EPITA
Advanced WEB Development
and
Advanced Database
2015 MI Fall Session

Teacher : Alexandra Champavert

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Session plan

- Introduction to databases
  - **Databases and management system**
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Before the Database management systems

- The first databases-like appeared in the 1950's years in order to make easier the bank and insurance process
- During almost 20 years, the data were stored in flat files
- The main langage used to process these flat files was the COBOL

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Birth of the DBMS Concept

- SQL language creation
- Strict splitting between the hardware hosting the data, and the data processing
- Data modeling
- Normalization creation
- Boyce-Codd normal form
- Database Management System Concept

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The database

- This is the place where the data is stored
- The data storage implies the information persistence
- The database isn't alive
- To be alive it needs a tool

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Database management

- A database is managed through a system
- This system is named Database Management System, otherwise DBMS
- The management system gives the interface between the data and the user
- The management system is in charge for updating the data

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

ACID Properties

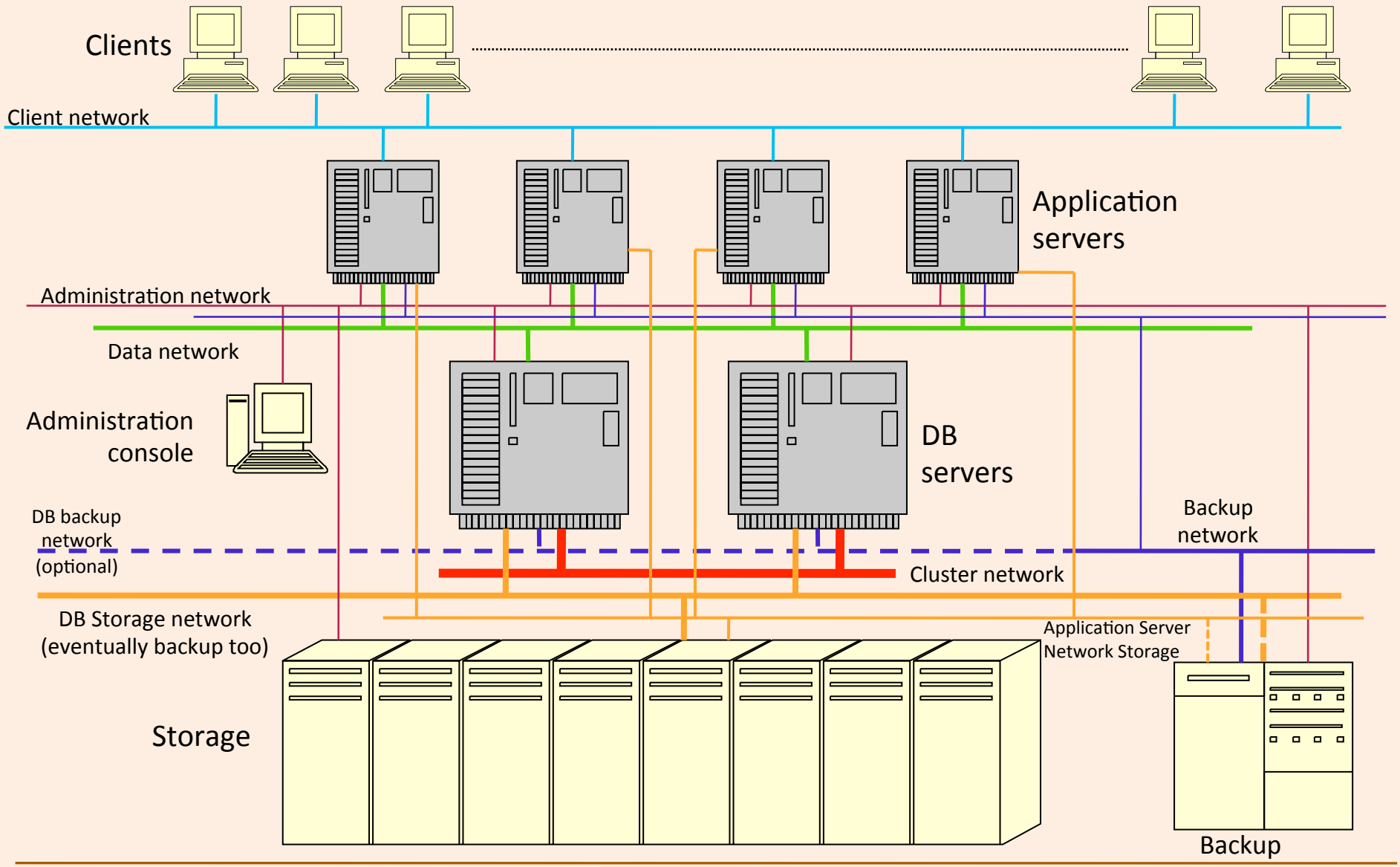
- Atomicity : A transaction is atomic if it can be totally commited or rollbacked
- Coherency : A transaction should not leave a database in an inconsistent state
- Isolation : A transaction can not view another running transaction
- Durability : As the transaction is commited, the modified data will not disapear

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Database Hardware Infrastructure

Infrastructure



Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The storage, central element of the architecture

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The storage

- … is made of :
  - Enclosures,
  - RAID disks (0, 1, 5, 10, 50, security),
  - Copper lines (SCSI),
  - Fibre Channel,
  - Memory cache,
  - Dedicated Operating System,
  - For some of them : backup batteries to prevent power cut

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The storage

- Three main storage types exist :
  - Internal storage (server)
  - NAS : Network Attached Storage
  - SAN : Storage Area Network

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Comparison of the main storage types

Storage types	Key points	Weak points
Internal	- Low range price - Needs no special place to reserve	- Poor number of slots → Poor number of disks - Non-evolving architecture - Storage and server is the same - Impossible to mutualize the storage
NAS	- Mutualized - Theoricaly no disks limitation - Growing capacity - Storage miroring possible - Middle range price	- Classical network technology - Proprietary filesystem making hard the raw devices management - Need of a secured place to store - Heavy electrical power supply
SAN	- Mutualized - Theoricaly no disks limitation - Growing capacity - Storage miroring possible - Dedicated network - Quasi-unlimited throughput - Mutualized backup	- Expensive !!! - Need of a secured place to store - Heavy electrical power supply

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

SAN IBM DS8700

 Reproduction interdite sans accord préalable

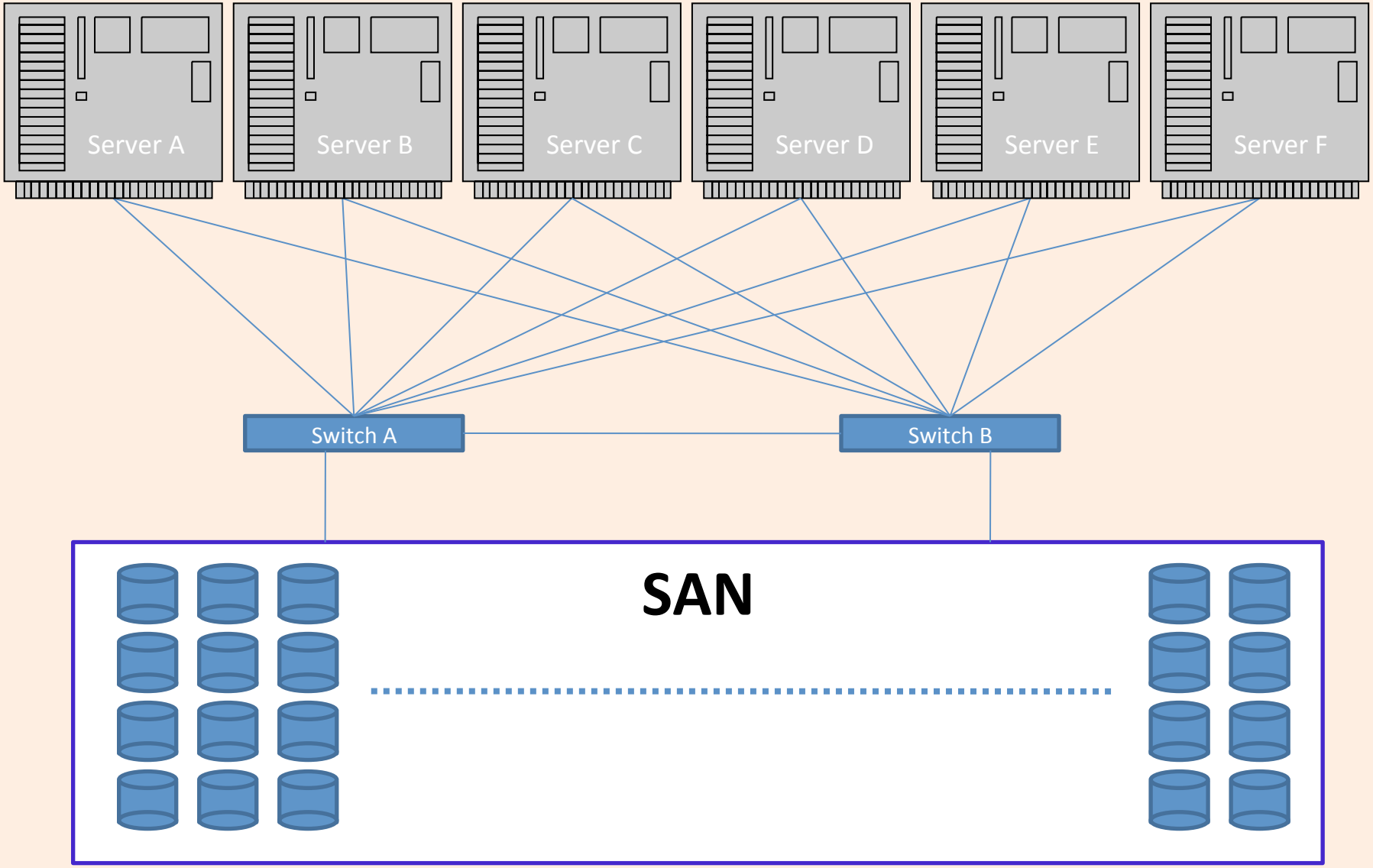
SAN Oracle-Sun Exadata

HP – XP 24000

EMC Symmetrix DMX-4

 Reproduction interdite sans accord préalable

Example of a SAN architecture



Physical considerations linked with the storage

- The number of disks is directly corelated to the throughtput with no-cache
- The cache has to be sized to accelerate the OLTP
- The logical units need to be created taking care of the class storage used
- The number of servers connected has a direct impact when sizing the storage

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Vocabulary (1/3)

- **I/O** : Input/Output on a peripheral
- **I/O/s** : I/O per second
- **I/O length** : Size of an I/O in KB, MB
- **I/O Wait** : Wait on a synchrone I/O having an impact on the process until the I/O is resolved
- **I/O Throughput** : Maximum number of I/Os and maximum throughput in bytes per second, or combination of these two measures
- **SAN switch** : Special equipment linking the SAN and the servers, with another SAN, a backup robot
- **FC or Fibre Channel** : Protocol used to communicate with the storage when fibre is used to connect the storage elements
- **RAID or Redondant Array of Inexpensive Disks** : Hardware or software architecture made up of many disks to ensure data security and I/O throughput
- **SAN Fabric** : SAN + FC + switches

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Vocabulary (2/3)

- **SCSI or Small Computer System Interface** : Communication standard between the computer and its peripherals
- **iSCSI or Internet SCSI** : Protocol used to connect peripherals like SANs with servers. The SCSI layer is encapsulated in the TCP/IP layer and is based on the client (server) / server (SAN) architecture
- **LUN or Logical Unit Number** : Pointer to a storage device. Designate too the storage
- **WWN or World Wide Name** : Unique identifier of a LUN on a SAN network
- **Zoning** : SAN splitting in multiple zones
- **I/O Latency** : Time necessary to acquire the I/O
- **SSD or Solid-State Drive** : Hardware storage made of flash-memory. Very fast. Main problem : Time and intensive usage destroy cells making holes in the matrix. Expensive.
- **HDD or Hard-Disk Drive** : Mecanical device with disks and read/write heads. Less fat than the SSD, but reliable. Cheap. Main problem : Mecanical parts = Fragility

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Vocabulary (3/3) – The most common RAID

- **RAID 0** : Disk striping → Disks agregate in the purpose to make a unique storage space, parallelizing reads and writes. No safety in case of a disk loss
- **RAID 1** : Disk mirroring → 2 or more disks are miror of each other. Writes are made simultaneously on the 2 disks, reads are made by the faster of the miror
- **RAID 5** : Disks agregate with parity between each disk. The loss of one disk doesn't imply the loss of the entire agregate. The lost disk can be hot unplugged and replaced immediatly by a new one who will be synchronized by the others
- **RAID 6** : Multiple parity. It can be possible to loose more than one disk depending of the security level choosen. Parity *n* equals possible loss of *n* disks without problem
- **RAID 0 + 1** : Priority for the RAID 0. One disk lost implies the loss of the entire miror
- **RAID 1 + 0** : Priority for the RAID 1. Much more robust. Only the loss of all disks of one side of the miror makes the loss of the miror

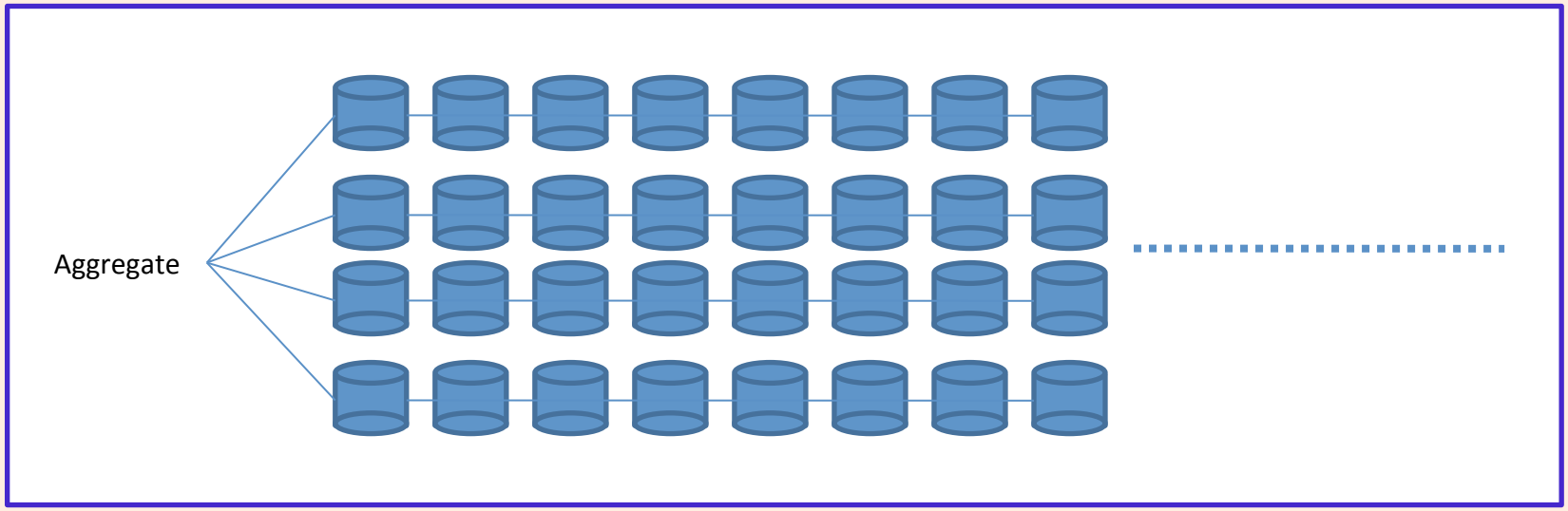
Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

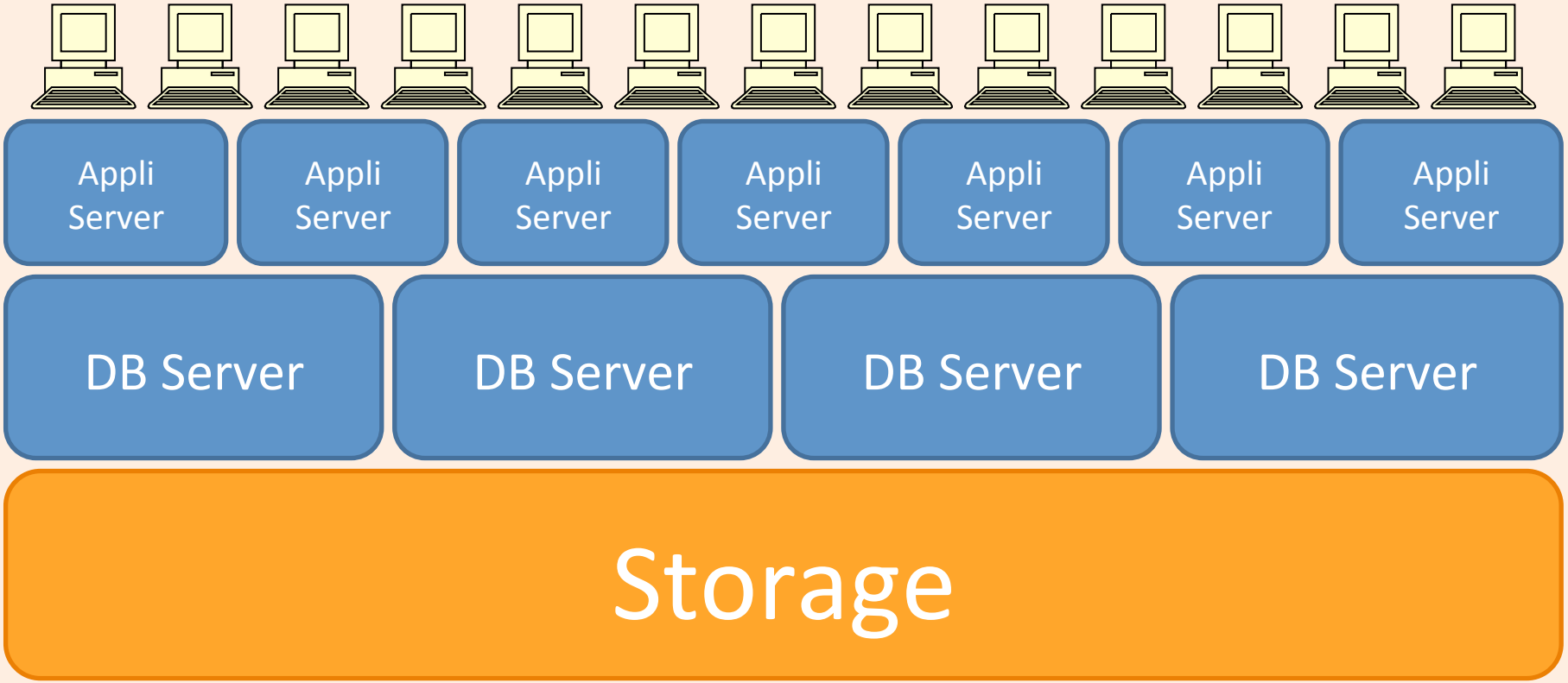
Some SAN disks architectures

- To major architecture exist :
  - vRAID 6

  - Clustered vRAID 5



From the storage performance to the client queries performance



Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Distribution rules of the data on the storage

- The better is to have aggregates with a wide number of disks to maximize I/O throughput
- The transaction logs have to be on the faster writing aggregates
- The data files need to be on the much more secure aggregates
- The distribution of the database files must follow the usage frequency of the data

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Complex architectures

- Replication of all or a part of a SAN
- Real-time replication between SANs
- SAN SSD (mega-cache) over SAN HDD

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The server: DBMS « engine »

IBM Power Server product line

HP Integrity Servers product line

Oracle-Sun product line

Hardware guiding the server performance

- Memory
- CPU
- Network card
- Storage card
- Low-latency network card (specific to cluster)
- Motherboard or cabinet

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Software guiding the server performance

- Operating system, calibration:
  - Memory management
  - Semaphore management
  - System cache management
  - Different buffers
  - IPC
  - …
- Drivers :
  - Have the good versions!!
  - Prefer the vendor drivers than the generic drivers coming with the operating system

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Server calibration when used for a RDBMS

- Choice of the hardware (PC-like or other)
- Choice of the architecture (64bit generally)
- … twice (Itanium, Opteron/Xeon)
- Choice of the operating system
- Multiple CPUs and multiple cores
- Memory sizing
- No unused daemon started
- Swap sizing
- Good parameters for the operating system for the RDBMS usage

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

From the hardware architecture to the RDBMS

 Reproduction interdite sans accord préalable

Link between the RDBMS and the hardware

- The hardware provides:
  - The storage
  - The computing power delivered by the server
  - The physical access through the server
- The RDBMS:
  - Leans on the server to process data
- An optimal configuration of a DBMS with th hardware has to be made depending on the application

What RDBMS for what usage?

MySQL™ (Oracle - Sun)

- Database of small to huge dimension
- « Free » for a personal usage
- Paying distribution when having to deploy on productions and dispose of a support
- Cluster with a master in read/write mode and read mode slaves
- InnoDB necessary to have the ACID properties of the DBMS
- Very fast if using the MyISAM tables (no transactional capability!)
- Mainly used for websites and now coming to critical professional websites and applications with heavy transactional load
- Present on an important choice of operating systems

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

PostgreSQL

- Database from small to very large size
- « Free » for a personal usage
- Support license
- Cluster license
- Fully ACID compliant
- Can easily be compared to the Oracle RDBMS for its performance and agility
- Found on a great number of operating systems
- Expertise in G.I.S.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Sybase™ ASE (Adaptive Server Enterprise)

- Commercial software
- Database small to heavy size
- Internal management of the blocks with sorted lines (Index Organized Tables) natively
- Can be found on multiple systems
- OLTP oriented even if some companies useit for decisional systems
- Clustering capabilities
- Simplified management administration
- Data replication

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Sybase™ IQ

- Commercial software
- Fully dedicated on decisional systems
- The blocks contain columns and these columns are systematically indexed
- To be used with very huge databases
- Fully ACID compliant
- Can be found on multiple operating systems
- Clustering capabilities

Oracle™ RDBMS

- Commercial software
- Polyvalent engine (OLTP, Decisional, Hybrid)
- From small to very huge databases
- Extremely adaptive making it very complex to parameter
- Fully ACID compliant
- Can be found on most of the operating systems
- Advanced cluster capabilities (leader)
- Advanced disaster recovery management
- Advanced replication functionnalities

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

IBM™ DB2™ UDB™

- Commercial software
- Polyvalent engine (OLTP, Decisional, Hybride)
- From small to very huge size
- Extremely adaptive making it complex to manage
- Fully ACID compliant
- Can be found on most of the operating systems
- Clustering capabilities (splitted databases)
- Disaster recovery capabilities
- Replication

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Microsoft™ SQL Server™

- Commercial software
- Born with the end of the collaboration with Sybase in version 5. Microsoft decided to make its own evolution of the engine
- From small to heavy databases
- Same block management as Sybase ASE (sorted lines inside the blocks)
- ONLY on Windows
- Essentially dedicated to OLTP
- Fully ACID compliant
- Clustering capabilities
- Easy management
- Replication

Conclusion

- The DBMS are extremely dependant of the hardware architecture
- During to conception phase of a new application on a new architecture, better is the knowledge of the architecture, better is the sizing of the architecture to comply with the needs of the application
- If it concerns an applicative evolution, the performance data from the actual production can permit to anticipate on the sizing of the new architecture
- The physical architecture have to be taken in consideration during the application creation, to be sure that the application will be efficient

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Session plan

- Introduction to databases
  - Databases and management system
  - **Normal forms**
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The conceptual model

- The nearest of the human
- It's not directly linked to a particular database kernel
- It's based on the entity-relationship concept

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Entity-Relationship

- An entity is a data collection
- A relationship is a link between two data collections
- It's particularly based on the Boyce-Codd normal form, and especially on the normal form key domain

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Normal Form

- It's a data structuration
- This form is normalized : It follows rules
- The goal is to normalize the data
- The normalization is physically limited to a unique database
- The normal forms as defined in the relational model are 6

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Role of the normal forms

- Avoid the transactional anomalies:
  - Data redundancy
  - Read anomaly
  - Write anomaly
  - Counter-performance

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

First Normal Form – 1NF (source Wikipedia)

- First normal form (1NF) is a property of a relation in a relational database. A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.
- Edgar Codd, in a 1971 conference paper, defined a relation in first normal form to be one such that none of the domains of that relation should have elements which are themselves sets.
- First normal form is an essential property of a relation in a relational database. Database normalization is the process of representing a database in terms of relations in standard normal forms, where first normal is a minimal requirement.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Second Normal Form – 2NF (source Wikipedia)

- Second normal form (2NF) is a normal form used in database normalization. 2NF was originally defined by E.F. Codd in 1971.
- A table that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form. Specifically: a table is in 2NF if and only if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the table. A non-prime attribute of a table is an attribute that is not a part of any candidate key of the table.
- Put simply, a table is in 2NF if and only if it is in 1NF and every non-prime attribute of the table is dependent on the whole of a candidate key.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Third Normal Form – 3NF (source Wikipedia)

- The third normal form (3NF) is a normal form used in database normalization. 3NF was originally defined by E.F. Codd in 1971. Codd's definition states that a table is in 3NF if and only if both of the following conditions hold:
- The relation R (table) is in second normal form (2NF)
- Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every superkey of R.
- A non-prime attribute of R is an attribute that does not belong to any candidate key of R. A transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X).
- A 3NF definition that is equivalent to Codd's, but expressed differently, was given by Carlo Zaniolo in 1982. This definition states that a table is in 3NF if and only if, for each of its functional dependencies X → A, at least one of the following conditions holds:
- X contains A (that is, X → A is trivial functional dependency), or
- X is a superkey, or
- Every element of A-X, the set difference between A and X, is a prime attribute (i.e., each attribute in A-X is contained in some candidate key)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Boyce-Codd Normal Form - BCNF (source Wikipedia)

- Boyce–Codd normal form (or BCNF or 3.5NF) is a normal form used in database normalization. It is a slightly stronger version of the third normal form (3NF). BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd to address certain types of anomaly not dealt with by 3NF as originally defined.
- Chris Date has pointed out that a definition of what we now know as BCNF appeared in a paper by Ian Heath in 1971. Date writes:
- "Since that definition predated Boyce and Codd's own definition by some three years, it seems to me that BCNF ought by rights to be called Heath normal form. But it isn't."
- If a relational scheme is in BCNF then all redundancy based on functional dependency has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce–Codd normal form if and only if for every one of its dependencies $X \rightarrow Y$, at least one of the following conditions hold:
- $X \rightarrow Y$ is a trivial functional dependency ($Y \subseteq X$)
- X is a superkey for schema R

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Fourth normal form – 4NF (source Wikipedia)

- Fourth normal form (4NF) is a normal form used in database normalization.
- Introduced by Ronald Fagin in 1977, 4NF is the next level of normalization after Boyce–Codd normal form (BCNF).
- Whereas the second, third, and Boyce–Codd normal forms are concerned with functional dependencies, 4NF is concerned with a more general type of dependency known as a multivalued dependency.
- A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies X \twoheadrightarrow Y, X is a superkey—that is, X is either a candidate key or a superset thereof.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Fifth Normal Form – 5NF (source Wikipedia)

- Fifth normal form (5NF), also known as project-join normal form (PJ/NF) is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships. A table is said to be in the 5NF if and only if every join dependency in it is implied by the candidate keys.
- A join dependency *{A, B, … Z} on R is implied by the candidate key(s) of R if and only if each of A, B, …, Z is a superkey for R.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Sixth Normal Form – 6NF (source Wikipedia)

- Sixth normal form (6NF) is a term in relational database theory, used in two different ways.
- A book by Christopher J. Date and others on temporal databases, defined sixth normal form as a normal form for databases based on an extension of the relational algebra.
- In this work, the relational operators, such as join, are generalized to support a natural treatment of interval data, such as sequences of dates or moments in time.
- Sixth normal form is then based on this generalized join, as follows:
  - A relvar R [table] is in sixth normal form (abbreviated 6NF) if and only if it satisfies no nontrivial join dependencies at all — where, as before, a join dependency is trivial if and only if at least one of the projections (possibly U_projections) involved is taken over the set of all attributes of the relvar [table] concerned.
- Any relation in 6NF is also in 5NF.
- Sixth normal form is intended to decompose relation variables to irreducible components. Though this may be relatively unimportant for non-temporal relation variables, it can be important when dealing with temporal variables or other interval data. For instance, if a relation comprises a supplier's name, status, and city, we may also want to add temporal data, such as the time during which these values are, or were, valid (e.g., for historical data) but the three values may vary independently of each other and at different rates. We may, for instance, wish to trace the history of changes to Status.
- For further discussion on Temporal Aggregation in SQL, see also Zimanyi. For a different approach, see TSQL2.

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Domain-Key Normal Form - FNDC (source Wikipedia)

- Domain/key normal form (DKNF) is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.
- A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

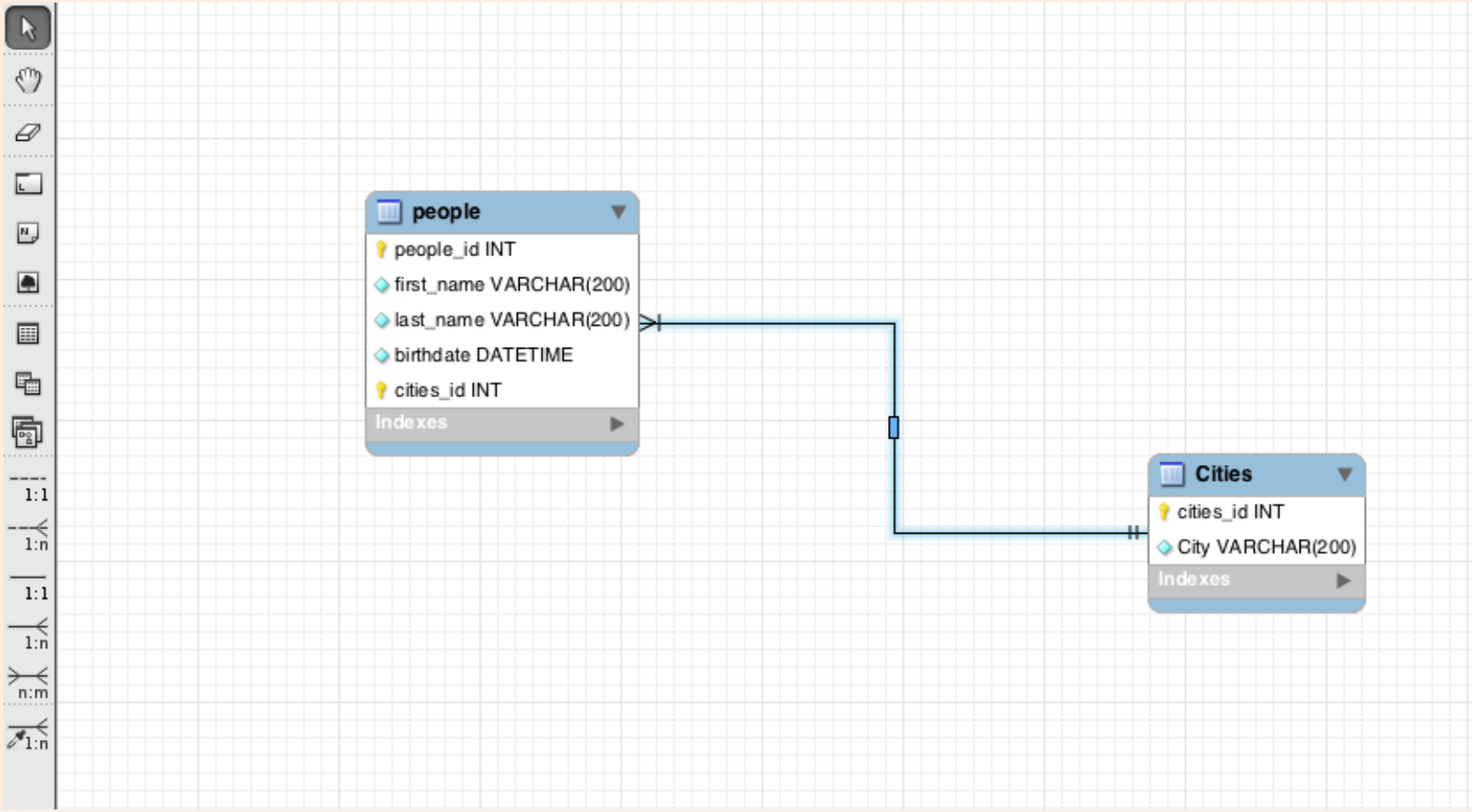
Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

MySQLWorkbench

- The tool
- Exercice

Data model example



Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - **HTML & CSS**
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

The HTML

What's the HTML

- HTML is derived from SGML
- HTML is the acronym for HyperText Markup Language
- HTML permits to define the content of a WEB page with links to other pages
- The main program to display HTML content is a WEB browser
- That WEB browser can be a CLI like Lynx

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Markups

- Like in XML, the markups have a particular syntax with a begin markup and (not all time) an end markup:

<keyword></keyword>

<keyword />

<keyword> (Correct… but… Bad!!!)

- If a keyword has no end markup, prefer to use the syntax “<keyword />” with the “/” indicating the end of the markup

Generic structure of an HTML page

```
<!doctype html>
<html>
<head>
</head>
<body>
</body>
</html>
```

What is HTML?

- HTML is made to display information on a terminal
- HTML doesn't need, normally, to format the presentation of the text
- HTML places objects but doesn't have to format them
- CSS is made for that

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Head of an HTML page

<head>
</head>

- This is the part where we define:
  - Character set
  - Metadata
  - Javascript programs
  - CSS
  - Title of the page
- No information except the title of the page will be displayed in the WEB browser

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Character set

- Most important part of a document
- Defines the coding of the page
- Needs to be chosen with care
- One's it's chosen, it has to be propagated to all the coding levels and data sources

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Metadata

- Used to pass variables to the browser, like character set…
- Used to indicate some informations for the search engines like Google, Bing, Yahoo!
- To link to another page with a timer (Refresh)
- … and many other things

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Body of an HTML Page

<body>
</body>

- This part contains all the page content:
  - Text
  - Images
  - Multimedia
- The content is ordered with:
  - Divs
  - Spans
  - Tables (try to forget it. Prefer the use of divs)
  - Headers
  - Paragraphs

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Comments

<!-- [...] -->

- Use this markup to introduce comments inside a page
- Can be used to deactivate temporarily a piece of HTML code

Divs

- This is the main container for all the objects
- It’s the better way to order them

Spans

- This object is used to place particular formats for the objects
- It can be used to place a special class for displaying information in a particular way

Tables

```
<table>
<thead>
</thead>
<tbody>
<tr>
<td>
</td>
</tr>
</tbody>
<tfooter>
</tfooter>
</table>
```

- Tables are less used with the succession of the divs
- Sometimes the tables are more adapted than the divs (accountancy table…)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Headers

<h1></h1>

<h2></h2>

[…]

<h6></h6>

- Headers are used for the chapters of the content, like in the books

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Paragraphs

- A paragraph is an object to format text like in the books

Address or Hyperlink and Names

[text]

- The address is an hyperlink to another page or to an element in the page

[text]

- Examples:

W3C

Introduction

1. Introduction

Lists

: ordered list (with numbers)

: unordered list (with bullets)

- Create lists of values
- Used to create menus in combination with CSS

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Images

- Insert an image in a page

Forms (1)

```
<form action="[...]" method="[get|post]">
</form>
```

- Forms are used to pass data to the server
- They contain fields and validation/cancellation buttons

Forms input + label (2)

<input type="[type]" name="[name]" id="[id]" value="[value]">

- … where type is:
  - text
  - date, datetime
  - password
  - radio
  - checkbox
  - submit
  - color
  - email
  - file
  - hidden
  - image
  - month
  - number
  - range
  - search
  - tel
  - url
  - week

<label for="[id]">Text of the label</label>

- Defines a label for an input field

Forms textarea (3)

<textarea name="[name]" id="[id]" rows="[number of rows]" cols="[number of columns]">[your text]</textarea>

- Defines a field to enter a long text

 Reproduction interdite sans accord préalable

Input select (4)

<select name="[name]" id="[id]" [multiple="multiple"] [size="number of rows displayed"]>

<option [default="default"] value="[value]">text option</option>

</select>

- This field is made to choose one or more (multiple) values in a list

Form fieldset (5)

<fieldset>[list of fields]</fieldset>

- Group fields in a box for styling purpose

Form button (6)

```
<button type="button" name="[name]" id="[id]"
value="[value]">Your text</button>
```

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - **PHP**
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

What is PHP?

- PHP is the acronym for "PHP: Hypertext processor"
- This is a scripting language made for WEB development
- It can be embedded inside HTML
- Easy to learn
- Taking part of C, Java and Perl

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Making a PHP program

```
<?php
echo "blablabla";
?>
```

This can be

```
<?php echo "a program"; ?>
```

written

```
<?php
echo "inside HTML";
?>
```

- These two tags can be used to do a php program file
- …or embedded inside HTML

Semi-column instructions separator

```
echo "blabla";
```

- As in C or Java, the semi-column is the instructions separator

Variables

```
$var_c = "blabla";
$var_i = 1;
```

- PHP is a light-typed language

 Reproduction interdite sans accord préalable

Variables

```
$a = "1";
$b = $a + 1;
$c = 2;
$d = $c + "15 pieces";
```

Particular variables

- $_POST: all the post values of a form with method post
- $_GET: all the get values of a form with method get
- $_SERVER: can return any server variable to use in the program such as "DOCUMENT_ROOT" who is the root of all the PHP code you use
- $_COOKIE: to use the cookies of the browser

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Operators

Operator precedence

- See: http://www.php.net/manual/en/language.operators.precedence.php

Arithmetic operators

- See: http://www.php.net/manual/en/language.operators.precedence.php

Assignment operators

- Assignment
  - Symbol "=":

```
$a = 3;
$b = $a * (3 + $a);
$c = "abcd";
```

  - Symbol "<operator>=":

```
$a += 1;
$b *= $a;
$d .= "efgh";
```

- Assignment by reference

```
$a = 3;
$b = &$a; // $a and $b are pointing on the same data
```

Modifying $a or $b modifies either of two

Bitwise operators

- They are used only on integers

$a & $b; // And

$a | $b; // Or

$a ^ $b; // Xor

~ $a ; // Not

$a << $b; // Shift left. $a is shifted left by $b bits

$a >> $b; // Shift right. $a is shifted right by $b bits

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Control operators

```
$a == $b; // Equality
$a === $b; // Identity
$a != $b; // Non-equality
$a <> $b; // Non-equality
$a !== $b; // Non-identity
$a < $b; // Less than
$a > $b; // Greater than
$a <= $b; // Less or equal
$a >= $b; // Greater or equal
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Incrementing/Decrementing operators

```
++$a; // Pre-increment
$a++; // Post-increment
--$a; // Pre-decrement
$a--; // Post-decrement
```

 Reproduction interdite sans accord préalable

Logical operators

$a and $b; // TRUE if $a and $b are TRUE

$a && $b; // same as preceding

$a or $b; // TRUE if either $a or $b are TRUE

$a || $b; // same as preceding

$a xor $b; // TRUE if one or the other is TRUE, but not both

! $a; // TRUE if $a is not TRUE. FALSE if $a is not FALSE

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

String operators

```
$a = "This is a ";
$b = $a . "concatenated string."; // $b is "This is a concatenated string."
$a = "This is a ";
$b .= "concatenated string."; // $b is "This is a concatenated string."
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Array operators

$a + $b; // Union of $a and $b

$a == $b; // TRUE if $a and $b have the same key/value pairs

$a === $b; // TRUE if $a and $b have the same key/value pairs in the same order and the same types

$a != $b; // TRUE if $a is not equal to $b

$a <> $b; // TRUE if $a is not equal to $b

$a !== $b; // TRUE if $a is not identical to $b

 Reproduction interdite sans accord préalable

Control structures

if

```
if(<expr>)
        <statement>
where <statement> is:
        <instruction>;
or
{
        <statements>
}
```

else

```
if(<expr>){
        <statements>
} else {
        <statements>
}
```

else if/elseif

```
if(<expr>){
        <statements>
} elseif(<expr>) {
        <statements>
} else {
        <statements>
}
```

while

```
while(<expr>)
        <statements>
```

or

```
while(<expr>):
        <statements>
endwhile;
```

do-while

```
do
        <statements>
while(<expr>);
```

for

```
for(<define var>;<expr>;<modify var>)
        <statements>
```

foreach

```
foreach(<array_expression> as $value)
        <statements>
```

or

```
foreach(<array_expression as $key => $value)
        <statements>
```

break

- This instruction is used to exit a control structure
- Used with a parameter exit the number of control structures corresponding to the value

```
break;
break 2; // Exit two levels
```

continue

- Used to continue to the next loop of a control structure
- The instructions following "continue" are skipped

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

switch

```
switch(<variable>){
        case <value1>:
                <statements>
                [break;]
        case <value2>:
                <statements>
                [break;]
        [default:
                <statements>
        ]
}
```

return

- Inside a php file is used to continue with the code who contains an include of the file

a.php

```
<?php
include("b.php");
echo "a";
?>
```

b.php

```
<?php
echo "b";
return;
?>
```

… executing a.php will display "ba"

include/require

- "include" in case of failure will produce a E_WARNING
- "require" in case of failure will produce a E_COMPILE_ERROR

```
include("a.php");
require("b.php");
```

 Reproduction interdite sans accord préalable

include_once/require_once

- Same as include/require but never include/require another time a file when it was already included/required

Functions

User-defined functions

```
function <name>(<arg1>, … , <argn>){
        <statements>
}
```

Function arguments

- That can be an array
- By default is passed by value
- With "&" the argument is passed by reference
- To have a default value just do "$var = <value>"

Returning value

return <value>;

- The <value> is returned to the variable associated with the function

```
function a(){
        return 10;
}

$var = a(); // $var will be 10
```

Variable functions

- A variable function is a function who is called with the evaluation of a variable

```
$func = "test";
$func(10);

function test($a){
        $b = $a + 10;
        return($b);
}
```

Classes and Objects

 Reproduction interdite sans accord préalable

Defining a basic class

```
class BasicClass{
        public $var1 = "My var one ";
        public $var2 = 2;

        public function myFunction(){
                return($this->$var1.$this->$var2);
        }
}
```

Properties of a class

- Properties are called "attributes" in other OOP languages

```
class BasicClass{
        public $var1 = "My var one ";
        public $var2 = 2;

        public function myFunction(){
                return($this->$var1.$this->$var2);
        }
}
```

Class constants

```
class BasicClass{
        const PI = 3.1415926535;
        public $var1 = " is Pi ";

        public function myFunction(){
                return(self::PI.$this->$var1);
        }
}

echo BasicClass::PI;
```

Autoloading classes

- The goal of this feature is to load automatically the classes defined in a php file

```
function __autoload($class_name){
        include $class_name.".php";
}

$a = new FirstClass();
$b = new SecondClass();
```

Constructor and destructor

```
void __construct ([ mixed $args [, $... ]] )

class BaseClass {
  function __construct() {
     print "In BaseClass constructor\n";
  }
}

void __destruct ( void )

class MyDestructableClass {
  function __construct() {
     print "In constructor\n";
     $this->name = "MyDestructableClass";
  }

  function __destruct() {
     print "Destroying " . $this->name . "\n";
  }
}
```

Visibility

- public: any method/property defined as public is always visible from anywhere
- protected: any method/property defined as protected can be accessed only within the class itself and by inherited and parent classes
- private: any method/property defined as private may only be accessed by the class

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Scope resolution operator

… So called "double colon"

```
class BasicClass{
        const PI = 3.1415926535;
        public static $my_stat = 2;
        public $var1 = " is Pi ";

        public function myFunction(){
                return(self::PI.$this->$var1);
        }
}

echo BasicClass::PI;
echo BasicClass::$my_stat;
```

Static

```
class BasicClass{
        const PI = 3.1415926535;
        public static $my_stat = 2;
        public $var1 = " is Pi ";

        public static function myFunction(){
                return(self::PI.$this->$var1);
        }
}

echo BasicClass::$my_stat;
BasicClass::myFunction();
$classname = "BasicClass";
$classname::myFunction();
```

Class abstraction (1)

- An abstract class may not be instantiated
- It can be only extended
- Can't contain private method/property
- All abstract method/property as to be defined in the child class
- If a method/property is protected it can be defined protected or public
- The signatures of the methods must match
- Optional arguments can be defined in the child class

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Class abstraction (2)

```
abstract class AbstractBasicClass{
        abstract protected static $my_stat;
        abstract public $var1;

        abstract public static function myFunction($a);
}

class BasicClass extends AbstractBasicClass{
        public static $my_stat = 2;
        public $var1 = " as a result.";

        public static function myFunction($a,$b = 10){
                return(($a * $b.$this->$var1);
        }
}
```

Object interface

- All methods declared in an interface must be public

```
interface iBasicTemplate{

        public static function myFunction();
}

class BasicClass implements iBasicTemplate{
        const PI = 3.1415926535;
        public static $my_stat = 2;
        public $var1 = " is Pi ";

        public static function myFunction(){
                return(self::PI.$this->$var1);
        }
}
```

Traits

- Traits is as a class but not a class

trait myFunctions{

Overloading

- Overloading is used to dynamically manage methods through magic methods

```
public void __set ( string $name , mixed $value )
public mixed __get ( string $name )
public bool __isset ( string $name )
public void __unset ( string $name )
```

Object iteration

- Browse the properties of a class

```
class BasicClass{
        public $var1 = "My var one ";
        public $var2 = 2;
        private $var3 = "10 items";
        protected $var4 = "protected property";

function iterateVisible() {
    echo "MyClass::iterateVisible:\n";
    foreach($this as $key => $value) {
        print "$key => $value\n";
    }

}
$class = new BasicClass();

foreach($class as $key => $value) {
   print "$key => $value\n";
}

$class->iterateVisible();
```

Final keyword

- The keyword "final" is used to prevent a child class to override the definition of a method

```
class BasicClass{
        public $var1 = "My var one ";
        public $var2 = 2;

        final public function myFunction(){
                return($this->$var1.$this->$var2);
        }
}
```

Object cloning

void __clone(void)

- This method is used to clone all or part of an object
- This method can't be called directly

```
class SubClass{

        static $instances = 0;
        public $instance;
        public function __construct(){
                $this->instance = ++self::$instances;
        }
        public function __clone(){
                $this->instance = ++self::$instances;
        }
}

class BasicClass{
        public $obj1;
        public $obj2;
        function clone(){
                $this->obj1 = clone $this->obj1;
        }
}
```

```
$obj = new BasicClass();
$obj->obj1 = new SubClass();
$obj->obj2 = new SubClass();
$obj2 = clone $obj;
```

- $obj will contain:
  - $obj2->obj1->instance: 1
  - $obj2->obj2->instance: 2
- $obj2 will contain:
  - $obj2->obj1->instance: 3
  - $obj2->obj2->instance: 2

Comparing objects

- The comparison operators to use are:
  - ===
  - !==

Static keyword

```
class A{
        public static function p(){
                echo __CLASS__;
        }
        public static function test(){
                self::p();
        }
}
class B extends A{
        public static function p(){
                echo __CLASS__;
        }
}

B::test(); // will produce "A"
```

```
class A{
        public static function p(){
                echo __CLASS__;
        }
        public static function test(){
                static::p();
        }
}
class B extends A{
        public static function p(){
                echo __CLASS__;
        }
}

B::test(); // will produce "B"
```

Namespaces

What are namespaces?

- That's almost the same as a filesystem representation
- It permits to have the same name for objects with different files using a notation indicating an absolute or relative reference to the object

namespace N1

N1\N2\$obj; //$obj in N1\N2

$obj; // $obj in N1 (not the same)

N2\$obj; // $obj in N1\N2 (same as first)

Declaring namespaces and sub-namespaces

- Declaring a namespace:

namespace N1;

- Declaring sub-namespace:

namespace N1\N2;

__NAMESPACE__ keyword

- __NAMESPACE__ keyword returns the current namespace
- It can be used to dynamically address objects with

```
namespace N1;
function get($classname){
        $c = __NAMESPACE__."\\".$classname;
        return new $c;
}
```

Using global space

```
namespace N1;
function fopen(){
        $f = \fopen(…);
}
```

Name resolution rules

- See documentation:

http://www.php.net/manual/en/language.namespaces.rules.php

Exceptions handling

How to handle exceptions

```
try{
        // code
} catch(e) {
} finally {
}
```

Generator

Generator

- Simplifies the iterators that are normally written inside a class
- One keyword does the whole work: yield

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Yield

- "yield" acts as "return" but just returns the current value and continue the loop where it is

Sample: http://www.php.net/manual/en/language.generators.overview.php

Predefined variables

$GLOBALS

- That variable is an array containing all the variables defined in the global scope of the current script

```
$a = 10;
function test(){
        $a = "abcd";
        echo $GLOBALS["a"]; // 10
        echo $a; // "abcd"
}
```

$_SERVER

- This array contains all the environment variables defined at server level
- See: http://www.php.net/manual/en/reserved.variables.server.php

$_GET

- This array contains the values send through the URL by a form or a link

http://myurl/myfile.php**?**a=10**&**b=azoirjaorha

$_GET["a"] → 10

$_GET["b"] → "azoirjaorha"

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

$_POST

- Same as $_GET but parameters can only be sent through a form

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

$_COOKIE

- This array contains all the cookies manipulated by the server

```
$ok_registered = $sql_ac->verifCookies(base64_decode($_COOKIE["gp"]), base64_decode($_COOKIE["tq"]));
```

 Reproduction interdite sans accord préalable

$REQUEST

- A super-global array containing:
  - $_GET
  - $_POST
  - $_COOKIE

$_FILE

- This array contains all the informations about the files to upload on the server, and the file(s) itself(themselves)

```
function telechargerFichier($composant,$taille_max){

  if(!empty($_FILES["fichier_file"]["tmp_name"]) && is_uploaded_file($_FILES["fichier_file"]["tmp_name"])){
    if(filesize($_FILES["fichier_file"]["tmp_name"]) < $taille_max * 1024){
      //$type = exif_imagetype($_FILES["media_file"]["tmp_name"]);
      $fichier_dir = $this->getInternalDirectory($composant);
      if(move_uploaded_file($_FILES["fichier_file"]["tmp_name"],$fichier_dir.$_FILES["fichier_file"]["name"])){
        return(sprintf("Transfer OK : %s",$fichier_dir.$_FILES["fichier_file"]["name"]));
      }else{
        return(sprintf("Transfer problem %s to %s",$_FILES["fichier_file"]["tmp_name"],$fichier_dir."/".
$_FILES["fichier_file"]["name"]));
      }
    }else{
      return(sprintf("File too big: %d Ko. Maximum authorized size: %d Ko.",filesize($_FILES["fichier_file"]["tmp_name"]),
$taille_max));
    }
  }else{
    return("Empty file or not loaded");
  }
}
```

$_SESSION

- Contains all the session variables, including session cookies
- See special functions to manipulate the session variables: http://www.php.net/manual/en/ref.session.php

$_ENV

- Contains all the environment variables of the server
- Depending on the server it's impossible to list these variables. See each server documentation for the accessible variables

$php_errormsg

- This variable contains the last error message generated by PHP

Database security

Design of the database

- Separate clearly the application logic from the data manipulation logic
- Use the good grants for the users such as read-only
- Use views instead of tables for the queries
- Don't allow connection ability from the client. Manage it server-side
- Encrypt critical data with modules such as mcrypt, mhash
- Protect from SQL injection

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Protect from SQL injection

- Don't use directly the informations sent by the server. That's the major security failure!
- Verify the information sent by the client before use it in a statement

Use bind variables!!

And to continue…

- Definitely look the documentation!
- Function reference: http://www.php.net/manual/en/funcref.php
- Security: http://www.php.net/manual/en/security.php

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - **PDO**
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

PDO

 Reproduction interdite sans accord préalable

What is PDO?

- PDO is the acronym for PHP Database Objects
- PDO is made to handle database objects:
  - Tables
  - Views
  - Rows
  - …
- It provides an abstraction layer to access any database kernel and offers specific methods and properties for each particular kernel
- PDO is a PHP class

Open and close a connection

- $db = new PDO(<connection string>,$user, $password);

$db = new PDO('mysql:host=localhost;dbname=groupe10', $user,$password);

[…] // Open the connection

$db = null; // Close the connection

Transactions

- To manage a transaction, three instructions:
  - PDO::beginTransaction()
  - PDO::commit();
  - PDO::rollBack();

Preparing a query

- **Use ONLY prepared statements!!**
- PDO::prepare(<statement>,[<options>])
- PDO::bindParam(<param_name>,<value>)
- PDO::execute([array(<parameters>)])

Preparing a query examples

- Example 1:

```
$stmt = $db->prepare("select col1,col2 from tab1 where col3 = :name");
$param[":name"] = "test";
$stmt->execute($param);
```

- Example 2:

```
$stmt = $db->prepare("select col1,col2 from tab1 where col3 = :name");
$stmt->bindParam(":name","test");
$stmt->execute();
```

Fetching resultset

- PDO::fetch(<fetch_style>[,<cursor_orientation>[,<curs or_offset>]])

```
$stmt = $db->prepare("select col1,col2 from tab1 where col3 = :name");
$param[":name"] = "test";
$stmt->execute($param);
while($row = $stmt->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT){
        printf("%s %s<br />",$row["col1"],$row["col2"];
}
```

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - **SQL language**
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax and jQuery
  - Using the database with jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

MySQL

SQL Language

 Reproduction interdite sans accord préalable

The SQL

- … is the acronym for Structured Query Language
- Uses classical words to express queries against databases
- Can be more or less rich depending on the database kernel
- Is a set of instructions classified in two major groups :
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Data Definition Language

- All the instructions to be able to create:
  - A database
  - A tablespace
  - A table
  - An index…

Data Manipulation Language

- All the instructions to be able to:
  - Insert data into tables
  - Update data into tables
  - Delete data into tables
  - Select data

Data Definition Language (DDL)

Instructions

- CREATE TABLE
- CREATE INDEX
- CREATE VIEW
- CREATE TABLESPACE
- See: http://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-definition.html

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Data Manipulation Language (DML)

The instructions

- … are:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

See: http://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-manipulation.html

SELECT

```
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
     [PARTITION partition_list]
   [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]]
```

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

JOIN

join_table:
  table_reference [INNER | CROSS] JOIN ***table_factor*** [***join_condition***]
 | ***table_reference*** STRAIGHT_JOIN ***table_factor***
 | ***table_reference*** STRAIGHT_JOIN ***table_factor*** ON ***conditional_expr***
 | ***table_reference*** {LEFT|RIGHT} [OUTER] JOIN ***table_reference join_condition***
 | ***table_reference*** NATURAL [{LEFT|RIGHT} [OUTER]] JOIN ***table_factor***

where

table_factor:
  tbl_name [PARTITION (***partition_names***)]
    [[AS] ***alias***] [***index_hint_list***]
 | ***table_subquery*** [AS] ***alias***
 | (***table_references***)

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

UNION

SELECT ...

UNION [ALL | DISTINCT] SELECT ...

[UNION [ALL | DISTINCT] SELECT ...]

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

Subqueries

- A subquery is a query returning a resultset to another query

select * from t1 where t1.a in (select t2.a from t2);

- It can be used in *select, update, insert, delete*
- Almost all subqueries can be rewritten as JOIN clauses

Subquery as scalar operand

select (select a from t1);

select lower(select city_name from cities) from cities_lower;

Subquery as a comparison

```
select * from t1 where a = (select max(a) from
t1);
```

Subquery EXISTS and NOT EXISTS

```
SELECT DISTINCT store_type FROM stores
  WHERE EXISTS (SELECT * FROM cities_stores
                WHERE cities_stores.store_type =
stores.store_type);

SELECT DISTINCT store_type FROM stores
  WHERE NOT EXISTS (SELECT * FROM cities_stores
                    WHERE cities_stores.store_type =
stores.store_type);
```

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

Correlated subquery

- Used to query only the rows of the query who have a result in the subquery

```
SELECT * FROM t1
  WHERE column1 = ANY (SELECT column1
FROM t2
                    WHERE t2.column2 = t1.column2);
```

Subquery as a pseudo-table

select b,sum(a) from (select a,b from t1 where a > 2) as t1a group by b;

HANDLER

- A handler is a particular object made to browse a particular table using its indexes
- It is much faster than a SELECT statement and can be reused
- Prefer handler to select for single tables

HANDLER ***tbl_name*** OPEN [[AS] ***alias***]

HANDLER ***tbl_name*** READ ***index_name*** { = | <= | >= | < | > } (***value1***,***value2***,...)
  [WHERE ***where_condition***] [LIMIT ...]
HANDLER ***tbl_name*** READ ***index_name*** { FIRST | NEXT | PREV | LAST }
  [WHERE ***where_condition***] [LIMIT ...]
HANDLER ***tbl_name*** READ { FIRST | NEXT }
  [WHERE ***where_condition***] [LIMIT ...]

HANDLER ***tbl_name*** CLOSE

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

INSERT

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] ***tbl_name***
[PARTITION (***partition_name***,...)]
[(***col_name***,...)]
{VALUES | VALUE} ({***expr*** | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE
col_name=***expr***
[, ***col_name***=***expr***] ...]

Or:

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] ***tbl_name***
[PARTITION (***partition_name***,...)]
SET ***col_name***={***expr*** | DEFAULT}, ...
[ON DUPLICATE KEY UPDATE
col_name=***expr***
[, ***col_name***=***expr***] ...]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] ***tbl_name***
[PARTITION (***partition_name***,...)]
[(***col_name***,...)]
{VALUES | VALUE} ({***expr*** | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE
col_name=***expr***
[, ***col_name***=***expr***] ...]

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

UPDATE

- Single-table syntax:

UPDATE [LOW_PRIORITY] [IGNORE] ***table_reference***

SET ***col_name1***={***expr1***|DEFAULT} [, ***col_name2***={***expr2***|DEFAULT}] ...

[WHERE ***where_condition***]

[ORDER BY ...]

[LIMIT ***row_count***]

- Multiple-table syntax:

UPDATE [LOW_PRIORITY] [IGNORE] ***table_references***

SET ***col_name1***={***expr1***|DEFAULT} [, ***col_name2***={***expr2***|DEFAULT}] ...

[WHERE ***where_condition***]

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

DELETE

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM ***tbl_name***

[PARTITION (***partition_name***,...)]

[WHERE ***where_condition***]

[ORDER BY ...]

[LIMIT ***row_count***]

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

Language structure

String literals

'this is a string'

"this is another string"

'this ' 'is' ' another string'

SELECT _charset_name'a string' COLLATE other_charset_name

SELECT N'national character set string'

make quotes:

SELECT 'hello','"hello"','hel''lo','\'hello';

SELECT "hello","'hello'","hel""lo","\"hello";

Number literals

1.2

-3

-4.56

+7.89

1.2E3

4.5E-6

-7.8E-9

Date and Time literals

'2014-04-23' (any punctuation delimiter can be used)
'20140423'
20140423

Possible formats are:
'YYYY-MM-DD HH-MM-SS[.nnnnnn]'
'YYYY-MM-DDTHH-MM-SS[.nnnnnn]'
'YYYYMDDHHMMSS[.nnnnnn]'
'YYMMDDHHMMSS[.nnnnnn]'
YYYYMDDHHMMSS[.nnnnnn]
YYMMDDHHMMSS[.nnnnnn]
YYMMDD[.nnnnnn]
YYYYMMDD[.nnnnnn]

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Hexadecimal literals

X'val'
x'val'
0xval
… where *val* is a combination of:
0 1 2 3 4 5 6 7 8 9 A B C D E F
or
0 1 2 3 4 5 6 7 8 9 a b c d e f
Usage:
SELECT 0x2F + 0;
→ 47
SELECT 0x416374;
→ 'Act'
SELECT HEX('Act');
→ '416374'

Boolean literals

SELECT TRUE, true, FALSE, false;

→ 1, 1, 0, 0

Bit-Field literals

```
CREATE TABLE t (b BIT(8));
INSERT INTO t SET b = b'11111111';
INSERT INTO t SET b = b'1010';
INSERT INTO t SET b = b'0101';
SELECT b+0, BIN(b+0), OCT(b+0), HEX(b+0) FROM t;
+------+----------+----------+----------+
| b+0  | BIN(b+0) | OCT(b+0) | HEX(b+0) |
+------+----------+----------+----------+
|  255 | 11111111 | 377      | FF       |
|   10 | 1010     | 12       | A        |
|    5 | 101      | 5        | 5        |
+------+----------+----------+----------+
```

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

NULL values

NULL

\N

Schema object names

Identifier qualifiers

col_name

tbl_name.col_name

db_name.tbl_name.col_name

`my-table` OK

my-table NOK

Identifier case sensitivity

table_name <> TABLE_NAME

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Reserved words

- See: http://dev.mysql.com/doc/refman/5.6/en/reserved-words.html

User-defined variables and dynamic SQL

```
SET @var_name = expr [,@var_name := expr]

SET @pi = 3.1415926535;
SELECT 2 * radius * @pi FROM all_radius;
```

Dynamic SQL:

```
SET @col_name = 'col1';
SET @stmt = CONCAT('SELECT ',@col_name,' FROM tab1');
PREPARE stmt from @stmt;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Expression syntax

- See: http://dev.mysql.com/doc/refman/5.6/en/expressions.html

Comment syntax and hints

```
# Comment 1
-- Comment 2
/* Comment 3 */
/*
This is
comment
4
*/
```

Hints:

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM tab1,table2 WHERE
…
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Globalization

 Reproduction interdite sans accord préalable

Globalization

- See: http://dev.mysql.com/doc/refman/5.6/en/expressions.html

Data types

Numeric types

- INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT
- DECIMAL, NUMERIC
- FLOAT, DOUBLE
- BIT

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Date and time types

- DATE, DATETIME, TIMESTAMP
- TIME
- YEAR

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

String types

- CHAR, VARCHAR
- BINARY, VARBINARY
- BLOB, TEXT
- ENUM
- SET

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

… and more

- See: http://dev.mysql.com/doc/refman/5.6/en/data-types.html

 Reproduction interdite sans accord préalable

Functions and operators

Function and operator reference

- See: http://dev.mysql.com/doc/refman/5.6/en/func-op-summary-ref.html

Control flow functions

- CASE value WHEN compare value THEN result […] ELSE result END
- CASE WHEN condition THEN result […] ELSE result END
- IF(expr1,res1,res2) → if expr1 is true then res1 else res2
- IFNULL(expr1,expr2) → if expr1 is null then expr2 else expr1
- NULLIF(expr1,expr2) → NULL if expr1 == expr2 else expr1

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Full-text search

- See: http://dev.mysql.com/doc/refman/5.6/en/fulltext-search.html

… and more

- http://dev.mysql.com/doc/refman/5.6/en/functions.html

Store InnoDB tables

Storing tables

set global innodb_file_per_table=<0|1>

create table <table>

data directory <absolute path to directory>

index directory <absolute path to directory>

 Reproduction interdite sans accord préalable

Partitioning

- Partitioning is made to split data across multiple pseudo-tables
- The execution plans benefit of that splitting
- Accessing a partition is same as accessing a table but with the data volume of the partition
- Multiple queries on multiple partition are not in concurrency if each query is on each partition

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Range partitioning

```
create table t1 (
id int not null,
data1 varchar(100),
data2 varchar(50)
)
partition by range(id) (
        partition p0 values less than (10),
        partition p1 values less than (20),
        partition p2 values less than (30),
        partition p3 values less than (40),
        partition p4 values less than (50),
        partition p5 values less than (60),
        partition pn values less than maxvalue
);
```

 Reproduction interdite sans accord préalable

List partitioning (integer values)

```
create table t1 (
id int not null,
data1 varchar(100),
data2 varchar(50)
)
partition by list(id) (
        partition pstart values in (1,2,3,4),
        partition pmiddle values in (11,12,13,15),
        partition pend values in (45,55,57,58)
);
```

Range columns partitioning

```
create table t1(
id1 int not null,
id2 int not null,
data1 varchar(10),
data2 varchar(10)
)
partition by range columns(id1,id2,data1)
        partition p0 values less than (2,10,'f'),
        partition p1 values less than (3,10,'g'),
        partition p2 values less than (5,100,'az'),
        partition pmax values less than
                (MAXVALUE,MAXVALUE,MAXVALUE)
);
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

List columns partitioning

```
create table t1(
region varchar(100) not null,
city varchar(100) not null
)
partition by list columns(dept)(
partition pregion01 values in('Paris','Seine-et-Marne','Yvelines','Essonne','Hauts-de-Seine','Val-d'Oise','Val-de-Marne',Seine-et-Oise'),
partition pregion02 values in('Corse-Nord','Corse-Sud'),
...
);
```

 Reproduction interdite sans accord préalable

[Linear] hash partitioning

```
create table products_references(
ref_id int not null,
product_name varchar(200) not null
quantity int not null
)
partition by [linear] hash(ref_id)
partitions 16;
```

Key partitioning

- Partitioning depending on a key in the table (primary, unique)

```
create table key1(
id int not null primary key,
name varchar(100)
)
partition by key()
partitions 10;
```

Subpartitioning

- … is a combination of a partitioning by:
  - range or list
- with a subpartitioning by:
  - hash or key
- named composite partitioning too

```
create table cp(
id int not null,
ref_id int not null
)
partition by range(id)
subpartition by hash(ref_id)
(
  partition p0 values less than (2)(
                    subpartition s0,
                    subpartition s1
  ),
  partition p1 values less than(3)(
                    subpartition s2,
                    subpartition s3
  )
);
```

Distribute data and index of partitions or subpartitions

- It is possible to distribute partitions and subpartitions across different storages

partition p1 values less than(2)

data directory = '/data/p1'

index directory = '/index/p1'

(same syntax for subpartitioning)

Managing partitions

- Managing partitions is essential to:
  - add partitions
  - reorganize partitions
  - drop partitions
  - archive old data
  - exchange partitions

Add partitions

```
alter table <table> add partition(
        partition <partition> values less
        than(<value>));
alter table <table> add partition(
        partition <partition> values in
        (<value list>));
```

Reorganize partitions

```
alter table <table> reorganize partition
        <partition list>
        into (<partition definition>);
alter table t1 reorganize partition p0,p1,p2,p3
        into(
          partition n0 values less than (1000),
          partition n1 values less than (MAXVALUE)
        );
```

Reorganize partitions by key or hash

- To reorganize these kind of partitions, use the keyword *coalesce*
- Coalescing partitions corresponds to change the number of partitions

```
alter table <table> coalesce
        partition <number of partitions>;
```

Exchanging partitions

- Exchanging a partition with a table is used to:
  - populate a new partition with table data
  - archive a partition
  - make a rotation between the partitions
- The table needs to have strictly the same structure as the partitioned table
- The source table may not have foreign keys and may not be referenced by another table
- The source table must not be partitioned

```
alter table <table_p> exchange partition <p>
        with table <table>;
```

Maintaining partitions (not subpartitions)

- alter table <table> rebuild partition
  <partition list>;
- alter table <table> optimize partition
  <partition list>;
- alter table <table> analyze partition
  <partition list>;
- alter table <table> repair partition
  <partition list>;
- alter table <table> check partition
  <partition list>;
- alter table <table> truncate partition
  <partition list>;
- alter table <table> truncate partition all;

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Informations for the partitions

- show create table <table>\G
- explain partitions select * from <table>\G

Partition pruning

- Pruning partition is the action of using only the partitions containing data when selecting them without browse the other partitions
- It can be done by using the good column in the WHERE clause with a list of values or a range (BETWEEN)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Partition selection

select * from <table> partition(<partition list>)

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - **Compound-Statement (Procedural Language)**
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

MySQL Compound-Statement
Procedural language for MySQL

 Reproduction interdite sans accord préalable

Procedures and functions

CREATE
[DEFINER = { ***user*** | CURRENT_USER }]
PROCEDURE ***sp_name*** ([***proc_parameter***[,...]])
[***characteristic*** ...] ***routine_body***

CREATE
[DEFINER = { ***user*** | CURRENT_USER }]
FUNCTION ***sp_name*** ([***func_parameter***[,...]])
RETURNS ***type***
[***characteristic*** ...] ***routine_body***

proc_parameter:
[IN | OUT | INOUT] ***param_name type***

func_parameter:
param_name type

type:
Any valid MySQL data type

characteristic:
COMMENT '***string***'
| LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }

routine_body:
Valid SQL routine statement

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

Procedure example

```
CREATE PROCEDURE title_count()
BEGIN
  SELECT 'Number of titles:', COUNT(*) FROM
titles;
END;
```

Function example

```
CREATE FUNCTION title_count() RETURNS int
BEGIN
  DECLARE count_titles int;
  SELECT COUNT(*) INTO count_titles FROM titles;
  RETURN count_titles;
END;
```

Blocks

- A block is defined as:

BEGIN … END

- Blocks can be nested:

```
BEGIN
        BEGIN
        END
END
```

- Blocks can be labeled:

```
my_label: BEGIN
END my_label
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Statement Label

[***begin_label***:] BEGIN
[***statement_list***]
END [***end_label***]

[***begin_label***:] LOOP
statement_list
END LOOP [***end_label***]

[***begin_label***:] REPEAT
statement_list
UNTIL ***search_condition***
END REPEAT [***end_label***]

[***begin_label***:] WHILE ***search_condition*** DO
statement_list
END WHILE [***end_label***]

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

DECLARE

- Used to declare variables inside a procedure or a function
- For each variable to declare, use the keyword DECLARE

```
CREATE PROCEDURE sp1()
BEGIN
        DECLARE a int;
        DECLARE b varchar(100);
[…]
END;
```

 Reproduction interdite sans accord préalable

Flow control statements

- Flow can be controled with:
  - CASE
  - IF
  - ITERATE
  - LEAVE
  - LOOP
  - REPEAT
  - RETURN
  - WHILE

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

CASE

CASE ***case_value***
WHEN ***when_value*** THEN ***statement_list***
[WHEN ***when_value*** THEN ***statement_list***] ...
[ELSE ***statement_list***]
END CASE

Or:

CASE
WHEN ***search_condition*** THEN ***statement_list***
[WHEN ***search_condition*** THEN ***statement_list***] ...
[ELSE ***statement_list***]
END CASE

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

IF

IF ***search_condition*** THEN ***statement_list***

[ELSEIF ***search_condition*** THEN ***statement_list***] ...

[ELSE ***statement_list***]

END IF

 Reproduction interdite sans accord préalable

ITERATE and LEAVE

- ITERATE label
- LEAVE label

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

LOOP

[***begin_label***:] LOOP

statement_list

END LOOP [***end_label***]

 Reproduction interdite sans accord préalable

REPEAT

[***begin_label***:] REPEAT

statement_list

UNTIL ***search_condition***

END REPEAT [***end_label***]

WHILE

[***begin_label***:] WHILE ***search_condition***

DO

statement_list

END WHILE [***end_label***]

CURSOR

- Cursors have to be defined after all declared variables and before any handler

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE a CHAR(16);
  DECLARE b, c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;

  OPEN cur1;
  OPEN cur2;

  read_loop: LOOP
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF done THEN
      LEAVE read_loop;
    END IF;
    IF b < c THEN
      INSERT INTO test.t3 VALUES (a,b);
    ELSE
      INSERT INTO test.t3 VALUES (a,c);
    END IF;
  END LOOP;

  CLOSE cur1;
  CLOSE cur2;
END;
```

Copyright 2014 – Oracle Corp

Reproduction interdite sans accord préalable

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - **Javascript**
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

What is Javascript?

- Javascript was first Livescript
- Renamed Javascript because quite similar as Java
- Light typed
- Object-Oriented but without the complexity of Java

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Where Javascript needs to be placed?

- Javascript is found in markups *<script></script>*
- It can be coded directly in these markups
- … or placed in a *.js* file
- In that last case the syntax is *<script src="myscript.js"></script>*
- … where *myscript.js* is your script
- Prefer to place the scripts in the *<head>* part of the HTML page

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Javascript and HTML objects events

- Javascript can be found inside objects

```
<input type="text" name="toto" id="toto"
onkeyup="{myFunctionForToto(this);}">
```

Javascript and DOM manipulations

- Javascript manipulates the DOM
- Every object of the DOM can be modified
- Javascript is made to create dynamically new objects
- … or to destroy objects

Literals

- Number literals:

10.5

100e10

-15

- String literals:

"This is a string"

'This is another string'

Variables

- They are defined with the keyword *var*
- Defining variables is mandatory

```
var x;
var y;
x = 10;
y = "Test";
```

- Variables have to be written as in Java, beginning with lowercase and uppercase for each word from the second

```
thisIsAVariable = "with a value";
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Data types

- Javascript uses dynamic data types

var a; // Undefined type

var a = 5; // Type is now numeric. Same variable

var a = "Test"; // Same variable, but now string

- Main types are:
  - Number
  - String
  - Boolean
  - Object

 Reproduction interdite sans accord préalable

Data types as objects

```
var a = new Number;
var b = new String;
var c = new Boolean;
var d = new Date();
```

 Reproduction interdite sans accord préalable

Date object

- new Date() // current date and time
- new Date(milliseconds) //milliseconds since 1970/01/01
- new Date(dateString)
- new Date(year, month, day, hours, minutes, seconds, milliseconds)
- … and many methods: see http://www.w3schools.com/jsref/jsref_obj_date.asp

Arrays

- Arrays are objects

```
var a = new Array();
a[0] = new Array(); // Multidimensional array
a[0][0] = "value";
a[1] = new Array();
a[1][0] = "another value";
var grades =
["Algorithmic","Mathematic","Database"];
var differentElements = ["1st element",2,true];
```

Number of elements in an array

```
var l = a.length;
```

Position of an element in an array

```
var p = grades.indexOf("Database");
```

Math object

- This objects contains constants and methods for mathematical operations
- Constants:

E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2

- Methods:

abs(x), acos(x), asin(x), atan(x), atan2(y,x), ceil(x), cos(x), exp(x), floor(x), log(x), max(x,y,z,…,n), min(x,y,z,…,n), pow(x,y), random(), round(x), sin(x), sqrt(x), tan(x)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Operators

- They are as in Java and other common languages
- Arithmetics:

+ - * / % ++ --

- Assignment:

= += -= *= /= %=

- Strings:

a = "this is";

b = " a concatenated string";

c = a + b;

Comparison and logical operators

- Comparison operators:

== === != !== > < >= <=

- Logical operators:

&& || !

Comments

- They are as in Java

// This is a comment

/* This is another one comment */

Case sensitivity

- Javascript is case sensitive

```
var thisVariable = "is not";
var thisvariable;
```

Instruction separator

- The instruction separator is the semicolon

```
alert("Next step");
s++;
```

Code block

- A code block is defined by curly brackets

```
{
        // This is a code block;
        a--;
}
```

- Code blocks are useful for control flow blocks

Conditions statements

```
if(<condition>){
        // instructions
}

if(<condition>){
        // instructions
}else{
        // instructions
}

if(<condition>){
        // instructions
}else if(<condition>){
        // instructions
}else{
        //instructions
}
```

Switch

```
switch(<expression>){
case n:
        // instructions
        [break;]
case m:
        // instructions
        [break;]
default:
        // instructions
}
```

For loop

```
for(<variable> = <value>;<ending condition>;<operation on the variable>){
        // instructions
}
```

While loop

```
while(<condition>){
        // instructions
}

do{
        // instructions
while(<condition>);
```

Break and continue

- *break* is used to leave the loop
- *continue* is used to go to the next iteration of the loop

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Errors management

```
try{
        // instructions
        throw "<message>";
}catch(<error>){
        // instructions
        // <error>.message returns the error message
        // in case of throw, <error> is the message
}
```

Regular expressions

- As in any other languages like awk and sed

/<pattern>/<modifiers>

Strict mode

- To protect the code, use strict mode:

"use strict";

- This disable the ability to create accidentally a global variable inside a local code
- This ensure the strict application of all the rules of the objects:
  - Impossible to delete a undeletable property
  - Impossible to write a read-only property
  - *with* is not allowed
  - …

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Javascript events

Events

- The events are linked to the objects present in a page
- An event is a kind of trigger on a specific action on a particular object
- Javascript handle a huge number of events
- This is the main usage of Javascript

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Main events

- onchange: When the state of an HTML element changes
- onclick: When the user clicks an HTML element
- onmouseover: When the mouse moves over an HTML element
- onmouseout: When the mouse moves away from an HTML element
- onkeydown: When the user pushes a keyboard key
- onload: When the browser has finished loading a page

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Complete list of events

- See: http://www.w3schools.com/jsref/dom_obj_event.asp

Some examples

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - **Ajax**
  - jQuery
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

What is Ajax?

- Ajax is the acronym for Asynchronous Javascript And XML
- Ajax is a toolset to create fast and dynamically WEB pages
- Ajax works with the events
- Ajax sends data to the server
- … and receives data, process it
- … to update the page content

Ajax objects

- XMLHttpRequest: object to exchange data asynchronously with the server
- Javascript/DOM: interaction with the page
- CSS: Style the data
- XML: transfer data between the client and the server

XMLHttpRequest

Var xmlhttp = new XMLHttpRequest(); // Defines an Ajax component

xmlhttp.onreadystatechange = function(){ … } // An unnamed function containing what to do when the result is returned by the serve

xmlhttp.open("[GET|POST]","<URL>",[true|false]); // Open a link with the server. The URL contains the file who will be processed server-side

xmlhttp.send(); // Sends the request

responseText vs responseXML

- Depending on the complexity of the returned data, use:
  - responseText for simple data
  - responseXML for complex data

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

responseXML

- Browsing the response can be like this:

```
var xml = xmlhttp.responseXML;
var i = 0;
while(xml.getElementByTagName(<tag>)[i]{
        // Process the element
        i++;
}
```

Ajax events

Source http://w3schools.com:

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readystate	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

PHP file returning data to Ajax

```
<?php
$res = "";
$res = process($_GET);
echo $res; // Echoing the variable is necessary
to return results to the Ajax caller
?>
```

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - **jQuery**
- Wordpress
  - Creating a plugin
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

What is jQuery?

- jQuery is a Javascript library made to simplify Javascript writing
- It simplifies Ajax writing too
- Easy to learn
- jQuery runs exactly same in all major browsers, including… IE 6!!

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Loading jQuery

- jQuery library needs to be loaded before used

```
<script src="jquery-1.11.0.min.js"></script>
```

- It's better to download the file in the same directory as all the other Javascript files of the application
- jQuery can be loaded from the WEB:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

jQuery syntax

$(selector).action()

- … where:
- $ defines jQuery
- (selector) is HTML element(s) (query)
- action() is the action to be performed on the element(s)

jQuery selectors

$("p") // All <p> elements of a page

$("#id") // Corresponds to the <*element* id="id">

$("p.classname") // All <p class="classname">

See: http://www.w3schools.com/jquery/jquery_ref_selectors.asp

… for the selectors reference

Trigger events

```
$(<element>).action(function(){
        // Code of the function
});
```

- Normally all the actions need to be encapsulated into:

```
$(document).ready()(function(){
        // All the methods
});
```

jQuery and Ajax

- jQuery simplifies Ajax
- Three methods are necessary to manage all Ajax needs:
  - $(selector).load(URL,data,callback)
  - $(selector).get(URL,data,callback)
  - $(selector).post(URL,data,callback)

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Some examples

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - **Creating a plugin**
  - Creating a widget

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Wordpress

- Wordpress is a Content Management System (CMS)
- Wordpress is a framework
- Wordpress is expandable through themes, plugins and widgets

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Wordpress structure

- wp-content
  - plugins
  - themes

Plugin structure

- plugin-name (directory)
  - plugin-name.php (mandatory file)
  - your own paths for all your program files

Plugin main file header

```
/*
Plugin Name: Module d'Administration des
Académiciens
Plugin URI: http://www.champavert-consulting.fr/
Description: Mise à jour des données des
académiciens présents dans la base acadmed
Version: 26.0
Author: Alexandra Champavert
Author URI: http://www.champavert-consulting.fr/
*/
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Shortcodes

- A shortcode is a code that can be inserted into a WP page: [shortcode-name]
- The WP PHP primitive corresponding to that shortcode is:

add_shortcode("shortcode-name",array(<php object>,"<php method of the object>"))

ex:

add_shortcode("maj_titres",array(& $maj_academiciens,"majTitres");

Class for WP for shortcodes

```
Class MAJAcademiciens {

[…]

 function majTitres(){

   $this->content = null;
// The code returning a string that can be displayed in the page containing the shortcode
   return $this->content;
  }

[…]
}
```

Wordpress and Ajax

- Needs to enqueue specific scripts inside WP:

```
wp_enqueue_script("alias_js_script",<js file>,array("jquery"),true);
```

- Needs to implement a standard action:

```
add_action("wp_ajax_nopriv_alias_method",array(<class variable>,"<method>"));
```

- … and a connected action:

```
add_action("wp_ajax_alias",array(<class variable>,"<method>"));
```

- … and finally the localization of the script:

```
wp_localize_script("alias","class name",array(
"ajaxurl" => admin_url("admin-ajax.php"),
"action" => "alias_method"));
```

Wordpress and Ajax example

```
wp_enqueue_script("maj_academiciens_cp_js_script",
plugin_dir_url(__FILE__)."js/cp_villes.js", array("jquery"),true);

add_action("wp_ajax_nopriv_maj_academiciens_cp_cpvsv",array(&
$maj_academiciens,"codesPostauxVillesSearchVilles"));

add_action("wp_ajax_maj_academiciens_cp_cpvsv",array(&
$maj_academiciens,"codesPostauxVillesSearchVilles"));

wp_localize_script("maj_academiciens_cp_js_script", "MAJAcademiciens",
array(
"ajaxurl" => admin_url("admin-ajax.php"),
"action" => "maj_academiciens_cp_cpvsv"
   )
);
```

Ajax inside the class for WP

```
class MAJAcademiciens {

function codesPostauxVillesSearchVilles(){

   $error_code = 0;
   […]
   header( "Content-Type: application/json" );
   echo json_encode(
      array(
         "error" => $error_code,
         "cpt_villes" => $cpt,
         "liste_villes" => $liste,
         "liste_codes_postaux" => $liste_cp
      )
   );
   exit;
 }
```

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

jQuery for WP for Ajax

```
jQuery("#search_villes").keyup(function(){
    jQuery.ajax({
        type: "POST",
        url: MAJAcademiciens.ajaxurl,
        datatype: "json",
        async: true,
        cache: false,
        timeout: 10000,
        data: {
            action: "maj_academiciens_cp_cpvsv",
            search_villes: jQuery("#search_villes:text").val()
        },
        success: function(data){
            if(data.error == 0){
                if(data.cpt_villes > 30){
//                  jQuery("p#show_liste_villes").html("<select name=\"liste_villes\" id=\"liste_villes\"></
select>");
                    jQuery("#liste_villes").empty();
                    jQuery("select#liste_villes").attr("size",1);
                    jQuery("select#liste_villes").css("display","none");
                    jQuery("p#show_villes").css("display","none");
                    jQuery("p#show_liste_villes").css("display","none");
                    jQuery("p#count_villes").css("display","inline-block");
                    jQuery("p#count_villes").html(data.cpt_villes + " communes");
                }else if(data.cpt_villes == 1){
                    jQuery("p#count_villes").css("display","none");
                    jQuery("p#show_villes").css("display","inline");
                    for(var i = 0; i < data.liste_villes.length;i++){
                        jQuery("#h_liste_villes").val(data.liste_villes[i].villes_id);
                        jQuery("#s_liste_villes").html(data.liste_villes[i].ville + " (" + data.liste_villes[i].no_dept + ")");
                    }
                    jQuery("#show_search_villes").css("display","none");
                    jQuery("#liste_villes").css("display","none");
                    jQuery("#s_liste_villes").css("display","inline");
                    jQuery("#search_codes_postaux").val("");
                    jQuery("#liste_codes_postaux").empty();
                    jQuery("#show_submit_insert_cp_ville").css("display","none");
                    jQuery("#show_codes_postaux").css("display","none");
                    jQuery("select#liste_codes_postaux").attr("size",1);
                    jQuery("select#liste_codes_postaux").css("display","none");
                    jQuery("p#count_codes_postaux").css("display","none");
                    jQuery("p#linked_codes_postaux").css("display","inline");
                    jQuery("#liste_villes").css("display","none");
jQuery("#show_search_codes_postaux").css("display","inline");
                    jQuery("select#liste_villes").attr("size",1);
                    liste_cp = "";
                    if(data.liste_codes_postaux.length){
                        for(var i = 0; i < data.liste_codes_postaux.length;i++){
                            liste_cp = liste_cp + "<div style=\"width: 200px\; position: relative; display: inline-block;\">" +
data.liste_codes_postaux[i].code_postal + "</div>";
                        }
                    }
                    jQuery("p#linked_codes_postaux").html(liste_cp);
                }else if(data.cpt_villes == 0){
                    jQuery("#liste_villes").empty();
                    jQuery("p#count_villes").css("display","inline");
                    jQuery("p#show_villes").css("display","none");
                    jQuery("p#show_liste_villes").css("display","inline");
                    jQuery("select#liste_villes").css("display","inline");
                    jQuery("select#liste_villes").attr("size",data.liste_villes.length);
                }else{
                    jQuery("#liste_villes").empty();
                    jQuery("#liste_villes").css("display","inline");
                    jQuery("p#count_villes").css("display","none");
                    jQuery("p#show_villes").css("display","inline");
                    jQuery("p#show_liste_villes").css("display","inline");
                    jQuery("select#liste_villes").css("display","inline");
                    jQuery("select#liste_villes").attr("size",data.liste_villes.length);
                    for(var i = 0; i < data.liste_villes.length;i++){
                        jQuery("select#liste_villes").append("<option value=\"" + data.liste_villes[i].villes_id +"\">" +
data.liste_villes[i].ville + " (" + data.liste_villes[i].no_dept + ")</option>");
                    }
                }
            }
            else{
                jQuery("p#count_villes").html("Liste trop longue");
            }
        },
        error: function(XMLHttpRequest,textStatus,errorThrown){
            jQuery("p#count_villes").html("Aucune commune");
        }
    });
});
```

Ajax, details

```
jQuery("#search_villes").keyup(function(){
    jQuery.ajax({
        type: "POST",
        url: MAJAcademiciens.ajaxurl,
        datatype: "json",
        async: true,
        cache: false,
        timeout: 10000,
        data: {
            action: "maj_academiciens_cp_cpvsv",
            search_villes: jQuery("#search_villes:text").val()
        },
        success: function(data){
        […]
                for(var i = 0; i < data.liste_villes.length;i++){
                    jQuery("select#liste_villes").append("<option value=\"" + data.liste_villes[i].villes_id +"\">" +
data.liste_villes[i].ville + " (" + data.liste_villes[i].no_dept + ")</option>");
                }
        […]
        },
        error: function(XMLHttpRequest,textStatus,errorThrown){
            jQuery("#" + count_ville + n).html("Aucune commune");
        }
    });
```

Using the WP database

```
function getCompteVilles($debut){
  global $wpdb;

  $stmt = $this->sqlInst["get-compte-villes"];
  $stmt = str_replace("[prefix]", $this->prefix_views, $stmt);
  $sql = $wpdb->prepare($stmt,$debut."%");
  // echo $sql;
  return($wpdb->get_results($sql));

 }
```

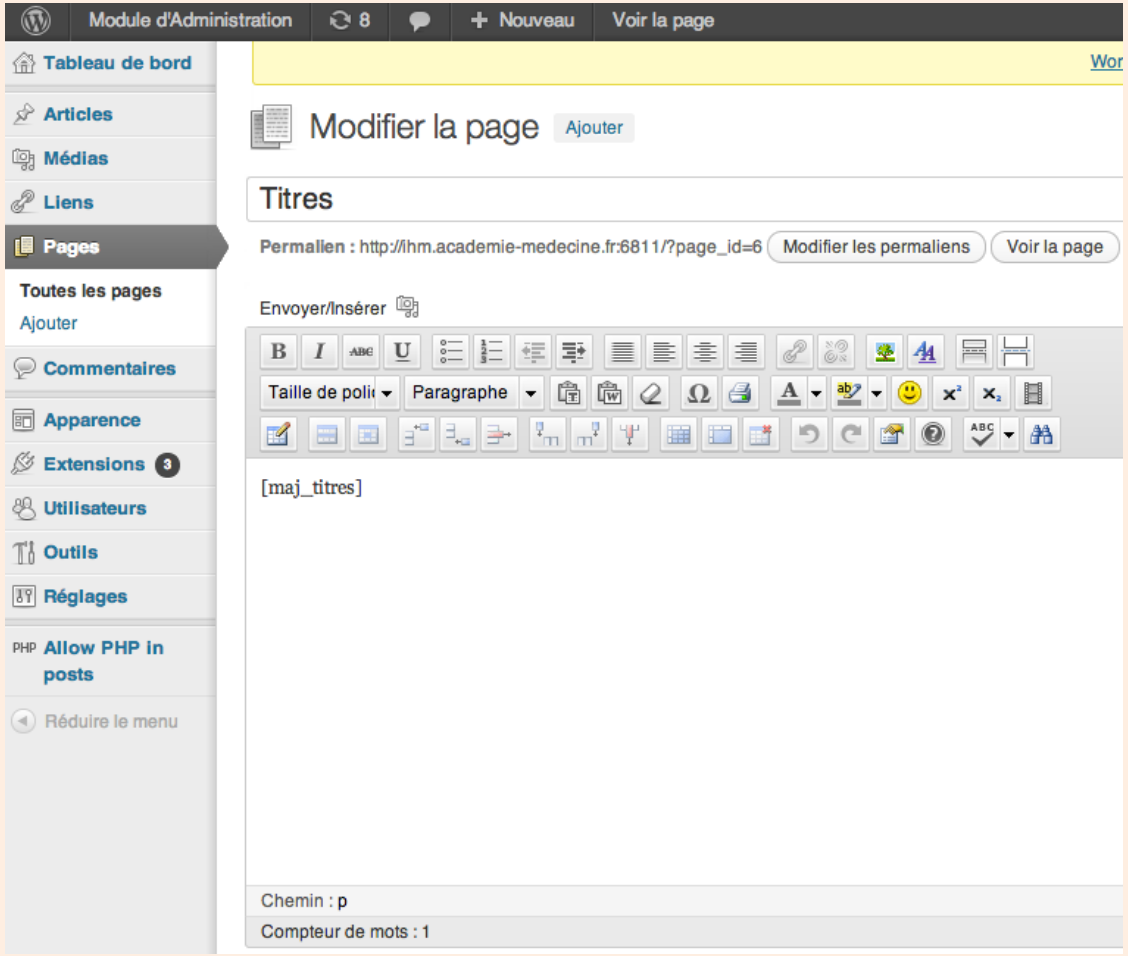
The query handled by this function is:

select count(1) as compte_villes from [prefix]v_villes where ville_description like %s /* +NO_PARSE */

How it appears in the administration panel?

☐	**Module d'Administration des Académiciens** Désactiver	Mise à jour des données des académiciens présents dans la base acadmed Version 0.1 I Par Alexandra Champavert I Aller sur le site de l'extension

How can it be called in a page (example)?



What is generated by the function inside the plugin?

Session plan

- Introduction to databases
  - Databases and management system
  - Normal forms
- Fundamentals
  - HTML & CSS
  - PHP
- Connecting from PHP to Databases
  - PDO
- MySQL
  - SQL language
  - Compound-Statement (Procedural Language)
- Advanced WEB development
  - Javascript
  - Ajax
  - jQuery
- Wordpress
  - Creating a plugin
  - **Creating a widget**

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

Widget structure

- Same as plugin structure

Widget main file header

- Same as plugin

Class definition

```
class LastPublicationsWidget extends WP_Widget
{
  var $arrayDataAchAnmPublications;
  var $arrayRes;

  function LastPublicationsWidget()
  {
    $widget_ops = array('classname' => 'LastPublicationsWidget',
'description' => 'Présente les dernières publications' );
    $this->WP_Widget('LastPublicationsWidget', 'Dernières
Publications', $widget_ops);
  }
```

Menu entry in the administration panel

```
function form($instance)
 {
  $instance = wp_parse_args( (array) $instance, array( 'title' => '' ) );
  $title = $instance['title'];
?>
 <p><label for="<?php echo $this->get_field_id('title'); ?>">Title: <input class="widefat" id="<?php echo $this-
>get_field_id('title'); ?>" name="<?php echo $this->get_field_name('title'); ?>" type="text" value="<?php echo
attribute_escape($title); ?>" /></label></p>
 <p>
  <label for="<?php echo $this->get_field_id('numberposts'); ?>">Number of posts:</label>
  <select id="<?php echo $this->get_field_id('numberposts'); ?>" name="<?php echo $this-
>get_field_name('numberposts'); ?>">
  <?php for ($i=1;$i<=20;$i++) {
      echo '<option value="'.$i.'"';
      if ($i==$instance['numberposts']) echo ' selected="selected"';
      echo '>'.$i.'</option>';
     } ?>
    </select>
 </p>
<?php
 }
```

Updating the data from admin panel

```
function update($new_instance, $old_instance)
 {
   $instance = $old_instance;
   $instance['title'] = $new_instance['title'];
   $instance['numberposts'] =
$new_instance['numberposts'];
   return $instance;
 }
```

The widget

```
function widget($args, $instance){
   extract($args, EXTR_SKIP);
   [...]
    if (!empty($title)){
     echo $title;
   }
   // All the data to display via echo or printf
}
```

Register the widget

```
add_action( 'widgets_init', create_function('', 'return
register_widget("LastPublicationsWidget");'));
```

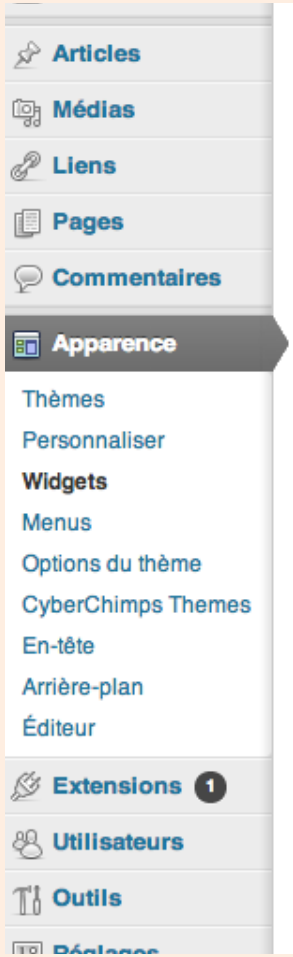
How it appears in the administration panel?

☐	**Module d'Administration des Académiciens** Désactiver	Mise à jour des données des académiciens présents dans la base acadmed Version 0.1 I Par Alexandra Champavert I Aller sur le site de l'extension

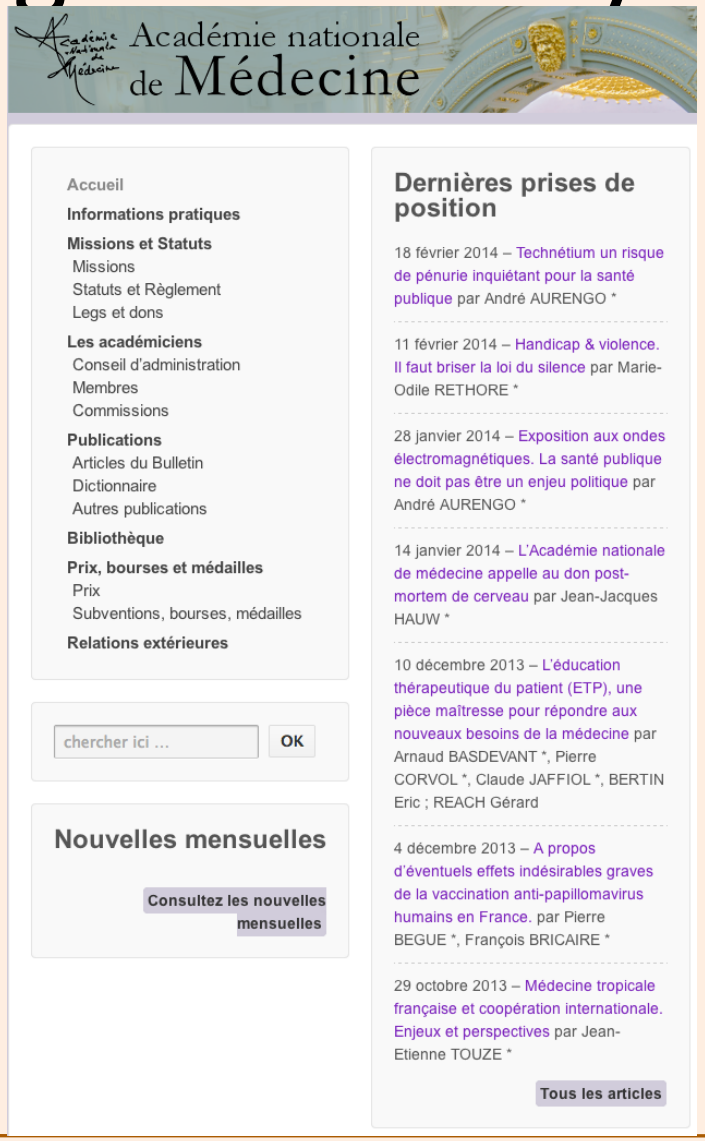
Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable

How can it be setup?



What is generated by the widget?



And now…

THE PROJECT!

Good luck!

Copyright 2014 – Alexandra Champavert

Reproduction interdite sans accord préalable