

# **Bases de Données 2**

## **1<sup>ère</sup> partie du module M3202**

### **Développement web 3**

# **MMI Vichy 2<sup>ème</sup> année**

# Quelques références

- Livres
  - MySQL 5 Le Guide Complet. Antoine Dinimant. MicroApplication
  - MySQL – Précis et Concis. George Reese. O'Reilly
  - Maîtrise de MySQL 5. Yves Darmailiac et Philippe Rigaux. O'Reilly
  - Administrez vos bases de données avec MySQL. Chantal Gribaumont. Simple IT
- Sites web
  - Site officiel en Français : <https://www.mysql.fr/>
  - Documentation officielle en Anglais de MySQL : <http://dev.mysql.com/doc/>
  - Site de Chantal Gribaumont sur openclassrooms:  
<https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql>

- **Etude de cas : bd shuttershop 4**
  - description 5,
  - cas d'utilisation 6,
  - diag. de classes 8,
  - normalisation 13,
  - diag. de classes 18
  - schéma relationnel 22
  - code SQL 23
  - utilisateurs 32
- **MySQL : Installation, Configuration 36**
  - dimensionnement 37
  - configuration 39
- **MySQL : éléments d'administration 47**
  - outils MySQL 50
  - structure du serveur 55
- **Gestion des utilisateurs 60**
  - création d'un utilisateur 63
  - connexion 65
  - privileges 72
- **Politique de sécurité 77**
- **MySQL : éléments d'administration (suite) 83**
- fonctionnement du serveur 84
- surveillance du serveur 86
- exemple de script 89
- sauvegardes 91
- MySQLdump 97
- restaurations 103
- bonnes pratiques 110
- **Compléments SQL 111**
  - types de données 114
  - opérateurs 120 et fonctions 126
  - agrégation et regroupement 142
  - sous-requêtes 151
  - mise à jour et ajout de données 157
    - ajout 158
    - mode strict 163
    - modification 164
    - suppression 165
    - remplacement 166
    - modifications multitable 168
  - vues 171
    - définition et utilité 172
    - création et gestion des vues 173
    - modification des données à travers les vues 176
    - exercice 178
- **SQL dynamique 182**
  - procédures stockées 185
  - variables 190
  - fonctions stockées 195
  - privilèges 197
  - tests 201
  - boucles 202
  - gestionnaires d'erreurs 203
  - curseurs 205
  - déclencheurs 207
- **Transactions 213**
  - notion de transaction 217
  - auto commit 219
  - verrous 221
  - niveaux d'isolation 225
  - exemple 228
- **Retour sur l'étude de cas 238**
  - exercices SQL 239
  - dimensionnement 242
  - utilisateurs 244
  - procédures stockées 248
- **Annexe 258**
  - phpMyAdmin 259

# Etude de cas

*Base de données  
pour la gestion de photos et de clients  
achetant les droits de ces photos*

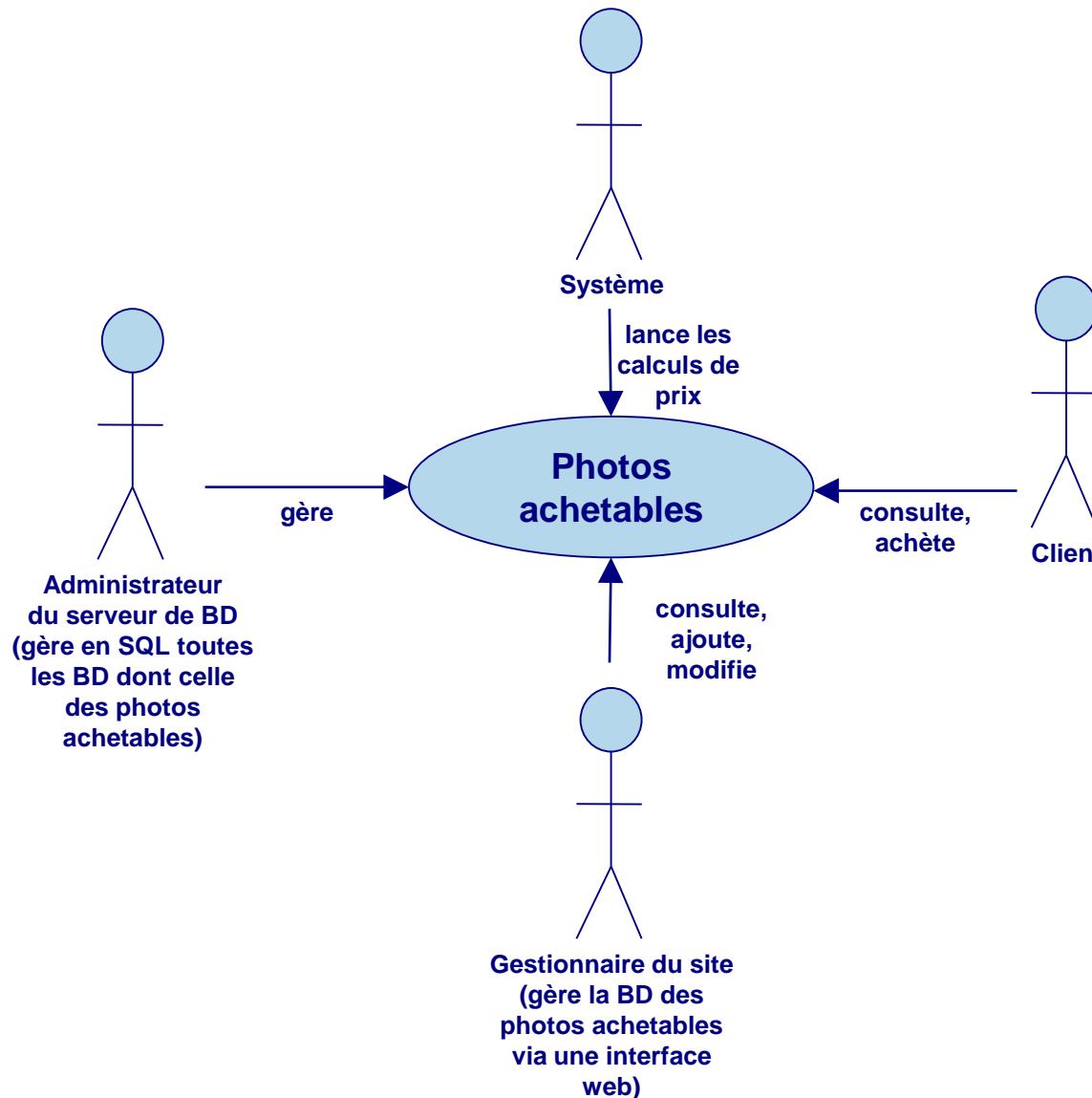
Extrait du projet d'un étudiant de l'université de Standford  
<http://infolab.stanford.edu/~ullman/fcdb/hernandez/PDA1.htm>

christophe.rey@uca.fr

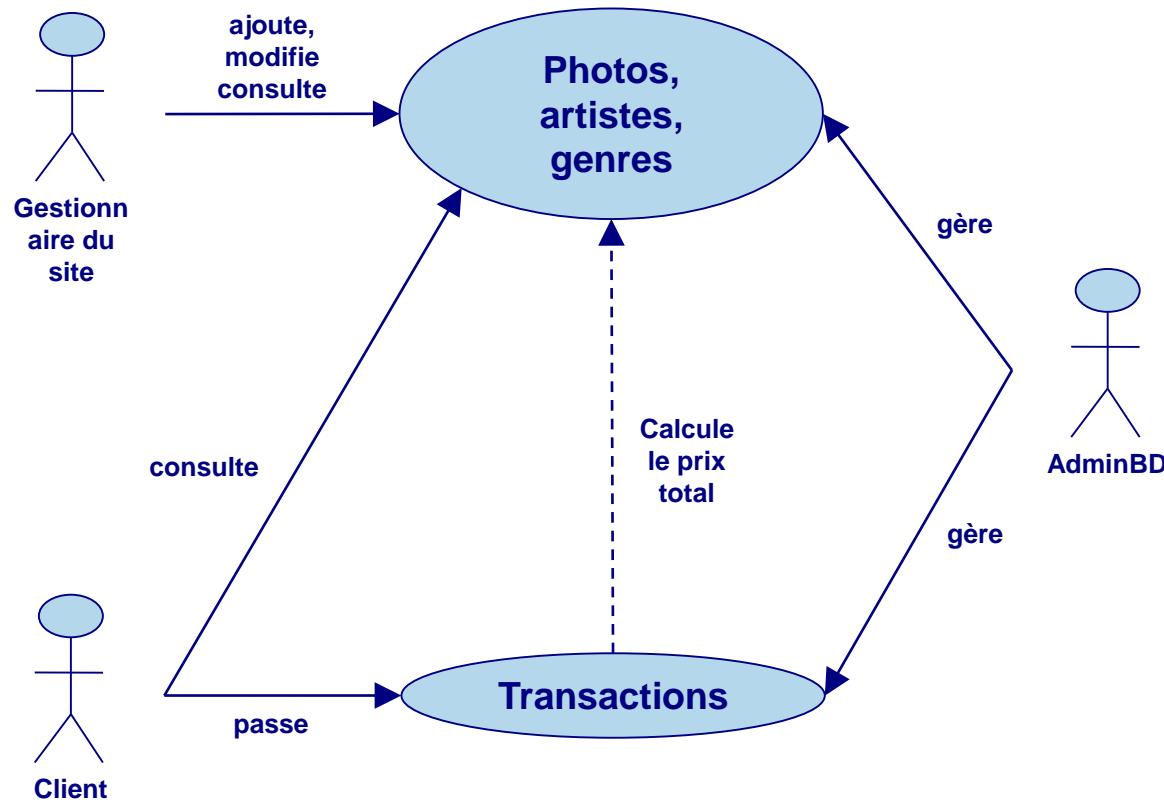
# Description

- Gestion d'une base de photos dont les droits peuvent être achetés sur internet. 2 grandes parties :
  - Le catalogue des photos, les genres, les photographes
  - Les transactions et les clients
- Photos :
  - le type (N&B ou couleur), le type de film, la vitesse d'obturation, le prix,...
- Genres :
  - paysages, portraits, abstraits
- Photographes et modèles (de portraits) :
  - info biographiques
- Clients :
  - nom d'utilisateur, mot de passe,...
- Transactions :
  - peuvent contenir plusieurs achats.
  - ont un prix total qui doit être calculé à partir des prix de chaque photo.

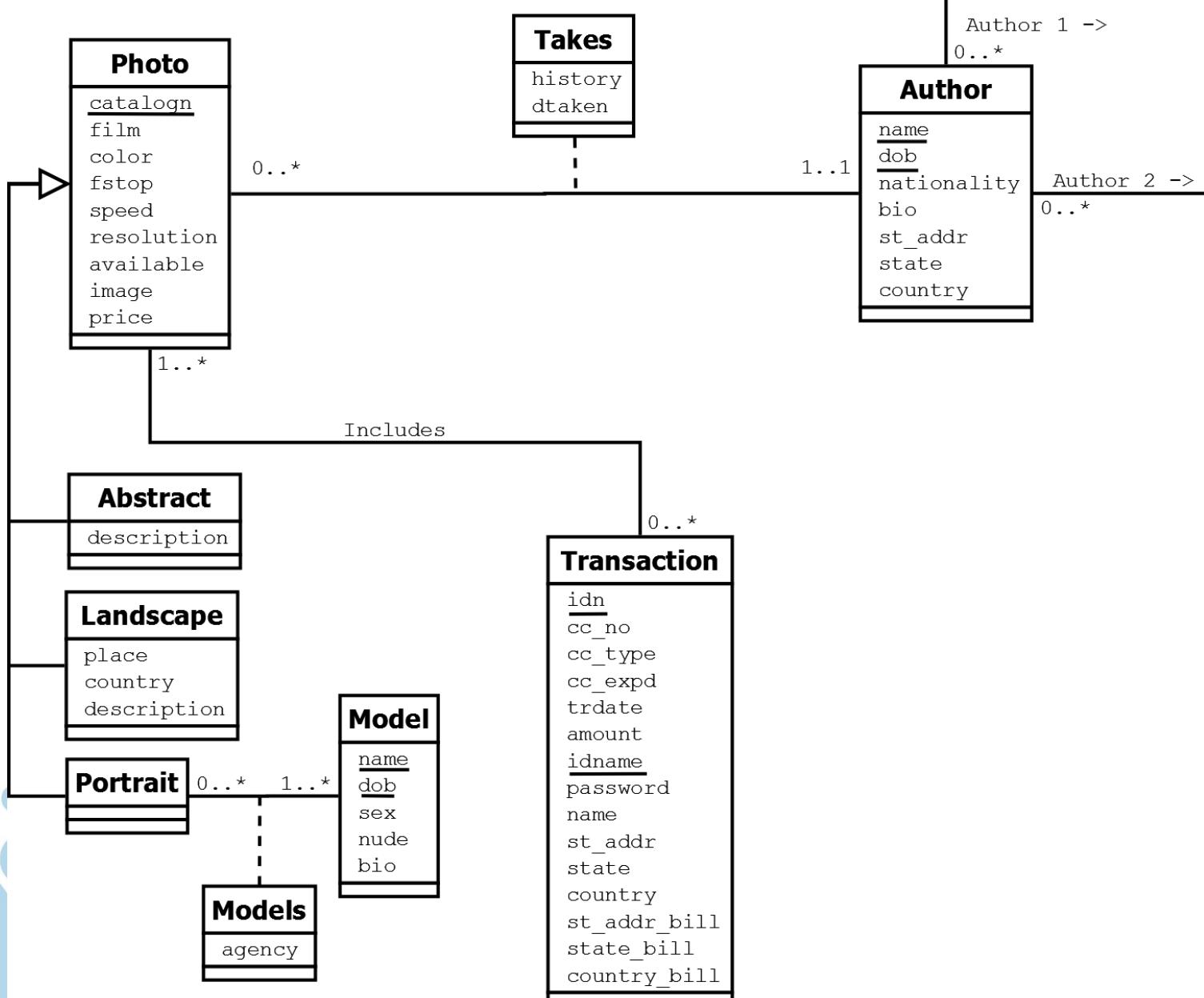
# Analyse : cas d'utilisation 1



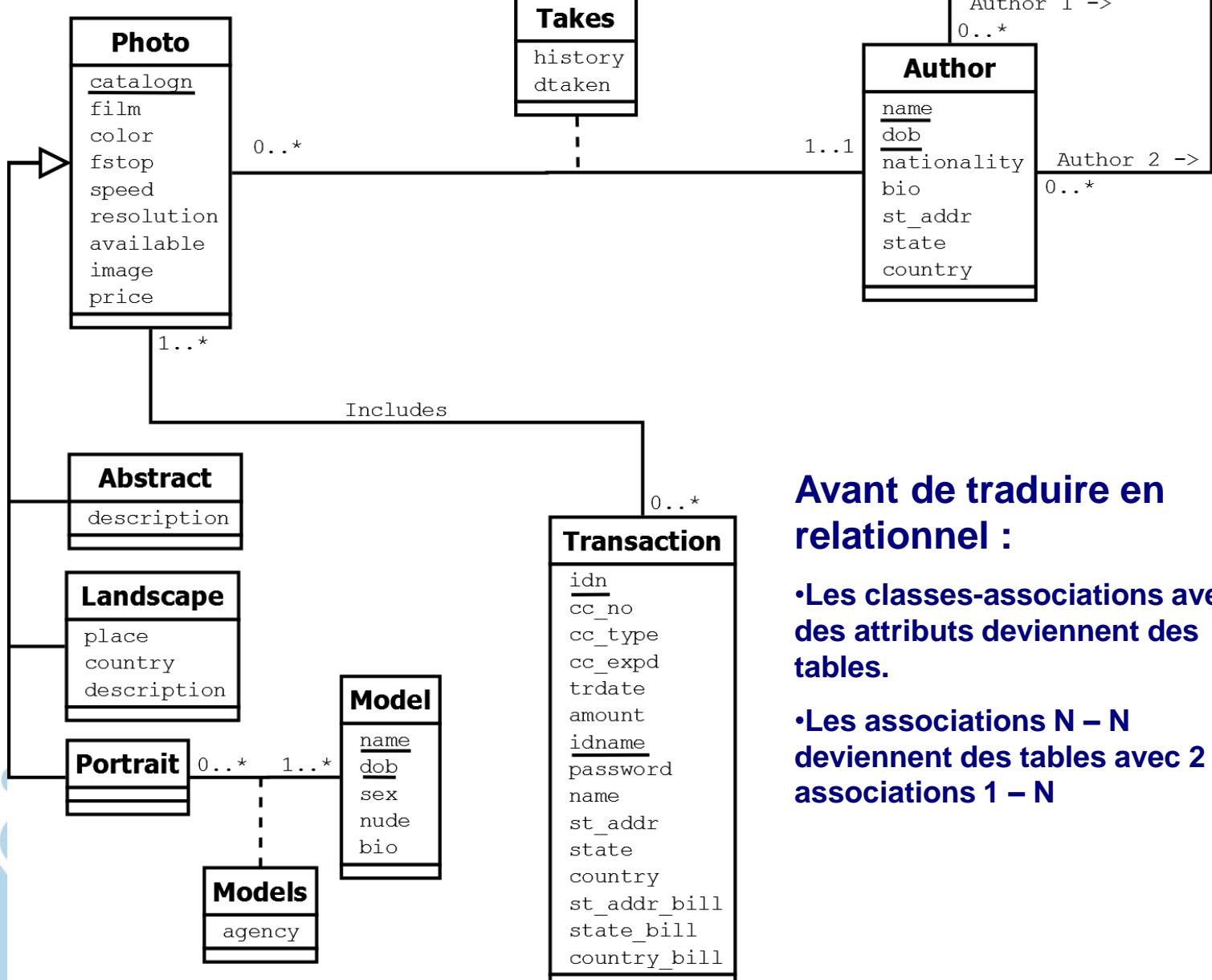
# Analyse : cas d'utilisation 2



# Analyse : diagramme de classes UML v1



# Analyse : diagramme de classes UML v1



Avant de traduire en relationnel :

- Les classes-associations avec des attributs deviennent des tables.
- Les associations N – N deviennent des tables avec 2 associations 1 – N

- catalogn : numéro de la photo dans le catalogue
- film : type de film utilisé (numérique ou argentique)
- color : couleurs 'C' ou noir et blanc 'B'
- fstop : focale
- speed : vitesse d'obturation
- resolution : nombre de pixels sur capteur si numérique
- available : disponibilité 'Y' ou 'N'
- image : nom du fichier image
- price : somme en Euros avec 2 chiffres après la virgule

- history : notes sur le moment où la photo a été prise
- dtaken : date à laquelle la photo a été prise

- name : nom du photographe
- dob : date de naissance du photographe
- nationality : nationalité du photographe
- bio : courte biographie du photographe
- st\_addr : adresse du photographe (numéro et rue)
- state : adresse du photographe (département)
- country : adresse du photographe (pays)

- description : description de la photo

- place : lieu du paysage de la photo
- country : pays
- description : description du paysage

- name : nom de la personne photographiée
- dob : date de naissance
- sex : sexe
- nude : photo de nu ou non
- bio : courte biographie de la personne photographiée

- agency : agence de mannequins à laquelle appartient la personne photographiée (le cas échéant)

- idn : numéro de la transaction
- cc\_no : numéro de la carte de crédit
- cc\_type : type de carte de crédit
- cc\_expd : date d'expiration de la carte de crédit
- trdate : date de la transaction
- amount : montant de la transaction
- idname : identifiant du client
- password : mot de passe du client
- name : nom (complet) du client
- st\_addr : adresse du client (nom de la rue et numéro)
- state : adresse du client (département)
- country : adresse du client (pays)
- st\_addr\_bill : adresse de facturation (nom de la rue et numéro)
- state\_bill : adresse de facturation (département)
- country\_bill : adresse de facturation (pays)

# Analyse : Schéma relationnel version 1

- Photo(catalogn, film, color, fstop, speed, image, resolution, available, price)
- Landscape(catalogn, place, country, description)  
CE1 : catalogn fait référence à Photo.catalogn
- Portrait(catalogn)  
CE1 : catalogn fait référence à Photo.catalogn
- Models(catalogn, name, dob, agency)  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (name,dob) fait référence à Model.(name, dob)
- Model(name, dob, sex, nude, bio)
- Abstract(catalogn, description)  
CE1 : catalogn fait référence à Photo.catalogn
- Takes(catalogn, name, dob, dtaken, history)  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (name,dob) fait référence à Author.(name, dob)  
(name,dob) NOT NULL
- Author(name, dob, nationality, bio, st\_addr, state, country)
- Influences(auth1\_name, auth1\_dob, auth2\_name, auth2\_dob)  
CE1 : (auth1\_name,auth1\_dob) fait référence à Author.(name, dob)  
CE2 : (auth2\_name,auth2\_dob) fait référence à Author.(name, dob)
- Includes(catalogn, idn, idname)  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (idn,idname) fait référence à Transaction.(idn,idname)
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)  
CE1 : (idn,idname) fait référence à Photo.(idn,idname)

# Analyse : Schéma relationnel version 1

- Schéma non normalisé (présence de dépendances fonctionnelles)
- Transaction n'est pas 2NF car :
  - Les infos du client sont mélangées avec celles de la transaction.
  - Présence de la Dépendance Fonctionnelle (DF) :  
 $\text{idname} \rightarrow (\text{password}, \text{name}, \text{st\_addr}, \text{state}, \text{country}, \text{st\_addr\_bill}, \text{state\_bill}, \text{country\_bill})$
- Landscape n'est pas en 3NF car :
  - $(\text{place}, \text{country}) \rightarrow \text{description}$
- De plus, dans Transaction,  $\text{idn} \rightarrow \text{idname}$   
Donc la clé primaire de Transaction n'est pas la plus petite possible (en nombre d'attributs).

# Analyse : Normalisation

## Version 1:

- Landscape(catalogn, place, country, description)  
CE1 : catalogn fait référence à Photo.catalogn
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)
- Version 1 normalisée = version 2
  - Landscape(catalogn, *place*, *country*)  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (*place*, *country*) fait référence à Location.(*place*,*country*)
  - Location(place, country, description)
  - Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname)  
CE1 : idname fait référence à Customer.idname
  - Customer(idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)

# Analyse : Normalisation

## Version 1:

- Landscape(catalogn, place, country, ~~description~~)  
CE1 : catalogn fait référence à Photo.catalogn
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, ~~idname~~, ~~password~~, ~~name~~, ~~st\_addr~~, ~~state~~, ~~country~~, ~~st\_addr\_bill~~, ~~state\_bill~~, ~~country\_bill~~)

## • Version 1 normalisée = version 2

- Landscape(catalogn, place, country)  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (place, country) fait référence à Location.(place,country)
- Location(place, country, description)
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname)  
CE1 : idname fait référence à Customer.idname
- Customer(idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)

# Analyse : Normalisation

## Version 1 :

- Landscape(catalogn, place, country, description)  
CE1 : catalogn fait référence à Photo.catalogn
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)
- Version 1 normalisée, autre possibilité = version 2'
  - Landscape(catalogn)  
CE1 : catalogn fait référence à Photo.catalogn
  - LocatedIn(catalogn, place, country)  
CE1 : catalogn fait référence à Landscape.catalogn  
CE2 : (place, country) fait référence à Location.(place,country)
  - Location(place, country, description)
  - Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd)
  - Buys(idn, idname)  
CE1 : idn fait référence à Transaction.idn  
CE2 : idname fait référence à Customer.idname
  - Customer(idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)

# Analyse : Normalisation

## Version 1 :

- Landscape(catalogn, place, country, description)  
CE1 : catalogn fait référence à Photo.catalogn
- Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd, idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)
- Version 1 normalisée, autre possibilité = version 2'
  - Landscape(catalogn)  
CE1 : catalogn fait référence à Photo.catalogn
  - LocatedIn(catalogn, place, country)  
CE1 : catalogn fait référence à Landscape.catalogn  
CE2 : (place, country) fait référence à Location.(place,country)
  - Location(place, country, description)
  - Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd)

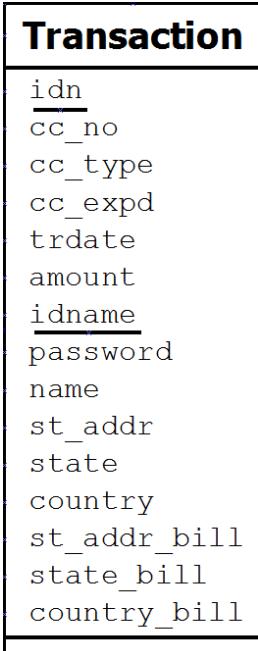
Met en valeur  
le lien entre les  
photos et les  
infos décrivant  
le lieu

Met en valeur  
le lien entre les  
transactions et  
les clients

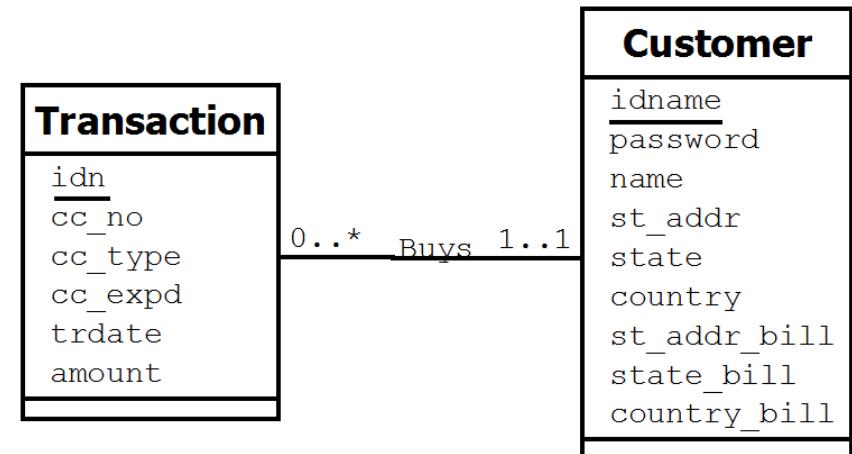
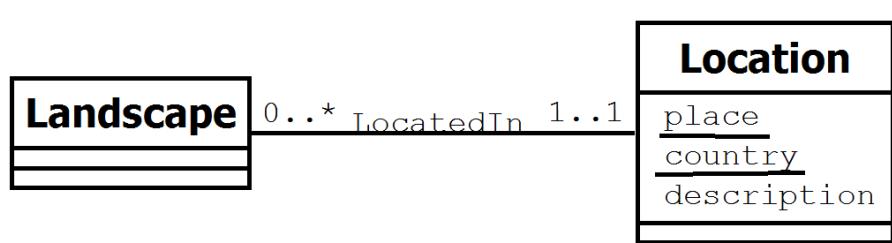
# Analyse : diagramme de classes UML normalisé

- Les versions 2 et 2' aboutissent au même diagramme de classes UML normalisé :

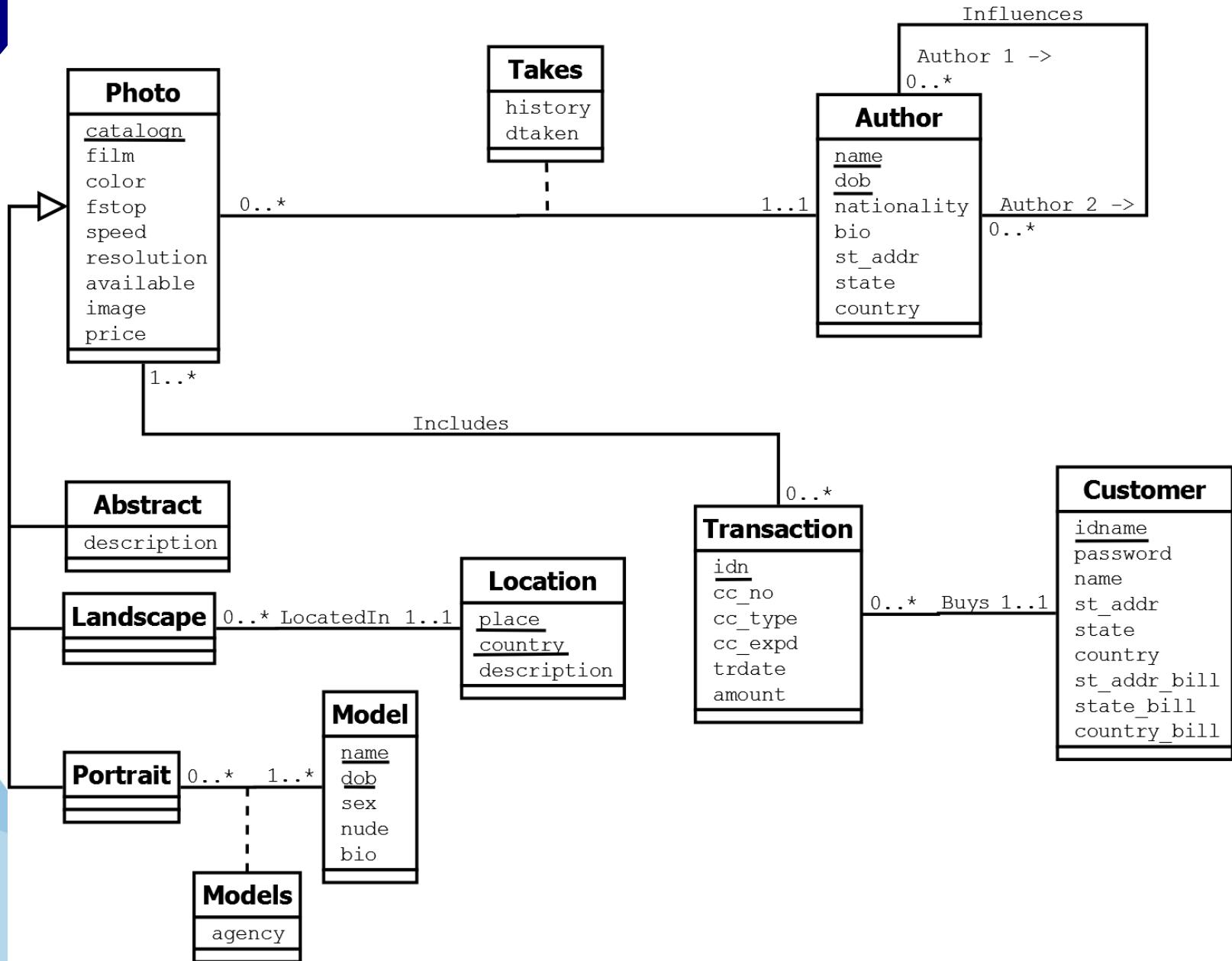
Avant normalisation



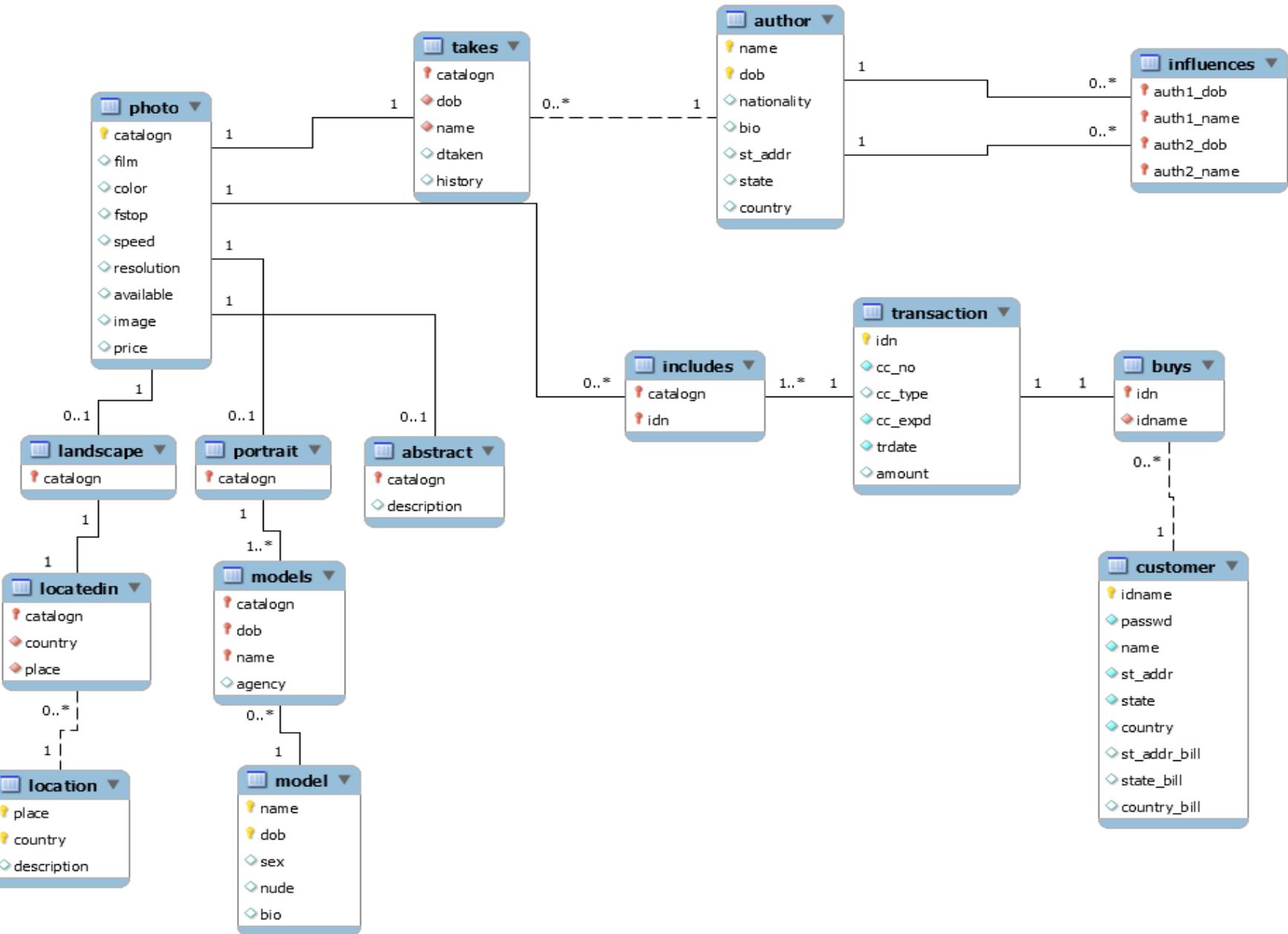
Après normalisation



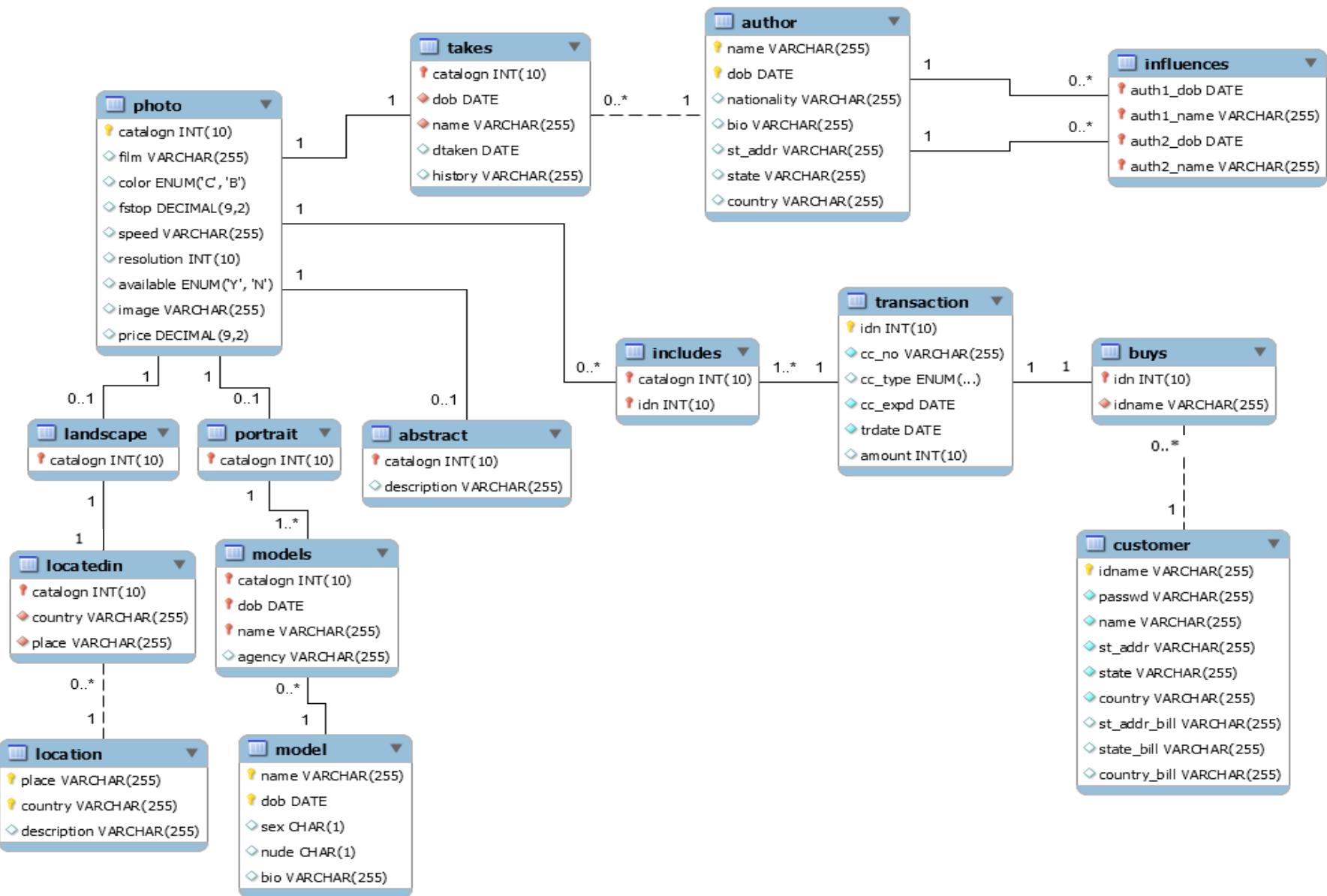
# Diagrammes de classes final (normalisé)



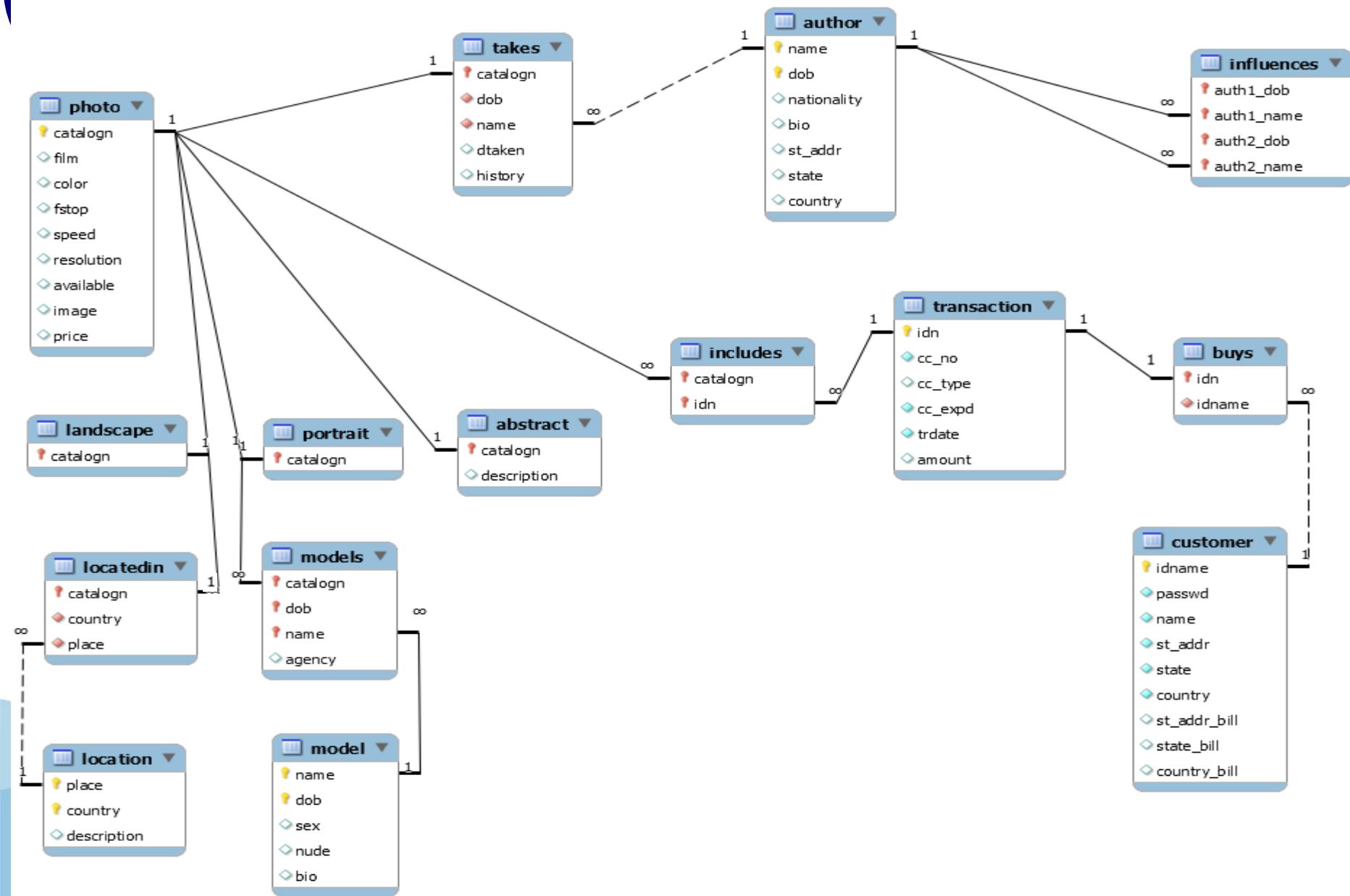
# Diagramme UML dans MySQL workbench



# Diagramme UML dans MySQL workbench



# Autre diagramme dans MySQL workbench



# Analyse : Schéma relationnel final (version 2')

- **Photo(catalogn, film, color, fstop, speed, image, resolution, available, price)**
- **Portrait(catalogn)**  
CE1 : catalogn fait référence à Photo.catalogn
- **Models(catalogn, name, dob, agency)**  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (name,dob) fait référence à Model.(name, dob)
- **Model(name, dob, sex, nude, bio)**
- **Abstract(catalogn, description)**  
CE1 : catalogn fait référence à Photo.catalogn
- **Takes(catalogn, name, dob, dtaken, history)**  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : (name,dob) fait référence à Author.(name, dob)
- **Author(name, dob, nationality, bio, st\_addr, state, country)**
- **Influences(auth1\_name, auth1\_dob, auth2\_name, auth2\_dob)**  
CE1 : (auth1\_name,auth1\_dob) fait référence à Author.(name, dob)  
CE2 : (auth2\_name,auth2\_dob) fait référence à Author.(name, dob)
- **Includes(catalogn, idn)**  
CE1 : catalogn fait référence à Photo.catalogn  
CE2 : idn fait référence à Transaction.idn
- **Landscape(catalogn)**  
CE1 : catalogn fait référence à Photo.catalogn
- **LocatedIn(catalogn, place, country)**  
CE1 : catalogn fait référence à Landscape.catalogn  
CE2 : (place, country) fait référence à Location.(place,country)
- **Location(place, country, description)**
- **Transaction(idn, cc\_no, trdate, amount, cc\_type, cc\_expd)**
- **Buys(idn, idname)**  
CE1 : idn fait référence à Transaction.idn  
CE2 : idname fait référence à Customer.idname
- **Customer(idname, password, name, st\_addr, state, country, st\_addr\_bill, state\_bill, country\_bill)**

# Commandes SQL de création (version 2')

- -- MySQL Workbench Forward Engineering
- SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;
- SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;
- SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';
- -----
- -- Schema mydb
- -----
- -----
- -- Schema shuttershop3
- -----
- -----
- -- Schema shuttershop3
- -----
- CREATE SCHEMA IF NOT EXISTS `shuttershop3` DEFAULT CHARACTER SET latin1 ;
- USE `shuttershop3` ;
- -----
- -- Table `shuttershop3`.`photo`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`photo` (  
    `catalogn` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT,  
    `film` VARCHAR(255) NULL DEFAULT NULL,  
    `color` ENUM('C', 'B') NULL DEFAULT NULL,  
    `fstop` DECIMAL(9,2) NULL DEFAULT NULL,  
    `speed` VARCHAR(255) NULL DEFAULT NULL,  
    `resolution` INT(10) UNSIGNED NULL DEFAULT NULL,  
    `available` ENUM('Y', 'N') NULL DEFAULT NULL,  
    `image` VARCHAR(255) NULL DEFAULT NULL,  
    `price` DECIMAL(9,2) NULL DEFAULT NULL,  
    PRIMARY KEY ( `catalogn` ),  
    INDEX `available` ( `available` ASC ))
- ENGINE = InnoDB
- AUTO\_INCREMENT = 30
- DEFAULT CHARACTER SET = latin1;

# Commandes SQL de création (version 2')

- -----  
• -- Table `shuttershop3`.`abstract`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`abstract` (  
•   `catalogn` INT(10) UNSIGNED NOT NULL,  
•   `description` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY (`catalogn`),  
•   INDEX `Abstract\_FKIndex1` (`catalogn` ASC),  
•   CONSTRAINT `abstract\_ibfk\_1`  
•     FOREIGN KEY (`catalogn`)  
•     REFERENCES `shuttershop3`.`photo` (`catalogn`)  
•     ON DELETE NO ACTION  
•     ON UPDATE NO ACTION)  
•   ENGINE = InnoDB  
•   DEFAULT CHARACTER SET = latin1;
- -----  
• -- Table `shuttershop3`.`author`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`author` (  
•   `name` VARCHAR(255) NOT NULL,  
•   `dob` DATE NOT NULL,  
•   `nationality` VARCHAR(255) NULL DEFAULT NULL,  
•   `bio` VARCHAR(255) NULL DEFAULT NULL,  
•   `st\_addr` VARCHAR(255) NULL DEFAULT NULL,  
•   `state` VARCHAR(255) NULL DEFAULT NULL,  
•   `country` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY (`name`, `dob`))  
•   ENGINE = InnoDB  
•   DEFAULT CHARACTER SET = latin1;

# Commandes SQL de création (version 2')

- -----
- -- Table `shuttershop3`.`transaction`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`transaction` (  
•   `idn` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT,  
•   `cc\_no` VARCHAR(255) NOT NULL,  
•   `cc\_type` ENUM('V', 'M', 'A', 'D') NULL DEFAULT NULL,  
•   `cc\_expd` DATE NOT NULL,  
•   `trdate` DATE NOT NULL,  
•   `amount` INT(10) UNSIGNED NULL DEFAULT NULL,  
•   PRIMARY KEY ( `idn` ))  
•   ENGINE = InnoDB  
•   AUTO\_INCREMENT = 6  
•   DEFAULT CHARACTER SET = latin1;
- -----
- -- Table `shuttershop3`.`customer`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`customer` (  
•   `idname` VARCHAR(255) NOT NULL,  
•   `passwd` VARCHAR(255) NOT NULL,  
•   `name` VARCHAR(255) NOT NULL,  
•   `st\_addr` VARCHAR(255) NOT NULL,  
•   `state` VARCHAR(255) NOT NULL,  
•   `country` VARCHAR(255) NOT NULL,  
•   `st\_addr\_bill` VARCHAR(255) NULL DEFAULT NULL,  
•   `state\_bill` VARCHAR(255) NULL DEFAULT NULL,  
•   `country\_bill` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY ( `idname` ))  
•   ENGINE = InnoDB  
•   DEFAULT CHARACTER SET = latin1;

# Commandes SQL de création (version 2')

```
-----  
-- Table `shuttershop3`.`buys`  
-----  
CREATE TABLE IF NOT EXISTS `shuttershop3`.`buys` (  
    `idn` INT(10) UNSIGNED NOT NULL,  
    `idname` VARCHAR(255) NOT NULL,  
    PRIMARY KEY (`idn`),  
    INDEX `Buys_FKIndex1` (`idn` ASC),  
    INDEX `Buys_FKIndex2` (`idname` ASC),  
    INDEX `lidname` (`idname` ASC),  
    CONSTRAINT `buys_ibfk_1`  
        FOREIGN KEY (`idn`)  
        REFERENCES `shuttershop3`.`transaction` (`idn`)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    CONSTRAINT `buys_ibfk_2`  
        FOREIGN KEY (`idname`)  
        REFERENCES `shuttershop3`.`customer` (`idname`)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = latin1;  
  
-----  
-- Table `shuttershop3`.`includes`  
-----  
CREATE TABLE IF NOT EXISTS `shuttershop3`.`includes` (  
    `catalogn` INT(10) UNSIGNED NOT NULL,  
    `idn` INT(10) UNSIGNED NOT NULL,  
    PRIMARY KEY (`catalogn`, `idn`),  
    INDEX `Includes_FKIndex1` (`catalogn` ASC),  
    INDEX `Includes_FKIndex2` (`idn` ASC),  
    CONSTRAINT `includes_ibfk_1`  
        FOREIGN KEY (`catalogn`)  
        REFERENCES `shuttershop3`.`photo` (`catalogn`)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    CONSTRAINT `includes_ibfk_2`  
        FOREIGN KEY (`idn`)  
        REFERENCES `shuttershop3`.`transaction` (`idn`)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = latin1;
```

# Commandes SQL de création (version 2')

```
• -----  
• -- Table `shuttershop3`.`influences`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`influences` (  
•     `auth1_dob` DATE NOT NULL,  
•     `auth1_name` VARCHAR(255) NOT NULL,  
•     `auth2_dob` DATE NOT NULL,  
•     `auth2_name` VARCHAR(255) NOT NULL,  
•     PRIMARY KEY (`auth1_dob`, `auth1_name`, `auth2_dob`, `auth2_name`),  
•     INDEX `Influences_FKIndex1` (`auth1_name` ASC, `auth1_dob` ASC),  
•     INDEX `Influences_FKIndex2` (`auth2_name` ASC, `auth2_dob` ASC),  
•     CONSTRAINT `influences_ibfk_1`  
•         FOREIGN KEY (`auth1_name` , `auth1_dob`)  
•             REFERENCES `shuttershop3`.`author` (`name` , `dob`)  
•             ON DELETE NO ACTION  
•             ON UPDATE NO ACTION,  
•     CONSTRAINT `influences_ibfk_2`  
•         FOREIGN KEY (`auth2_name` , `auth2_dob`)  
•             REFERENCES `shuttershop3`.`author` (`name` , `dob`)  
•             ON DELETE NO ACTION  
•             ON UPDATE NO ACTION)  
•     ENGINE = InnoDB  
•     DEFAULT CHARACTER SET = latin1;  
  
• -----  
• -- Table `shuttershop3`.`landscape`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`landscape` (  
•     `catalogn` INT(10) UNSIGNED NOT NULL,  
•     PRIMARY KEY (`catalogn`),  
•     INDEX `Landscape_FKIndex1` (`catalogn` ASC),  
•     CONSTRAINT `landscape_ibfk_1`  
•         FOREIGN KEY (`catalogn`)  
•             REFERENCES `shuttershop3`.`photo` (`catalogn`)  
•             ON DELETE NO ACTION  
•             ON UPDATE NO ACTION)  
•     ENGINE = InnoDB  
•     DEFAULT CHARACTER SET = latin1;
```

# Commandes SQL de création (version 2')

- -----
- -- Table `shuttershop3`.`location`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`location` (  
•   `place` VARCHAR(255) NOT NULL,  
•   `country` VARCHAR(255) NOT NULL,  
•   `description` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY (`place`, `country`))  
• ENGINE = InnoDB  
• DEFAULT CHARACTER SET = latin1;
- -----
- -- Table `shuttershop3`.`locatedin`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`locatedin` (  
•   `catalogn` INT(10) UNSIGNED NOT NULL,  
•   `country` VARCHAR(255) NOT NULL,  
•   `place` VARCHAR(255) NOT NULL,  
•   PRIMARY KEY (`catalogn`),  
•   INDEX `LocatedIn\_FKIndex1` (`catalogn` ASC),  
•   INDEX `LocatedIn\_FKIndex2` (`place` ASC, `country` ASC),  
•   INDEX `lloc` (`place` ASC, `country` ASC),  
•   CONSTRAINT `locatedin\_ibfk\_1`  
•       FOREIGN KEY (`catalogn`)  
•       REFERENCES `shuttershop3`.`landscape` (`catalogn`)  
•       ON DELETE NO ACTION  
•       ON UPDATE NO ACTION,  
•   CONSTRAINT `locatedin\_ibfk\_2`  
•       FOREIGN KEY (`place`, `country`)  
•       REFERENCES `shuttershop3`.`location` (`place`, `country`)  
•       ON DELETE NO ACTION  
•       ON UPDATE NO ACTION)  
• ENGINE = InnoDB  
• DEFAULT CHARACTER SET = latin1;

# Commandes SQL de création (version 2')

- -----  
• -- Table `shuttershop3`.`model`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`model` (  
•   `name` VARCHAR(255) NOT NULL,  
•   `dob` DATE NOT NULL,  
•   `sex` CHAR(1) NULL DEFAULT NULL,  
•   `nude` CHAR(1) NULL DEFAULT NULL,  
•   `bio` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY (`name`, `dob`))  
• ENGINE = InnoDB  
• DEFAULT CHARACTER SET = latin1;
  
- -----  
• -- Table `shuttershop3`.`portrait`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`portrait` (  
•   `catalogn` INT(10) UNSIGNED NOT NULL,  
•   PRIMARY KEY (`catalogn`),  
•   INDEX `Portrait\_FKIndex1`(`catalogn` ASC),  
•   CONSTRAINT `portrait\_ibfk\_1`  
•     FOREIGN KEY (`catalogn`)  
•       REFERENCES `shuttershop3`.`photo`(`catalogn`)  
•       ON DELETE NO ACTION  
•       ON UPDATE NO ACTION)  
• ENGINE = InnoDB  
• DEFAULT CHARACTER SET = latin1;

# Commandes SQL de création (version 2')

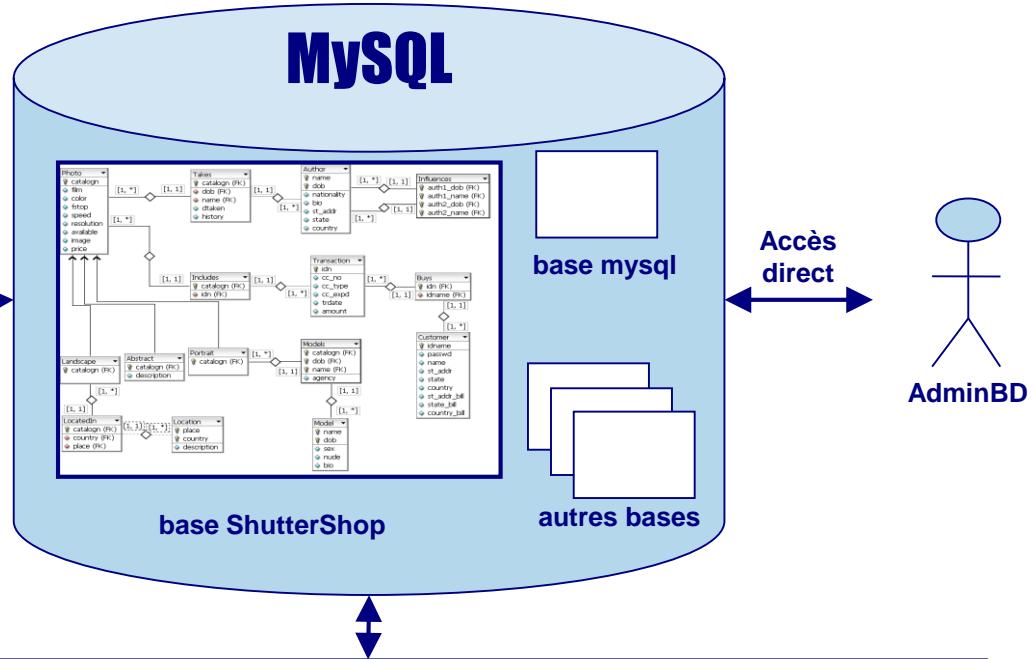
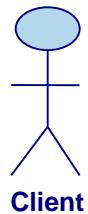
```
• -----  
• -- Table `shuttershop3`.`models`  
• -----  
• CREATE TABLE IF NOT EXISTS `shuttershop3`.`models` (  
•     `catalogn` INT(10) UNSIGNED NOT NULL,  
•     `dob` DATE NOT NULL,  
•     `name` VARCHAR(255) NOT NULL,  
•     `agency` VARCHAR(255) NULL DEFAULT NULL,  
•     PRIMARY KEY (`catalogn`, `dob`, `name`),  
•     INDEX `Portrait_has_Model_FKIndex1` (`catalogn` ASC),  
•     INDEX `Portrait_has_Model_FKIndex2` (`name` ASC, `dob` ASC),  
•     CONSTRAINT `models_ibfk_1`  
•         FOREIGN KEY (`catalogn`)  
•             REFERENCES `shuttershop3`.`portrait` (`catalogn`)  
•             ON DELETE NO ACTION  
•             ON UPDATE NO ACTION,  
•     CONSTRAINT `models_ibfk_2`  
•         FOREIGN KEY (`name`, `dob`)  
•             REFERENCES `shuttershop3`.`model` (`name`, `dob`)  
•             ON DELETE NO ACTION  
•             ON UPDATE NO ACTION)  
•     ENGINE = InnoDB  
•     DEFAULT CHARACTER SET = latin1;
```

# Commandes SQL de création (version 2')

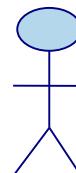
- -----
- -- Table `shuttershop3`.`takes`
- -----
- CREATE TABLE IF NOT EXISTS `shuttershop3`.`takes` (  
•   `catalogn` INT(10) UNSIGNED NOT NULL,  
•   `dob` DATE NOT NULL,  
•   `name` VARCHAR(255) NOT NULL,  
•   `dtaken` DATE NULL DEFAULT NULL,  
•   `history` VARCHAR(255) NULL DEFAULT NULL,  
•   PRIMARY KEY (`catalogn`),  
•   INDEX `Takes\_FKIndex1` (`catalogn` ASC),  
•   INDEX `Takes\_FKIndex2` (`name` ASC, `dob` ASC),  
•   INDEX `Itakes` (`name` ASC, `dob` ASC),  
•   CONSTRAINT `takes\_ibfk\_1`  
•     FOREIGN KEY (`catalogn`)  
•       REFERENCES `shuttershop3`.`photo` (`catalogn`)  
•       ON DELETE NO ACTION  
•       ON UPDATE NO ACTION,  
•   CONSTRAINT `takes\_ibfk\_2`  
•     FOREIGN KEY (`name` , `dob`)  
•       REFERENCES `shuttershop3`.`author` (`name` , `dob`)  
•       ON DELETE NO ACTION  
•       ON UPDATE NO ACTION)  
•   ENGINE = InnoDB  
•   DEFAULT CHARACTER SET = latin1;
- SET SQL\_MODE=@OLD\_SQL\_MODE;  
• SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  
• SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

# Analyse : utilisateurs

**Site web avec interface de consultation et d'achat en ligne, « front office » (par ex. applet Java, pages php,...)**



**Programme de consultation et de modification, « back office » (par ex. application Java, php,...)**



Gestionnaire du site

# Analyse : 3 utilisateurs

- Un administrateur BD:
  - Tous les droits
  - Toutes les bases
  - Toutes les tables
- PAS d'utilisateur anonyme
- Utilisateur front office « interface consultation et achat » :
  - Pour le client
  - Base ShutterShop uniquement
  - Consultation des tables : Photo, Portrait, Models, Model, Abstract, Takes, Author, Landscape, LocatedIn, Location
  - Consultation des tables Includes, Transaction, Buys, Customer limitée aux données concernant l'utilisateur courant
- Utilisateur back office « interface de gestion de la base ShutterShop » :
  - Pour le gestionnaire
  - Base ShutterShop uniquement
  - Toutes tables
  - Consultation, modification, ajout, suppression
  - A le droit de créer un utilisateur ayant tous les privilèges du gestionnaire sauf celui de créer d'autres gestionnaires : un gestionnaire principal qui crée des gestionnaires secondaires.

# Conception

- Ajouts de contraintes relatives aux données des tables
  - Définitions
  - Requêtes tests
- Ajouts de contraintes relatives au fonctionnement de la bd :
  - Achat d'une ou plusieurs photos : on veut changer la cardinalité N de Photo dans son association à Includes pour ne permettre qu'un seul achat d'une même photo (cardinalité max à 1). D'où, à chaque achat de photo :  
Après calcul du montant de la transaction, rendre la photo indisponible
  - A l'insertion d'un paysage, on veut s'assurer que la photo n'a pas déjà été insérée en tant que portrait.
  - ...
- Requêtes courantes → vues
  - Afficher dans une table les autoportraits avec les données techniques des photos et les biographies des auteurs.
  - ...

## Les applications

- interfaces utilisatrices des objets de la base.
- Ex. Front et back offices
- Une application ne doit pas pouvoir faire une manipulation non prévue dans la base :
  - Pas de consultation de données non prévue (par ex. les utilisateurs qui consultent les données d'autres utilisateurs).
  - Pas de modification, ajout, suppression non prévue.
- Permet une plus grande sécurité

# MySQL v.5.7 sous windows Workbench 6.3

## Installation, configuration

## Serveur

- Si possible dédier un serveur à MySQL
- Mieux vaut 1 serveur http « moyen » et un serveur MySQL « moyen » qu'un « gros » serveur hébergeant http et MySQL
- Stockage physique
  - Disques rapides
  - Disques en RAID
    - Réplication des données
    - Accès parallèles → plus rapide
  - Contrôleurs de disque avec beaucoup de cache et rapides
  - Voir baies de stockage
- Mémoire
  - Environ 1 ou 2 Mo de RAM par utilisateur
  - Prévoir de faire des index sur les colonnes souvent utilisées
  - 4 Go de RAM : suffit dans la majorité des cas (BD de sites web)

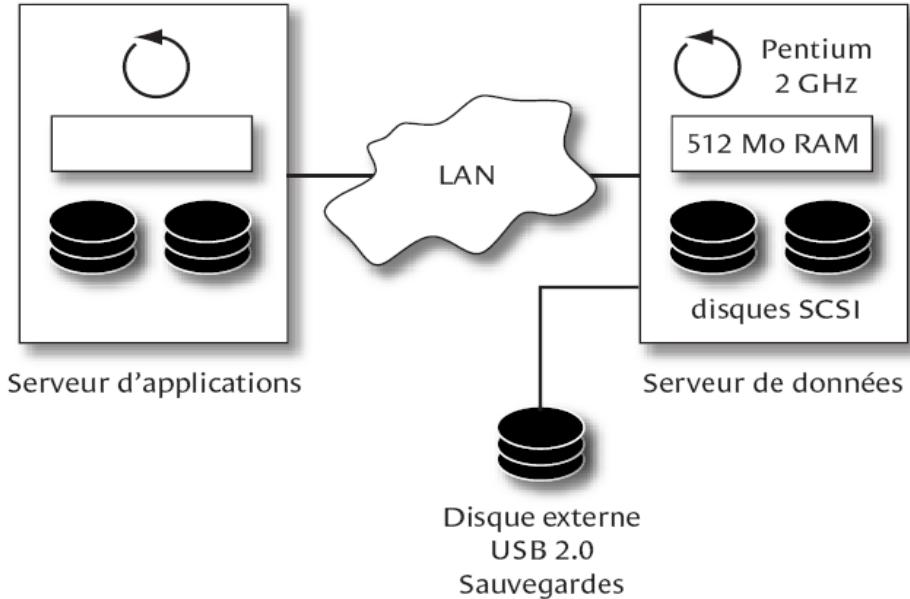
# Dimensionnement du serveur 2/2

- Processeur

- Pas de gros besoin en puissance de calcul processeur
- Dépend du type de requête (jointure, agrégats, regroupement)
- Pas besoin de cache important dans le processeur

- Sauvegardes

- Sur disques
- Sur bandes
  - Si on archive pendant longtemps et qu'on veut garder d'anciennes versions
  - Si on veut réutiliser le support de sauvegarde
- Disques USB :
  - Que quelques sauvegardes



**Exemple de dimensionnement pour une petite PME aux besoins informatiques restreints**



# MySQL v5.7 sous Windows

- Adresse : dev.mysql.com
  - Version : MySQL 5.7 Community Server (gratuit)
  - Système : Windows (de nombreux sont disponibles)
  - Configuration en bref (via le Configuration Wizard) :
    - Dimensionnement : « Developper machine »
    - Gestion des transactions : « Transactional Database only »
    - Répertoire d'installation
    - Nombre de connexions simultanées : 15
    - Paramètres réseau par défaut
    - « Strict mode » validé
    - Jeu de caractères standard
    - Installer Mysql comme un service démarré automatiquement
- Utilisateurs : Changer le mot de passe root, ne pas permettre l'accès root distant et ne pas créer d'utilisateur anonyme

# MySQL v5.7 sous Windows

- Installation en bref (avec la version portable compressée) :
  - Téléchargez le fichier zip mysql-5.7.14-win32.zip.
  - Décompressez ce fichier par exemple dans C:\mysql.
  - Ouvrez une fenêtre de commandes DOS (taper « cmd » dans le menu principal de windows)
  - Ajouter à la variable système PATH le chemin contenant les fichiers exécutables de MySQL avec la commande :  
`set PATH=%PATH%;"C:\mysql\bin"`
  - Avant toute première utilisation, initialisez le serveur pour qu'il crée la structure de base des données avec la commande :  
`mysqld --initialize --console`
  - Notez le mot de passe root qui est généré aléatoirement.
  - Lancez le serveur MySQL en tâche de fond avec la commande :  
`mysqld.exe --console`
  - Dans une autre fenêtre DOS, changez le mot de passe avec :  
`mysqladmin --u root --p password`
  - Tout est prêt, vous pouvez commencer à utiliser MySQL.

# Fichier de configuration

- Pour fixer précisément les valeurs des variables systèmes utiles au fonctionnement du serveur
- Fichier texte « my.ini »
- Crée à l'installation ou d'après le fichier « my-default.ini »
- Dans le répertoire d'installation
- Toute indication du fichier de configuration sera ignorée si une indication du même type est donnée en ligne de commande.
- Exemple :  
« password = toto » dans le fichier my.ini ignoré si le client se connecte avec « mysql --u truc --p secret »

# Fichier de configuration

- Le fichier de configuration « my.ini » a la forme suivante :
- La rubrique [mysqld] (ou [server]) contient les variables et options destinées au serveur.
- Dans [client] : éléments communs à tous les exécutables et utilitaires clients MySQL
- Possible de paramétrer un programme client (mysql, mysqldump, etc.) de manière individuelle en spécifiant son nom entre crochets.

<http://dev.mysql.com/doc/refman/5.7/en/server-options.html>

```
[mysqld]
param1=valeur1
param2=valeur2
param3=valeur3
...
```

```
[client]
param4=valeur4
param5=valeur5
...
```

```
[mysql]
param6=valeur6
...
```

```
[mysqldump]
param7=valeur7
...
```

```
[ . . . ]
```

# Fichier de configuration - exemple

```
[client]
port=3306

[mysql]
default-character-set=latin1

[mysqld]
port=3306
basedir="C:/MySQL/"
datadir="C:/MySQL/Data/"
default-character-set=latin1
default-storage-engine=INNODB
sql-mode="STRICT_TRANS_TABLES,
           NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"
max_connections=100
query_cache_size=8M
table_cache=256
tmp_table_size=5M
thread_cache_size=8

***** INNODB Specific options *****
#skip-innodb
innodb_additional_mem_pool_size=2M
innodb_flush_log_at_trx_commit=1
innodb_log_buffer_size=1M
innodb_buffer_pool_size=8M
innodb_log_file_size=10M
innodb_thread_concurrency=8
```

# Modes SQL du serveur

- Dans le fichier my.ini
- Mode SQL : ensemble de paramètres généraux affectant le fonctionnement global du serveur MySQL, ou seulement celui d'une session.
- Exemple
  - `sql-mode="STRICT_TRANS_TABLES,  
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"`
  - `STRICT_TRANS_TABLES` = insertion de données reconnues par le serveur
  - `NO_AUTO_CREATE_USER` = pas de création automatique de nouveaux utilisateurs
  - `NO_ENGINE_SUBSTITUTION` = contrôle du moteur utilisé à la création d'une table
- **[http://dev.mysql.com/doc/refman/5.7/en/  
sql-mode.html](http://dev.mysql.com/doc/refman/5.7/en/sql-mode.html)**

# Avec le workbench

On peut modifier facilement le fichier my.ini

- Voir l'icône



Options File

- Toutes les options sont disponibles et commentées !

Root Mysql

## Options File

Locate option:

	Value	Description
<input type="checkbox"/> default_tmp_storage_engine	InnoDB	The default storage engine (table type) for TEMPORARY tables
<input type="checkbox"/> init-file	<input type="text"/> ...	Read SQL statements from this file at startup
<input type="checkbox"/> lock_wait_timeout	31536000	Timeout for metadata locks
<input type="checkbox"/> old		Cause the server to revert to certain behaviors present in older versions
<input type="checkbox"/> session_track_gtids	OFF	Enables a tracker which can be configured to track different GTIDs.
<input checked="" type="checkbox"/> session_track_schema		Whether to track schema changes
<input type="checkbox"/> session_track_state_change		Whether to track session state changes
<input type="checkbox"/> session_track_system_variables	time_zone, autocommit, character_set_client, character_set_result	Session variables to track changes for
<input type="checkbox"/> stored_program_cache	256	Sets a "soft" upper limit for number of cached stored routines per connection; stored functions are cached separately; this variable sets size for both of them
<b>Firewall</b>		
<input type="checkbox"/> mysql_firewall_max_query_size	4096	Maximum size of recorded statements

# Variables système

- Variables dont la valeur sert au fonctionnement de MySQL
  - Exemples : taille de certains caches mémoires, noms des répertoires importants,...
  - Fixées
    - Par défaut
    - Au lancement de mysqld sur la même ligne de commandes
    - Dans le fichier my.ini
  - Leur valeur peut être
    - Globale (toujours la même pour tout le monde)
    - Liée à une session (peut varier entre plusieurs utilisateurs)
  - Peuvent souvent être modifiées à chaud (durant l'exécution du serveur) par la commande SET
  - La modification à chaud nécessite le privilège SUPER.

# MySQL v.5.7 sous windows Workbench 6.3

## Quelques éléments d'administration

# Administration MySQL

- Quelques outils
- Structure générale des données d'un serveur MySQL
- Gestion des utilisateurs
- Surveillance régulière du serveur
- Définition d'une politique de sécurité
- Sauvegardes des données
- Restaurations des données

# **MySQL v.5.7 sous windows**

## **Workbench 6.3**

**Outils et strucure de**  
**MySQL**

# Outils MySQL : « MySQL programs »

## **mysqld.exe**

le serveur MySQL qui tourne en tâche de fond

- **mysql.exe**

client par défaut en ligne de commandes DOS.

Pratique pour des commandes ponctuelles ou bien dans des scripts système.

- **mysqladmin.exe**

pour les tâches d'administration (création, suppression de BD, rechargement des droits des utilisateurs, sauvegarde des tables sur disque, ouverture de fichiers log, récupération d'infos diverses sur le serveur et ses paramètres).

- **mysqlcheck.exe**

pour la maintenance et l'optimisation des tables

# Outils MySQL : « MySQL programs »

## **mysqlimport.exe**

pour l'importation de données dans des tables à partir de fichiers texte avec la commande « LOAD DATA INFILE »

- **mysqlpump.exe et mysqldump.exe**  
sauvegardent des données dans un fichier texte .sql
- **mysqlshow.exe**  
affiche des informations diverses sur un serveur MySQL
- Autres programmes :  
<http://dev.mysql.com/doc/refman/5.7/en/programs-overview.html>

# Outils MySQL : mysqladmin

- Documentation : <http://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html>
- Exécution de mysqladmin :  
C:\mysql\bin>  
mysqladmin [options] command [command-arg] [command [command-arg]] ...
- Commandes de mysqladmin
  - create db\_name : Create a new database named db\_name.
  - debug : Tell the server to write debug information to the error log.
  - drop db\_name : Delete the database named db\_name and all its tables.
  - extended-status : Display the server status variables and their values.
  - flush-hosts : Flush all information in the host cache.
  - flush-logs [log\_type ...] : Flush all logs.
  - flush-privileges : Reload the grant tables (same as reload).
  - flush-status : Clear status variables.
  - flush-tables : Flush all tables.
  - flush-threads : Flush the thread cache.
  - kill id,id,... : Kill server threads.
  - password new\_password : Set a new password.
  - ping : Check whether the server is available.
  - processlist : Show a list of active server threads.
  - reload : Reload the grant tables.
  - refresh : Flush all tables and close and open log files (flush-tables followed by flush-logs)
  - shutdown : Stop the server.
  - start-slave : Start replication on a slave server.
  - status : Display a short server status message.
  - stop-slave : Stop replication on a slave server.
  - variables : Display the server system variables and their values.
  - version : Display version information from the server.

# Outils MySQL : exemples

Regarde si le serveur mysql fonctionne ou non :

```
mysqladmin -u root -p ping
```

Option « -u » avec comme valeur « root » pour dire sous quel utilisateur on veut se connecter pour exécuter la commande

Option « -p » sans valeur pour dire qu'on va entrer le mot de passe après avoir appuyé sur Entrée

Commande « ping » sans valeur pour dire qu'on veut faire un test de la connexion réseau avec le serveur MySQL

- Ferme la session de l'utilisateur d'identifiant de connexion 34

```
mysqladmin -u root -p kill 34
```

Option « -u » avec comme valeur « root » pour dire sous quel utilisateur on veut se connecter pour exécuter la commande

Option « -p » sans valeur pour dire qu'on va entrer le mot de passe après avoir appuyé sur Entrée

Commande « kill » avec la valeur « 34 » pour dire qu'on veut tuer la connexion de numéro 34 au serveur MySQL

# Outils MySQL : autres exemples

- Regarde si le serveur mysql fonctionne ou non :  
`mysqladmin -u root -p ping`
- Ferme le serveur mysql :  
`mysqladmin -u root -p shutdown`
- Ferme la session de l'utilisateur test  
`mysqladmin -u root -p processlist`  
`mysqladmin -u root -p kill 34`
- Crée la base de données essai  
`mysqladmin -u root -p create essai`
- Se connecte à mysql en mode texte :  
`mysql -u root -password='toto'`
- Exercices :
  - Je veux supprimer la base essai.
  - Je veux me connecter en tant qu'utilisateur 'test' avec le mot de passe 'test'.
  - Je veux examiner la structure de la base ShutterShop.

# Le répertoire des données

## « data directory »

- Contient toutes les données gérées par le serveur MySQL
- Contient des fichiers de log écrits par le serveur
- Sous-répertoires du data directory
  - Un par base de données (ex. shuttershop)
  - « mysql »  
contient les informations requises par le serveur pour fonctionner.
  - « performance\_schema »  
fournit les informations utiles pour surveiller l'exécution interne du serveur.
  - « sys »  
fournit un ensemble d'objets pour une meilleure interprétation des données de la BD performance\_schema
  - « ndbinfo »  
contient des informations spécifiques à MySQL Cluster
- information\_schema est une BD standard, mais qui n'a pas de sous-répertoire dans le data directory.

# La BD mysql

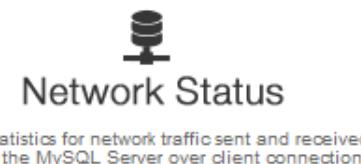
- Tables des données relatives aux droits globaux des utilisateurs
  - user
  - db
  - tables\_priv
  - columns\_priv
  - procs\_priv
  - proxies\_priv
- Tables décrivant les objets systèmes
  - event
  - func
  - plugin
  - Proc
- Tables contenant les logs du serveur
- Tables contenant le système d'aide côté serveur
- Tables contenant le mécanisme des zones temporelles
- Tables permettant la replication des données
- Tables permettant l'optimisation du système
- Tables diverses

# La BD mysql

Fonction	Description
<i>user</i>	Table stockant les utilisateurs, leur mot de passe, et les droits d'accès globaux.
<i>db</i>	Table stockant les droits d'accès d'un utilisateur sur une base de données.
<i>host</i>	Table permettant de restreindre les droits en cas de connexion à partir d'un ordinateur particulier.
<i>func</i>	Les fonctions créées par un utilisateur.
<i>proc</i>	Les procédures stockées.
<i>tables_priv</i>	Table stockant les droits d'accès d'un utilisateur sur les tables d'une base de données.
<i>columns_priv</i>	Table stockant les droits d'accès d'un utilisateur sur les colonnes d'une table.
<i>procs_priv</i>	Table stockant les droits d'accès d'un utilisateur sur les procédures stockées.

# La BD performance\_schema

- Contient des données permettant la surveillance de bas niveau de tout ce qui se passe dans le serveur MySQL
- Récupération de ces données par des requêtes SELECT.
- Synthèse dans le Workbench :



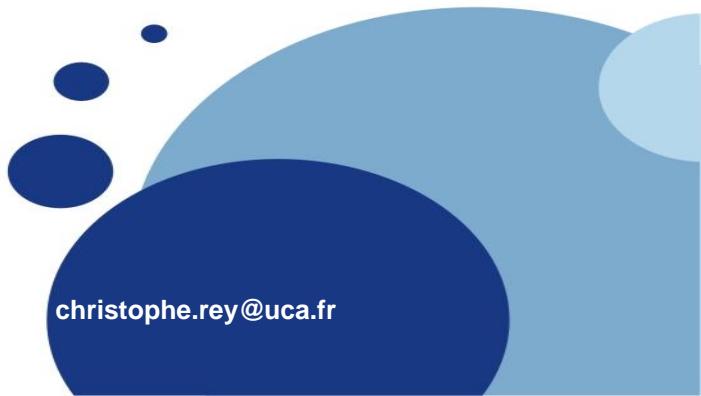
- → pratique pour détecter des problèmes (surcharge mémoire, processeur, réseau,...)

# La BD information\_schema

- Ensemble de tables en lecture seule
  - Requêtes SELECT
  - Pas de requête UPDATE, INSERT, DELETE
- Contient des données décrivant le serveur MySQL (des « meta data ») : c'est le dictionnaire de données
- Contient aussi les droits précis des utilisateurs par rapport a certaines BD, tables et attributs (cf la BD mysql)
- Exemples :
  - Le nom des BD
  - Le nom des tables d'une BD
  - Le type de données des colonnes
  - ...
- Aucun fichier physique associé dans le data directory

**MMI Vichy  
S3  
BD2**

# Gestion des utilisateurs



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

## Différents types d'utilisateurs

- Les utilisateurs du serveur MySQL (qui se connectent directement au serveur)
- Les utilisateurs d'une application liée au serveur
  - c'est l'application qui se connecte au serveur MySQL
  - c'est l'application qui est un utilisateur du serveur
- **Attribution des droits d'accès**
  - Création des utilisateurs (droit de se connecter au serveur seulement)
  - Attribution des droits d'accès sur les BD, tables, ...  
→ notion de « privilège »
- **Privilège = droit de faire une action particulière sur un élément particulier (BD, table, colonne)**

# Généralités

- Base mysql = gère connexions et privilèges
- Par défaut à l'installation du serveur
- Modifier cette base mysql est DANGEREUX !!!
- Autres vision des droits dans la base information\_schema
- DCL : Data Control Language
  - Méthode conseillée pour gérer les utilisateurs
  - CREATE USER : pour créer un utilisateur
  - RENAME USER : pour renommer un utilisateur
  - DROP USER : pour supprimer un utilisateur
  - SET PASSWORD : pour affecter ou modifier un mot de passe
  - GRANT : pour créer un utilisateur ou attribuer des privilèges
  - REVOKE : pour enlever des privilèges
- DML : Data Manipulation Language sur la base mysql
  - Très délicat (mais aussi plus puissant) : attention aux erreurs
  - INSERT
  - UPDATE
  - DELETE

# Création d'un utilisateur

• Chaque utilisateur a :

- Un compte utilisateur

- Un nom d'utilisateur (login)
  - Un hôte (machine à partir de laquelle il se connecte)
  - Exemple : 'login'@'host'

- Un password

• Création d'un compte (2 possibilités) :

- CREATE USER 'login'@'host' IDENTIFIED BY 'password';

- GRANT USAGE ON \*.\* TO 'login'@'host' IDENTIFIED BY 'password';

- Facilité :

- '' : chaîne vide pour un utilisateur anonyme  
==> éviter de créer des utilisateurs anonymes !!
  - % : remplace toute chaîne dans un nom d'hôte

- Mot de passe :

SET PASSWORD FOR 'login'@'host' = PASSWORD('truc');

• Connexion :

- La table user est consultée : elle contient tous les triplets (login, password, hôte).
- Si connexion réussie, une session de travail démarre. Elle prend fin à la déconnexion de l'utilisateur.

• Destruction et renommage :

- DROP USER 'login'@'localhost';
- RENAME USER moi@localhost TO moi@toto;

# Exemples de création d'un utilisateur

- CREATE USER 'martin'@'mt32.yahoo.com'  
IDENTIFIED BY 'Fr56t,Z';
- CREATE USER 'martin'@'%.yahoo.com'  
IDENTIFIED BY 'Fr56t,Z';
- CREATE USER toto  
IDENTIFIED BY 'rpm896ThA';
- CREATE USER toto@'%'  
IDENTIFIED BY 'rpm896ThA';
- GRANT USAGE ON \*.\* TO toto  
IDENTIFIED BY 'rpm896ThA';
- SET PASSWORD FOR 'toto' =  
PASSWORD('rpm896ThA');

# Règle de connexion

- Si le login et l'hôte peuvent correspondre à plusieurs comptes alors :
  - MySQL détermine d'abord l'hôte (puis le login)
  - MySQL privilégie l'information la plus spécifique avant la moins spécifique
    - Par ex. : localhost de préférence à %.
    - Par ex. : toto de préférence à " (le compte anonyme)
- Connaître son compte de connexion
  - `SELECT CURRENT_USER();`  
Donne le compte avec lequel l'utilisateur est effectivement connecté.
  - `SELECT USER();`  
Donne le compte avec lequel l'utilisateur a voulu se connecter.

# Exemple

## Question n°1:

Pierre se connecte en local à la base de donnée. Quel sera son mot de passe ?

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Exemple

## Question n°1:

Pierre se connecte en local à la base de donnée. Quel sera son mot de passe ?

Aucun mot de passe ne sera nécessaire : puisqu'il est en local, le compte auquel il se connecte est forcément le dernier (le compte anonyme).

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Exemple

## Question n°2:

Sonia se connecte depuis chez elle à la base de données (via le client mysql, IP : 192.164.0.34). Quels seront ses privilèges ?

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Exemple

## Question n°2:

Sonia se connecte depuis chez elle à la base de données (via le client mysql, IP : 192.164.0.34). Quels seront ses privilèges ?

Sonia se connecte au compte ayant '%' comme hôte. Aucun mot de passe ne lui sera demandé, et elle aura les privilèges Select et Update.

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Exemple

## Question n°3:

Pierre se connecte depuis l'adresse 192.168.12.9 en donnant son mot de passe 'toto'. Que se passe-t-il ?

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Exemple

## Question n°3:

Pierre se connecte depuis l'adresse 192.168.12.9 en donnant son mot de passe 'toto'. Que se passe-t-il ?

Pierre ne peut se connecter car le compte correspondant à sa connexion (le deuxième en partant du haut) ne requiert aucun mot de passe.

Utilisateur	Password	Droit
pierre@'%'	admin	All
pierre@192.168.%	"	Select,Insert
sonia@'%'	"	Select,Update
pierre@192.168.12.8	pierre	Select,Insert, Delete,Update
sonia@192.165.%	sandra	Select,Insert, Delete,Update
"@localhost	"	Select

# Attribution et niveau de privilèges

## Attribution de privilèges avec la commande

```
GRANT privilège[,privilège, ...] ON composant  
TO nom_utilisateur [IDENTIFIED BY mot_de_passe]  
[WITH GRANT OPTION]
```

### 4 niveaux de privilèges

- Global
  - Portée : toutes colonnes de toutes tables de toutes bases
  - EX. : GRANT SELECT ON \*.\* TO 'moi'@'%';
- Base
  - Portée : toutes colonnes de toutes tables de la base Auteur
  - EX. : GRANT SELECT ON Auteur.\* TO 'moi'@'%';
- Table
  - Portée : toutes colonnes de la table Romancier de la base Auteur
  - EX. : GRANT SELECT ON Auteur.Romancier TO 'moi'@'%';
- Colonne
  - Portée : colonnes nom et prénom de la table Romancier de la base Auteur
  - EX. : GRANT SELECT (nom, prénom) ON Auteur.Romancier TO 'moi'@'%';

# Catégories de privilèges



Droit	Description
USAGE	Droit de connexion
CREATE	Toutes les commandes de création <b>CREATE DATABASE, CREATE TABLE</b>
DROP	Destruction de bases ou de tables <b>DROP DATABASE, DROP TABLE , DROP VIEW</b>
GRANT OPTION	Permet à un utilisateur d'accorder à d'autres utilisateurs les droits qu'il possède sur des bases, des tables ou des procédures stockées.
REFERENCES	Permet de référencer une table, ou les tables d'une base
ALTER	Modification d'une table <b>ALTER DATABASE, ALTER TABLE</b>
DELETE	Destruction de lignes dans une table
INDEX	Création ou suppression d'index sur une table <b>CREATE INDEX, DROP INDEX</b>
SELECT	Recherche dans une table
UPDATE	Modification de lignes dans une table
CREATE VIEW	Création de vues (voir page 175)
SHOW VIEW	Inspection de la définition d'une vue <b>SHOW CREATE VIEW, SHOW CREATE TABLE</b>
CREATE TRIGGER	Création de triggers (voir page 385)
CREATE ROUTINE	Création de procédures stockées <b>CREATE PROCEDURE, CREATE FUNCTION</b>
ALTER ROUTINE	Modification ou destruction de procédures stockées <b>ALTER PROCEDURE, ALTER FUNCTION</b>
EXECUTE	Exécution de procédures stockées
ALL [PRIVILEGES]	Tous les privilèges
LOCK TABLES	Droit de verrouiller des tables

Droit	Description
FILE	Accès ou écriture de fichier sur le serveur <b>SELECT ... INTO FILE, LOAD DATA INFILE</b>
CREATE TEMPORARY TABLE	Création de tables temporaires
CREATE USER	Création d'un utilisateur avec droit de connexion + <b>RENAME USER, REVOKE ALL</b>
PROCESS	Consultation des processus (ou <i>threads</i> ) de MySQL <b>SHOW FULL PROCESSLIST</b>
RELOAD	Rechargement du serveur <b>FLUSH</b>
REPLICATION CLIENT	Droit lié au système de réplication (non présenté dans ce livre) <b>SHOW MASTER STATUS, SHOW SLAVE STATUS</b>
REPLICATION SLAVE	Droit lié au système de réplication (non présenté dans ce livre)
SHOW DATABASES	Droit de consulter les bases
SUPER	Droit d'interrompre des <i>threads</i>
SHUTDOWN	Droit d'arrêter le serveur <b>avec mysqladmin shutdown</b>

Commandes : kill,  
purge master logs, set  
global, mysqladmin  
debug,...

## Droits d'administration

← Droits d'accès aux objets  
(tables, colonnes,...)

# Délégation de privilèges

- Déléguer un privilège : l'accorder à un autre utilisateur.
- On a un privilège  $\Leftrightarrow$  un utilisateur qui a ce même privilège nous l'a délégué.
- On a un privilège sur un niveau → on a le même privilège sur tous les niveaux inférieurs.  
BD > table > colonne
- On ne peut déléguer un privilège que si :
  - On a soi-même le privilège que l'on veut déléguer
  - Et si on a le privilège de déléguer des privilèges.
- Syntaxe :
  - GRANT privilège ON niveau TO user@host;
  - Pour accorder tous les privilèges (moi@localhost devient un second super administrateur) :  
GRANT ALL ON \*.\* TO moi@localhost;
  - Pour accorder le droit de déléguer :  
GRANT USAGE ON ... TO ... WITH GRANT OPTION;

# Privilèges (fin)

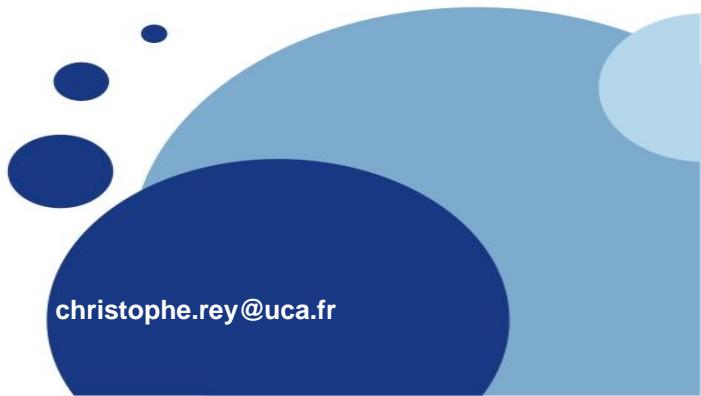
- Afficher les privilèges :
  - SHOW GRANTS FOR moi@localhost;
  - SHOW GRANTS; -- pour l'utilisateur en cours
- Révoquer les privilèges :
  - REVOKE DELETE ON Auteur.Romancier FROM moi@localhost ;
  - REVOKE GRANT OPTION ON Auteur.Romancier FROM moi@localhost ;
  - REVOKE ALL PRIVILEGES, GRANT OPTION FROM moi@localhost ;

# Remarques

- Droits sensibles :  
GRANT, RELOAD, SHUTDOWN, PROCESS et FILE  
→ root only !
- Droits inutiles pour les applications :  
CREATE, DROP, ALTER, INDEX
- CREATE et DROP :  
pour les tables ET les BD correspondantes
- Pour un utilisateur humain :  
SELECT, UPDATE, DELETE, INSERT, EXECUTE
  - ou bien
  - SELECT

**MMI Vichy  
S3  
BD2**

# **Politique de sécurité**



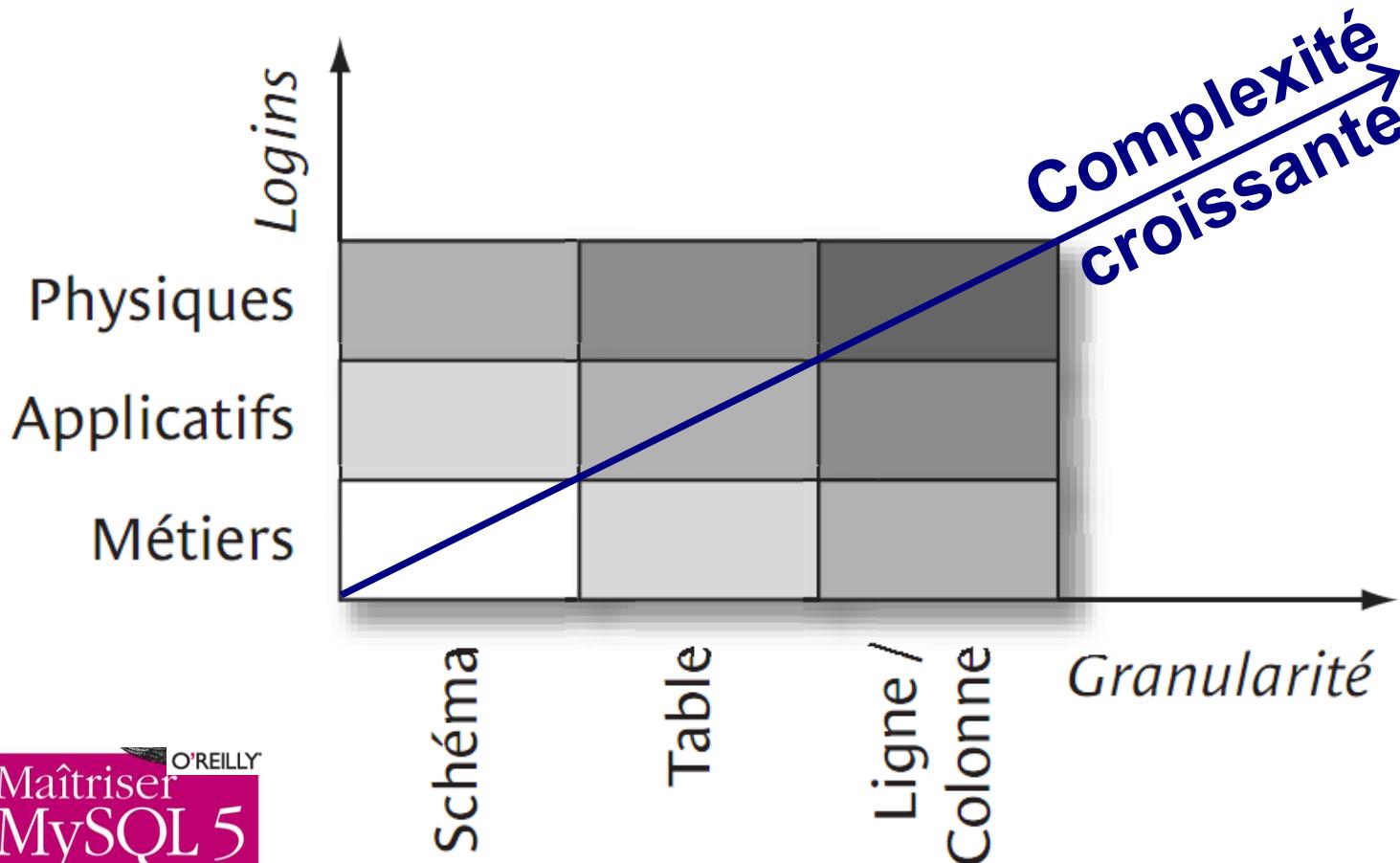
**christophe.rey@uca.fr**

# Définir une politique de sécurité

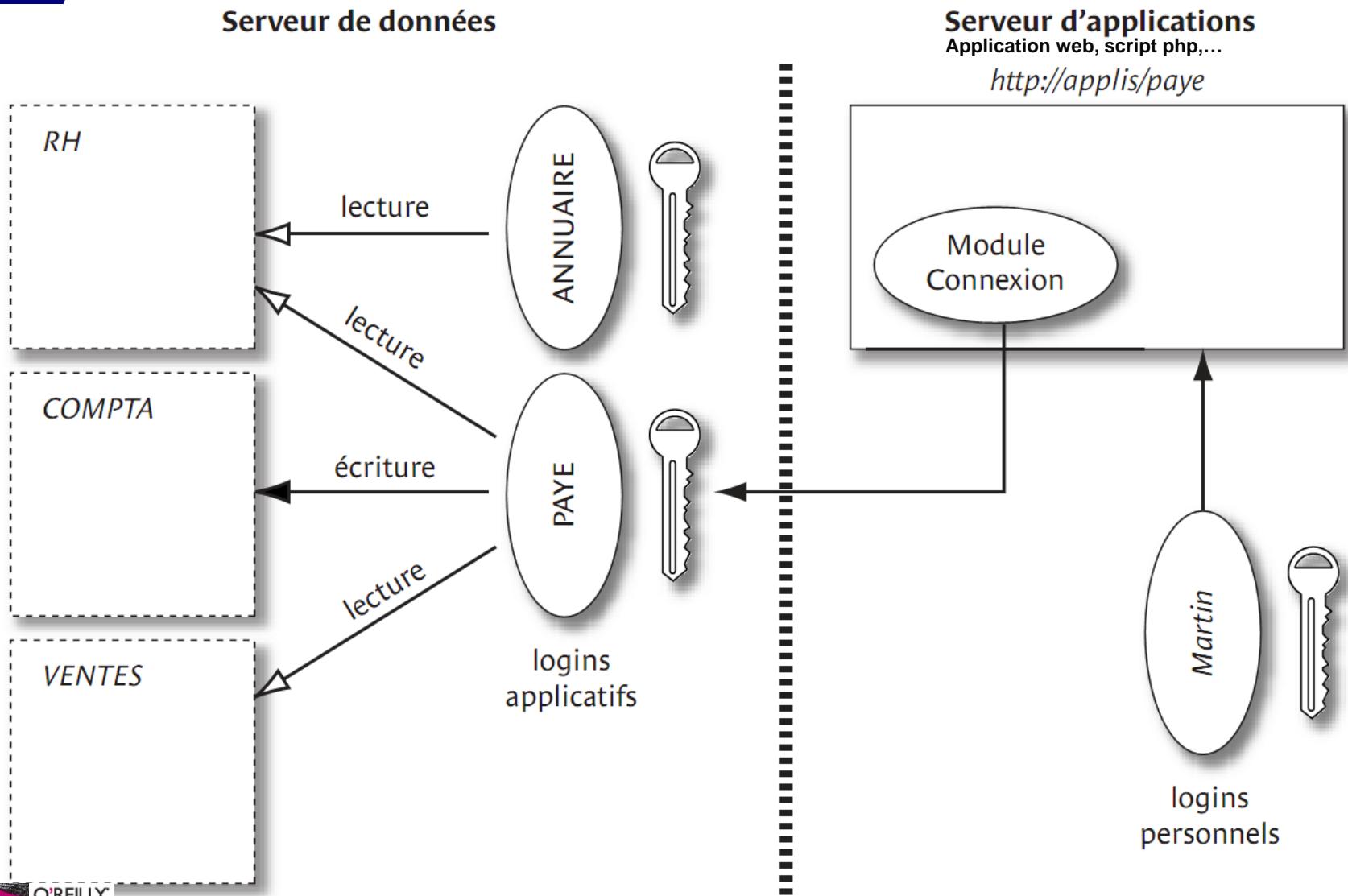
- Si possible, utiliser uniquement des utilisateurs à nom unique (c'est-à-dire avec '%' en hôte).
- Plus la politique de sécurité est précise (beaucoup de cas particuliers), plus elle est complexe à définir et à gérer.
- Définir un administrateur ne pouvant se connecter qu'en local (localhost) peut être très pratique.
- Notion de login / granularité

Logins « métiers »	Chaque métier dispose d'un accès partagé à la base : <i>RH, COMPTA, etc.</i>
Logins « applicatifs »	Chaque application dispose d'un accès à la base : <i>PAYE, ANNUAIRE, etc.</i> <small>Les personnes utilisatrices de l'application ne peuvent pas se connecter au serveur.</small>
Logins « physiques »	Chaque personne physique dispose de son propre accès à la base.
Granularité schéma	Les logins accèdent à des schémas entiers en lecture ou en écriture.
Granularité table	Les logins accèdent à des tables en lecture ou en écriture en fonction des besoins.
Granularité ligne / colonne	Les logins n'accèdent que certaines colonnes voire certaines lignes de tables.

# Définir une politique de sécurité



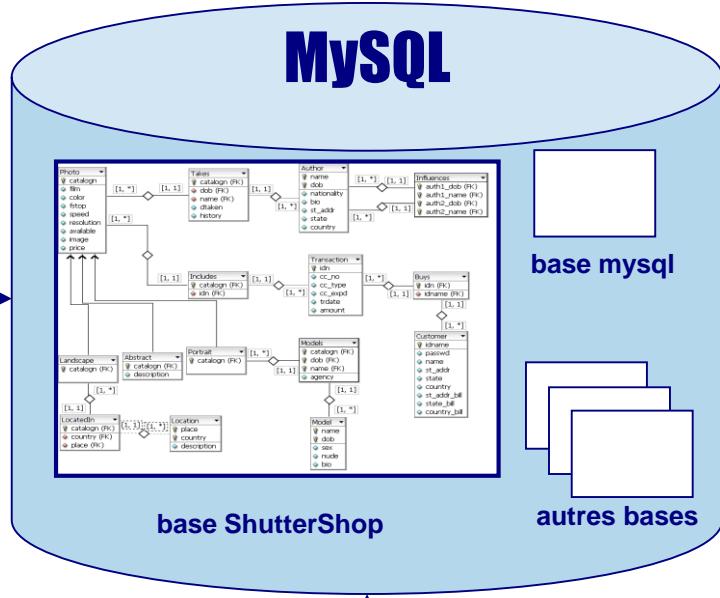
# Définir une politique de sécurité



# Etude de cas

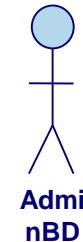


Quels priviléges ?  
Sur quels éléments ?



Uniquement en localhost ?

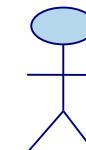
Accès direct



Quels priviléges ?  
Sur quels éléments ?

**Back office**

Quel type d'accès ?

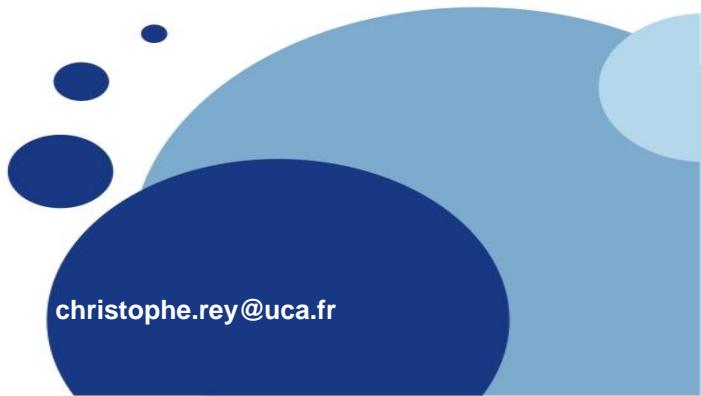


# Définir une politique de sécurité

- Buts de la sécurité :
  1. Se protéger des erreurs
  2. Se protéger de la malveillance
- La malveillance :  
gestion au niveau humain aussi !
- Sécurité totale
  - 2 aspects : politique + confiance
  - 2 niveaux : serveur de données + application  
→ sensibiliser les développeurs
- Politique trop complexe, des règles trop fines, trop strictes:
  - Grande difficulté de gestion
  - Des erreurs et incompréhensions à prévoir
  - Des failles qui apparaissent

**MMI Vichy  
S3  
BD2**

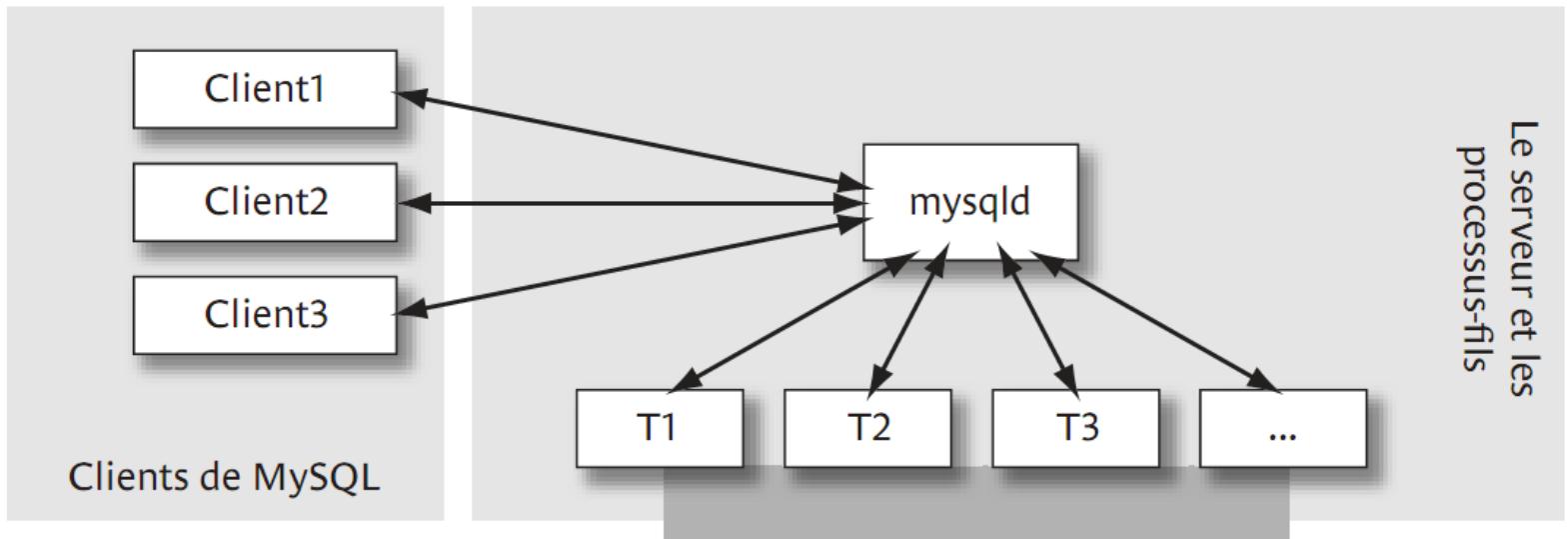
# **Elements d'administration**



**christophe.rey@uca.fr**

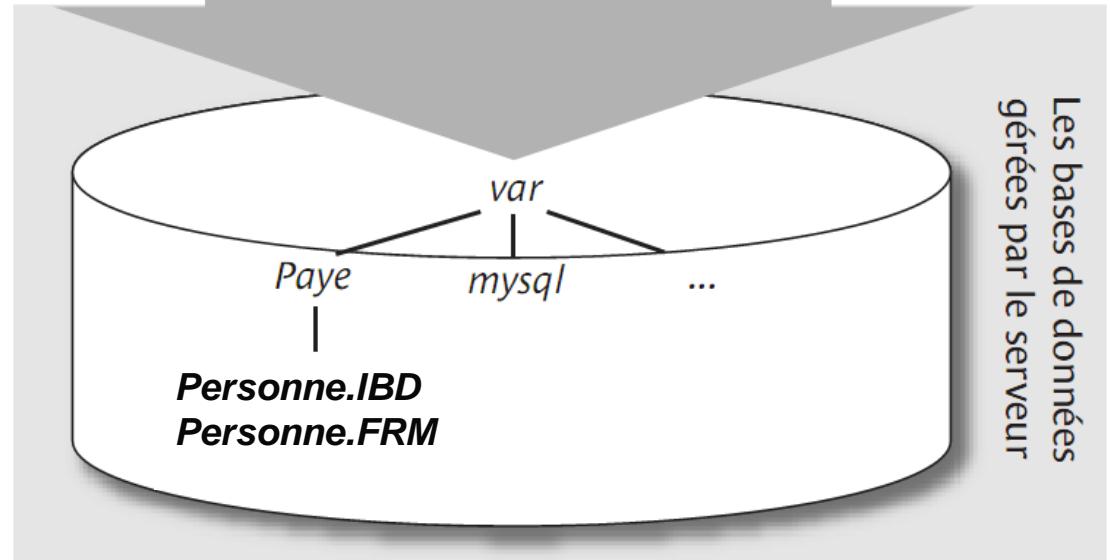
# Fonctionnement du serveur

Le serveur et les processus-fils

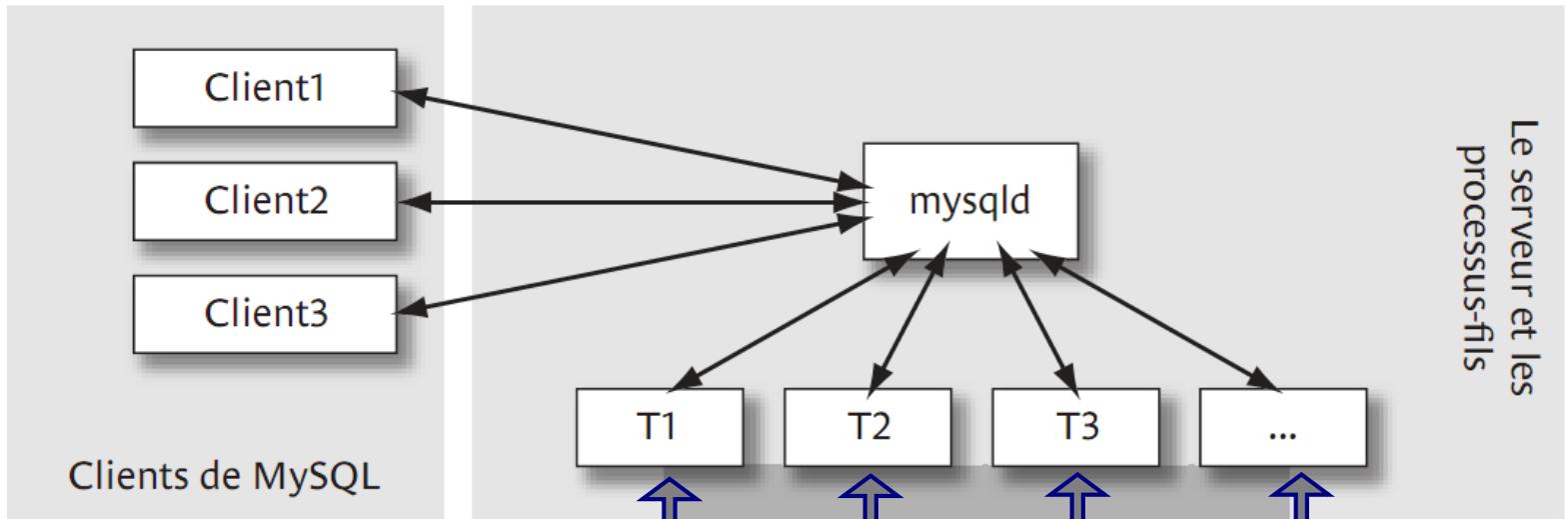


- Une connexion ≈ un processus-fils de mysqld, « thread »
- Si le moteur de traitement des requêtes est InnoDB : 2 fichiers par table  
IBD : contenu  
FRM : description

Les bases de données gérées par le serveur

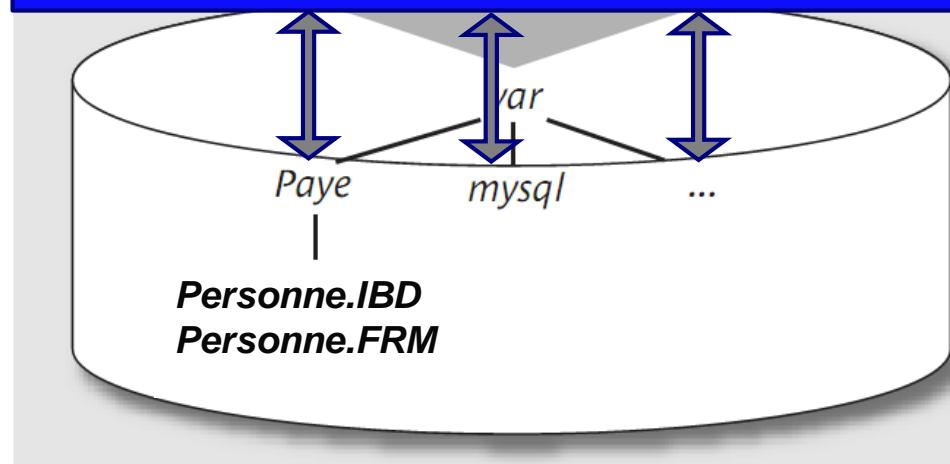


# Fonctionnement du serveur



- E/S : entrées / sorties : copie des données entre le disque et la mémoire
- Temps d'E/S : déterminants pour les performances

## Mémoire vive affectée au serveur



# Surveillance du serveur

- But : éviter les problèmes et pannes (ralentissement, arrêt du serveur, perte de données, modifications non voulues,...)
- A disposition :
  - Outils d'administration MySQL (mysqladmin,...)
  - MySQL Workbench !
  - Outils du système d'exploitation
    - Créer des scripts système exécutant régulièrement des commandes mysqladmin et les mettant en forme de manière claire.
    - Envoi automatique de mails à l'administrateur.
- Exemples de points à surveiller
  - Présence et contenu de fichiers .err dans le répertoire de MySQL  
➔ des erreurs sont peut-être survenues
  - Espace disponible sur le serveur : ne pas le laisser trop proche de 0
  - Consommation des ressources
    - Le CPU ne doit pas fonctionner à 100% sur de longues périodes.
    - La mémoire swap (sur disque) ne doit pas trop être utilisée.  
➔ sous-dimensionnement du serveur, requêtes problématiques,...

# Surveillance du serveur

## Surveillance des threads avec mysqladmin

- status : infos générales, avec le nombre de threads en cours en particulier
- processlist : list des threads en cours et infos associées
- kill : arrêter un thread d'après son numéro
- Synchronisation disques et mémoire
  - Technique du caching :
    - Exécution de modifications sur des données en mémoire
    - Mises à jour des données sur disques au bout d'un certain temps
  - Problème éventuel
    - Si on veut tout de suite prendre en compte des mises à jour
    - Si panne avant écriture sur disque
  - Solutions
    - Synchronisation forcée entre mémoire et disque par les commandes **flush** de mysqladmin.
    - Si panne, alors restauration par utilisation des journaux (logs) de transaction et/ou de mise à jour (cf après).  
« Transaction » = opération effectuée sur le serveur MySQL (typiquement un INSERT, UPDATE ou DELETE sur une table).

# Agenda de l'administrateur

- Tâches précédentes à faire régulièrement en présentiel
  - Début de matinée
  - Début d'après-midi
  - Fin de journée
- Tâches à planifier
  - Envoi automatique de mails dans la nuit par exemple
  - Opérations de vérification des tables aux heures creuses  
→ scripts DOS

## Vérification de toutes les tables de toutes les BD

```
select concat('USE ', ' ', RTRIM(table_schema), ';' ),
           'CHECK TABLE', concat(table_schema, '.', table_name),
'MEDIUM', ';'
from information_schema.tables;
```

```
C:\...>mysql --skip-column-names -u root -proot < GenCheck.sql
> Check.sql
```

```
...
USE shuttershop; CHECK TABLE      shuttershop.abstract      MEDIUM ;
USE shuttershop; CHECK TABLE      shuttershop.author       MEDIUM ;
USE shuttershop; CHECK TABLE      shuttershop.buys        MEDIUM ;
USE shuttershop; CHECK TABLE      shuttershop.customer    MEDIUM ;
...
```

```
C:\... >mysql --skip-column-names -u root -proot < Check.sql >
CheckRes.txt
```

- CheckRes  
.txt

```
...
shuttershop.abstract   check   status  OK
shuttershop.author     check   status  OK
shuttershop.buys       check   status  OK
shuttershop.customer   check   status  OK
shuttershop.includes   check   status  OK
...
```

**MMI Vichy  
S3  
BD2**

# Sauvegardes / restaurations de données



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

## Pertes de données

- Pannes
- Erreurs humaines

Exemple :

```
mysql> DELETE FROM Paye; WHERE id = 67;
Query OK, 0 rows affected (0.02 sec)

ERROR 1064: You have an error in your SQL syntax
          near 'WHERE id = 0' at line 1
mysql> select * from Paye;
Empty set (0.00 sec)
```



➔ la table Paye a été effacée !!

- Exécution défectueuse d'un programme ➔ corruption de données
- Sauvegarde
  - Ouverte / Fermée : pendant que le serveur fonctionne / ou non
  - Format binaire / texte : copie des fichiers de données et des logs / copie des instructions SQL de création des tables et d'insertion des données (avec mysqlpump ou mysqldump)
  - Totale / incrémentale / partielle

# Politique de sauvegarde

## Critères :

- Disponibilité du serveur : sauvegarde ouverte ou fermée
- Format :
  - Si petites bases et restauration non critique → texte
  - Sinon → binaire + logs
- Tolérance de perte → fréquence de sauvegarde
- Temps de restauration → rapidité du média de sauvegarde
- Volume des sauvegardes → coût des media de stockage
  - Bases volumineuses → sauvegardes incrémentales (avec logs) ou partielles
  - Sinon → sauvegarde totale
- Exemple : sauvegarde fermée incrémentale binaire
  - Sauvegarde complète une fois par semaine sur bande, et sauvegarde du journal de mises à jours (logs) une fois par jour sur un disque lui-même monté en RAID avec un autre disque.
  - Adaptée à
    - Une entreprise qui éteint ses serveurs la nuit.
    - Volumes raisonnables (< 200Go)
    - → temps de restauration acceptables (au max 24h de m̄aj à faire)
    - Cas extrême : perte d'une journée si perte du journal des mises à jour
- → protéger aussi le journal des mises à jour !

# Sauvegarde fermée incrémentale binaire

- Première étape : sauvegarder les données en sauvegarde fermée
- Mode opératoire de la sauvegarde des données en mode fermé
  - Arrêter le serveur
  - Copier les fichiers de données sur un media de sauvegarde (disque)
  - Copier les fichiers logs (journaux des modifications ayant eu lieu sur le serveur) sur un autre disque (en RAID avec un troisième disque)
  - Redémarrer le serveur
  - Archivage de la sauvegarde disque sur bande
- Possibilité de réaliser un script à exécution régulière
- Toujours tester la procédure !
- Exemple :
  - Script save.bat

```
echo Sauvegarde du
%DATE:~6,4%-%DATE:~3,2%-%DATE:~,2% %TIME:~0,8%
mysqladmin shutdown -u root --password=...
xcopy /E /Y c:\mysql c:\temp\
"C:\mysql\bin\mysqld" --defaults-file="C:\MySQL\my.ini" &
```

- Planification par le planificateur de tâches : save.bat > save.txt  
➔ LIRE save.txt régulièrement (par ex. tous les matins) !

# Sauvegarde incrémentale binaire

- Journaux des mises à jour (logs) : fichiers qui enregistrent toute mise à jour effectuée sur la base en format binaire (et pas en texte SQL)
- Activé par l'option **log-bin** dans le fichier de configuration (dans la section du serveur)  
Par exemple :  
`log-bin='d:/SauveMySQL/UNIV-018-bin'`  
indique que
  - Le fichier UNIV-018-bin.index contient tous les noms des fichiers journaux :
    - UNIV-018-bin.index.000001
    - UNIV-018-bin.index.000002
    - ...
  - Les fichiers journaux sont stockés sur d:\SauveMySQL\
- Affichage des logs en SQL (et pas en binaire) : outil mysqlbinlog  
`mysqlbinlog UNIV-018-bin.index.000001`
- Exécution des logs :  
`mysqlbinlog UNIV-018-bin.index.000001 | mysql -u root -password=...`

# Sauvegarde incrémentale binaire

## Fichiers du journal des logs: petits fichiers

- Faciles à sauvegarder
- Sauvegardables plusieurs fois par jour
- Procédure de sauvegarde incrémentale (ouverte ou fermée) :
  - Les fichiers de données ont été copiés il y a quelques temps (en sauvegarde fermée).
  - On met à jour les journaux par :  
`mysqladmin -u ... --password=... flush-logs`  
Le dernier fichier de log est fermé et un autre est tout de suite réouvert pour les prochaines modifications.
  - On copie les fichiers log (sauf le dernier qui vient d'être ouvert) sur le media de sauvegarde.
  - Dédier un disque pour les fichiers log est une bonne idée.

# Sauvegardes au format texte

## Inconvénients

- Bien plus longues à réaliser
- Données pas toutes sauvegardées exactement au même moment → perte de cohérence possible

## Avantages

- Simples à réaliser
- Bonne résistance aux corruptions de données
- Permettent de charger les données dans d'autres bases, voire d'autres SGBD

## Cas où on peut les envisager

- Faible volume de données (quelques centaines de Mo)
- Tolérance aux pertes de données

## Outil : mysqldump ou mysqlpump

# Mysqldump - options



Fonction	Description
<code>-A</code>	Sauvegarde toutes les bases du serveur.
<code>-t</code>	Sauvegarde des lignes, mais pas des commandes de création de table.
<code>-T répertoire</code>	Écrit dans <i>répertoire</i> deux fichiers pour chaque table. Le fichier <i>table.sql</i> contient la commande de création, et le fichier <i>table.txt</i> les lignes.
<code>-c</code>	Produit des ordres <code>INSERT</code> complets, avec la liste des attributs.
<code>-x</code>	Verrouille les tables avant sauvegarde
<code>-l</code>	Verrouille les tables avant la sauvegarde pour éviter des mises à jour simultanées.
<code>-u, -p, -h</code>	Les options habituelles pour, respectivement, l'utilisateur, le mot de passe et le nom de l'hôte de <code>mysqld</code> .

## Données et structure

L'option `-T` est très pratique pour réutiliser séparément les données sauvegardées. Elle créera, pour chaque table de la base *Credit*, deux fichiers dans le répertoire `/tmp` :

- `nomTable.sql` contient l'instruction SQL qui recréera la table ;
- `nomTable.txt` contient les données qu'on pourra charger avec la commande `LOAD DATA` sous *mysql*.

### Données ou structure

- sauvegarde des ordres de création :

```
mysqldump --no-data -A > backupTextDML.sql
```

- sauvegarde des données seules :

```
mysqldump --no-create-info --all-databases > backupTextDML.sql
```

# Mysqldump usages

## Exemples

- mysqldump –u root --password=root –x –A > save.sql
- mysqldump –u root --password=root shuttershop -T d:/temp/mysqlsave/
- mysqldump -u root --password=root -t shuttershop > save.txt
- Utiliser mysqldump
  - en sauvegarde « pseudo fermée » de préférence (serveur arrêté, puis redémarré sans connexion réseau et en lecture seule).
  - sinon en sauvegarde ouverte (serveur en fonctionnement) mais avec l'option -x pour interdire tout accès autre qu'en lecture.  
→ risque de perturber le fonctionnement des applications accédant au serveur

# Mysqldump – assurer la cohérence

Pour assurer la cohérence avec mysqldump, on peut, avant la sauvegarde textuelle :

- couper les connexions réseau entre mysql et les applications : option `--skip-networking` de mysqld
- mettre les tables en lecture seule : option `--read-only` de mysqld
- verrouiller les tables (ni lecture ni écriture ne sont possibles) option `-x` de mysqldump
- Exemple de commandes DOS
  - `mysqladmin -u root -p... shutdown`
  - `mysqld --skip-networking`
  - `mysqldump -u root -p... -x -A > save.sql`
  - `mysqladmin -u root -p... shutdown`
  - `mysqld --console`

# Sauvegardes binaires ouvertes

- Utile quand contraintes fortes sur a BD
  - Disponibilité totale de la base (le serveur ne doit jamais s'arrêter)
  - Gros volumes de transactions
  - Exigence de restauration très rapide
  - Trop de données à sauvegarder pour le temps de sauvegarde qu'on peut se permettre
- Processus complexe : hors sujet pour ce cours

# Sauvegardes : conclusion

Exemple d'une politique de sauvegarde adaptée à de nombreux contextes :

- sauvegardes binaires fermées journalières, par copie de fichiers sur un disque
- Archivage de la sauvegarde journalière par copie sur bande
- Sauvegarde incrémentale des journaux toutes les heures
- Sauvegardes textes toutes les semaines pour palier le cas de corruption des fichiers binaires
- Possibilité de compresser les sauvegardes journalières

# Restaurations

- Les procédures de restauration doivent être :
  - Documentées
  - Testées
- 3 techniques
  - Restauration physique des fichiers binaires de la base
  - Appliquer les journaux de logs (mysqlbinlog)
  - Recréer les données et/ou tables à partir des sauvegardes texte
- 2 scenarios principaux
  - Défaillance complète du serveur de données
  - Défaillance partielle
    - Perte de fichiers
    - Modifications malencontreuses de données

# Défaillance complète

- Remonter un autre serveur (administrateur réseau)
- Réinstaller (la bonne version de) MySQL
- Reprendre la sauvegarde binaire complète la plus récente
- Reprendre les journaux juste postérieurs à cette sauvegarde
- Redémarrer le serveur (sans le réseau et en lecture seule)
- Appliquer les modifications présentes dans les journaux à partir de la date qui suit immédiatement celle de la sauvegarde
- → utiliser mysqlbinlog avec l'option  
--start-datetime

# Défaillance complète (suite)

- Appliquer les modifications présentes dans les journaux à partir de la date qui suit immédiatement celle de la sauvegarde :

```
mysqlbinlog –u root –p... UNIV-018-bin.index.00000*  
--start-datetime=... > ExecLogs.sql
```

```
mysql –u root –p... < ExecLogs.sql
```

- Arrêter le serveur
- Redémarrer le serveur en mode normal (avec réseau et plus en lecture seule)

# Défaillance complète – petites bases

- Sauvegardes textuelles journalières (avec mysqldump)
- Restauration
  - Réinstallation de MySQL
  - Exécution des commandes SQL sauvegardées  
`mysql –u root < sauvegardeTxt.sql`
- ➔ Potentiellement perte de données

# Défaillance partielle

- En cas de
  - Perte de fichiers
  - Modifications malencontreuses
- Si beaucoup de données perdues → restauration complète
- Sinon restauration des données perdues seulement
  - Avantage : permet une plus grande disponibilité du serveur
  - Inconvénients : plus compliqué, plus stressant

# Perte de fichiers

## Restauration de fichiers binaires en 3 étapes

- Identification des fichiers perdus
- Restauration (copie) des fichiers sauvegardés correspondants
  - Si tables MyISAM, alors il faut récupérer 3 fichiers par table
  - Si tables InnoDB
    - Si option `innodb_file_per_table` activée, alors récupérer 2 fichiers par table
    - Sinon, impossible de faire une récupération partielle
- Application des journaux sur ces tables
  - ➔ étape difficile : extraire des journaux les logs ne concernant que les tables perdues
    - Travail sur la sortie de `mysqlbinlog`
    - Réalisation de scripts complexes
    - Indispensable : tester ces scripts !

# Modifications malencontreuses

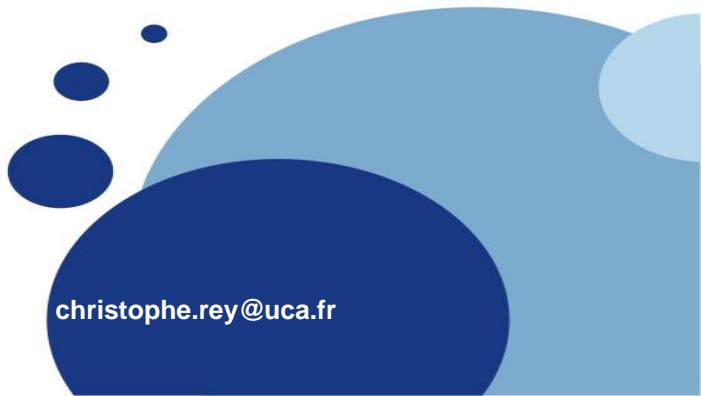
- Récupération partielle à partir d'une sauvegarde textuelle en 3 étapes
  - Suppression (temporaire) des contraintes d'intégrité référentielle pointant vers la table à récupérer
  - Récupérer et exécuter, dans la sauvegarde textuelle qui précède l'erreur, la partie concernant la table
  - Récupérer dans les journaux les modifications postérieures à la sauvegarde et ne concernant que la table en cours de restauration
    - ➔ scripts assez complexes
  - Restaurer les contraintes d'intégrité
    - ➔ résultat pas toujours cohérent

## Organisation

- Conserver tous les scripts (sauvegarde / restauration)
- Commenter les scripts et laisser une documentation sur leur utilisation
- Classer les scripts dans une arborescence bien définie
- Procédures de sauvegarde et de restauration
  - Les tester régulièrement → réserver un disque pour les tests
  - Si la base de données change, toujours mettre à jour les procédures.

**MMI Vichy  
S3  
BD2**

# Compléments SQL et MySQL



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

- **Les types de données de MySQL**
- **Les opérateurs et les fonctions de MySQL**
- **L'agrégation et le regroupement**
- **Sous requêtes**
- **Mise à jour et ajout des données**
  - Modification simple
  - Modifications combinées
  - Modifications multitable
  - Intégrité des données et mode strict
- **Les vues**
  - Définition et utilité
  - Création et gestion des vues
  - Modification des données à travers les vues

# Catégories d'instructions SQL

## DDL : Data Definition Language

- Pour créer les bd
- CREATE TABLE, ALTER TABLE, DESC, USE, ...

## DML : Data Manipulation Language

- Pour interroger et modifier les données
- SELECT, INSERT INTO, ...

## DCL : Data Control Language

- Pour gérer les utilisateurs et leurs privilèges
- CREATE USER, GRANT, REVOKE, ...

## TCL : Transaction Control Language

- Pour gérer les accès concurrents aux données
- START TRANSACTION, SET ISOLATION LEVEL, ...

## SQL procédural (embedded)

- Pour gérer les procédures et fonctions stockées
- CREATE PROCEDURE, WHILE ... DO, ...

# Les types de données de MySQL

# Les types numériques

Type	Argument	Description	Plage de valeurs	Options	Exemples	Taille en octets
Decimal	Précision (obli.), décimales (opt.)	Nombre en virgule fixe	Précision max de 65 chiffres significatifs et 30 après la virgule	Unsigned, Zerofill	Decimal(5,2) Unsigned	Dépend des arguments
Float		Nombre en virgule flottante, simple précision	Précis à environ 7 chiffres après la virgule		Float Float(6) Float(5,2)	4
Double		Nombre en virgule flottante, double précision	Précis à environ 15 chiffres après la virgule		Double(4) Unsigned	8
Tinyint	Nombre de chiffre à l'affichage (opt.)	Nombre entier	-128 à 127 ou 0 à 255 avec Unsigned	Unsigned, Zerofill		1
Smallint			-32768 à 32767 ou 0 à 65535 avec Unsigned		Smallint Unsigned	2
Medium int			-8 à +8 millions ou 0 à 16 millions			3
Int			-2 à +2 milliards ou 0 à 4 milliards		Int(6)	4
Bigint			-9.10^18 à +9.10^18 ou 0 à 18.10^18			8

# Les types numériques

## Si UNSIGNED

- Uniquement positifs
- Espace de valeurs double
- Ex. TINYINT UNSIGNED : entiers entre 0 et 255

Avec MySQL, lorsque l'on déclare un type pour un champ, on peut également y associer un mode "d'affichage".

Ainsi à l'ensemble de ces types on peut associer le mot clé **ZEROFILL** pour compléter le nombre par des zéros jusqu'à avoir un affichage sur N chiffres où N est le nombre de chiffres du plus grand nombre positif du type sélectionné. Et on peut également fixer cette valeur de N en l'indiquant entre parenthèse :

Affichage du nombre 23	
Int	23
Int ZEROFILL	0000000023
Int (4) ZEROFILL	0023

### Ce qu'il faut retenir :

Avant de choisir un type entier, il faut se poser les questions suivantes :

- Ai-je besoin de stocker des nombres négatifs ?
- Quel est le plus grand nombre que je serai amené à stocker ?

# Les dates

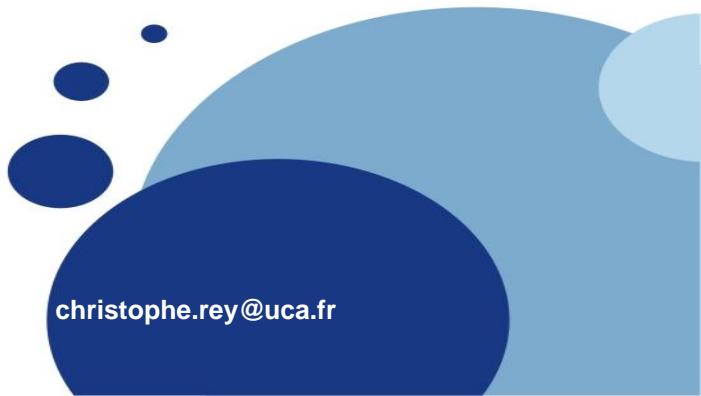
Type	Min	Max	Précision	Format
Date	1 janvier 1000	31 Décembre 9999	jour	AAAA-MM-JJ
Datetime (date et heure)	1 Janvier 1000 00h00m00s	31 Décembre 9999 23h59m59s	Seconde	AAAA-MM-JJ HH:MM:SS
Timestamp (date et heure avec une plage de validité réduite)	1 Janvier 1970 00h00m00s	31 Décembre 2037 23h59m59s	Seconde	AAAAMMJJHHMMSS
Time (heure, durée)	-838 h 59 m 59s	838 h 59 m 59 s	Seconde	HH:MM:SS
Year	1901	2155	Année sur 4 chiffres	AAAA
Year(2)	70 (i.e. 1970)	69 (i.e. 2069)	Année sur 2 chiffres	AA

# Les types textuels

Type	Description	Taille
Char(x)	Texte de taille fixe x caractères (complété par des espaces si nécessaire)	x caractères (chaîne limitée à 255 car.)
Varchar(x)	Texte de taille variable d'au maximum x car	(chaîne limitée à 65535 car.)
Tinytext	Texte potentiellement très long	(chaîne limitée à 255 car.)
Text		(chaîne limitée à 65535 car.)
Mediumtext		(chaîne limitée à 16777215 car.)
Longtext		(chaîne limitée à 4294967295 car.)
Enum	Un texte parmi une liste de n textes	n vaut 65535 au maximum Ex. enum('Y','N')
Set	Sous ensemble de texte parmi une liste de m textes	m vaut au plus 64 Ex. set('Toi', 'Moi', 'Eux')

**MMI Vichy  
S3  
BD2**

# **Les opérateurs et les fonctions**



**christophe.rey@uca.fr**

## Une trentaine d'opérateurs (de calcul ou de recherche)

- Calcul numérique
- Calcul temporel INTERVAL
- Comparaison
- Quelques 200 fonctions
  - ➔ aperçu des plus utilisées
  - La syntaxe des fonctions
  - Les fonctions mathématiques
  - Les fonctions textuelles
  - Les fonctions de comparaison
  - Extraction d'éléments de date/heure
  - Les fonctions d'information
  - ...

# Les opérateurs de calcul numérique

Opérateur	Description	Exemple	Résultat affiché
+	Addition	SELECT 5+2;	7
-	Soustraction	SELECT 5-2;	3
-	Mise en négatif	SELECT -5;	-5
*	Multiplication	SELECT 5*2;	10
/	Division	SELECT 5/2;	2.5000
Div	Division entière (décimales omises)	SELECT 5 Div 3;	1

# L'opérateur de calcul temporel INTERVAL

- L'opérateur de calcul temporel est nommé **INTERVAL**.
- La durée d'INTERVAL peut-être spécifiée par des mots-clés qui permettent de définir l'unité de la durée choisie.

Exemple:

```
SELECT '2000-01-01' + INTERVAL 20 YEAR AS nom_colonne ;
```

Date de départ

Durée de l'INTERVAL

Unité de la durée

Résultat:

nom_colonne
2020-01-01

# Les opérateurs de comparaison

- Ecrire des conditions (cf clauses **where** et **having**).

Opérateur(s)	Renvoi "true" si...
<> ou !=	les deux valeurs ne sont pas égales
<	la valeur de gauche est strictement inférieure à celle de droite
>	la valeur de gauche est strictement supérieure à celle de droite
<=	la valeur de gauche est strictement inférieure ou égale à celle de droite
>=	la valeur de gauche est strictement supérieure ou égale à celle de droite
BETWEEN..AND	la valeur testée est située entre deux valeurs données
IN	la valeur testée se situe dans une liste valeurs données
NOT IN	la valeur testée ne se situe pas dans une liste de valeurs données
LIKE	la valeur de gauche correspond à celle de droite (celle de droite peut utiliser le caractère % pour simuler n'importe quel nombre de caractère, et _ pour un seul caractère)
NOT LIKE	les deux valeurs ne correspondent pas
REGEXP ou RLIKE	la valeur de gauche correspond à l'expression régulière donnée
NOT REGEXP	la valeur de gauche ne correspond pas à l'expression régulière donnée
EXISTS	la requête imbriquée qui suit renvoie au moins une ligne en résultat
NOT EXISTS	la requête imbriquée qui suit ne renvoie aucune ligne dans son résultat

# NULL, l'absence de valeur

## Comparaison avec NULL :

- NULL ne répond vrai à aucune condition, sauf IS NULL et IS NOT NULL  
Spécifique à MySQL l'opérateur  $<=>$  (cf exemple après)
- Exemple :  
une valeur d'attribut numérique à NULL
  - N'est pas égale à 12
  - N'est pas différente de 12
- Toute comparaison avec NULL (sauf IS NULL, ou IS NOT NULL) renvoie inconnu (ni vrai, ni faux)
- Calcul avec NULL
  - Tout calcul avec NULL a un résultat NULL.
  - Exception : fonctions d'agrégation :
    - Elles ignorent les NULL.
    - Si que des NULL, leur résultat est NULL (sauf Count() qui renvoie alors 0).
- Essayer d'avoir le moins de NULL possibles (par exemple en prévoyant une valeur par défaut).

# Exemples opérateurs de comparaison



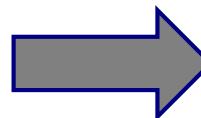
Opérateur	Description	Exemples	Résultat
=	Egal	'Austen' = 'AUSTEN' 'Austen' = NULL Null = Null	Vrai Inconnu Inconnu
Binary	Rend un texte sensible à la casse	'Austen' = BINARY 'AUSTEN'	Faux
<=>	Egal, compatible avec NULL	'Austen' <=> 'AUSTEN' 'Austen' <=> NULL Null <=> Null	Vrai Faux Vrai
!= ou <>	Différent	'Austen' <> 'AUSTEN' 'Austen' != NULL 'Austen' <=> BINARY 'AUSTEN'	Faux Inconnu Vrai
< >	Inférieur strictement Supérieur strictement	5 < 12 'Austen' < 'Arnaud'	Vrai Faux
<= >=	Inférieur ou égal Supérieur ou égal	'Austen' < 'AUSTEN' 'Austen' <= 'AUSTEN' 'Austen' > BINARY 'AUSTEN' 'Austen' > NULL 'Austen' < NULL	Faux Vrai Vrai Inconnu Inconnu
IS NULL IS Not NULL	Est NULL N'est pas NULL	NUL IS NULL 0 IS NULL	Vrai Faux
BETWEEN Not BETWEEN	Est entre N'est pas entre	4 BETWEEN 2 AND 5 'Austen' BETWEEN 'A' AND 'B'	Vrai Vrai
IN Not IN	Est dans la liste N'est pas dans la liste	2 IN (1, 2, 5) 4 IN (1, 2, 5)	Vrai Faux
LIKE Not LIKE	Comme Pas comme	'Austen' LIKE 'Aust%' 'Tim' LIKE 'T_m' 'Tom' LIKE 'T_m'	Vrai Vrai Vrai

# Fonctions de MySQL – Principe 1

- Syntaxe :
  - `SELECT MAFONCTION(colonne1)  
FROM table1;`
- Fonctionnement
  - Etape 1 : On extrait colonne1 de table1 (c'est une projection)
  - Etape 2 : MAFONCTION est appliquée à chaque valeur de la colonne colonne1 de la table table1
- Exemple :
  - `SELECT ROUND(colonne1)  
FROM table1;`
  - La fonction ROUND : arrondie à l'entier le plus proche

table1

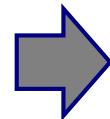
colonne1	colonne2	...
2.6	...	...
5.1	...	...
2.2	...	...



ROUND(colonne1)
3
5
2

# Fonctions de MySQL – Principe 2

- Syntaxe :
  - `SELECT MAFONCTION(valeur);`
- Fonctionnement
  - On fait faire un calcul directement sur une valeur connue.
- Exemple :
  - `SELECT ROUND(2.125);`



ROUND(2.125)

2

# Récapitulatif fonctions mathématiques

Fonction	Description	Exemple	Résultat
Abs(x)	Valeur absolue de x notée $ x $	Abs(5) Abs(-2)	5 2
Ceil(x)	Arrondi à l'entier supérieur de x	Ceil(2.1) Ceil(-2.1)	3 -2
Floor(x)	Arrondi à l'entier inférieur de x	Floor(2.1) Floor(-2.1)	2 -3
Power(x,p)	x à la puissance p, noté $x^p$	Power(5,2)	25
Rand()	Tirage d'un Float aléatoire entre 0 et 1	Rand()	0.42 (par exemple) Le résultat change à chaque fois.
Round(x,d)	Arrondi de x au nombre le plus proche ayant d décimales	Round(2.12,1) Round(-2.18,1)	2.1 -2.2
Sqrt(x)	Racine carrée de x	Sqrt(25)	5
Truncate(x, d)	Ne garde que d décimales à x (supprime les autres)	Truncate(2.12,1) Truncate(-2.18,1)	2.1 -2.1

# Les fonctions sur les chaînes de caractères

## La fonction CHAR\_LENGTH (ou CHARACTER\_LENGTH ou LENGTH)

- Compter le nombre de caractères présents dans une chaîne
- Exemple:

SELECT monchamp, CHAR\_LENGTH(monchamp) FROM  
matable;

monchamp	CHAR_LENGTH(monchamp)
CedricCarmie	12
JauffreyMirand	14

- Les fonctions UPPER et LOWER

- Mettre en majuscules ou minuscules
- Exemple:

SELECT monchamp, LOWER(monchamp) FROM matable;

monchamp	LOWER(monchamp)
CedricCarmie	cedriccarmie
JauffreyMirand	jauffreymirand

# Récapitatif fonctions sur les chaînes

Fonction	Description	Exemple	Résultat
Char_Length(texte)	Nb de caractères dans texte	Char_Length('Austen')	6
Concat(texte1, texte2, texte 3, ...)	Accole (concatène) texte1 avec texte2 avec texte3 avec ... pour former un nouveau texte.	Concat ('Jane', 'Austen')	'JaneAusten'
Concat_WS(sep, texte1, texte2, texte 3, ...)	Concatène avec le séparateur sep.	Concat _WS(' ; ', 'Jane', 'Austen')	'Jane ; Austen'
Locate(texte, grandtexte)	Recherche texte dans grandtexte et renvoie sa position (le premier caractère est numéroté 1).	Locate('ste','Jane Austen') Locate('toto','Jane Austen')	8 0
Locate(texte, grandtexte,début)	Idem au-dessus mais commence la recherche au caractère numéroté début.	Locate('ste','Jane Austen',3) Locate('ste','Jane Austen',9) Locate('Jan','Jane Austen',1)	8 0 1
Lower(texte)	Renvoie texte en minuscules	Lower('Jane Austen')	'jane austen'
Replace(texte,s1,s2)	Dans texte, remplace s1 par s2	Replace('Jane Austen','ste','--')	'Jane Au---n'
Substring(texte FROM début FOR nb_carac)	Renvoie la sous-chaîne de caractères de texte qui commence au caractère numéroté début et qui est de longueur nb_carac	Substring('Jane Austen',1,4)	'Jane'
Substring(texte FROM début)	Renvoie la sous-chaîne de caractères de texte qui commence au caractère numéroté début et qui s'arrête à la fin de texte	Substring('Jane Austen',6)	'Austen'
Trim(texte)	Elimine les espaces au début et à la fin de texte (variantes Rtrim et Ltrim)	Trim(' Jane Austen ')	'Jane Austen'
Upper(texte)	Renvoie texte en majuscules	Upper('Jane Austen')	'JANE AUSTEN'

# Les fonctions sur les dates

## La fonction CURRENT\_DATE()

- Avoir la date courante
- Exemple : `SELECT CURRENT_DATE();`

	Retour
CURRENT_DATE	2007-03-19

- La fonction CURRENT\_TIMESTAMP()
- Avoir la date et l'heure courante
- Exemple : `SELECT CURRENT_TIMESTAMP();`

	Retour
CURRENT_TIMESTAMP	2007-03-19 13:22:26

- La fonction DATE\_FORMAT(datetime, format)
- Conversion de date en texte
- CF transparent qui suit

# Rappel sur les types temporels

Type	Min	Max	Précision	Format
Date	1 janvier 1000	31 Décembre 9999	jour	AAAA-MM-JJ
Datetime (date et heure)	1 Janvier 1000 00h00m00s	31 Décembre 9999 23h59m59s	Seconde	AAAA-MM-JJ HH:MM:SS
Timestamp (date et heure avec une plage de validité réduite)	1 Janvier 1970 00h00m00s	31 Décembre 2037 23h59m59s	Seconde	AAAAMMJJHHMMSS
Time (heure, durée)	-838 h 59 m 59s	838 h 59 m 59 s	Seconde	HH:MM:SS
Year	1901	2155	Année sur 4 chiffres	AAAA
Year(2)	70 (i.e. 1970)	69 (i.e. 2069)	Année sur 2 chiffres	AA

# Type « INTERVAL expr unit »

unit	Format de expr	Exemples d'expr
MICROSECOND	MICROSECONDS	25
SECOND	SECONDS	4
MINUTE	MINUTES	6
HOUR	HOURS	489
DAY	DAYS	12
WEEK	WEEKS	3
MONTH	MONTHS	48
QUARTER	QUARTERS	4
YEAR	YEARS	25
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'	'4.23'
MINUTE_MICROSECOND	'MINUTES:SECONDS.MICROSECONDS'	'12:5.1238'
MINUTE_SECOND	'MINUTES:SECONDS'	'45:26'
HOUR_MICROSECOND	'HOURS:MINUTES:SECONDS.MICROSECONDS'	'1:25:5.201'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'	'5:23:15'
HOUR_MINUTE	'HOURS:MINUTES'	'9:48'
DAY_MICROSECOND	'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS'	'3 14:58:09.54224'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'	'3 14:58:12'
DAY_MINUTE	'DAYS HOURS:MINUTES'	'8 15:41'
DAY_HOUR	'DAYS HOURS'	'7 23'
YEAR_MONTH	'YEARS-MONTHS'	'10-02'

# Récapitulatif fonctions sur les dates

<https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html>

Fonction	Description
<a href="#">ADDDATE()</a>	Add time values (intervals) to a date value
<a href="#">ADDTIME()</a>	Add time
<a href="#">CONVERT_TZ()</a>	Convert from one time zone to another
<a href="#">CURDATE()</a>	Return the current date
<a href="#">CURRENT_DATE(), CURRENT_DATE</a>	Synonyms for CURDATE()
<a href="#">CURRENT_TIME(), CURRENT_TIME</a>	Synonyms for CURTIME()
<a href="#">CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP</a>	Synonyms for NOW()
<a href="#">CURTIME()</a>	Return the current time
<a href="#">DATE()</a>	Extract the date part of a date or datetime expression
<a href="#">DATE_ADD()</a>	Add time values (intervals) to a date value
<a href="#">DATE_FORMAT()</a>	Format date as specified
<a href="#">DATE_SUB()</a>	Subtract a time value (interval) from a date
<a href="#">DATEDIFF()</a>	Subtract two dates
<a href="#">DAY()</a>	Synonym for DAYOFMONTH()
<a href="#">DAYNAME()</a>	Return the name of the weekday
<a href="#">DAYOFMONTH()</a>	Return the day of the month (0-31)
<a href="#">DAYOFWEEK()</a>	Return the weekday index of the argument
<a href="#">DAYOFYEAR()</a>	Return the day of the year (1-366)

# Récapitulatif fonctions sur les dates

<https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html>

Fonction	Description
<a href="#">EXTRACT()</a>	Extract part of a date
<a href="#">FROM_DAYS()</a>	Convert a day number to a date
<a href="#">FROM_UNIXTIME()</a>	Format Unix timestamp as a date
<a href="#">GET_FORMAT()</a>	Return a date format string
<a href="#">HOUR()</a>	Extract the hour
<a href="#">LAST_DAY</a>	Return the last day of the month for the argument
<a href="#">LOCALTIME(), LOCALTIME</a>	Synonym for NOW()
<a href="#">LOCALTIMESTAMP,</a>	
<a href="#">LOCALTIMESTAMP()</a>	Synonym for NOW()
<a href="#">MAKEDATE()</a>	Create a date from the year and day of year
<a href="#">MAKETIME()</a>	Create time from hour, minute, second
<a href="#">MICROSECOND()</a>	Return the microseconds from argument
<a href="#">MINUTE()</a>	Return the minute from the argument
<a href="#">MONTH()</a>	Return the month from the date passed
<a href="#">MONTHNAME()</a>	Return the name of the month
<a href="#">NOW()</a>	Return the current date and time
<a href="#">PERIOD_ADD()</a>	Add a period to a year-month
<a href="#">PERIOD_DIFF()</a>	Return the number of months between periods
<a href="#">QUARTER()</a>	Return the quarter from a date argument
<a href="#">SEC_TO_TIME()</a>	Converts seconds to 'HH:MM:SS' format
<a href="#">SECOND()</a>	Return the second (0-59)
<a href="#">STR_TO_DATE()</a>	Convert a string to a date

# Récapitulatif fonctions sur les dates

<https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html>

Fonction	Description
<u>SUBDATE()</u>	Synonym for DATE_SUB() when invoked with three arguments
<u>SUBTIME()</u>	Subtract times
<u>SYSDATE()</u>	Return the time at which the function executes
<u>TIME()</u>	Extract the time portion of the expression passed
<u>TIME_FORMAT()</u>	Format as time
<u>TIME_TO_SEC()</u>	Return the argument converted to seconds
<u>TIMEDIFF()</u>	Subtract time
<u>TIMESTAMP()</u>	With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments
<u>TIMESTAMPADD()</u>	Add an interval to a datetime expression
<u>TIMESTAMPDIFF()</u>	Subtract an interval from a datetime expression
<u>TO_DAYS()</u>	Return the date argument converted to days
<u>TO_SECONDS()</u>	Return the date or datetime argument converted to seconds since Year 0
<u>UNIX_TIMESTAMP()</u>	Return a Unix timestamp
<u>UTC_DATE()</u>	Return the current UTC date
<u>UTC_TIME()</u>	Return the current UTC time
<u>UTC_TIMESTAMP()</u>	Return the current UTC date and time
<u>WEEK()</u>	Return the week number
<u>WEEKDAY()</u>	Return the weekday index
<u>WEEKOFYEAR()</u>	Return the calendar week of the date (1-53)
<u>YEAR()</u>	Return the year
<u>YEARWEEK()</u>	Return the year and week

# Les fonctions sur les dates

## La fonction ADDDATE(date, intervalle)

- Ajouter n'importe quelle valeur (année, mois, jour, minute, seconde) à la date courante
- Exemple :

	monchamp	retour
ADDDATE(monchamp, INTERVAL 1 SECOND)	2001-01-01 20:00:00	2001-01-01 20:00:01
ADDDATE(monchamp, INTERVAL 1 MONTH)	2001-01-01 20:00:00	2001-02-01 20:00:00
ADDDATE(monchamp, INTERVAL "01:02:03" HOUR_SECOND)	2001-01-01 20:00:00	2001-01-01 21:02:03

- Les fonctions DAYOFWEEK(champ), DAYOFTYEAR(champ)

- Extraire des informations de dates
- Exemple :

	monchamp	retour
DAYOFWEEK(monchamp)	2002-01-01	3
DAYOFTYEAR(monchamp)	2002-01-01	1

# Récapitulatif fonction DATE\_FORMAT()

**Tableau 8.15 : Principaux codes de formats temporels (liste intégrale dans la description de DATE\_FORMAT() dans les documentations 5.0 et 5.1)**

Élément temporel	En nombre	En anglais	Exemples	Résultats
Secondes	%s ou %S		'%H:%i' '%k:%i'	09:45 9:45
Minutes	%i		'%kh%i'	9h45
Heures	%H (sur deux chiffres) ou %k (sans zéro initial)			
Jour de la semaine	%w (de 0 pour dimanche à 6 pour samedi)	%W (entier) ou %a (abrégé)	'%W %e' '%M' '%a %d' '%YM%m'	Thursday 9 March Thu 09 2006M03
Jour du mois	%d (sur deux chiffres) ou %e (sans zéro initial)			
Mois	%m (sur deux chiffres) ou %c (sans zéro initial)	%M (entier) ou %b (abrégé)		
Année	%Y (quatre chiffres) ou %y (deux chiffres)			

## Récapitulatif fonctions de conversion de dates

**Tableau 8.14 : Principales fonctions de conversion et formatage temporels**

Type de conversion	Fonction	Exemple	Résultat
DateTime local en Timestamp Unix	Unix_Timestamp(date et/ou heure MySQL)	Unix_Timestamp('2006-03-09 09:45:42')	1141893942
Time en nombre de secondes écoulées depuis minuit	Time_To_Sec(sec(heure))	Time_To_Sec('09:45:42')	35142
Nombre de secondes écoulées depuis minuit en Time	Sec_To_Time(nombre de secondes)	Sec_To_Time(35142)	09:45:42
Nombres en Time	Maketime(heures, minutes, secondes)	Maketime(9, 45, 42)	09:45:42
DateTime en texte	Date_Format(datetime MySQL, format)	Date_Format(Now(), '%W %d %M %H:%i %s sec.')	Thursday 09 March 09h45 42 sec
Texte en DateTime	Str_To_Date(texte, format)	Str_To_Date('March 9th 2006', '%M %D %Y')	2006-03-09
Conversion implicite de DateTime en nombre	Curdate() + 0	Curdate() + 0	20060309

# Récapitatif extraction de dates

**Tableau 8.16 : Quelques extractions d'éléments de date/heure**

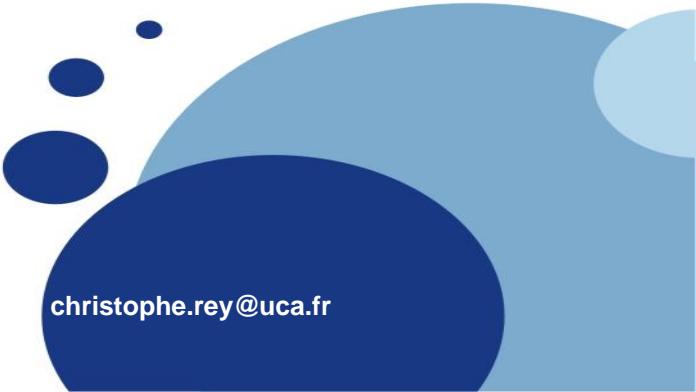
Type d'extraction	Avec Extract()	Avec les autres fonctions	Résultat
Jour de la semaine		DayName(Now())	Thursday
Année	Extract(Year From Now())	Year(Now())	2006
Année et mois	Extract(Year_Month From Now())		200603
Heure complète	Extract(Hour_Second From Now())		094542
		Time(Now())	09:45:42

# Quelques autres fonctions utiles

Fonction	Rôle
Benchmark()	Répète le calcul d'une expression et permet d'en estimer la durée de calcul
Current_User()	Nom et machine de l'utilisateur
Database()	BD en cours d'utilisation
Found_Rows()	Nombre de lignes récupérées par un SELECT
Last_Insert_Id()	Dernier ID généré automatiquement
Row_Count()	Nombre de lignes modifiées par la dernière requête
Version	Version de MySQL

**MMI Vichy  
S3  
BD2**

# **L'agrégation et le regroupement**



**christophe.rey@uca.fr**

# L'agrégation et le regroupement

## Agrégation :

- Agréger les valeurs de plusieurs lignes
- Exemple : procéder à des sommes, des comptages ou à d'autres opérations
- Fonctions d'agrégation classiques
  - Sum() : Effectue une somme (colonne numérique)
  - Avg() : Effectue une moyenne (colonne numérique)
  - Max() : Donne la plus grande valeur
    - Colonne numérique
    - Date ou heure (Valeur la plus récente)
    - Texte (Dernière valeur par ordre alphabétique)
  - Min() : Donne la plus petite valeur numérique, la date/heure la plus ancienne ou la première valeur texte par ordre alphabétique.
  - Count() : Compte le nombre de lignes, ne prend pas en compte les NULL.
  - Count(\*) : Compte le nombre de lignes, et prend en compte les valeurs NULL

# L'agrégation et le regroupement

## Autres fonctions d'agrégation

- VARIANCE() : Variance sur une population.
- VAR\_SAMP() : Variance sur un échantillon.
- STD(), STDDEV() : Ecart-type sur une population.
- STDDEV\_SAMP() : Ecart-type sur un échantillon

# L'agrégation et le regroupement

- Exemples

```
SELECT COUNT(IDEleve)
      AS Nb_Eleve_MM
FROM Eleves ;
```

Nb_Eleve_MM
6

## Eleves

IDEleve	Nom	Age	IdGroupe	...
101	Durant	18	5	...
105	Dupond	19	4	...
45	Mercier	17	5	...
423	Labard	20	1	...
54	Terrier	18	4	...
912	Felipe	23	4	...

```
SELECT MAX(Age)
      AS Eleve_le_plus_vieux
FROM Eleves ;
```

Eleve_le_plus_vieux
23

# L'agrégation et le regroupement

## Regroupement

- Regrouper des lignes dont les valeurs de certaines colonnes sont égales
- Appliquer des fonctions d'agrégation sur ces groupes de lignes.
- Exemple :
  - Nombre d'élèves dans chaque groupe de TP.
    - Regrouper les élèves appartenant au même groupe
    - Utiliser le Count(\*) sur la table des élèves,
    - Ajouter dans la requête une clause Group By, après un From et un éventuel Where
    - SELECT IdGroupe,  
COUNT(\*) AS nb\_Etudiants  
FROM Eleves  
GROUP BY IdGroupe ;

## Eleves

IDEleve	Nom	Age	IdGroupe	...
101	Durant	18	5	...
105	Dupond	19	4	...
45	Mercier	17	5	...
423	Labard	20	1	...
54	Terrier	18	4	...
912	Felipe	23	4	...

IdGroupe	Nb_Etudiants
1	1
4	3
5	2

# Clause HAVING

- Conditions sur chaque groupe d'un GROUP BY  
→ clause having

- Exemple :

- Compter le nombre d'étudiants par groupe
- Ne prendre en compte que les groupes constitués d'au moins 3 étudiants
- ```
SELECT IdGroupe,
       COUNT(*) AS nb_Etudiants
  FROM Eleves
 GROUP BY IdGroupe
 HAVING COUNT(*) >= 3
```

- On peut faire la même requête avec l'alias :

...

```
HAVING nb_Etudiants >= 3
```

**Eleves**

| IDEleve | Nom     | Age | IdGroupe | ... |
|---------|---------|-----|----------|-----|
| 101     | Durant  | 18  | 5        | ... |
| 105     | Dupond  | 19  | 4        | ... |
| 45      | Mercier | 17  | 5        | ... |
| 423     | Labard  | 20  | 1        | ... |
| 54      | Terrier | 18  | 4        | ... |
| 912     | Felipe  | 23  | 4        | ... |

| IdGroupe | Nb_Etudiants |
|----------|--------------|
| 4        | 3            |

# Clause HAVING

- Cohabitation de WHERE et HAVING
  - WHERE et HAVING sont parfaitement compatibles.

- Exemple :
  - Pour les étudiants de plus de 20 ans uniquement, grouper les étudiants par IdGroupe
  - SELECT IdGroupe,  
COUNT(\*) AS nb\_Etudiants  
FROM Eleves  
WHERE Age >= 20  
GROUP BY IdGroupe  
HAVING nb\_Etudiants >= 2

**Eleves**

| IDEleve | Nom     | Age | IdGroupe | ... |
|---------|---------|-----|----------|-----|
| 101     | Durant  | 21  | 5        | ... |
| 105     | Dupond  | 19  | 4        | ... |
| 45      | Mercier | 20  | 5        | ... |
| 423     | Labard  | 20  | 1        | ... |
| 54      | Terrier | 18  | 4        | ... |
| 912     | Felipe  | 23  | 4        | ... |

| IdGroupe | Nb_Etudiants |
|----------|--------------|
| 5        | 2            |

# La fonction GROUP\_CONCAT()

- Pour concaténer les valeurs d'une colonne d'un groupe.
- Possède 2 options
  - Order By
  - Separator
- Exemple 1 :

```
SELECT id_groupe, GROUP_CONCAT(prenom) AS prenoms  
FROM Table1  
GROUP BY id_groupe ;
```

| <b>id_groupe</b> | <b>prenom</b> |
|------------------|---------------|
| 1                | Jauffrey      |
| 1                | Cédric        |



| <b>id_groupe</b> | <b>prenoms</b> |
|------------------|----------------|
| 1                | JauffreyCédric |

**Table1**

# La fonction GROUP\_CONCAT()

- Exemple 2 :

```
SELECT id_groupe,  
       GROUP_CONCAT(prenom ORDER BY prenom SEPARATOR ' ET ')  
AS prenoms  
FROM Table1  
GROUP BY id_groupe ;
```

| id_groupe | prenom   |
|-----------|----------|
| 1         | Jauffrey |
| 1         | Cédric   |

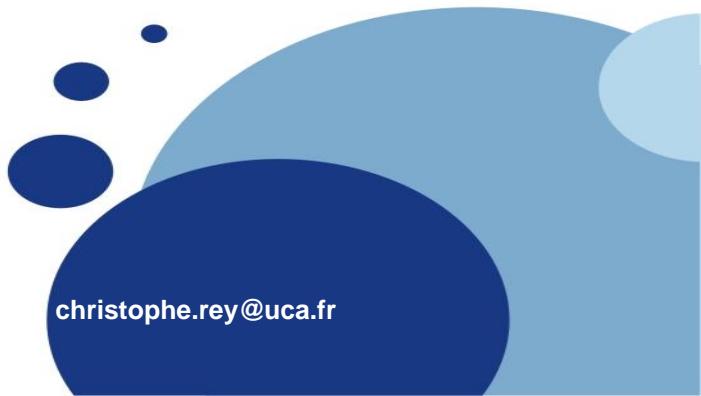


| id_groupe | prenom             |
|-----------|--------------------|
| 1         | Cédric ET Jauffrey |

**Table1**

**MMI Vichy  
S3  
BD2**

# **Les sous-requêtes**



**christophe.rey@uca.fr**

# Les sous-requêtes

- Aussi appelées requêtes imbriquées
- Ecriture plus intuitive que des jointures
- Certaines requêtes exprimables uniquement avec des sous-requêtes.
- Le résultat d'une sous-requête peut être :
  - Un scalaire (table d'une seule colonne et une seule ligne)
  - Une ligne
  - Une colonne (ou liste)
  - Une table
- Plusieurs utilisations
  - À la place d'une valeur ou d'un constructeur de ligne (par exemple dans une condition du WHERE)
  - En tant que table dérivée (dans la clause FROM)
  - Avec un opérateur (IN, ANY, SOME, ALL)

# Exemples de sous-requêtes

## Sous-requête renvoyant un scalaire

- Nom et prénom des auteurs les plus jeunes
- `SELECT Nom, Prenom FROM Auteur WHERE Date_naissance = (SELECT max(Date_naissance) FROM Auteur);`

## • Sous-requête renvoyant une ligne

- Nom et prénom des employés dont le salaire et les primes sont supérieures à la moyenne
- `SELECT Nom, Prenom FROM Auteur WHERE (sal,primes) > (SELECT avg(sal), avg(primes) FROM employe);`

## • Sous-requête renvoyant une colonne

- Nom et prénom des auteurs européens
- `SELECT Nom, Prenom FROM Auteur WHERE nationality IN (SELECT country FROM pays WHERE continent = 'Europe');`

# Exemples de sous-requêtes

## Sous-requête renvoyant une colonne

- Nom et prénom des employés ayant un salaire supérieur aux salaires de tous les vendeurs.
- `SELECT Nom, Prenom FROM employé WHERE sal > ALL (SELECT sal FROM employé WHERE métier = 'vendeur');`

## • Sous-requête renvoyant une table (plus rare)

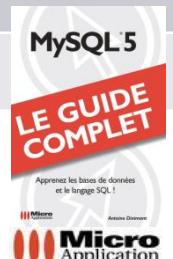
- Nom, prénom et prime des employés dont la prime est supérieure à la prime moyenne.
- `SELECT e.Nom, e.Prenom, avgp.prime  
FROM employé as e,  
(SELECT avg(primes) FROM employé) as avgp  
WHERE e.Prime > avgp.prime;`

# Corrélation dans les sous-requêtes

- Réutiliser une table de la « sur-requête » dans la sous-requête
- Impossible dans l'autre sens (pas d'utilisation d'une table d'une sous-requête dans la sur-requête)
- Exemple
  - Nom et prénom des employés ayant un salaire supérieur à la moyenne des salaires des employés de même métier.
  - ```
SELECT Nom, Prenom
      FROM employé as e1
 WHERE sal > (SELECT avg(sal)
                  FROM employé as e2
                 WHERE e1.métier = e2.métier);
```

# Résumé utilisation des sous-requêtes

Utilisation	Type de sous-requête	Corrélation possible
A la place d'une valeur quelconque	Scalaire	Oui
Comparée à un constructeur de ligne	Ligne	Oui
En tant que table dérivée dans le FROM	Table	Non
Avec un opérateur de liste (In, Any, Some, All)	Liste (colonne unique)	Oui
Comparaison de type : <i>Constructeur de ligne In sous-requête</i>	Table (même nombre de colonnes que le constructeur)	Oui
Avec Exists ou Not Exists	Indifférent	Oui

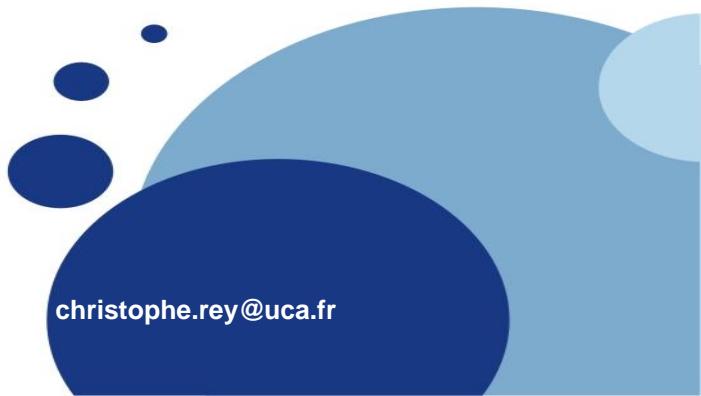


Apprenez les bases de données et le langage SQL !

Micro Application

**MMI Vichy  
S3  
BD2**

# **Modifications des données**



**christophe.rey@uca.fr**

## 3 façons

- **INSERT INTO ... VALUES...**
  - insert into table (colonne1, colonne2,...)  
values (liste de valeurs), ... , (liste de valeurs);
  - insert into table  
values (liste de valeurs), ... , (liste de valeurs);  
**Dans ce cas :**
    - La liste des valeurs doit comporter des valeurs pour tous les attributs de la table.
    - Il faut faire attention à l'ordre des colonnes dans la table.
    - On peut insérer plusieurs lignes à la fois (une liste de valeurs par ligne).
- **INSERT INTO ... SET ...**
  - insert into table set  
attribut1='toto', attribut2='loulou', ...;  
**Dans ce cas :**
    - On ne peut insérer qu'une seule ligne à la fois.
    - La requête peut être facilement transformée en requête UPDATE.
- **INSERT INTO ... SELECT ...**
  - insert into table1 (att1, att2,...)  
select (col1, col2, ...) from table2 where ... ;  
**Dans ce cas :**
    - On doit avoir autant d'attributs col que d'attributs att, avec des types respectifs qui correspondent.

# Ajout de données avec insert

## Exemple avec INSERT INTO ... SELECT ...

- On souhaite avoir dans une même table les données d'un client et celles de ses achats.
- On suppose :
  - customer\_photos (name,nationality,catalogn,image,price)
- On remplit customer\_photos :
  - ```
INSERT INTO customer_photos (name, nationality ,  
                           catalogn,image, price)  
SELECT DISTINCT c.name, c.country , p.catalogn ,  
               p.image , p.price  
FROM customer as c , photo as p , includes as i ,  
      transaction as t , buys as b  
WHERE (p.catalogn=i.catalogn) and (c.idname=b.idname)  
      and (t.idn=i.idn) and (t.idn = b.idn);
```
- Attention : customer\_photos est à jour juste après la requête, mais ne le reste pas automatiquement !  
→ peut-être faudrait-il faire une vue plutôt.

# Colonnes muettes

Correspondent aux attributs non renseignés au cours d'un insert

- Si l'attribut non renseigné est une CP avec auto-increment
  - Alors un numéro automatique est attribué à la colonne
- Sinon
  - Si l'attribut a été créé avec une valeur par défaut
    - Alors la valeur par défaut est donnée à la ligne ajoutée pour l'attribut concerné
  - Sinon
    - Si l'attribut accepte les NULL
      - Alors NULL est mis dans l'attribut de la ligne ajoutée
    - Sinon
      - Si le mode strict est activé
        - Alors la valeur par défaut correspondant au type de l'attribut muet est ajoutée, et MySQL émet un avertissement

# Récapitulatif colonnes muettes

| Valeur par défaut ?                      | Colonne obligatoire          | Strict mode | Résultat                                                                                                                                                                                                            |
|------------------------------------------|------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Colonne clé avec auto_increment          |                              |             | Numéro automatique                                                                                                                                                                                                  |
| Colonne avec valeur par défaut explicite |                              |             | Valeur par défaut                                                                                                                                                                                                   |
| Colonne sans valeur par défaut explicite | Non (NULL)<br>Oui (NOT NULL) | Oui<br>Non  | NULL<br>Refus de la ligne entière<br>Valeur par défaut implicitement fixée par le type de colonne (0 pour les colonnes numériques, chaîne vide pour les types chaîne, etc...) MySQL émet un avertissement (warning) |



# Remarques sur l'insertion

- Ajout explicite de la valeur par défaut :  
`insert into table  
set att1 = DEFAULT, ... ;`
- Sur une colonne auto\_increment : NULL, 0 et DEFAULT induisent l'attribution d'un numéro automatique.
- Récupérer le dernier id attribué (par ex. sur une colonne auto\_increment)
  - `select LAST_INSERT_ID();`
  - A appeler juste après l'insertion.
  - Ne concerne que la session en cours (ne prend pas en compte les insertions d'autres sessions).

# Le mode STRICT

## Avec le mode strict

- MySQL ne remplace plus les valeurs interdites (par ex. par une contrainte d'intégrité) par des valeurs par défaut.
- MySQL refuse les instructions violant les contraintes d'intégrité.
- Valeur du mode dans la variable @@SQL\_MODE
  - STRICT\_TRANS\_TABLES : tables InnoDB en mode strict seulement
  - STRICT\_ALL\_TABLES : toutes tables en mode strict
  - Sinon pas de mode strict du tout
- Modification du mode
  - Dans le fichier my.ini :  
`sql-mode="STRICT_ALL_TABLES,..."`
  - En cours de session :
    - Pour la session en cours :  
`SET SESSION SQL_MODE = 'STRICT_ALL_TABLES,...'`
    - Pour toutes les sessions :  
`SET GLOBAL SQL_MODE = 'STRICT_ALL_TABLES,...'`
- Autres options :  
voir <http://dev.mysql.com/doc/refman/5.0/fr/server-sql-mode.html>

# La modification avec update

Syntaxe :

UPDATE table

SET col1 = val1,col2 = val2, ...

WHERE

Conditions\_sélectionnant\_les\_lignes\_à\_modifier

ORDER BY Ordre\_de\_modification

LIMIT Nombre\_maximal\_de\_lignes\_à\_modifier ;

- UPDATE et SET sont obligatoires
- WHERE, ORDER BY et LIMIT sont facultatives
- Si pas de WHERE : modification de toutes les lignes de la table

# La suppression avec DELETE

Syntaxe :

```
DELETE
  FROM table
 WHERE
  Conditions_sélectionnant_les_lignes_à_supprimer
 ORDER BY Ordre_de_suppression
 LIMIT Nombre_maximal_de_lignes_à_supprimer ;
```

- DELETE et FROM sont obligatoires
- WHERE, ORDER BY et LIMIT sont facultatives
- Si ni WHERE, ni LIMIT : suppression de toutes les lignes
- Autre solution pour supprimer toutes les lignes d'une table :

```
TRUNCATE TABLE table;
```

# Remplacer avec REPLACE

Mêmes syntaxes que INSERT :

- REPLACE ... VALUES...
- REPLACE ... SET ...
- REPLACE ... SELECT ...
- Si valeur de CP existe déjà dans la table  
Alors la ligne est supprimée et une nouvelle ligne est insérée avec cette valeur et les nouvelles valeurs
- Sinon (valeur de CP n'existe pas ou n'est pas mentionnée)  
le REPLACE revient à un INSERT
- Les colonnes muettes sont traitées comme dans un INSERT.

# Remplacement partiel

- L'option ON DUPLICATE KEY UPDATE d'un INSERT  
INSERT INTO table  
SET ...  
ON DUPLICATE KEY UPDATE col1=Val1, ... ;  
• Si la CP est nouvelle, alors INSERT, sinon UPDATE.

# Modification multitable avec UPDATE

- Pour modifier des valeurs d'une table A selon un critère relatif à une table B.
- Exemple :
  - Modification de la disponibilité des photos présentes dans une transaction :
  - UPDATE PHOTO as P, INCLUDES as I  
SET P.available = 'N'  
WHERE (P.catalogn = I.catalogn);
- Ni ORDER BY, ni LIMIT dans un UPDATE multitable
- Exercice :
  - UPDATE author, takes, influences  
SET author.st\_addr='vichy'  
WHERE (author.name=takes.name) AND (author.dob=takes.dob)  
AND (author.name=influences.auth1\_name)  
AND (author.dob=influences.auth1\_dob)  
AND (author.name=influences.auth2\_name)  
AND (author.dob=influences.auth2\_dob) ;
- Qu'exprime cette requête ?

# Suppression multitables avec DELETE

- Pour supprimer des lignes d'une table A selon un critère relatif à une table B → implique souvent de supprimer des lignes dans A et B à la fois (à cause des CE)
- Syntaxe 1 :

```
DELETE tables_ où_ supprimer
FROM autres_tables_ et_ jointures_ éventuelles
WHERE conditions_de_suppression_et_de_jointure;
```
- Syntaxe 2 :

```
DELETE
FROM tables_ où_ supprimer
USING toutes_tables_ où_ supprimer_et_ autres
WHERE conditions_de_jointure_et_de_suppression
```

# Suppression multitables avec DELETE

## Exemple :

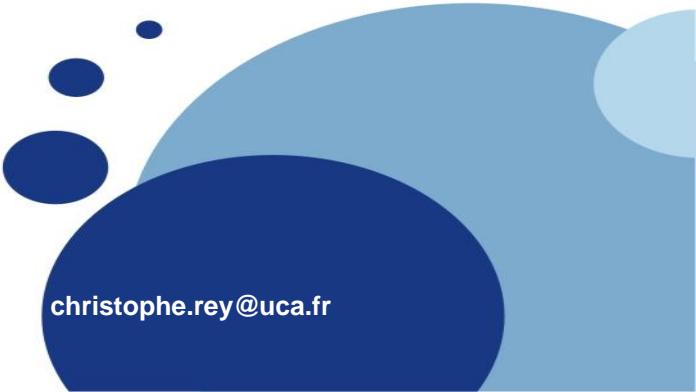
- Supprimons de la base toutes les photos de type abstrait qui ont été vendues
  - DELETE p  
FROM photo as p  
inner join includes as i on p.catalogn=i.catalogn  
inner join abstract as a on p.catalogn=a.catalogn  
;

## Exercices :

- Supprimer les auteurs n'ayant aucune photo.
- Supprimer les portraits et les modèles datant de plus de 50 ans.

**MMI Vichy  
S3  
BD2**

# Les vues



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

# Introduction

## Définition :

- Les vues sont des tables virtuelles issues de l'interrogation d'autres tables.
- Ou encore des requêtes préenregistrées et utilisables comme des tables dans d'autres requêtes.

## Utilité :

- Conserver les principales requêtes sur la base pour les rendre facilement réutilisables.
- Ajouter des restrictions aux données : notamment permettre de restreindre l'accès de certains utilisateurs à certaines lignes de certaines tables.

## Concrètement :

- Une vue est assimilée à une table :
  - A un nom
  - « Show tables » montre les vues avec les tables
  - « Desc » donne une description du schéma de la vue comme de celui d'une table
- Une vue est fondamentalement une requête SELECT :
  - Ne contient pas de donnée,
  - Les calcule à la volée.
  - Les données sont donc toujours à jour.

# Commandes et privilèges nécessaires

|                                                                                                                         | Commandes SQL                                                           | Privilèges requis                                                                |
|-------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <b>Création d'une vue</b><br>(la requête ne doit pas afficher des colonnes de même nom<br>➔ les renommer si nécessaire) | CREATE VIEW nom_vue<br>AS<br>SELECT ... FROM ... WHERE ... ;            | CREATE VIEW                                                                      |
| <b>Modification d'une vue</b><br>(revient à remplacer complètement la requête ➔ CREATE OR REPLACE VIEW)                 | ALTER VIEW nom_vue<br>AS<br>SELECT ... FROM ... WHERE ... ;             | CREATE VIEW ET DROP<br>Et être le créateur de la vue ou avoir le privilège SUPER |
| <b>Remplacement d'une vue</b> (ou création si elle n'existe pas déjà)                                                   | CREATE OR REPLACE VIEW nom_vue<br>AS<br>SELECT ... FROM ... WHERE ... ; | CREATE VIEW ET DROP<br>Et être le créateur de la vue ou avoir le privilège SUPER |
| <b>Visualiser la requête source d'une vue</b>                                                                           | SHOW CREATE VIEW nom_vue;                                               | SHOW VIEW                                                                        |
| <b>Suppression d'une vue</b>                                                                                            | DROP VIEW nom_vue ;                                                     | DROP                                                                             |

# Exemple

- Afficher toutes les infos des photos de type portrait
- ```
CREATE VIEW details_portrait
AS
SELECT ph.* , a.name AS author_name, a.nationality,
       a.dob AS author_dob,
       m.name AS model_name ,
       m.dob AS model_dob,
       m.sex,m.bio,models.agency
FROM photo AS ph ,portrait AS prt ,author AS a,
      model AS m, takes,models
WHERE (ph.catalogn=prt.catalogn) and
      (prt.catalogn=models.catalogn)
      and (m.name=models.name)
      and (m.dob=models.dob)
      and(ph.catalogn=takes.catalogn)
      and(a.name=takes.name)
      and(a.dob=takes.dob);
```

## La requête source

- un SELECT unique
- Pas de sous-requête dans le FROM
- Peut utiliser une autre vue dans le FROM
- Pas de variable système (@@...)
- Doit pouvoir être exécutée au moment de l'utilisation de la vue :
  - Valide
  - Définie avec des objets déjà existants
- Une vue ne peut pas lire de table temporaire, ni être elle-même temporaire.
- Pas d'index sur une vue.

# Vues modifiables et insérables

- Les vues peuvent permettre
  - La mise à jour des données qu'elles exposent
  - L'insertion de nouvelles données dans les tables à partir desquelles elles sont définies
- Contraintes rendant une vue non-modifiable
  - Clauses GROUP BY ET HAVING
  - Les fonctions d'agrégation : SUM(), COUNT(), ...
  - Le mot-clé DISTINCT, ou aussi UNION
  - Les sous-requêtes scalaires dans le SELECT
  - Une vue non modifiable dans le FROM
  - Une sous-requête corrélée dans le WHERE
- Autres contraintes pour une vue modifiable
  - La modification ne peut porter que sur une seule table à la fois.
  - On ne peut pas modifier les valeurs d'une colonne calculée avec une fonction.
- Exemple :

```
UPDATE details_portrait
SET price=price*2
WHERE (resolution > 12000);
```

La modification se fait réellement dans la table photo (une vue ne contient pas de données).

# Vues modifiables et insérables

Contraintes supplémentaires pour une vue insérable et supprimable

- Pas de doublon sur les noms des colonnes
- La vue doit présenter l'ensemble des colonnes obligatoires n'ayant pas de valeur par défaut.
- Pas de colonne calculée.
- Pas de jointure externe.
- Insertion sur une seule table à la fois.

# Exercice sur les vues

- Sujet :  
On voudrait que les photographes de nationalité française et leurs photos puissent être gérés (supprimés, modifiés, ajoutés) uniquement par l'administrateur nommé julien ([julien@localhost](mailto:julien@localhost)) et que ce dernier ne puisse pas avoir accès aux autres photographes et à leurs photos.
- Question :  
comment faire avec les vues ?

# Solution (1/3)

- Etape 1 : créer deux vues :
  - La vue « authors\_fr » qui regroupe tout les photographes français.
  - La vue « photo\_fr » qui regroupe toutes les photos des photographes français.
- Etape 2 : modifier les droits de julien@localhost
  - Supprimer tous les droits qu'il avait sur l'ensemble de la table « author » et les reporter sur la vue « authors\_fr ».
  - Supprimer tous les droits qu'il avait sur la table photo et les reporter sur la vue « photo\_fr »

# Solution (2/3)

- Etape 1 : créer deux vues :

- La vue « authors\_fr » qui regroupe tous les photographes français.

```
CREATE VIEW authors_fr
```

```
AS
```

```
SELECT * FROM author  
WHERE(nationality='french');
```

- La vue « photo\_fr » qui regroupe toutes les photos des photographes français.

```
CREATE VIEW photo_fr
```

```
AS
```

```
SELECT photo.catalogn as catalogn_fr, film, color, fstop,  
speed, resolution ,available , price  
FROM photo, takes, author  
WHERE(photo.catalogn=takes.catalogn)and  
(takes.name=author.name) and (takes.dob=author.dob)  
And (author.nationality="french");
```

# Solution (3/3)

## Etape 2 : modifier les droits de julien@localhost

- Supprimer tous les droits qu'il avait sur l'ensemble de la table « author » et les reporter sur la vue « authors\_fr ».

```
REVOKE SELECT ON shuttershop.author  
FROM julien@localhost;
```

```
GRANT SELECT ON shuttershop.author_fr  
TO julien@localhost ;
```

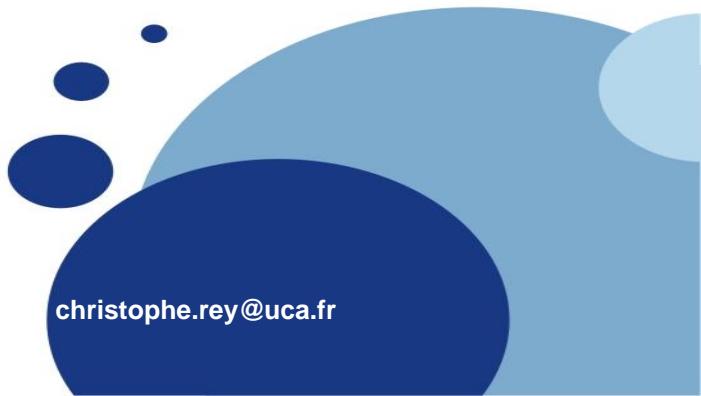
- Supprimer tous les droits qu'il avait sur la table photo et les reporter sur la vue « photo\_fr »

```
REVOKE SELECT ON shuttershop.photo  
FROM julien@localhost;
```

```
GRANT SELECT ON shuttershop.photo_fr  
TO julien@localhost;
```

**MMI Vichy  
S3  
BD2**

# SQL Dynamique



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

# Qu'est-ce que c'est ?

- Possibilité d'aller plus loin que des requêtes déclaratives
  - Un vrai langage de programmation inséré dans MySQL et facilement utilisable avec des requêtes SQL
    - Algorithmique
      - Variables
      - Curseurs (résultats de requêtes)
      - Tests
      - Boucles
      - Gestion des erreurs
    - Programmation
      - Procédures
      - Fonctions
      - Déclencheurs (sorte de programmation événementielle liée aux insertions, modifications et suppression dans des tables)

- Laisser au concepteur/administrateur de la base de données le soin de créer des procédures et fonction adaptées aux données et au schéma.
- Les programmes accédant à la base de données deviennent utilisatrices de ces procédures et fonctions.
- Résultats :
  - Plus grande fiabilité :
    - le concepteur et/ou administrateur connaît la bd mieux que les développeurs des programmes qui l'utilisent.
    - La cohérence de la bd est renforcée.
  - Plus grande sécurité :
    - l'utilisation de la bd est beaucoup mieux encadrée
    - les privilèges et droits d'accès peuvent être plus stricts et finement définis
  - Plus grande utilisabilité de la base
    - Les procédures restent les mêmes que les programmes qui accèdent à la base soient écrits en Java, php, C++, ...
  - Plus grande efficacité de développement
    - Les développeurs d'applications accédant à la bd ne sont plus obligés de connaître la bd très précisément. La bd est presque une API de plus à manipuler.
  - Plus grande efficacité de traitement :
    - Beaucoup moins d'échanges client-serveur et donc meilleures performances réseau

# Procédures stockées

## Procédure stockée

- Ensemble d'instruction SQL dynamique et de requêtes SQL
- Prend des paramètres
- Ne renvoie pas de résultat, mais peut modifier des paramètres
- Syntaxe

```
CREATE PROCEDURE <nom_procédure> ([[<qualif>] <param1> <type1>[,...]])  
[    LANGUAGE SQL  
|    [NOT] DETERMINISTIC  
|    SQL SECURITY {DEFINER | INVOKER}  
|    COMMENT 'string'  
]  
<corps de la procédure>
```

- Syntaxe des blocs BEGIN ... END;

```
[<étiquette>:] BEGIN  
    [<déclaration des variables>]  
    [<déclaration des curseurs>]  
    [<déclaration des handlers>]  
    <instructions>  
END;
```

# Exemple sur la base shuttershop

Procédure qui insère une nouvelle ligne dans la table transaction

- Entrées :
  - les informations sur la carte de crédit
  - le login du client
  - Remarque : le numéro de la transaction est incrémenté automatiquement, la date est obtenue dynamiquement et le montant sera mis à null),
- Sorties :
  - le numéro de la transaction (-1 si erreur)
  - Stockée dans un paramètre
- Instructions :
  - Ajoute une nouvelle ligne à la table transaction
  - Mettre à jour la table buys (ajout d'une ligne)

```
CREATE PROCEDURE create_trans (
    IN p_cc_no varchar(255),
    IN p_cc_type ENUM('V','M','A','D'),
    IN p_cc_expd date,
    OUT p_idn INT,
    IN p_idname varchar(255))
BEGIN
    SET p_idn = -1;

    insert into
    transaction(cc_no,cc_type,cc_expd,trdate)
    values (p_cc_no,p_cc_type,p_cc_expd,now());

    select last_insert_id()
    into p_idn;

    insert into buys(idn, idname)
    values (p_idn, p_idname);

END;
```

# Remarque

- Création d'une procédure stockée sous le client texte mysql
  - ➔ changement de délimiteur de requête :
    - on remplace le ; par un autre symbole
    - Utiliser DELIMITER pour cela
- Exemple :

DELIMITER !!!! /\* Permet de définir une balise de fin de procédure \*/

/\* Prototype de la procédure \*/  
CREATE PROCEDURE *nom\_de\_procedure*  
(*nom\_du\_parametre* TYPE\_VAR)

BEGIN  
/\* Opérations à effectuer \*/  
END; !!!!!

## De type entrée :

- Type par défaut
- Préfixés par IN
- Valeurs nécessaires à la procédure pour effectuer ses traitements

## • De type sortie :

- Préfixés par out
- Variables qui stockent les résultats de la procédure (utilisables par la session qui appelle la procédure)

## • De type entrée et sortie

- Préfixés par INOUT

## • Convention de nommage

- Les paramètres peuvent être préfixés par « p\_ »
- Les variables par « v\_ »

# Caractéristiques optionnelles

## DETERMINISTIC :

- si la routine donne toujours le même résultat pour les mêmes entrées (NOT DETERMINISTIC sinon)
- Utile pour la réPLICATION des données

## SQL SECURITY INVOKER

- Si la routine n'est invocable qu'avec les privilèges de l'utilisateur (ou du créateur si DEFINER à la place de INVOKER)

## LANGUAGE SQL

- Dans le futur, possibilité d'écrire des routines en php (on aura alors LANGUAGE PHP)

## COMMENT

- Pour ajouter un commentaire visible après un SHOW PROCEDURE STATUS

## Par défaut :

NOT DETERMINISTIC, SQL SECURITY DEFINER

# Variables dans une procédure ou fonction

- Les variables locales ne sont pas prefixées par @ ni @@
- Déclarées en début d'un bloc (après un BEGIN) par DECLARE nom\_variable type\_variable;
- L'affectation est réalisée par
  - SET variable = valeur ;
  - SET variable = (SELECT ...);
  - SELECT att1, att2  
INTO variable1, variable2  
FROM ... WHERE ... ;
- Exemples de déclaration de variables :

```
DECLARE v_aujourd'hui
```

```
DATE DEFAULT CURDATE();
```

```
DECLARE v_pi
```

```
DECIMAL(7,5) DEFAULT 3.14116;
```

```
DECLARE v_compteur
```

```
INTEGER NOT NULL DEFAULT 1;
```

```
DECLARE v_nom
```

```
VARCHAR(30);
```

# Autres variables

Variables utilisateurs (hors procédures et fonctions, dans une session)

- Préfixées par @
- Pas besoin d'être déclarées ni explicitement typées
- Exemple :

```
SET @tva = 0,196;  
SELECT objet, prix * (1+@tva)  
FROM Marchandise;
```

- Variables systèmes

- Préfixées par @@
- De session : @@variable ou @@session.variable
- Globales : @@global.variable

# Récapitulatif variables

Variable	Ecriture	Portée
Variable (système) globale	<code>@@global.variable</code>	Valeur par défaut des variables de session des futures sessions
Variable (système) de session	<code>@@variable</code> ou <code>@@session.variable</code>	La session en cours
Variable utilisateur	<code>@variable</code>	
Variable locale	<code>variable</code>	A l'intérieur d'une routine (procédure ou fonction); doit être déclarée en début de routine

# Autres commandes utiles

- Modification d'une procédure  
`ALTER PROCEDURE ...`  
et même syntaxe que `CREATE PROCEDURE`
- Suppression d'une procédure  
`DROP PROCEDURE nom_procédure;`
- Suppression conditionnelle  
`DROP PROCEDURE IF EXISTS nom_procédure;`
- Appel d'une procédure dans une session
  - `CALL nom_procédure(liste_valeurs_et_variables)`
  - Exemple :  
`CALL create_trans('I56','M',2008-03-11,@resultat,'Dupont');`  
`SELECT @resultat;`
  - Les paramètres OUT et INOUT doivent être des variables utilisateurs de la session courante afin de pouvoir sauvegarder et réutiliser les résultats de la procédure.

# Sorties d'une procédure

## 3 sortes

- Via un paramètre fourni par l'appelant et modifié par la procédure (cf exemple précédent)
- Création et remplissage d'une table temporaire par la procédure et consultation de cette table par l'appelant (qui doit connaître le nom de la table)
- En fin de procédure se trouve un SELECT (pas SELECT ... INTO... ) dont le résultat est affiché à l'écran pour l'appelant.

# Fonctions stockées

## Fonction stockée

- Ensemble d'instruction SQL dynamique et de requêtes SQL
- Prend des paramètres
- Renvoie un unique résultat de type scalaire (une valeur)
- Syntaxe

CREATE FUNCTION <nom\_fonction>

(liste des paramètres)

RETURNS type\_valeur\_de\_retour

[ mêmes options que pour les procédures]

<corps de la fonction>

- La dernière instruction du corps doit être :  
RETURN nom\_variable\_de\_type\_scalaire;
- S'utilisent comme les procédures prédéfinies de MySQL (dans un SELECT)

# Exemple

- Fonction qui compte et renvoie le nombre de photos de la table photo
- Code source :

```
DELIMITER |||
CREATE FUNCTION comptePhotos () RETURNS int
BEGIN
    DECLARE v_nbp int;
    select count(*)
        into v_nbp
        from photo;
    RETURN v_nbp;
END |||
DELIMITER ;
```
- Appel par

```
select comptePhotos();
```

# SQL dynamique – Récapitulatif privilèges

Ordre	Effet	Privilège
Create procedure Create function	Crée une procédure stockée ou une fonction	Create routine sur la BD
Call	Exécute une procédure stockée	
Utilisation de la fonction dans une requête	Exécute la fonction qui renvoie son résultat à la requête appelante	Execute, ou être le créateur de la routine

# SQL dynamique – Récapitulatif privilèges

Ordre	Effet	Privilège
Drop Procedure Drop Function	Supprime la routine, provoque une erreur si elle n'existe pas	
Drop Procedure If Exists Drop Function If Exists	Supprime la routine, sans provoquer d'erreur si elle n'existe pas	
Alter Procedure Alter Function	Permet de changer les options de la routine (mais pas son code source)	Alter Routine ou bien être le créateur de la routine
Show Procedure Status Show Function Status	Liste les procédures ou fonctions, avec notamment leur créateur, leur SQL Security et les commentaires	
Show Create Procedure Show Create Function	Affiche le code de création (sans réécriture), ainsi que le @@sql_mode	Pour le code de création, SELECT sur la table de privileges mysql.proc ou bien être le créateur de la routine



# Autres commandes utiles

- Modification d'une fonction  
`ALTER FUNCTION ...`  
et même syntaxe que `CREATE FUNCTION`
- Suppression d'une fonction  
`DROP FUNCTION nom_fonction;`
- Suppression conditionnelle  
`DROP FUNCTION IF EXISTS nom_function;`
- Appel d'une fonction : dans un `SELECT`

# Limites des fonctions

- Pas de transaction dans les fonctions
- Pas d'ordre DDL sauf CREATE TEMPORARY TABLE
- Pas de requête affichant un résultat à l'écran
  - Pas de SELECT (mais les SELECT...INTO ... sont possibles)
  - Pas de SHOW, DESC, EXPLAIN,...
  - Que des paramètres IN (ni OUT, ni INOUT)
- Les mêmes limites s'appliquent aux déclencheurs (cf après).

# Tests

if

- IF condition THEN instructions  
ELSE instructions  
END IF;
- IF condition1 THEN instructions  
ELSEIF condition2 THEN instructions  
ELSEIF ... THEN  
...  
ELSE instructions  
END IF;

• case

- CASE variable  
WHEN valeur1 THEN instructions  
WHEN valeur2 THEN instructions  
...  
ELSE instructions  
END CASE;
- CASE  
WHEN condition1 THEN instructions  
WHEN condition2 THEN instructions  
...  
ELSE instructions  
END CASE;

# Boucles

while

- WHILE condition DO  
instructions  
END WHILE;
- repeat
  - REPEAT  
instructions  
UNTIL condition  
END REPEAT;
- loop, leave et iterate :
  - A EVITER ABSOLUMENT !!!
  - Basée sur les branchements et les étiquettes
  - Programmation « sale »
  - TOUT peut être fait proprement avec while et repeat.

# Gestionnaire d'erreurs (handler)

- Si une erreur survient lors d'une requête

- MySQL renvoie un code différent suivant le type d'erreur
- Résultat de SELECT vide (alors qu'on en attendait un) ou multiple (alors qu'on n'en attendait qu'un seul)

- Déclaration en début de procédure d'un handler

- Permet d'intercepter les erreurs (d'un SELECT ou d'un FETCH par ex.)
- Permet de faire se terminer la procédure proprement

- `DECLARE type_handler HANDLER`

- `FOR type_ou_numero_erreur`

- `BEGIN`

- `instructions`

- `END;`

- 2 type\_handler possibles

- EXIT : sort de la procédure après exécution des instructions du handler

- CONTINUE : continue l'exécution de la procédure après exécution des instructions du handler

# Types d'erreurs

Chaque erreur est décrite

- Par un numéro spécifique
- Par un statut : chaîne de caractères normalisée précédée de SQLState
- Par un des 3 mots clés :
  - SQLWarning
  - Not Found
  - SQLException
- Par une condition nommée :  
`DECLARE 'LIGNE ABSENTE' CONDITION FOR SQLState '02000';`
- Voir la documentation officielle pour les correspondance entre numéros et le SQLState correspondant.
- ATTENTION : toute requête SQL ne ramenant pas de ligne déclenche le handler → faire attention à bien positionner le handler

# Curseurs dans procédures

- Si un SELECT renvoie plusieurs lignes, alors on traite le résultat ligne à ligne par un curseur.
- Utilisation des curseurs :
  - Déclaration du curseur
  - Ouverture du curseur
  - Exécution du FETCH tant que le curseur ramène des lignes
  - Fermeture du curseur
- Gestionnaire d'arrêt pour détecter que le curseur ne ramène plus de ligne
- Limites des curseurs MySQL
  - Read-only : ne permettent pas la modification des données
  - Non-scrolling : vont de la première à la dernière ligne sans possibilité de revenir en arrière
  - Asensitive : si des modifications ont lieu sur les données en cours de lecture par le curseur, celles-ci ne sont pas transmises au curseur

# Curseurs : schéma d'utilisation

## Avant le BEGIN

- DECLARE nom curseur CURSOR FOR  
SELECT ... FROM ... WHERE ... ;
- DECLARE CONTINUE HANDLER  
FOR NOT FOUND  
SET v\_fin curseur = TRUE;
- BEGIN
  - ...
  - OPEN nom curseur;
  - FETCH nom curseur INTO variable;
  - WHILE v\_fin curseur = FALSE DO  
instructions avec variable  
FETCH nom curseur INTO variable;
  - END WHILE;
  - CLOSE nom curseur;

# Triggers (déclencheurs) - Généralités

- Qu'est-ce qu'un trigger?  
Un déclencheur est une procédure associé à une table, qui s'active automatiquement lorsqu'un événement particulier survient.
- Intérêts
  - Gestion des redondances
  - Enregistrements automatique de certains événements
  - Spécification de contraintes complexes (des prix qui ne peuvent qu'augmenter,...)
  - Toute règle liée à l'environnement d'exécution (restrictions sur les utilisateurs,...)
- Quelle syntaxe? (privilège nécessaire)

Pour créer un trigger (Trigger, Super) :

```
CREATE TRIGGER trigger_nom trigger_temps trigger_action  
ON table_nom FOR EACH ROW trigger_cmd;
```

Pour effacer un trigger (Trigger, Super) :

```
DROP TRIGGER table_nom.trigger_nom;
```

Pour afficher la liste des triggers (accès à la base) :

```
SHOW TRIGGERS
```

```
SHOW TRIGGER LIKE 'table_nom';
```

# Triggers - Construction

**CREATE TRIGGER** *trigger\_nom trigger\_temps trigger\_action*  
**ON** *table\_nom* **FOR EACH ROW** *trigger\_cmd*

- *Trigger\_nom* : est le nom de votre déclencheur.
- *Trigger\_temps* : **BEFORE** ou **AFTER** : indique au déclencheur quand il doit s'activer p/r à l'événement qui le déclenche.
- *Trigger\_action* : **INSERT**, **UPDATE** ou **DELETE** : indique le type d'événement qui active le déclencheur.
- **FOR EACH ROW** : rappelle que le trigger agira pour chaque ligne affectée par l'événement.
- *Trigger\_cmd* : est la commande à exécuter lorsque le déclencheur s'active.

# Triggers – Pseudo-tables

Pseudo-tables disponibles selon l'événement.

	INSERT	UPDATE	DELETE
BEFORE	<b>New</b> [données qui vont être insérées]	<b>New</b> [données après modification] <b>Old</b> [données avant modification]	<b>Old</b> [données qui vont être supprimées]
AFTER	<b>New</b> [données qui viennent d'être insérées]		<b>Old</b> [données qui viennent d'être supprimées]

Attention : Old et New ne sont pas de vraies tables.

# Triggers - Remarques

- Conditions d'utilisation :
  - Il doit faire référence à une table permanente, et pas une table TEMPORARY ou une Vue (View).
  - Il ne peut pas y avoir deux déclencheurs pour une même table avec les mêmes configuration de moment(trigger\_temps) et de commande(trigger\_cmd).
- Induit un comportement dynamique de la bd
  - Permet des instructions générées automatiquement en cascade
  - Attention aux boucles infinies
- Les triggers ont leur propres mécanismes d'atomicité
  - À un niveau plus fin que ceux des transactions
  - → pas d'interférence avec les transactions

# Triggers - Exemple

- On veut créer un déclencheur, pour qu'une procédure « AchatPhotographie » s'exécute automatiquement avant une nouvelle insertion dans la table Transaction.
- ```
CREATE TRIGGER transaction_insert
BEFORE INSERT
ON Transaction
FOR EACH ROW
CALL AchatPhotographie(New.*);
```

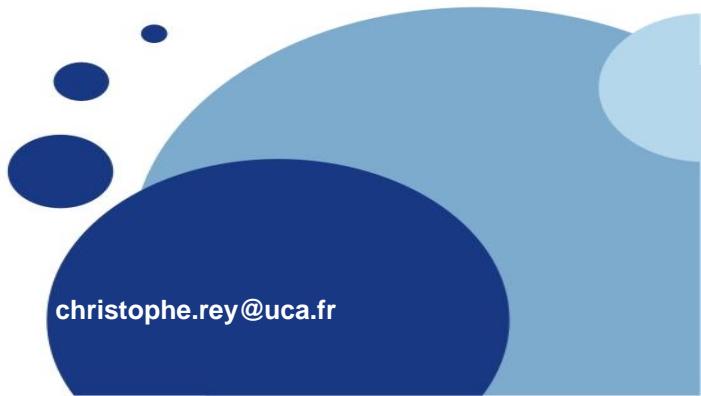
# Triggers - Exercices

- Créez un trigger qui à l'insertion d'une nouvelle photo appelle la procédure « verif\_photo\_insert » qui vérifie que la photo est bien disponible, et si ce n'est pas le cas, la rend disponible.

```
CREATE TRIGGER photo_insert
  BEFORE INSERT
  ON Photo
  FOR EACH ROW
  CALL verif_photo_insert();
```

**MMI Vichy  
S3  
BD2**

# **Les transactions**



**christophe.rey@uca.fr**

# Les différents moteurs de MySQL

Des noms de moteurs: MyISAM, Memory, InnoDB, BerkleyDB...

- **Intérêt:**

Meilleur performance, adapter aux besoins de l'utilisateur.

- **Raison:**

Certains moteurs s'adaptent mieux que d'autres à différentes tâches. Il est très facile de changer de moteur.

- **Exemple d'utilisation:**

Lecture plus rapide des résultats, optimisation de l'utilisation de l'espace disponible, sécurité accrue, écriture simultanées dans les tables...

- **Choisir un moteur:**

« CREATE TABLE test ( id integer not null primary key )  
ENGINE=innodb ; »

- **Changer de moteur pour une table:**

« ALTER TABLE test CHANGE TYPE=InnoDB »

## Généralités

- Evolution du moteur ISAM, Hérite de ses propriétés avec des extensions supplémentaires

- Moteur par défaut de MySQL

- Orienté application Web

## Avantages

- Verrouillage de table manuellement

- Simple d'utilisation et rapide

- Utilisation du FullText: recherche multicritère (exemple)

- Gain de place sur les disques et sur la mémoire lors des MAJ

## Inconvénients

- Ne supporte pas les transactions et les clés étrangères

- Obligation d'effectuer des opérations de maintenance

## Généralités

- Moteur plus perfectionné que MyISAM adapté à des modèles entités-associations plus complexes.
- Développé à partir de l'API MySQL++

## Avantages

- Gestion des transactions
- Possibilité retour en arrière
- Reprise après panne plus efficace
- Permet un verrouillage au niveau des enregistrements
- Gestion des clés étrangères
- Mise à jours et suppression en cascade (triggers)

## Inconvénients

- Affichage plus lent que MyISAM car plus de contraintes à vérifier
- Gestion mémoire plus coûteuse

# Les transactions (avec InnoDB)

- **Une transaction = ensemble de requêtes**
- **Une transaction permet d'assurer la bonne exécution d'une requête SQL (sans interférence avec d'autres transactions ayant lieu en même temps sur les mêmes données).**
- **Processus de gestion de requêtes ACID**
  - Atomicité: toutes les requêtes sont validées ou aucune
  - Cohérence : les données restent cohérentes
  - Isolation : indépendance p/r autres requêtes
  - Durabilité : leurs effets persistent
- **Moteur « Innodb » uniquement**
- **Mots clés :**
  - **START TRANSACTION** : début d'une transaction
  - **COMMIT** : validation d'une transaction (on ne peut plus annuler les manipulations effectuées après validation)
  - **ROLLBACK** : annulation des manipulations de la transaction

# Les transactions (avec InnoDB)

## • Une transaction = ensemble de requêtes

Buts des transactions en résumé :

- Donner l'impression à l'utilisateur du serveur qu'il est seul à accéder aux données
- Donner la possibilité d'annuler les mises à jour effectuées pour revenir à un état précédent cohérent (sans erreur). Sorte de Undo (ou Ctrl+Z)

- COMMIT : validation d'une transaction (on ne peut plus annuler les manipulations effectuées après validation)
- ROLLBACK : annulation des manipulation de la transaction

# Mode autocommit

- Mode de fonctionnement par défaut de toute session MySQL :
  - Toute requête est automatiquement validée.
  - Fonctionnement dû à la variable @@autocommit valant 1 (vrai) par défaut
  - Si on la met à 0, alors il faut valider les transactions par COMMIT.
  - Récapitulatif : modes d'ouverture et de fermeture des transactions

| Instruction                                    | Avec AUTOCOMMIT                                         | Sans AUTOCOMMIT                                                               |
|------------------------------------------------|---------------------------------------------------------|-------------------------------------------------------------------------------|
| Start transaction;<br>Begin;<br>Begin work;    | Ouvre une transaction                                   | Valide les précédentes requêtes puis ouvre une transaction                    |
| Start transaction with<br>consistent snapshot; | Ouvre une transaction et prend un instantané (snapshot) | Valide les précédentes requêtes, ouvre une transaction et prend un instantané |

# Mode autocommit

## Récapitulatif (suite)

| Instruction                                                     | Avec AUTOCOMMIT                                                                               | Sans AUTOCOMMIT                                                                                                   |
|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| N'importe quelle requête;                                       | Si une transaction est ouverte, la requête en fait partie, sinon elle est exécutée et validée | Si une transaction est ouverte, la requête en fait partie, sinon elle en ouvre une nouvelle dont elle fait partie |
| Commit;                                                         | Valide et ferme la transaction                                                                |                                                                                                                   |
| Une requête DDL ou autre ordre causant une validation implicite | Valide et ferme la transaction                                                                |                                                                                                                   |
| Rollback;                                                       | Annule et ferme la transaction                                                                |                                                                                                                   |
| Commit and chain;                                               | Valide la transaction puis en ouvre une nouvelle                                              |                                                                                                                   |
| Rollback and chain;                                             | Annule la transaction puis en ouvre une nouvelle                                              |                                                                                                                   |
| Commit release;                                                 | Valide la transaction puis ferme la session                                                   |                                                                                                                   |
| Rollback release;                                               | Annule la transaction puis ferme la session                                                   |                                                                                                                   |

# Les verrous

- Le verrouillage consiste à « bloquer » ou non un (ou plusieurs) enregistrement(s) lors d'une transaction.
- Ceci permet d'éviter certains problèmes de ralentissement et de conserver une base de données cohérente.
- Type de verrous : partagés ou exclusifs
- Principes d'utilisation
  - Plusieurs verrous partagés (appartenant à différents utilisateurs) peuvent être mis sur les mêmes lignes
  - Un seul verrou exclusif (tous utilisateurs confondus) peut être mis sur un groupe de lignes d'une table.
- Des verrous sont mis
  - Au cours d'un INSERT, UPDATE ou DELETE automatiquement (de type exclusif)
  - Au cours d'un SELECT par l'utilisateur si celui-ci a ajouté à la fin du SELECT :
    - Lock in Share Mode :
      - verrou partagé, autorise la lecture mais interdit les modifs pour les autres sessions
      - Exemple : `SELECT * FROM photo LOCK IN SHARE MODE;`
    - For Update :
      - verrou exclusif, interdit lecture et modifs pour les autres sessions
      - Exemple : `SELECT * FROM photo WHERE id='2' FOR UPDATE;`

# Utilisation des verrous

## Des verrous sont mis :

- Au cours d'un **INSERT, UPDATE ou DELETE** automatiquement (de type exclusif)
- Au cours d'un **SELECT** par l'utilisateur si celui-ci a ajouté à la fin du **SELECT** :
  - **Lock in Share Mode :**
    - verrou partagé pour la lecture
    - Exemple : **SELECT \* FROM photo LOCK IN SHARE MODE;**
  - **For Update :**
    - verrou exclusif
    - Exemple : **SELECT \* FROM photo WHERE id='2' FOR UPDATE;**

## Les verrous sont levés :

- à la fin de la transaction durant laquelle ils ont été mis (qu'elle soit validée ou annulée).

## Interblocage (deadlock)

- Quand 2 utilisateurs différents ont mis un verrou partagé sur les mêmes lignes
- Et quand ils veulent chacun transformer ce verrou en verrou exclusif
- Chacun attend que l'autre enlève son verrou partagé pour placer son verrou exclusif → situation qui ne bouge plus.
- Situation détectée par MySQL et résolue après un temps d'attente.

# Verrous récapitulatif

| Instruction                    | Verrou   | Portée du verrou              | Commentaire                                                                                              |
|--------------------------------|----------|-------------------------------|----------------------------------------------------------------------------------------------------------|
| Select                         | Aucun    |                               | Sauf avec le niveau d'isolation serializable qui ajoute automatiquement la clause « Lock in share mode » |
| Select ... lock in share mode; | Partagé  | Lignes lues et futures lignes |                                                                                                          |
| Select... for update;          | Exclusif | Lignes lues et futures lignes |                                                                                                          |

# Verrous récapitulatif

| Instruction  | Verrou   | Portée du verrou                    | Commentaire                                                                                                                  |
|--------------|----------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Insert ... ; | Exclusif | Lignes insérées                     | Si une contrainte de clé étrangère est mise en jeu, un verrou partagé est posé sur les lignes lues dans la table référencée. |
| Update ... ; |          | Lignes modifiées et futures lignes  |                                                                                                                              |
| Delete ... ; |          | Lignes supprimées et futures lignes |                                                                                                                              |

# L'isolation

• **Intérêt :** *Empêcher des phénomènes indésirables dus à des interférences entre utilisateurs accédant en même temps aux mêmes données.*

- **Lecture sale** : Une transaction lit des données écrites par une transaction concurrente non validée.
- **Lecture non reproductible**: Une transaction relit des données qu'elle a lu précédemment et trouve que les données ont été modifiées par une autre transaction (validée depuis la lecture initiale) : la lecture n'est pas cohérente entre le début et la fin de la transaction.
- **Lecture fantôme**: Si on est dans le cas où une transaction possède la propriété de lecture cohérente, alors quel que soit le moment dans la transaction, la même requête produit les mêmes résultats. Et ce même si les données ont été modifiées ou supprimées entre temps. C'est ce qu'on appelle la lecture de données dites fantômes.

# Les niveaux d'isolation

Il existe quatre niveaux d'isolation permettant d'empêcher ces phénomènes de se produire :  
**variables @@tx\_isolation @@global.tx\_isolation**

| Niveau d'isolation | Lecture sale | Lecture non reproductible | Lecture fantôme |
|--------------------|--------------|---------------------------|-----------------|
| Read Uncommitted   | Possible     | Possible                  | Possible        |
| Read Committed     | Impossible   | Possible                  | Possible        |
| Repeatable Read    | Impossible   | Impossible                | Possible        |
| Serializable       | Impossible   | Impossible                | Impossible      |

- Attention :
  - le niveau **SERIALIZABLE** n'empêche pas les interblocages (deadlocks)  
Possibilité de les éviter par pose de verrous exclusifs lors des SELECT en ajoutant FOR UPDATE à la fin.

# Récapitulatif niveaux d'isolation

## Résumé

| Niveau d'isolation   | Description                                                                                                                 | Type d'erreur évité                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| 0 - Read Uncommitted | Pas d'isolation : chaque session voit les modifications des autres sessions même si elles ne sont pas validées              | Aucun                                                                       |
| 1 - Read Committed   | Isolation de bas : les modifications non validées ne sont visibles que dans leur session                                    | Lecture sale (dirty read)                                                   |
| 2 - Repeatable Read  | Niveau d'isolation par défaut : le précédent plus l'imposition de la lecture cohérente (consistent read)                    | Lecture sale et lecture non reproductible (unrepeatable read)               |
| 3 - Serializable     | Comme le précédent plus l'application des transactions les unes après les autres (quand elles concernent les mêmes données) | Lecture sale et lecture non reproductible et lecture fantôme (phantom read) |

- Modifier le niveau d'isolation :

```
SET GLOBAL TRANSACTION ISOLATION LEVEL niveau;
SET SESSION TRANSACTION ISOLATION LEVEL niveau;
```

niveau est le nom du niveau en lettres

# Cas pratique : Shuttershop

Problèmes engendrés par les différents niveaux d'isolation :

- Serializable
  - Interblocage
- Repeatable Read
  - La même image peut-être achetée 2 fois
- Read Committed
  - La même image peut-être achetée 2 fois
- Read Uncommitted
  - Une photo disponible peut-être vu comme non disponible
- Regardons différents scénarios d'achat...

# Shuttershop – Scénario d'achat

| <i>Instant :</i> <b><i>En mode Read Uncommitted</i></b> |                                                       |                                               |
|---------------------------------------------------------|-------------------------------------------------------|-----------------------------------------------|
|                                                         | Utilisateur A                                         | Utilisateur B                                 |
| t1                                                      | Ouvre une transaction et achète une photo             |                                               |
| t2                                                      |                                                       | Ouvre une transaction et achète la même photo |
| t3                                                      | A voit l'achat de B, même si il n'a pas été validé... |                                               |
| t4                                                      |                                                       | B voit l'achat de A et annule sa transaction  |
| t5                                                      | ...A annule sa transaction                            |                                               |
| Résultat : la photo n'a pas été achetée                 |                                                       |                                               |

# Shuttershop – Scénario d'achat

| <i>Instant :</i>                       | <i>En mode Read Committed</i>                        |                                                                                                                  |
|----------------------------------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
|                                        | Utilisateur A                                        | Utilisateur B                                                                                                    |
| t1                                     | Ouvre une transaction et achète une photo            |                                                                                                                  |
| t2                                     |                                                      | Ouvre une transaction et achète la même photo                                                                    |
| t3                                     | A ne voit pas l'achat de B car il n'a pas été validé |                                                                                                                  |
| t4                                     |                                                      | A n'a pas encore validé sa transaction donc B est autorisé à acheter la photo<br>-> B valide l'achat de la photo |
| t5                                     | A valide sa transaction                              |                                                                                                                  |
| Résultat : la photo est achetée 2 fois |                                                      |                                                                                                                  |

# Shuttershop – Scénario d'achat

| <i>Instant :</i> | <i>En mode Repeatable Read (avec instantané)</i>                                          |                                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Utilisateur A                                                                             | Utilisateur B                                                                                                                                 |
| t1               | Ouvre une transaction et achète une photo                                                 |                                                                                                                                               |
| t2               |                                                                                           | Ouvre une transaction et achète la même photo (car la transaction de A n'étant pas validée n'apparaît pas à B quand il ouvre sa transaction). |
| t3               | A ne voit pas l'achat de B car il succède à celui de A<br>-> A valide l'achat de la photo |                                                                                                                                               |
| t4               |                                                                                           | A l'instant t2, A n'avait pas validé sa transaction donc B est autorisé à acheter la photo<br>-> B valide l'achat de la photo                 |

Résultat : la photo est achetée 2 fois

|                                               | Utilisateur A                                                                                                                                                                                                                                                             | Utilisateur B                                                                                                                                  |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| t1                                            | Ouvre une transaction et veut acheter une photo. A la lecture de la table, il pose un verrou (partagé) sur l'enregistrement concerné et verrouille une deuxième fois (avec un verrou exclusif) les achats (au moment de la modification de la disponibilité de la photo). |                                                                                                                                                |
| t2                                            |                                                                                                                                                                                                                                                                           | Ouvre une transaction et essaie d'acheter la même photo. Le verrou l'empêche d'aller plus loin. Il doit attendre la fin de la transaction de A |
| t3                                            | A valide sa transaction                                                                                                                                                                                                                                                   |                                                                                                                                                |
| t4                                            |                                                                                                                                                                                                                                                                           | B est autorisé à « essayer d'acheter » la photo. Elle n'est plus disponible<br>-> B annule l'achat de la photo                                 |
| t5                                            |                                                                                                                                                                                                                                                                           |                                                                                                                                                |
| Résultat : la photo n'est achetée qu'une fois |                                                                                                                                                                                                                                                                           |                                                                                                                                                |

# Shuttershop – Scénario d'achat

**Instant :**

***En mode Serializable***

|    | Utilisateur A                                                                                                                                                                                                                                                      | Utilisateur B                                                                                                                                                              |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t1 | Ouvre une transaction et veut acheter une photo. A la lecture de la table, il pose un verrou (partagé) sur l'enregistrement concerné                                                                                                                               |                                                                                                                                                                            |
| t2 |                                                                                                                                                                                                                                                                    | Ouvre une transaction et veut acheter une photo. A la lecture de la table, il pose un deuxième verrou (partagé) sur l'enregistrement concerné (le premier est celui de A). |
| t3 | A essaie de verrouiller une deuxième fois (avec un verrou exclusif) les achats (au moment de la modification de la disponibilité de la photo). Il n'y arrive pas car B a aussi un verrou partagé sur les mêmes données. Il attend que B enlève son verrou partagé. |                                                                                                                                                                            |
| t4 |                                                                                                                                                                                                                                                                    | B est dans la même situation que A.                                                                                                                                        |
| t5 | MySQL détecte l'interblocage et décide d'annuler la transaction de B qui est le dernier à avoir entamé sa transaction.                                                                                                                                             |                                                                                                                                                                            |
| t6 | A valide sa transaction                                                                                                                                                                                                                                            |                                                                                                                                                                            |

Résultat : la photo n'est achetée qu'une fois

# Shuttershop – Scénario d'achat

| <i>Instant :</i> | <i>En mode Serializable avec FOR UPDATE ajouté en fin des SELECT</i>                                                                                    |                                                                                                                                                                                                                                                                    |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Utilisateur A                                                                                                                                           | Utilisateur B                                                                                                                                                                                                                                                      |
| t1               | Ouvre une transaction et veut acheter une photo. A la lecture de la table, il pose un verrou (exclusif) sur l'enregistrement concerné                   |                                                                                                                                                                                                                                                                    |
| t2               |                                                                                                                                                         | Ouvre une transaction et veut acheter une photo. A la lecture de la table, il veut poser un deuxième verrou (exclusif) sur l'enregistrement concerné (le premier est celui de A). Il n'y arrive pas à cause du verrou de A. Il attend que A ait enlevé son verrou. |
| t3               | A peut faire ses achats puisqu'il a le verrou exclusif. Il modifie donc la disponibilité de la photo et valide sa transaction (son verrou est détruit). |                                                                                                                                                                                                                                                                    |
| t4               |                                                                                                                                                         | B peut continuer sa transaction : il se rend compte que la photo n'est plus disponible et il annule sa transaction.                                                                                                                                                |

Résultat : la photo n'est achetée qu'une fois

# Exercice : Transaction sur Shuttershop

- Exercice : Réaliser une transaction d'achat d'une photo par un client
- Information dans le cas où le client 1 achète la photo 2 :
  - **cc\_no = 706565794**
  - **cc\_type = D**
  - **cc\_expd = 2009-10-08**
  - **trdate = 2008-03-25**
  - **amount = 125\$**
- 1/ Quelles requêtes sont à exécuter ?
- 2/ Ecrire et tester la transaction sous Toad.

# Exercice : Transaction sur Shuttershop

Exemple de la requête de la transaction :

```
SET @@autocommit = 0;
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

```
START TRANSACTION;
```

```
USE shuttershop;
```

```
UPDATE photo SET available='N' WHERE available='Y' AND catalogn='2';
```

```
INSERT INTO transaction (idn, cc_no, cc_type, cc_expd, trdate, amount) SELECT  
MAX(idn)+1,'706565794', 'D', '2009-10-08', '2008-03-25', '125' FROM transaction;
```

```
INSERT INTO buys (idn,idname) SELECT idn, '1' from transaction where idn in  
(SELECT idn FROM transaction WHERE cc_no='706565794' AND cc_type='D' AND  
cc_expd='2009-10-08' AND trdate='2008-03-25' AND amount = '125');
```

```
INSERT INTO includes (catalogn, idn) SELECT '2',idn from transaction where idn in  
(SELECT idn FROM transaction WHERE cc_no='706565794' AND cc_type='D' AND  
cc_expd='2009-10-08' AND trdate='2008-03-25' AND amount = '125');
```

# Pour aller plus loin...

**Avec le mode Serializable, on peut garantir la cohérence de la base de données ainsi qu'un déroulement logique des transactions au sein du SGBD.**

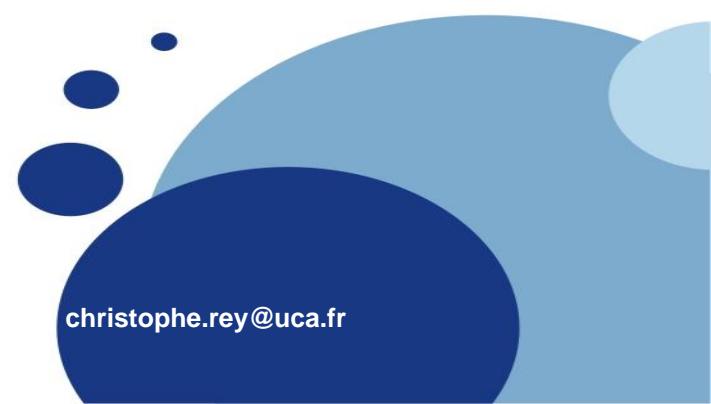
**Cependant, on pourrait se demander pourquoi ce niveau n'est pas utilisé comme niveau d'isolation par défaut dans MySQL.**

**A votre avis, pourquoi ?**

**Car trop contraignant et pas adapté à tous les cas (seuls les bd subissant de nombreuses transactions sur les mêmes données ont un intérêt à un niveau aussi élevé).**

**MMI Vichy  
S3  
BD2**

# **Retour sur l'étude de cas**



**christophe.rey@uca.fr**

## Opérateurs (calcul et comparaison)

- Donner les auteurs nés entre 1921 et 1969
- Donner les noms des modèles commençant par M
- Donner l'auteur le plus jeune de la base
- Donner le prix total des photos de l'auteur le plus jeune
- Donner le prix total TTC de chaque photo (multiplier le prix de chacune par 1.196)
- Donner les lieux qui ont une description
- Fonctions (mathématiques et textuelles)
  - Donner le prix TTC des photos en arrondissant à 1 chiffre après la virgule
  - Donner les nom, prénom et date de naissance de chaque auteur sous la forme « Nom : \_\_\_\_\_, Prénom : \_\_\_\_\_, né le \_\_\_\_\_ »
  - Donner les noms des photos en majuscules

## Fonctions (date)

- Donner la date courante
- Donner l'heure courante
- Donner la date et l'heure courante
- Transformer la date courante en le nombre de secondes depuis le 1<sup>er</sup> janvier 1970
- Transformer un nombre de secondes depuis 1<sup>er</sup> janvier 1970 en date
- Donner le jour de la semaine du 8 mai 1945
- Extraire l'année de la date 2006-12-04
- Nombre de jours entre le 8 mai 1945 et le 25 mars 2008

## • Fonctions (générales)

- Donner le nom et la machine de l'utilisateur
- Donner le nom et la machine donnés par l'utilisateur lors de sa connection
- Donner le dernier id généré par auto incrémentation
- Donner le nombre de lignes modifiées par la dernière requête

## Regroupement – Agrégation

- Donner le nombre d'auteurs par nationalité
- Compter le nombre de lignes de la table abstract
- Donner le prix moyen des photos disponibles et celui des photos indisponibles
- Donner le montant total des achats pour chaque client
- Donner la date du premier achat pour chaque client
- Donner le nom du client ayant acheté la photo la plus chère
- Modification
  - Insérer une nouvelle photo dans la base
  - Supprimer les clients n'ayant aucun rien acheté
  - Modifier le nom du client ayant acheté la photo n° 3
- Vues
  - Créer une vue affichant les photos et leurs auteurs
  - Créer une vue affichant toutes les photos avec leur type (abstraite, portrait, paysage)

## Caractéristiques attendues du site de vente de photos

- Quelques centaines de clients enregistrés
- Maximum 50 clients en accès simultané
- Quelques milliers de photos
- Durée de vie planifiée du serveur : 4 ans
- Disponibilité du serveur de bases de données : totale mais avec possibilité de le fermer quelques heures dans la semaine
- Politique de sauvegarde
  - Hebdomadaire, fermée, binaire et textuelle
  - Journalière (la nuit), ouverte, incrémentale (les journaux) avec verrouillage des tables
- Dimensionnement
  - Serveur dédié
    - Processeur mono core suffisant
    - 2 disques en RAID 100 Go
  - Media de sauvegarde
    - 1 disque 500 Go pour les fichiers, 1 disque de 250 Go pour les journaux
    - Archivage sur DVD

# Exercice

- Définissez la politique de sauvegarde de votre projet
- Dimensionner le serveur de votre projet
- Faire un script de sauvegarde des fichiers binaires (tables et journaux)
- Faire un script de sauvegarde textuelle
- Faire un script de restauration
- Tester les scripts

# Exercice - utilisateurs

PAS d'utilisateur anonyme :

- Révoquer tous les droits des utilisateurs anonymes
- Révoquer tous les droits des utilisateurs ne précisant pas l'hôte
- Détruire l'utilisateur anonyme "@%"
- Utilisateur « interface de gestion de la base ShutterShop » :
  - Nom : admin\_interface
  - Pour le gestionnaire
  - Base ShutterShop
  - Toutes tables
  - Consultation, modification, ajout, suppression
  - A le droit de créer un utilisateur ayant tous les privilèges du gestionnaire sauf celui de créer d'autres gestionnaires

- Utilisateur « cust\_interface »

- Droit de visualiser tout le contenu des tables Model, Author, Location, Influences, Models, Takes, LocatedIn  
➔ GRANT SELECT sur ces tables
- Droit de visualiser les photos disponibles et les informations associées  
➔ GRANT SELECT sur des vues des tables Photo, Portrait, Abstract, Landscape
- Droit de consultation des tables Includes, Transaction, Buys, Customer  
limitée aux données concernant l'utilisateur courant  
➔ création d'une procédure prenant en paramètre le nom de l'utilisateur, et GRANT EXECUTE sur la base shuttershop
- Droit d'insérer et de modifier des données de client dans la table Customer
- Droit d'insérer des données de transaction dans les tables Includes, Transaction, Buys mais avec des contrôles (cf scenario achat photo)  
➔ création d'une procédure
- Exercice : droit d'enlever une photo d'une transaction et de supprimer une transaction  
➔ création de 2 procédures avec un trigger pour mise en cohérence

# Création et droits de cust\_interface

```
GRANT USAGE ON *.* TO 'cust_interface' @'localhost' IDENTIFIED  
BY PASSWORD '*E237E836449E591611B289F9B19B1BDF152B5E37';
```

```
GRANT EXECUTE ON `shuttershop`.*
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`model`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`author`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`takes`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`locatedin`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`models`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`influences`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT SELECT ON `shuttershop`.`location`
```

```
    TO 'cust_interface' @'localhost';
```

```
GRANT INSERT, UPDATE ON `shuttershop`.`customer`  
    TO 'cust_interface' @'localhost';
```

# Vues et droits pour cust\_interface

```
CREATE VIEW av_photo AS
SELECT * FROM photo where available = 'Y';
```

```
CREATE VIEW av_abstract AS
SELECT a.* FROM photo as p, abstract as a
WHERE (p.catalogn = a.catalogn) and (p.available='Y');
```

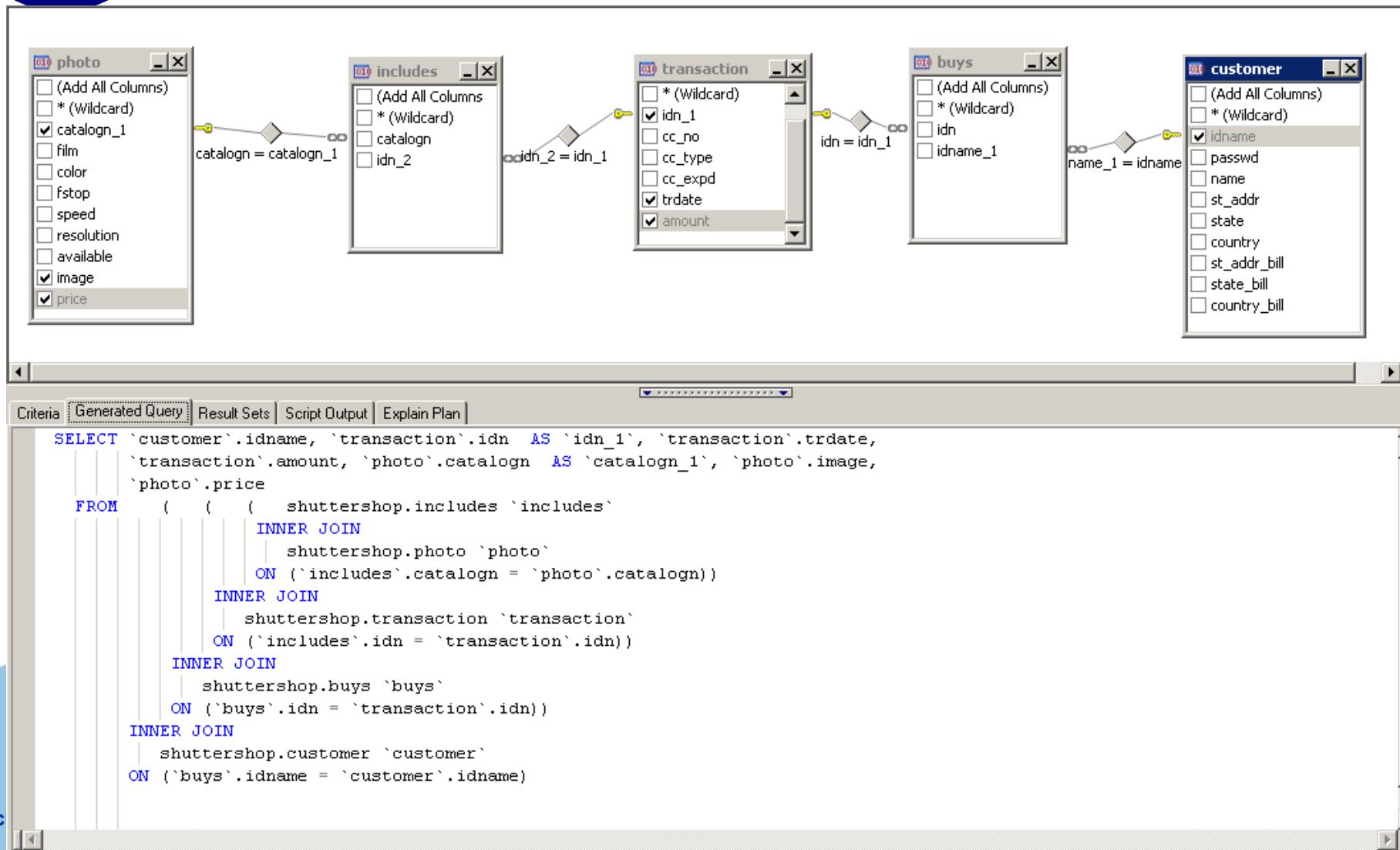
```
CREATE VIEW av_landscape AS
SELECT l.* FROM photo as p, landscape as l
WHERE (p.catalogn = l.catalogn) and (p.available='Y');
```

```
CREATE VIEW av_portrait AS
SELECT pr.* FROM photo as p, portrait as pr
WHERE (p.catalogn = pr.catalogn) and (p.available='Y');
```

```
GRANT SELECT ON `shuttershop`.`av_photo`
TO 'cust_interface' @'localhost';
GRANT SELECT ON `shuttershop`.`av_abstract`
TO 'cust_interface' @'localhost';
GRANT SELECT ON `shuttershop`.`av_landscape`
TO 'cust_interface' @'localhost';
GRANT SELECT ON `shuttershop`.`av_portrait`
TO 'cust_interface' @'localhost';
```

# Droit de voir ses transactions 1/2

Création de la requête (avec TOAD) :



# Droit de voir ses transactions 2/2

Transformation en procédure :

```
CREATE PROCEDURE consult_trans (p_cust_login varchar(255))
BEGIN
    SELECT `customer`.idname as 'Nom client',
           `transaction`.idn as `Numéro transaction`,
           `transaction`.trdate as 'Date',
           `transaction`.amount as 'Montant',
           `photo`.catalogn as `Numéro photo`, *
           `photo`.image as 'Nom photo',
           `photo`.price as 'Prix photo'
      FROM (((shuttershop.includes `includes` INNER JOIN shuttershop.photo `photo`
              ON (`includes`.catalogn = `photo`.catalogn))
              INNER JOIN shuttershop.transaction `transaction`
              ON (`includes`.idn = `transaction`.idn))
              INNER JOIN shuttershop.buys `buys`
              ON (`buys`.idn = `transaction`.idn))
              INNER JOIN shuttershop.customer `customer`
              ON (`buys`.idname = `customer`.idname)
 WHERE `customer`.idname = p_cust_login;
END;
```

# Scenario achat photo 1/2

- Point de départ : le client a trouvé (par ex.) 3 numéros de photos qu'il souhaite acquérir (et qui étaient disponibles au moment où il a consulté le catalogue).
- Etape 1 : si le client n'est pas enregistré, alors il doit le faire  
→ appel par l'interface client d'une procédure reg\_verif()
  - Entrées : un login de client, son mot de passe
  - Sorties : 0 (erreur : client existant mais mauvais mot de passe), 1 (validation : client existant et bon mot de passe), 2 (client inexistant)
  - Instructions :
    - Vérification que l'utilisateur n'existe pas déjà
    - Si non, retourner 2
    - Si oui, vérification que son mot de passe est le bon
      - Si non : fin, retourner 0
      - Si oui : fin, retourner 1
- Etape 2 : Créer d'une nouvelle transaction  
→ appel par l'interface client d'une procédure create\_trans()
  - Entrées : les informations sur la carte de crédit (le numéro de la transaction est incrémenté automatiquement, la date est obtenue dynamiquement et le montant sera mis à null), le login du client
  - Sorties : le numéro de la transaction (-1 si erreur)
  - Instructions :
    - Ajoute une nouvelle ligne à la table transaction
    - Mettre à jour la table buys (ajout d'une ligne)

# Scenario achat photo 2/2

- Etape 3 : pour chaque numéro de photo choisi (par ex. stocké dans un tableau php), ajouter la photo à la transaction  
→ appel par l'interface client d'une procédure add\_photo()
  - Entrées : le numéro de la transaction et celui de la photo
  - Sorties : message (erreur ou validation)
  - Instructions :
    - Vérification que la photo est toujours disponible
      - Si oui on continue, mais on posant un verrou exclusif sur la ligne de la table photo
      - Si non on sort de la procédure avec un message d'erreur type « la photo n° 345 n'est plus disponible »
    - Ajout d'une nouvelle ligne à la table includes
    - Mise à jour du montant de la transaction
    - Mise à jour de la disponibilité de la photo
    - Faire tout cela au sein d'une transaction MySQL (niveau d'isolation par défaut Repeatable\_read)
  - S'il veut par la suite acheter une autre photo, alors il le fera dans une autre transaction et recommencera les 3 étapes.

# Procédure reg\_verif()

```
CREATE PROCEDURE reg_verif (IN p_customer_login varchar(255), IN p_passwd
varchar(255), OUT p_res int)
BEGIN
    DECLARE v_exist int;
    DECLARE v_goodpassword int;

    select count(*)
    into v_exist
    from customer
    where idname = p_customer_login;

    IF v_exist = 0 then
        SET p_res = 2;
    ELSE
        select count(*)
        into v_goodpassword
        from customer
        where (idname = p_customer_login) AND (passwd = p_passwd);

        IF v_goodpassword = 0 THEN
            SET p_res = 0;
        ELSE
            SET p_res = 1;
        END IF;
    END IF;
END;
```

# Procédure create\_trans()

```
CREATE PROCEDURE create_trans (
    IN p_cc_no varchar(255),
    IN p_cc_type ENUM('V','M','A','D'),
    IN p_cc_expd date,
    OUT p_idn INT,
    IN p_idname varchar(255))

BEGIN
    SET p_idn = -1;

    insert into transaction(cc_no,cc_type,cc_expd,trdate)
    values (p_cc_no,p_cc_type,p_cc_expd,now());

    select last_insert_id()
    into p_idn;

    insert into buys(idn, idname)
    values (p_idn, p_idname);

END;
```

# Procédure add\_photo() 1/2

```
CREATE PROCEDURE add_photo (
    IN p_idn int,
    IN p_catalogn int,
    OUT p_message varchar(255))

BEGIN
    declare v_photo enum('Y','N');
    declare v_amount int;
    declare v_price int;
    declare exit handler for SQLEXCEPTION

    set p_message = 'Erreur dans add_photo';

    -----
    -- debut de la transaction Mysql d'ajout d'une photo a une transaction de la base
    start transaction;

    -----
    -- Verification que la photo est toujours disponible et pose d'un verrou exclusif
    -- sur la ligne correspondante de la table photo
    select available into v_photo from photo where catalogn=p_catalogn FOR UPDATE;

    if v_photo = 'N' then
        set p_message = concat('La photo ', p_catalogn, ' n est plus disponible');
    else
        -----
        -- insertion d'une nouvelle ligne dans la table includes avec le numero de la photo
        -- et celui de la transaction. On pourrait tester ici si la photo n'est pas deja dans la
        -- transaction afin de ne pas l'y insérer deux fois.
        insert into includes (catalogn, idn) values (p_catalogn, p_idn);
```

# Procédure add\_photo() 2/2

```
-- -----  
-- Mise a jour du montant de la transaction  
-- Recuperation du montant actuel  
select amount into v_amount from transaction where idn = p_idn;  
  
-- recuperation du prix de la photo  
select price into v_price from photo where catalogn = p_catalogn;  
  
-- si c'est la première photo de la transaction  
if v_amount is null then  
    update transaction set amount = v_price where idn = p_idn;  
-- s'il y a deja des photos dans la transaction  
else  
    update transaction set amount = v_amount + v_price where idn = p_idn;  
end if;  
  
-- -----  
-- mise a jour de la disponibilite de la photo  
update photo set available = 'N' where catalogn = p_catalogn;  
  
• set p_message = concat('La photo ', p_catalogn, ' a ete ajoutee a la transaction ',p_idn, '.');  
end if;  
  
-- -----  
-- Fin de la transaction MySQL et liberation des verrous  
commit;  
END;
```

# Exercice

Pour l'utilisateur cust\_interface, ajoutez :

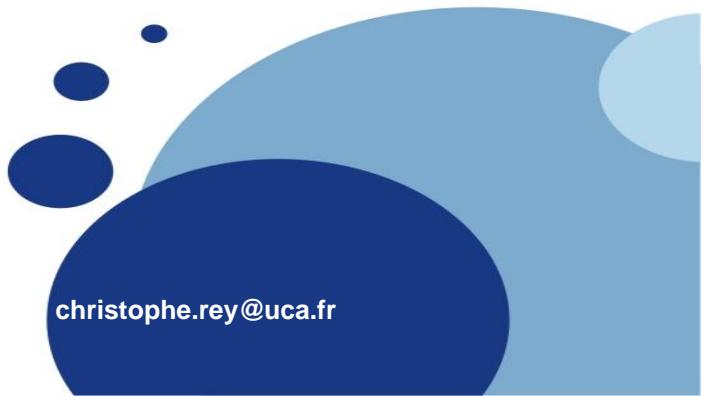
- Une procédure remove\_photo()
  - Entrées : le numéro de la photo, le numéro de la transaction d'achat de la photo
  - Sortie : 1 si succès, 0 sinon
  - Instructions
    - Début de la transaction
    - Vérifier que la photo est bien liée à la transaction en posant un verrou exclusif sur la ligne concernée de la table includes
    - Si non, alors retourner 0
    - Si oui, supprimer la ligne de includes et mettre à jour
      - » Le montant de la transaction
      - » La disponibilité de la photo
    - Et retourner 1 après la fin de la transaction
- Une procédure de suppression d'une transaction  
Principe de fonctionnement : vider la transaction de chacune de ses photos
  - Récupérer toutes les photos dans un curseur
  - Appeler remove\_photo() pour chacune
  - Supprimer la ligne de la table transaction en mettant à jour la table buys.

# Exercice

- Reprendre l'utilisateur « admin\_interface »
- Fixer ses droits
- Créer les éventuelles vues, procédures, triggers, ...., nécessaires à cet utilisateur

**MMI Vichy  
S3  
BD2**

# Annexes



[christophe.rey@uca.fr](mailto:christophe.rey@uca.fr)

# phpMyAdmin

- Interface d'administration de MySQL en php
- Permet l'administration via le web de sa BD
- [http://www.phpmyadmin.net/home\\_page/index.php](http://www.phpmyadmin.net/home_page/index.php)
- Nécessite d'être installé dans un serveur web qui sait interpréter les pages php
- Par exemple : easyphp  
<http://www.easyphp.org/index.php>
- Tutorial sur [www.developpez.com](http://www.developpez.com/) :  
<http://cyberzoide.developpez.com/php4/mysql/>

# phpMyAdmin – configuration basique avec easypHP

- S'assurer que le super utilisateur MySQL (root) a bien un mot de passe  
→ on n'a alors plus accès à phpmyadmin directement
- Enlever tous les utilisateurs créés par défaut autres que root@localhost et root@127.0.0.1
- En tant que root, créer dans MySQL l'utilisateur phpadmin en lui donnant
  - Le droit de visualiser les tables de la base mysql
  - Et un mot de passe

```
grant select on mysql.* to phpadmin identified by  
'phpadmin';
```
- Script de configuration de phpMyAdmin :  
<http://127.0.0.1/home/mysql/scripts/setup.php>
  - Créer un répertoire config à la racine du répertoire de phpMyAdmin (dans EasyPHP)
  - Choisir puis « Add server »
    - Authentication type : http
    - Config author : root
    - phpMyAdmin control user : phpadmin
    - phpMyAdmin control user : le mot de passe de phpadmin
  - Choisir « Save »
  - Copier le fichier généré config.inc.php dans le répertoire racine de phpMyAdmin
  - Supprimer le répertoire config

# phpMyAdmin – fichier de configuration

- Fichier config.inc.php généré :

```
<?php
/*
 * Generated configuration file
 * Generated by: phpMyAdmin 2.9.1.1 setup script by Michal
 * ÄŒihaÃ™ <michal@cihar.com>
 * Version: $Id: setup.php 9484 2006-10-03 13:11:22Z nijel $
 * Date: Wed, 12 Mar 2008 13:30:18 GMT
 */

/* Servers configuration */
$i = 0;

/* Server localhost (http) [1] */
$i++;
$cfg['Servers'][$i]['host'] = 'localhost';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['connect_type'] = 'tcp';
$cfg['Servers'][$i]['compress'] = false;
$cfg['Servers'][$i]['controluser'] = 'phpadmin';
$cfg['Servers'][$i]['controlpass'] = 'phpadmin';
$cfg['Servers'][$i]['auth_type'] = 'http';

/* End of servers configuration */

?>
```