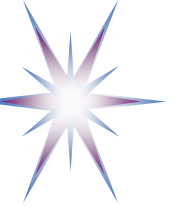
- Identify important entities and their relations

2. Logical design

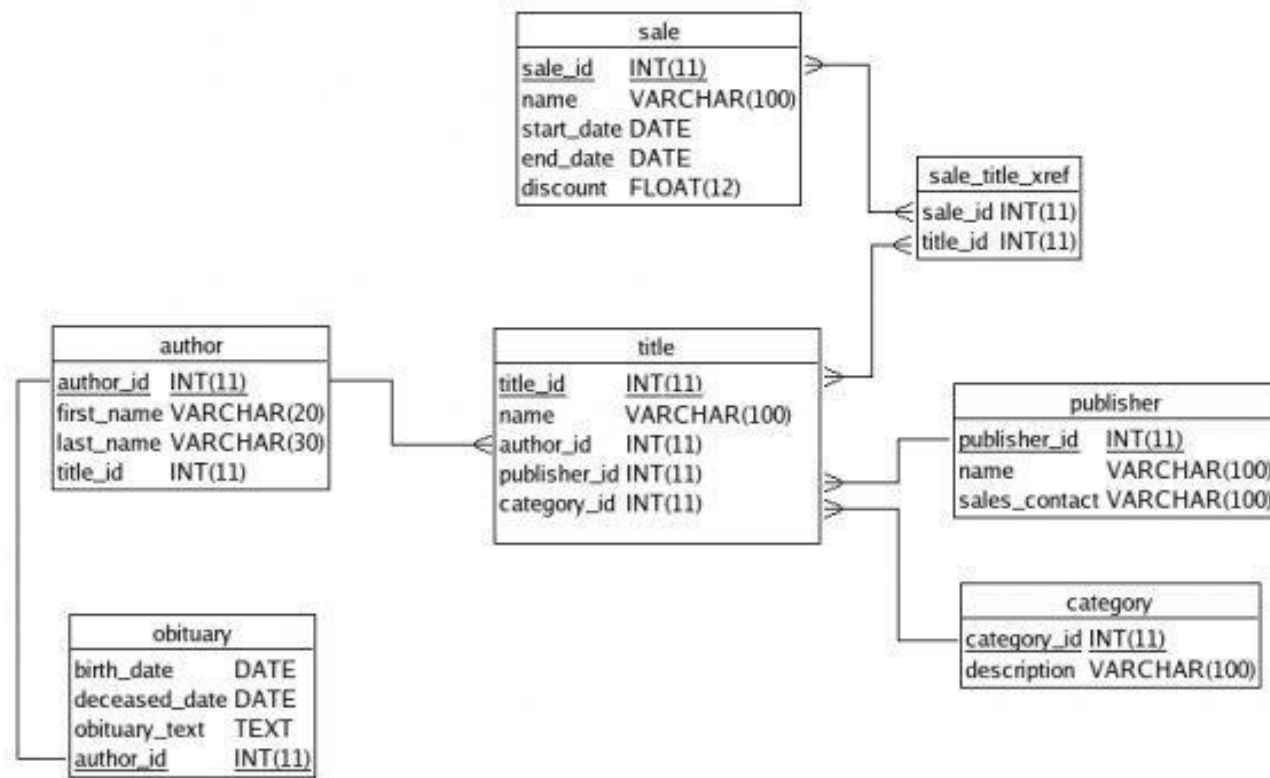
- Specification of attributes of entities

3. Physical design

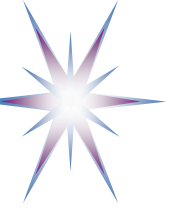
- Representation of the logical design with machines
 - In this course: “relational database”



Relational database – Relational datamodel



- Relational model: tables with references
- Optimisation through Indexes and data constraints
- Support for “Access control”
- Examples: **MySQL**, Postgres, Oracle, MS SQL Server, MS Access, ...



Relational Database design

- Tables (including columns and rows)
- Relations
- Indexes

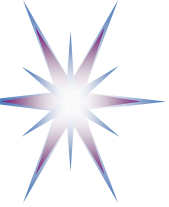


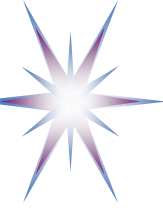
Table: columns and rows

Column or field

Table title								Property (attribute)
Property 1	Property 2	Property 3	Property 4	Property 5	Property 6	Property 7	Property 8	
Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	

Row or record or entity

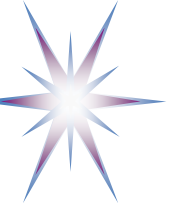
Element (data value)



Step-by-step Example: Book shop sell data

Books sell							
Author 1	Author 2	Title	ISBN	Price	Client name	Client address	Purchase date
Robin Nixon		Learning PHP, MySQL, and JavaScript	596157135	24.99	Don Johnson	4 New York Plaza, New York, NY 10004	Jan 03 2008
David Sklar	Adam Trachtenberg	PHP Cookbook	596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Danny Goodman		Dynamic HTML	596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Hugh E. Williams	David Lane	PHP and MySQL	596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
David Sklar	Adam Trachtenberg	PHP Cookbook	596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Rasmus Lerdorf	Kevin Tatroe & Peter MacIntyre	Programming PHP	596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

- There is a lot wrong with this table!



Good data model

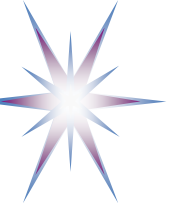
When is a data model "good"?

If there is:

- Logical reflection of entities in tables
- Relations between tables with indexes ("keys")

with the aim of:

- Retaining data integrity: trusted data
- Preventing inefficiencies in space/storage and time/performance in accessing data and working with data
- Consistent presentation of relations between entities
- Maintainable and scalable, in particular with the future growth



Good data model

When is a data model "good"?

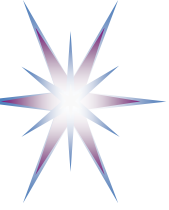
If there is:

- Logical ref
- Relations k

Good approach is continuously asking:

with the aim of “What questions/requests should my database (and application) answer”

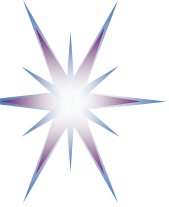
- Retaining data integrity: trusted data
- Preventing inefficiencies in space/storage and time/performance in accessing data and working with data
- Consistent presentation of relations between entities
- Maintainable and scalable, in particular with the future growth



What is the goal?

- Balance between a "pure" logical design and "practical" physical design.



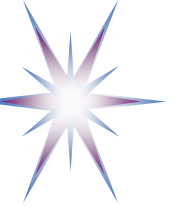


Normalisation

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Fourth normal form (4NF)
- Fifth normal form (5NF)

Advice:

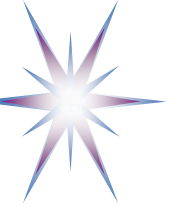
- Start from using standard approaches
- Use common sense as you build your understanding and experience



First Normal Form

A table is in the first normal form if:

- each column has a name and a suitable type
- order of rows or columns has no meaning
- there are no duplicates in rows or columns
- all columns have one value ("atomic")
- each row is unique (has a "primary key")



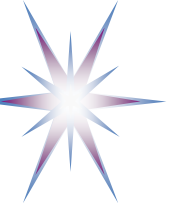
Example: Bookshop sell data

Different columns but
the same data (type)

Books sell							
Author 1	Author 2	Title	ISBN	Price	Client name	Client address	Purchase date
Robin Nixon		Learning PHP, MySQL, and JavaScript	596157135	24.99	Don Johnson	4 New York Plaza, New York, NY 10004	Jan 03 2008
David Sklar	Adam Trachtenberg	PHP Cookbook	596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Danny Goodman		Dynamic HTML	596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Hugh E. Williams	David Lane	PHP and MySQL	596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
David Sklar	Adam Trachtenberg	PHP Cookbook	596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Rasmus Lerdorf	Kevin Tatroe & Peter MacIntyre	Programming PHP	596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

Multiple values:
2 authors in one column

There is no primary key in this table: a customer could, for example, purchase multiple copies of a book on the same date



Example: Bookshop sell data

Books sell

Title	ISBN	Price	Client name	Client address	Purchase date
Learning PHP, MySQL, and JavaScript	596157135	24.99	Don Johnson	4 New York Plaza, New York, NY 10004	Jan 03 2008
PHP Cookbook	596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Dynamic HTML	596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
PHP and MySQL	596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
PHP Cookbook	596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Programming PHP	596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

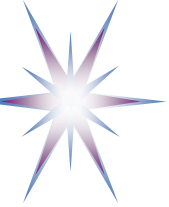
Authors

596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Authors moved to a separate table: a book written by several authors appears in several rows with the same ISBN, different author.

A relationship has been created between two tables.

In this case, n:m ("many to many"): a book can have multiple authors, and an author may have written several books.



Second Normal Form (2NF)

A table is in the second normal form if:

- The table is in the first normal form, and
- There are no repeating subsets in the rows.

In other words:

- Elements that belong to each other are placed in the same table.

Or vice versa:

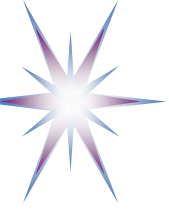
- Elements that do not belong together are placed in different tables.

In other words:

- Each table represents entities with only the attributes directly associated with them. Reference is made to related entities.

One last time, but now more formally:

- All non-primary-key fields are completely dependent on the primary key.



Example: Bookshop sell data

Books sell

Title	ISBN	Price	Client name	Client address	Purchase date
Learning PHP, MySQL, and JavaScript	596157135	24.99	Don Johnson	4 New York Plaza, New York, NY 10004	Jan 03 2008
PHP Cookbook	596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Dynamic HTML	596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
PHP and MySQL	596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
PHP Cookbook	596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Programming PHP	596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

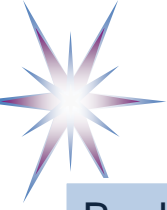
Authors

596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Repetition of the same data (: subsets).

- To split to separate tables:
Book titles, Customers, Purchases
- But then mutual references must be made to maintain the relationships!

These repetitions are "keys":
we allow them.



Example: Bookshop sell data

Books

ISBN	Title	Price
596157135	Learning PHP, MySQL, and JavaScript	24.99
596101015	PHP Cookbook	44.99
596527403	Dynamic HTML	59.99
596005436	PHP and MySQL	44.95
596006815	Programming PHP	39.99

Books sell

Client numr	ISBN	Date
1	596157135	Jan 03 2008
2	596101015	Mar 03 2009
3	596527403	Dec 19 2008
3	596101015	Dec 19 2008
4	596527403	Dec 19 2008
5	596006815	Jan 16 2009

Authors

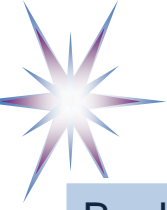
ISBN	Author
596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Clients (customers)

Client number	Name	Address	Place	State	Zip
1	Don Johnson	4 New York Plaza	New York	NY	10004
2	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
3	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
4	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
5	David Miller	3647 Cedar Lane	Waltham	MA	02154

Book titles, client numbers, purchases also moved to separate tables

Client address are split (atomicity)



Example: Bookshop sell data

Books

ISBN	Title	Price
596157135	Learning PHP, MySQL, and JavaScript	24.99
596101015	PHP Cookbook	44.99
596527403	Dynamic HTML	59.99
596005436	PHP and MySQL	44.95
596006815	Programming PHP	39.99

Books sell

Client numr	ISBN	Date
1	596157135	Jan 03 2008
2	596101015	Mar 03 2009
3	596527403	Dec 19 2008
3	596101015	Dec 19 2008
4	596527403	Dec 19 2008
5	596006815	Jan 16 2009

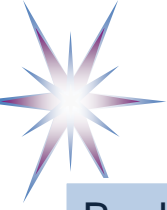
Authors

ISBN	Author
596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Clients (customers)

Client number	Name	Address	Place	State	Zip
1	Don Johnson	4 New York Plaza	New York	NY	10004
2	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
3	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
4	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
5	David Miller	3647 Cedar Lane	Waltham	MA	02154

These repetitions are "keys": we allow them.



Example: Bookshop sell data

Books

ISBN	Title	Price
596157135	Learning PHP, MySQL, and JavaScript	24.99
596101015	PHP Cookbook	44.99
596527403	Dynamic HTML	59.99
596005436	PHP and MySQL	44.95
596006815	Programming PHP	39.99

Books sell

Client numr	ISBN	Date
1	596157135	Jan 03 2008
2	596101015	Mar 03 2009
3	596527403	Dec 19 2008
3	596101015	Dec 19 2008
4	596527403	Dec 19 2008
5	596006815	Jan 16 2009

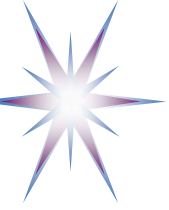
Authors

ISBN	Author
596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Clients (customers)

Client number	Name	Address	Place	State	Zip
1	Don Johnson	4 New York Plaza	New York	NY	10004
2	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
3	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
4	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
5	David Miller	3647 Cedar Lane	Waltham	MA	02154

These tables are not necessary 1NF: duplicates must be prevented (in other words: there is no primary key).
Solution: to add key column.



Third Normal Form (3NF)

A table is in third normal form if:

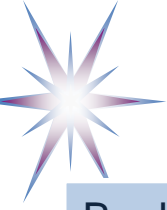
- The table is in second normal form, and
- A property can not be obtained from other properties.

In other words:

- All non-key properties are independent of all other non-key properties

Once again, but now more formally:

- There are no transitive dependencies



Example: Bookshop sell data

Books

ISBN	Title	Price
596157135	Learning PHP, MySQL, and JavaScript	24.99
596101015	PHP Cookbook	44.99
596527403	Dynamic HTML	59.99
596005436	PHP and MySQL	44.95
596006815	Programming PHP	39.99

Books sell

Client numr	ISBN	Date
1	596157135	Jan 03 2008
2	596101015	Mar 03 2009
3	596527403	Dec 19 2008
3	596101015	Dec 19 2008
4	596527403	Dec 19 2008
5	596006815	Jan 16 2009

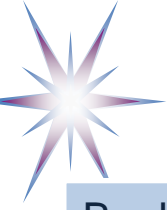
Authors

ISBN	Author
596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Clients (customers)

Client number	Name	Address	Place	State	Zip
1	Don Johnson	4 New York Plaza	New York	NY	10004
2	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
3	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
4	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
5	David Miller	3647 Cedar Lane	Waltham	MA	02154

Zip is directly related to State and place/city. F.e., Postcode table in Nederland.



Example: Bookshop sell data

Books

ISBN	Title	Price
596157135	Learning PHP, MySQL, and JavaScript	24.99
596101015	PHP Cookbook	44.99
596527403	Dynamic HTML	59.99
596005436	PHP and MySQL	44.95
596006815	Programming PHP	39.99

Books sell

Client numr	ISBN	Date
1	596157135	Jan 03 2008
2	596101015	Mar 03 2009
3	596527403	Dec 19 2008
3	596101015	Dec 19 2008
4	596527403	Dec 19 2008
5	596006815	Jan 16 2009

Authors

ISBN	Author
596157135	Robin Nixon
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E. Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

Clients (customers)

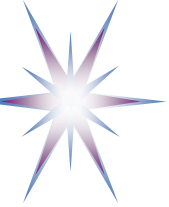
Client number	Name	Address	Zip
1	Don Johnson	4 New York Plaza	10004
2	Emma Brown	1565 Rainbow Road	90014
3	Darren Ryder	4758 Emily Drive	23219
4	Earl B. Thurston	862 Gregory Lane	40601
5	David Miller	3647 Cedar Lane	02154

Zip codes

Zip	Place	State
10004	New York	NY
90014	Los Angeles	CA
23219	Richmond	VA
40601	Frankfort	KY
02154	Waltham	MA

New table Zipcodes.

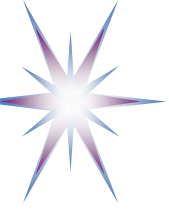
Reason not to do this:
Extra table costs money
and performance.



Another example

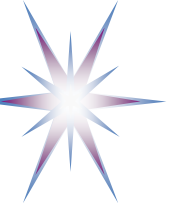
- Computable attributes should not be placed in the table
 - Table below is not 3NF: TotalIncBTW can be calculated from TotalExBTW

Orders			
Order Nr	Client Nmr	TotalExBTW	TotalIncBTW
1987123	2121	12,34	14,68
2234789	2135	100,00	119,00
3231414	2652	1,24	1,48
4456456	5323	23,28	27,70
5346778	3356	32,43	38,59



How to define tables

- One table for each type of object
- Provided that:
 - Relationship is not 1: 1
 - 2 or more attributes required per object



Example: Tables

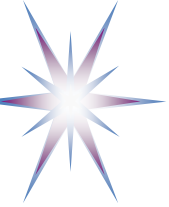
- Relation between buildings and apartments 1:n

Apartment Buildings

Address	Year built	Service
Lodewijk laan 304-450	1970	Roof asphalt
Braambos 525-630	1984	CV kettle renew
Kennedy laan 251-365	2001	

Tenants

Name	House number	Building
Jaansen	305	
Pitersen	306	
Klaassen	252	

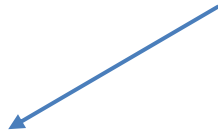


Example: Tables

- Relation between buildings and apartments 1:n

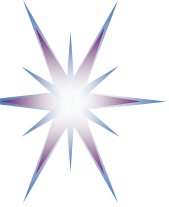
Apartment Buildings				Tenants		
ID	Address	Year built	Service	Name	House number	Building
1	Lodewijk laan 304-450	1970	Roof asfalt	Jaansen	305	1
2	Braambos 525-630	1984	CV kettle renew	Pitersen	306	1
3	Kennedy laan 251-365	2001		Klaassen	252	3

“primary key”



“foreign key”
(reference key)





MySQL Command Line Shell (CLI)

Login to the server with SSH:

```
$ mysql -u <username> -p
```

Enter password: *<password> e.g. Ed!sonD5*

Mysql can connect to remote server

```
$ mysql -h hostname -u username -p
```

Exit – quit or exit

```
$ mysql exit Bye
```

List databases:

```
mysql > show databases;
```

Select database:

```
mysql > use <databasename>;
```

List tables in a selected database:

```
mysql > show tables;
```

```
mysql > drop table <tablename>;
```

SQL statements terminated with ; semicolon:

```
mysql > select * from <table>;
```

```
bitnami@debian: ~/Desktop/mysqlsampledatabase
File Edit View Search Terminal Help
+-----+
| Database |
+-----+
| information_schema |
| classicmodels |
| mysql |
| performance_schema |
| sys |
| testdb01 |
+-----+
6 rows in set (0.00 sec)

mysql> use testdb01
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Product_Type
-> ;
+-----+-----+
| PRODUCT_TYPE_CD | NAME |
+-----+-----+
| ACCOUNT | Customer Accounts |
| INSURANCE | Insurance Offerings |
| LOAN | Individual and Business Loans |
+-----+-----+
3 rows in set (0.00 sec)

mysql> C
```

All SQL command are case insensitive
However, attribute may be case sensitive in some MySQL tools.

Sample MySQL Database testdb01 customers

Data model

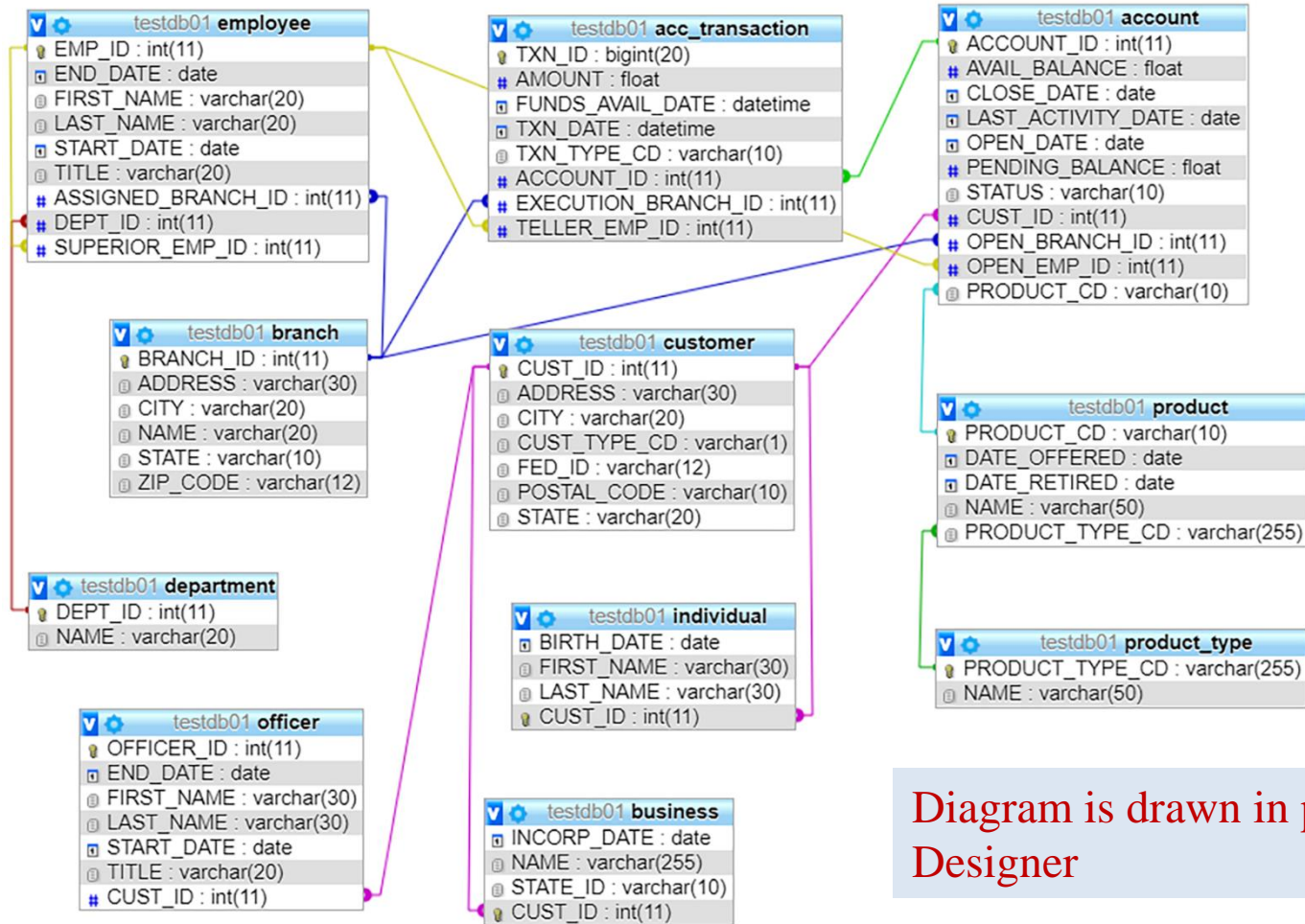


Diagram is drawn in phpMyAdmin Designer

- Learning database testdb01 is available in <http://voorbeeldcode.science.uva.nl>

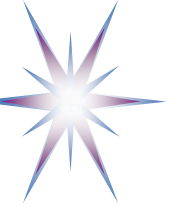
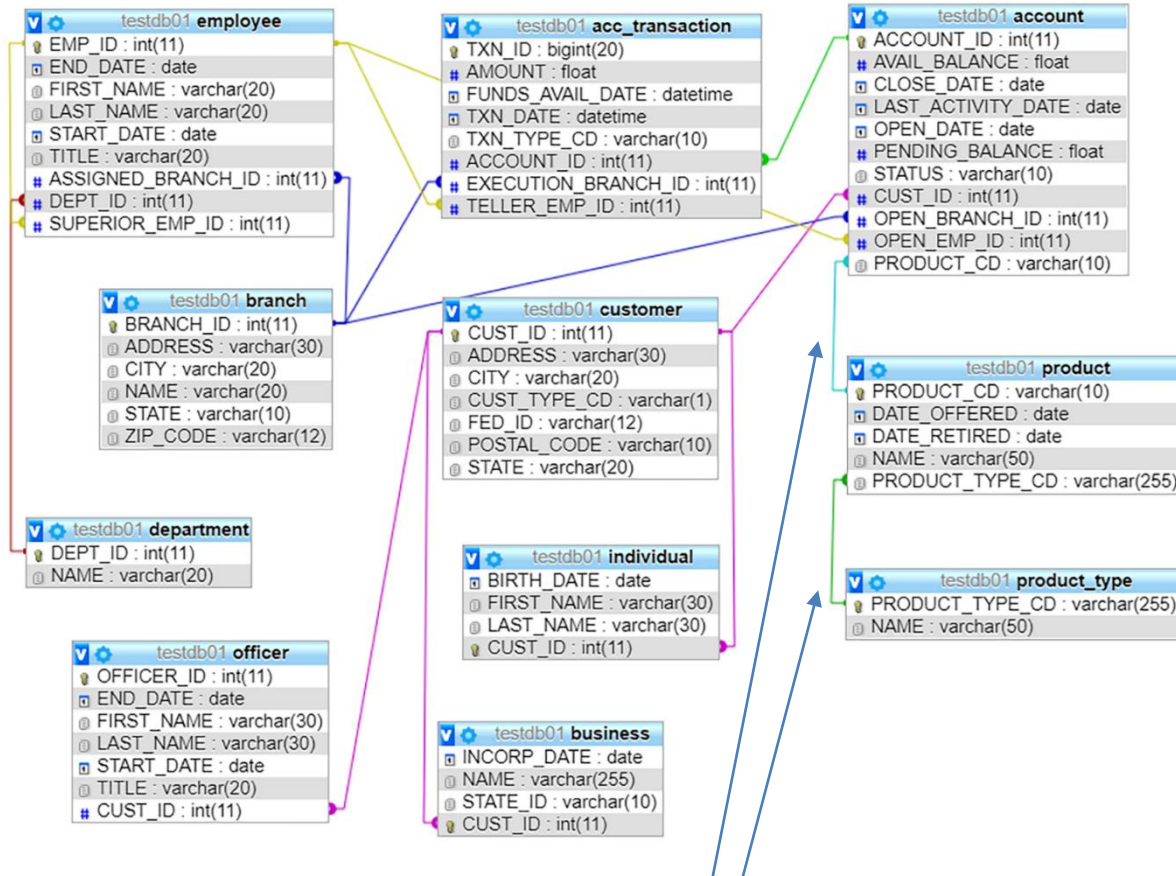


Table keys



- Each Product has Product_type
- Each Product_type may have multiple Products
- Each Product may be linked to multiple Accounts

- **Key or index** unlocks access to table
 - Each key consists of one or more fields, or field prefix
- A **candidate key** is a field, or combination of fields, that uniquely identifies a database record and a primary key (PK).
- A **primary key (PK)** is a candidate key that has been designated to identify unique records in the table throughout the database structure.
- **Foreign key** is a primary key of one table used in another table
- **Relation between two tables** is created by assigning a common field to the two tables
 - This common field must be a primary key to one table
- Foreign keys also allow **cascading deletes and updates**

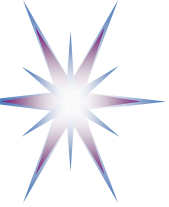
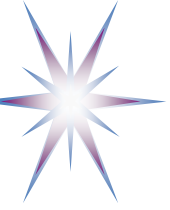


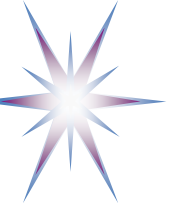
Table details: column/data types

STRING	CHAR	BYTE STRING	BINARY	Length/Values Default Attributes Index NullAUTO_INCREMENT
	VARCHAR		VARBINARY	
	TINYTEXT		TINYBLOB	
	TEXT		BLOB	
	MEDIUMTEXT		MEDIUMBLOB	
	LONGTEXT		LOB	
INTEGER	TINYINT	DATE & TIME	DATE	VARCHAR can hold numbers and characters of variable length
	SMALLINT		DATETIME	
	MEDIUMINT		YEAR	
	INT		TIME	
	BIGINT		TIMESTAMP	
DECIMAL	FLOAT	SETS	ENUM	Importance of the correct column type selection <ul style="list-style-type: none">E.g., maximum valueCan cause service disruption
	DOUBLE		SET	
	DECIMAL			
MySQL		Data types		



Part 2. SQL commands practice

- Use <http://w3schools.com/sql/> for exercises

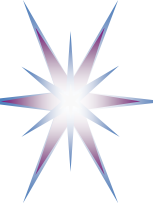


SQL: Structured Query Language

Most used commands:

- Select row: `SELECT ... ;`
- Change row: `UPDATE ... ;`
- Add row: `INSERT ... ;`
- Delete row: `DELETE ... ;`

<http://dev.mysql.com/doc/refman/5.5/en/sql-syntax-data-manipulation.html>



Template exercises – General commands

Login to the server with SSH:

```
$ mysql -u <username> -p
```

Enter password: <password>

Mysql can connect to remote server

```
$ mysql -h hostname -u username -p
```

Exit – quit or exit

```
$ mysql exit Bye
```

List databases:

```
mysql > show databases;
```

Select database:

```
mysql > use <databasename>;
```

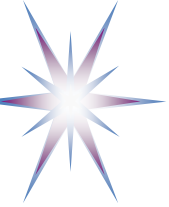
List tables in a selected database:

```
mysql > show tables;
```

```
mysql > drop table <tablename>;
```

SQL statements terminated with ; semicolon:

```
mysql > select * from <table>;
```



INSERT

Add a row

Two methods, e.g.:

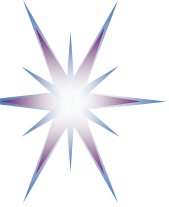
1. **INSERT INTO users**
SET username="jansen",
password="secret#";
2. **INSERT INTO users**
(username, password)
VALUES ("jansen", "secret#");

```
mysql> SELECT * FROM Product_type;
```

```
+-----+-----+
| PRODUCT_TYPE_CD | NAME                                |
+-----+-----+
| ACCOUNT          | Customer Accounts                  |
| INSURANCE        | Insurance Offerings                |
| LOAN             | Individual and Business Loans      |
+-----+-----+
3 rows in set (0.01 sec)
```

```
mysql> INSERT INTO Product_type
      -> SET Product_type_cd="RISK", NAME="Ris assessment";
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO Product_type SET Product_type_cd="RISK",
NAME="Ris assessment
";sql> SET Product_type_cd="RISK", NAME="Ris assessment";
mysql> SELECT * FROM Product_type;
+-----+-----+
| PRODUCT_TYPE_CD | NAME                                |
+-----+-----+
| ACCOUNT          | Customer Accounts                  |
| INSURANCE        | Insurance Offerings                |
| LOAN             | Individual and Business Loans      |
| RISK             | Ris assessment                     |
+-----+-----+
4 rows in set (0.00 sec)
```



UPDATE

Change one or more rows

Example:

- UPDATE users
SET password="secret#"
WHERE username="jansen"
LIMIT 1;

Only one row
to be changed

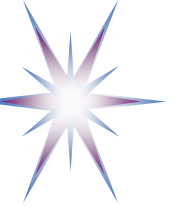
```
mysql> SELECT * FROM Product_type;
+-----+-----+
| PRODUCT_TYPE_CD | NAME                                |
+-----+-----+
| ACCOUNT          | Customer Accounts                  |
| INSURANCE        | Insurance Offerings                |
| LOAN             | Individual and Business Loans      |
+-----+-----+
3 rows in set (0.01 sec)

mysql> INSERT INTO Product_type
-> SET Product_type_cd="RISK", NAME="Ris assessment";
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Product_type SET Product_type_cd="RISK",
NAME="Ris assessment
";sql> SET Product_type_cd="RISK", NAME="Ris assessment";
mysql> SELECT * FROM Product_type;
+-----+-----+
| PRODUCT_TYPE_CD | NAME                                |
+-----+-----+
| ACCOUNT          | Customer Accounts                  |
| INSURANCE        | Insurance Offerings                |
| LOAN             | Individual and Business Loans      |
| RISK             | Ris assessment                     |
+-----+-----+
4 rows in set (0.00 sec)

mysql> UPDATE Product_type
-> SET NAME="Risk assessment"
-> WHERE Product_type_cd="RISK"
-> LIMIT 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Product_type;
+-----+-----+
| PRODUCT_TYPE_CD | NAME                                |
+-----+-----+
| ACCOUNT          | Customer Accounts                  |
| INSURANCE        | Insurance Offerings                |
| LOAN             | Individual and Business Loans      |
| RISK             | Risk assessment                    |
+-----+-----+
4 rows in set (0.00 sec)
```



DELETE

Delete one or more rows

Example:

- DELETE FROM users
WHERE username="jansen"
LIMIT 1;

Only one row
to be changed

```
mysql> SELECT * FROM Product_type;
```

PRODUCT_TYPE_CD	NAME
ACCOUNT	Customer Accounts
INSURANCE	Insurance Offerings
LOAN	Individual and Business Loans
RISK	Risk assessment

4 rows in set (0.00 sec)

```
mysql> DELETE FROM Product_type WHERE Product_type_cd="RISK"  
LIMIT 1;
```

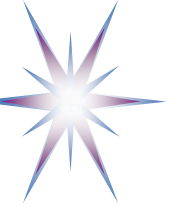
Query OK, 1 row affected (0.00 sec)

```
mysql> SELECT * FROM Product_type;
```

PRODUCT_TYPE_CD	NAME
ACCOUNT	Customer Accounts
INSURANCE	Insurance Offerings
LOAN	Individual and Business Loans

3 rows in set (0.00 sec)

```
mysql>
```



SELECT

```
mysql> help select
```

```
Name: 'SELECT'
```

```
Description:
```

```
Syntax:
```

```
SELECT
```

```
[ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
[LIMIT [{offset,} row_count | row_count OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
      [CHARACTER SET charset_name]
      export_options
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

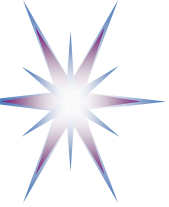
```
[...]
```

Fetch information from database

- The most widely used and the most rich instruction

Example:

- SELECT name, surname
FROM users_table
WHERE userid= 1016123;

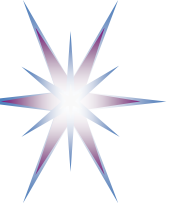


SELECT

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp;
```

Alias

```
bitnami@debian: ~  
File Edit View Search Terminal Help  
+-----+-----+-----+  
| 1 | Michael | Smith | 3 |  
| 2 | Susan   | Barker | 3 |  
| 3 | Robert  | Tyler  | 3 |  
| 4 | Susan   | Hawthorne | 1 |  
| 5 | John    | Gooding | 2 |  
| 6 | Helen   | Fleming | 1 |  
| 7 | Chris   | Tucker  | 1 |  
| 8 | Sarah   | Parker  | 1 |  
| 9 | Jane    | Grossman | 1 |  
| 10 | Paula   | Roberts | 1 |  
| 11 | Thomas  | Ziegler | 1 |  
| 12 | Samantha | Jameson | 1 |  
| 13 | John    | Blake   | 1 |  
| 14 | Cindy   | Mason   | 1 |  
| 15 | Frank   | Portman | 1 |  
| 16 | Theresa | Markham | 1 |  
| 17 | Beth    | Fowler  | 1 |  
| 18 | Rick    | Tulman  | 1 |  
+-----+-----+-----+  
18 rows in set (0.00 sec)  
  
mysql> mysql> SELECT Emp.Emp_Id ,Emp.First_Name  
,Emp.Last_Name ,Emp.Dept_Id FROM  
Emplo  
mysql>
```



SELECT – LIMIT

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp  
LIMIT 10;
```

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp  
LIMIT 10, 5;
```

Other SQL versions use TOP attribute

```
SELECT TOP 10 * FROM Badges  
WHERE name LIKE 'a%';
```

```
mysql> Select Emp.Emp_Id, Emp.First_Name, Emp.Last_Name,  
Emp.Dept_Id From   Empl  
oyee Emp LIMIT 10;
```

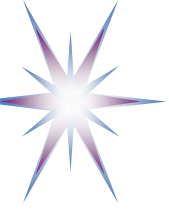
Emp_Id	First_Name	Last_Name	Dept_Id
1	Michael	Smith	3
2	Susan	Barker	3
3	Robert	Tyler	3
4	Susan	Hawthorne	1
5	John	Gooding	2
6	Helen	Fleming	1
7	Chris	Tucker	1
8	Sarah	Parker	1
9	Jane	Grossman	1
10	Paula	Roberts	1

10 rows in set (0.00 sec)

```
mysql> Select Emp.Emp_Id, Emp.First_Name, Emp.Last_Name,  
Emp.Dept_Id From   Empl  
oyee Emp LIMIT 10,5;
```

Emp_Id	First_Name	Last_Name	Dept_Id
11	Thomas	Ziegler	1
12	Samantha	Jameson	1
13	John	Blake	1
14	Cindy	Mason	1
15	Frank	Portman	1

5 rows in set (0.00 sec)



SELECT – WHERE ... LIKE

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp;
```

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp  
WHERE  Emp.Last_Name  
LIKE  'T%';
```

```
mysql> Select Emp.Emp_Id, Emp.First_Name,  
Emp.Last_Name, Emp.Dep  
t_Id From Employee Emp;
```

Emp_Id	First_Name	Last_Name	Dept_Id
1	Michael	Smith	3
2	Susan	Barker	3
3	Robert	Tyler	3
4	Susan	Hawthorne	1
5	John	Gooding	2
6	Helen	Fleming	1
7	Chris	Tucker	1
8	Sarah	Parker	1
9	Jane	Grossman	1
10	Paula	Roberts	1
11	Thomas	Ziegler	1
12	Samantha	Jameson	1
13	John	Blake	1
14	Cindy	Mason	1
15	Frank	Portman	1
16	Theresa	Markham	1
17	Beth	Fowler	1
18	Rick	Tulman	1

18 rows in set (0.00 sec)

```
mysql> Select Emp.Emp_Id, Emp.First_Name, Emp.Last_Name,  
Emp.Dept_Id  
-> From Employee Emp WHERE Emp.Last_Name LIKE "T%";
```

Emp_Id	First_Name	Last_Name	Dept_Id
3	Robert	Tyler	3
7	Chris	Tucker	1
18	Rick	Tulman	1

3 rows in set (0.00 sec)



SELECT – ORDER BY

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp  
ORDER BY Emp.Last_Name;
```

```
Select Emp.Emp_Id,  
       Emp.First_Name,  
       Emp.Last_Name,  
       Emp.Dept_Id  
From   Employee Emp  
ORDER BY Emp.Last_Name DESC  
LIMIT 10;
```

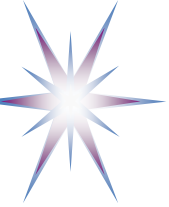
```
mysql> Select Emp.Emp_Id,      Emp.First_Name,  
Emp.Last_Name,      Emp.Dep  
t_Id From      Employee Emp ORDER BY Emp.Last_Name;
```

Emp_Id	First_Name	Last_Name	Dept_Id
2	Susan	Barker	3
13	John	Blake	1
6	Helen	Fleming	1
17	Beth	Fowler	1
5	John	Gooding	2
9	Jane	Grossman	1
4	Susan	Hawthorne	1
12	Samantha	Jameson	1
16	Theresa	Markham	1
14	Cindy	Mason	1
8	Sarah	Parker	1
15	Frank	Portman	1
10	Paula	Roberts	1
1	Michael	Smith	3
7	Chris	Tucker	1
18	Rick	Tulman	1
3	Robert	Tyler	3
11	Thomas	Ziegler	1

18 rows in set (0.00 sec)

```
mysql> Select Emp.Emp_Id,      Emp.First_Name,  
Emp.Last_Name,      Emp.Dep  
t_Id From      Employee Emp ORDER BY Emp.Last_Name Desc  
LIMIT 10;
```

Emp_Id	First_Name	Last_Name	Dept_Id
11	Thomas	Ziegler	1
3	Robert	Tyler	3
18	Rick	Tulman	1
7	Chris	Tucker	1
1	Michael	Smith	3
10	Paula	Roberts	1
15	Frank	Portman	1
8	Sarah	Parker	1
14	Cindy	Mason	1
16	Theresa	Markham	1



SELECT - DISTINCT

SELECT DISTINCT

Product_Type_Cd
from Product;

Selects only distinct,
not repeating records

```
mysql> Select Pro.Product_Cd  
->      ,Pro.Name  
->      ,Pro.Product_Type_Cd  
-> From    Product Pro;
```

Product_Cd	Name	Product_Type_Cd
AUT	auto loan	LOAN
BUS	business line of credit	LOAN
CD	certificate of deposit	ACCOUNT
CHK	checking account	ACCOUNT
MM	money market account	ACCOUNT
MRT	home mortgage	LOAN
SAV	savings account	ACCOUNT
SBL	small business loan	LOAN

8 rows in set (0.00 sec)

```
mysql> Select Pro.Product_Type_Cd from Product Pro;
```

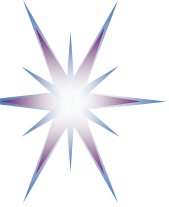
Product_Type_Cd
ACCOUNT
ACCOUNT
ACCOUNT
ACCOUNT
LOAN
LOAN
LOAN
LOAN

8 rows in set (0.00 sec)

```
mysql> Select Distinct Pro.Product_Type_Cd from Product Pro;
```

Product_Type_Cd
ACCOUNT
LOAN

2 rows in set (0.00 sec)



SELECT - WHERE

```
mysql> Select * From Product Pro Where Pro.Product_Type_Cd = 'LOAN';
```

PRODUCT_CD	DATE_OFFERED	DATE_RETIRED	NAME	PRODUCT_TYPE_CD
AUT	2000-01-01	NULL	auto loan	LOAN
BUS	2000-01-01	NULL	business line of credit	LOAN
MRT	2000-01-01	NULL	home mortgage	LOAN
SBL	2000-01-01	NULL	small business loan	LOAN

```
4 rows in set (0.00 sec)
```

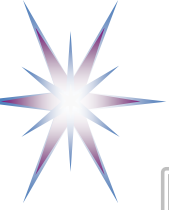
```
mysql> Select Product_cd, Date_Offered, Name, Product_type_cd From Product Pro Where Pro.Product_Type_Cd = 'LOAN';
```

Product_cd	Date_Offered	Name	Product_type_cd
AUT	2000-01-01	auto loan	LOAN
BUS	2000-01-01	business line of credit	LOAN
MRT	2000-01-01	home mortgage	LOAN
SBL	2000-01-01	small business loan	LOAN

```
4 rows in set (0.00 sec)
```

Select * From Product Pro Where Pro.Product_Type_Cd = 'LOAN';

Select Product_cd, Date_Offered, Name, Product_type_cd From Product Pro Where Pro.Product_Type_Cd = 'LOAN';



SELECT – WHERE (case sensitive)

```
mysql> Select * From Product Pro Where Pro.Product_Type_Cd = 'LOAN';
```

PRODUCT_CD	DATE_OFFERED	DATE_RETIRED	NAME	PRODUCT_TYPE_CD
AUT	2000-01-01	NULL	auto loan	LOAN
BUS	2000-01-01	NULL	business line of credit	LOAN
MRT	2000-01-01	NULL	home mortgage	LOAN
SBL	2000-01-01	NULL	small business loan	LOAN

4 rows in set (0.00 sec)

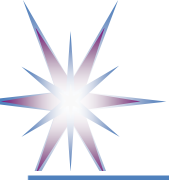
```
mysql> Select Product_cd, Date_Offered, Name, Product_type_cd From Product Pro Where Pro.Product_Type_Cd = 'LOAN';
```

Product_cd	Date_Offered	Name	Product_type_cd
AUT	2000-01-01	auto loan	LOAN
BUS	2000-01-01	business line of credit	LOAN
MRT	2000-01-01	home mortgage	LOAN
SBL	2000-01-01	small business loan	LOAN

4 rows in set (0.00 sec)

Select * From product Pro Where Pro.product_type_cd = 'LOAN';

Select product_cd, date_offered, name, product_type_cd
From product Pro Where Pro.product_type_cd = 'LOAN';

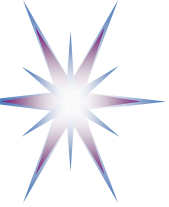


SELECT – Aggregate functions

Name	Description
AVG()	Return the average value of the argument
BIT_AND()	Return bitwise and
BIT_OR()	Return bitwise or
BIT_XOR()	Return bitwise xor
COUNT(DISTINCT)	Return the count of a number of different values
COUNT()	Return a count of the number of rows returned
GROUP_CONCAT()	Return a concatenated string
MAX()	Return the maximum value
MIN()	Return the minimum value
STD()	Return the population standard deviation
STDDEV_POP()	Return the population standard deviation
STDDEV_SAMP()	Return the sample standard deviation
STDDEV()	Return the population standard deviation
SUM()	Return the sum
VAR_POP()	Return the population standard variance
VAR_SAMP()	Return the sample variance
VARIANCE()	Return the population standard variance

Examples

- `SELECT MAX(temp)FROM temperature;`
- `SELECT AVG(temp)FROM temperature;`



SELECT – COUNT (*)

SELECT COUNT(*)
From Employee Emp
WHERE Emp.Dept_Id='1';

SELECT COUNT(*)
From Employee
WHERE Dept_Id
IS NOT NULL;

SELECT COUNT(*)
From Employee
WHERE Dept_Id !=0;

```
mysql> Select Emp.Emp  
Emp.Last_Name,  
t_Id From Employee  
Emp.Last_Name;
```

Emp_Id	First_Name
2	Susan
13	John
6	Helen
17	Beth
5	John
9	Jane
4	Susan
12	Samantha
16	Theresa
14	Cindy
8	Sarah
15	Frank
10	Paula
1	Michael
7	Chris
18	Rick
3	Robert
11	Thomas

```
18 rows in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) From Employee Emp WHERE  
Emp.Dept_Id='1';
```

COUNT(*)
14

```
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) From Employee WHERE Dept_Id IS NOT  
NULL;
```

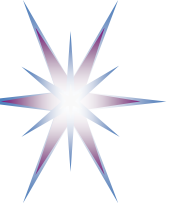
COUNT(*)
18

```
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) From Employee WHERE Dept_Id != 0;
```

COUNT(*)
18

```
1 row in set (0.00 sec)
```



SELECT – GROUP BY

```
SELECT dept_id, count(*)  
From employee  
GROUP BY dept_id;
```

```
SELECT dept_id, count(*)  
From employee  
GROUP BY dept_id  
HAVING dept_id != '1';
```

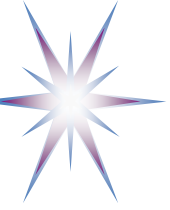
```
SELECT dept_id, count(*)  
From employee  
GROUP BY dept_id  
HAVING Count(*) < 5;
```

```
mysql> SELECT Dept_Id, count(*) From Employee GROUP BY  
Dept_Id HAVING Dept_I  
d != 1;  
+-----+-----+  
| Dept_Id | count(*) |  
+-----+-----+  
| 2 | 1 |  
| 3 | 3 |  
+-----+-----+  
2 rows in set (0.01 sec)
```

```
mysql> SELECT Dept_Id, count(*)  
-> From Employee  
-> GROUP BY Dept_Id  
-> HAVING Count(*) < 5;  
+-----+-----+  
| Dept_Id | count(*) |  
+-----+-----+  
| 2 | 1 |  
| 3 | 3 |  
+-----+-----+  
2 rows in set (0.00 sec)
```

```
mysql> Select Emp  
Emp.Last_Name,  
t_Id From Empl  
Emp.Last_Name;  
+-----+-----+  
| Emp_Id | First  
+-----+-----+  
| 2 | Susan  
| 13 | John  
| 6 | Helen  
| 17 | Beth  
| 5 | John  
| 9 | Jane  
| 4 | Susan  
| 12 | Samantha  
| 16 | Theresa  
| 14 | Cindy  
| 8 | Sarah  
| 15 | Frank  
| 10 | Paula  
| 1 | Michael  
| 7 | Chris  
| 18 | Rick  
| 3 | Robert  
| 11 | Thomas  
+-----+-----+  
18 rows in set (0.00 sec)
```

```
| Grossman | 1 |  
| Hawthorne | 1 |  
| Jameson | 1 |  
| Markham | 1 |  
| Mason | 1 |  
| Parker | 1 |  
| Portman | 1 |  
| Roberts | 1 |  
| Smith | 3 |  
| Tucker | 1 |  
| Tulman | 1 |  
| Tyler | 3 |  
| Ziegler | 1 |
```



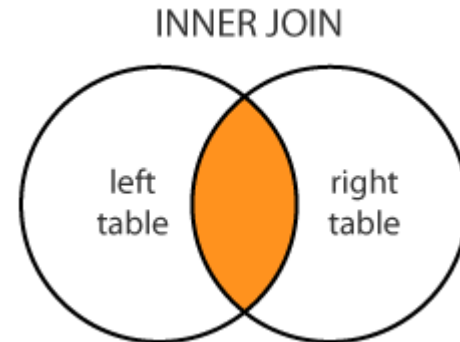
SELECT – JOIN (INNER JOIN)

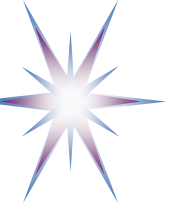
```
SELECT groups.id, title FROM groups  
JOIN projects ON groups.project_id=  
projects.id;
```

```
SELECT firstname, lastname, group_id,  
title  
FROM students  
JOIN groups ON students.group_id=  
groups.id JOIN projects  
ON groups.project_id= projects.id  
ORDER BY title, group_id;
```

```
SELECT title, COUNT(groups.id)  
FROM groups  
JOIN projects ON  
groups.project_id= projects.id  
GROUP BY title ORDER BY  
COUNT(groups.id) DESC;
```

```
SELECT {column-names}  
FROM table-name1  
JOIN table-name2  
ON column-name1 = column-name2  
WHERE condition
```

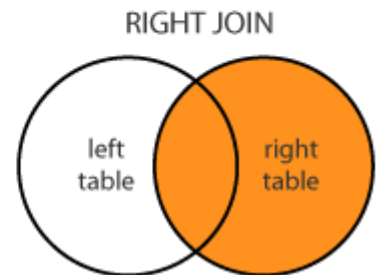
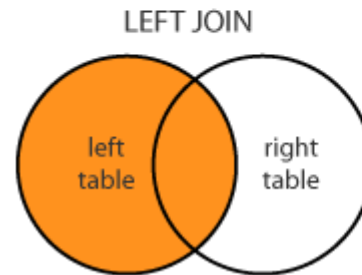
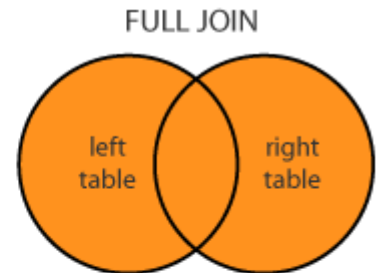
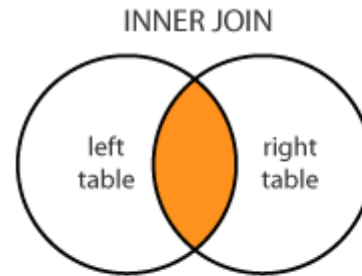


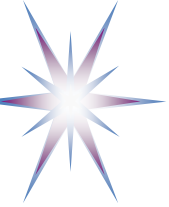


SELECT – LEFT JOIN, RIGHT JOIN

```
SELECT title, COUNT(groups.id)
FROM groups
LEFT JOIN projects
ON groups.project_id= projects.id
GROUP BY title ORDER BY
COUNT(groups.id) DESC;
```

```
SELECT title, COUNT(groups.id)
FROM groups
RIGHT JOIN projects
ON groups.project_id= projects.id
GROUP BY title
ORDER BY COUNT(groups.id)
DESC;
```





SELECT

```
mysql> help select
```

```
Name: 'SELECT'
```

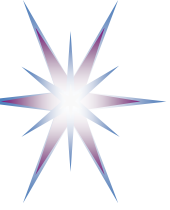
```
Description:
```

```
Syntax:
```

```
SELECT
```

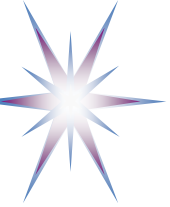
```
  [ALL | DISTINCT | DISTINCTROW ]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
  [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
  [CHARACTER SET charset_name]
  export_options
  | INTO DUMPFILE 'file_name'
  | INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

```
[...]
```



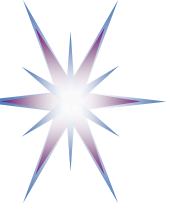
Part 3. SQL getting started practice

- Installing and using SQL database
- Experimenting with simple examples above



Options SQL database installation for exercises

- LAMP machine (Linux – Apache – MySQL – PHP)
 - The most popular platform for web servers and web applications
 - Install Bitnami LAMP server on VirtualBox
- AWS Aurora
 - Cloud based Petabyte scale database compatible with MySQL
- MySQL workbench
 - Convenient client and tool work with SQL databases



Access to MySQL database on LAMP machine

Three methods to access database:

1. With **phpMyAdmin**

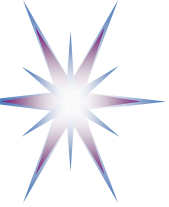
- Easy to use web interface
- Tip: use this to build your database!
- "Designer" tool is useful for defining relations and creating graphical schemes
- Useful for testing and debugging "queries"

2. With a **mysql** command line shell

- Useful to quickly test and debug queries

3. From **PHP** code (optional)

- Access your database from PHP programs



Accessing LAMP Server

phpMyAdmin

- Example: `http://ec2-54-77-161-187.eu-west-1.compute.amazonaws.com/phpmyadmin/`
- Useful tool to manage and build your database
- Login with your username and password

MySQL CLI

- Simple and intuitive
- Try SQL examples above

Login to the server with SSH:

```
$ mysql -u <username> -p
```

Enter password: `<password>`

MySQL can connect to remote server

```
$ mysql -h hostname -u username -p
```

Exit – quit or exit

```
$ mysql exit Bye
```

List databases:

```
mysql > show databases;
```

Select database:

```
mysql > use <databasename>;
```

List tables in a selected database:

```
mysql > show tables;
```

phpMyAdmin

Welcome to phpMyAdmin

Language

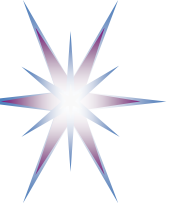
English

Log in

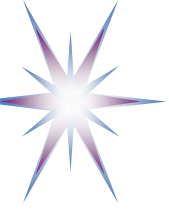
Username:

Password:

Go



AWS Aurora MySQL/PostgreSQL



Accessing MySQL/Aurora

The screenshot shows the MySQL Workbench interface with the following components:

- Navigator:** Includes sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup), and SCHEMAS (Filter objects).
- Query Editor:** Contains a query with four lines:

```
1 show databases;
2 use testdb01;
3 INSERT INTO product_type SET product_type_cd="RISK", NAME="Ris assessment";
4 select * from product_type;
```
- Result Grid:** Displays the results of the query execution. The first two queries returned 5 rows and 0 rows respectively. The third query failed with Error Code: 1146. The fourth query returned 4 rows.
- Action Output:** A table showing the execution details of each query step.

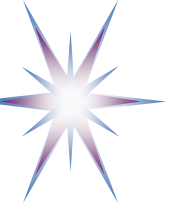
#	Time	Action	Message	Duration / Fetch
2	00:52:33	show databases	5 row(s) returned	0.047 sec / 0.000 sec
3	00:52:56	use testdb01	0 row(s) affected	0.047 sec
4	00:53:16	INSERT INTO Product_type SET Product_type_cd="RISK", NAM...	Error Code: 1146. Table 'testdb01.Product_type' doesn't exist	0.031 sec
5	00:53:56	show tables	11 row(s) returned	0.032 sec / 0.000 sec
6	00:54:57	INSERT INTO product_type SET product_type_cd="RISK", NAM...	1 row(s) affected	0.032 sec
7	00:56:28	select * from product_type LIMIT 0, 1000	4 row(s) returned	0.031 sec / 0.000 sec

The screenshot shows the MySQL Workbench Welcome screen with the following content:

- Welcome to MySQL Workbench**
- MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.**
- MySQL Connections:** A list of connections including 'Local instance wampmysqld64' and 'AWS Aurora'.

MySQL workbench <https://dev.mysql.com/downloads/workbench/>

- MySQL workbench connected to Aurora DB
- Use DBMS connector URL
bdittut01-cluster-1.cluster-c8pzxehfuebq.eu-west-1.rds.amazonaws.com
- Port 3306



Deploying Aurora cluster

The screenshot displays the MySQL Workbench interface with the 'Setup New Connection' dialog box open. The connection is named 'Aurora Tutor02' and uses the 'Standard (TCP/IP)' method. The parameters are as follows:

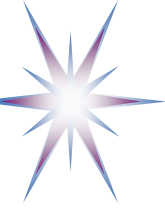
- Hostname: rehfebq.eu-
- Username: admin
- Password: (Stored in Vault)
- Default Schema: (Empty)

A success message dialog box is overlaid on top of the setup dialog, stating: 'Successfully made the MySQL connection'. It provides the following information:

- Host: bdittut02.c8pzhfebq.eu-west-1.rds.amazonaws.com
- Port: 3306
- User: admin
- SSL: enabled with DHE-RSA-AES256-SHA

The message concludes with: 'A successful MySQL connection was made with the parameters defined for this connection.' The 'Test Connection' button is highlighted in the setup dialog.

- Cluster name bdittut02
- admin :: bdit2018aurora



Deploying Aurora cluster

The screenshot displays the AWS Management Console for the Amazon RDS service, specifically the page for an Aurora MySQL instance named 'bdittut02' in the eu-west-1 region. The console shows the instance's summary, CloudWatch metrics, and various configuration options.

Amazon RDS

- Dashboard
- Instances**
- Clusters
- Performance Insights PREVIEW
- Snapshots
- Reserved instances
- Subnet groups
- Parameter groups
- Option groups
- Events
- Event subscriptions
- Notifications

bdittut02 Instance actions

Summary

Engine Aurora MySQL 5.6.10a	DB instance class info db.t2.small	DB instance status available	Pending maintenance none
--------------------------------	---	---------------------------------	-----------------------------

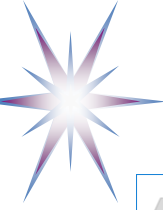
CloudWatch (38) Refresh Add instance to compare Monitoring Last Hour

Legend: bdittut02

CPU Utilization (Percent)

DB Connections (Count)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)



Deploying Aurora cluster

The screenshot displays the AWS Management Console for Amazon RDS. The left sidebar contains navigation links: Dashboard, Instances (highlighted), Clusters, Performance Insights, Snapshots, Reserved instances, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, and Notifications. The main content area is titled 'Connect' and shows the endpoint 'bdittut02.c8pzhfuebq.eu-west-1.rds.amazonaws.com', port '3306', and 'Publicly accessible: Yes'. Below this, 'Security group rules (2)' are listed in a table:

Security group	Type	Rule
rds-launch-wizard-3 (sg-72a33808)	CIDR/IP - Inbound	146.50.146.132/32
rds-launch-wizard-3 (sg-72a33808)	CIDR/IP - Outbound	0.0.0.0/0

Below the table, the 'Details' section is visible with tabs for Configurations, Security and network, Instance and IOPS, and Maintenance details. The bottom of the console shows a footer with the URL 'https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#SecurityGroups:search=rds-launch-wizard-3' and a taskbar with several document icons.

Example running SQL query in Aurora remotely

The screenshot shows the MySQL Workbench interface with the following components:

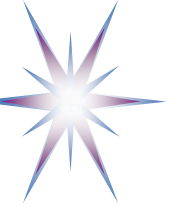
- Navigator:** Shows the database structure including Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, and Data Import/Restore.
- Query Editor:** Contains the following SQL script:

```
5 exit;
6 show databases;
7 use mysql;
8 show tables;
9 exit;
10
11 select * from employee;
12 select * from employee limit 5;
13
14 Select product_cd, date_offered, name, product_type_cd From product Pro Where Pro.product_type_cd = 'LOAN';
```
- Result Grid:** Displays the results of the last query (Query 14) in a table with 4 columns: product_cd, date_offered, name, and product_type_cd. The results are:

product_cd	date_offered	name	product_type_cd
AUT	2000-01-01	auto loan	LOAN
BUS	2000-01-01	business line of credit	LOAN
MRT	2000-01-01	home mortgaoe	LOAN
SBL	2000-01-01	small business loan	LOAN
- SQLAdditions:** Provides a syntax reference for the SELECT statement.
- Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
19	22:27:08	select * from employee limit 5	5 row(s) returned	0.078 sec / 0.000 sec
20	22:29:08	Select product_cd, date_offered, name, product_type_cd From pro...	4 row(s) returned	0.047 sec / 0.000 sec

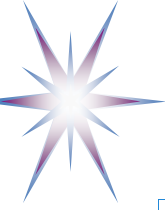
Select product_cd, date_offered, name, product_type_cd From product Pro
Where Pro.product_type_cd = 'LOAN';



MySQL CLI to Aurora

MySQL client installed

- `mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert`

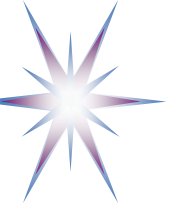


AWS DynamoDB

The screenshot shows the AWS DynamoDB console interface. The left sidebar contains navigation links for DynamoDB, Dashboard, Tables, Backups, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main content area displays the 'Music' table, which is active and has a partition key of 'Artist (String)' and a sort key of 'Song (String)'. The 'Items' tab is selected, showing a list of items. The table has columns for Artist, Song, Album, Year, and Genre. Two items are visible: John Lennon (Imagine, 1971, Soft rock) and Pink Floyd (Money, 1973, The Drk Side ...).

Artist	Song	Album	Year	Genre
John Lennon	Imagine	Imagine	1971	Soft rock
Pink Floyd	Money	The Drk Side ...	1973	

- Document based and key-value database
- <https://qwiklabs.com/focuses/2376?locale=en>



Additional materials
