

Vincent Dechef & Kevin Marra

2016 – 2017

Application Informatique (JEE)

Anne Vandevorst

Rapport de Projet JEE

Partie PL/SQL

# La base de données

## Schéma

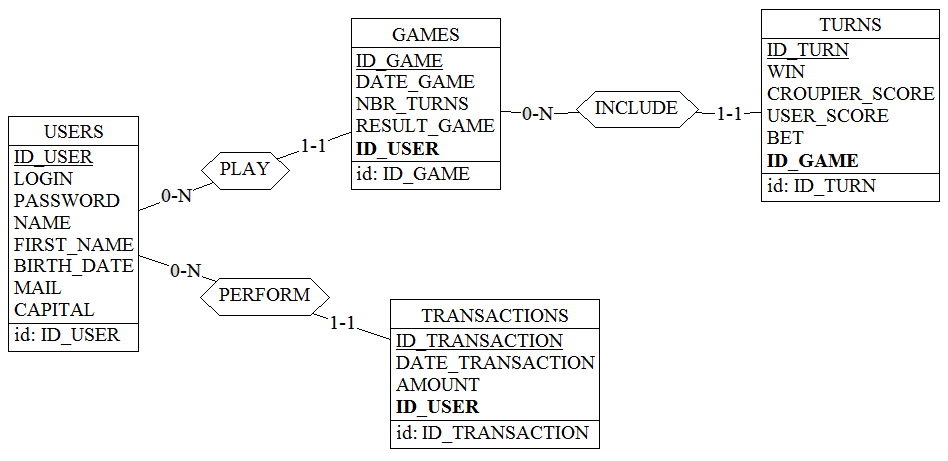


Figure 1: Schéma de la base de données

### Explication

Un utilisateur (USERS) possède un identifiant (LOGIN), un mot de passe (PASSWORD) et une quantité de jetons (CAPITAL) ainsi que des information classique (NAME, FIRST\_NAME, BIRTH\_DATE, MAIL).

Il peut jouer une ou plusieurs parties (GAMES) qui auront une date (DATE\_GAME), un nombre de tours (NBR\_TURNS), un résultat représentant le nombre de jetons perdus ou gagnés (RESULT\_GAME).

Chaque partie est composée de tour (TURNS) qui seront gagné ou perdu (WIN), on gardera le score du croupier et le score de l’utilisateur en mémoire (respectivement CROUPIER\_SCORE et USER\_SCORE) ainsi que la mise du joueur lors de ce tour (BET).

L’utilisateur pourra à tout moment acheter ou vendre ses jetons et nous garderons trace des transaction (TRANSACTIONS) qui auront une date (DATE\_TRANSACTION) et un montant positif ou négatif selon le type de transaction (AMOUNT).

# Script de création de la base de données

/\*==============================================================\*/

/\* Nom de SGBD : ORACLE Version 11g \*/

/\* Nom de DB : CASINO \*/

/\* Date de création : 10/12/2016 15:47:00 \*/

/\*==============================================================\*/

DROP USER blackjack CASCADE;

CREATE USER blackjack IDENTIFIED BY blackj;

GRANT CONNECT, RESOURCE, DBA to BLACKJACK;

CONNECT blackjack/blackj

/\*==============================================================\*/

/\* Table : USERS \*/

/\*==============================================================\*/

create table USERS (

ID\_USER INTEGER not null,

LOGIN VARCHAR2(100) not null,

PASSWORD VARCHAR2(100) not null,

NAME VARCHAR2(100),

FIRST\_NAME VARCHAR2(100),

BIRTH\_DATE VARCHAR2(100),

MAIL VARCHAR2(100),

CAPITAL INTEGER not null,

constraint PK\_USERS primary key (ID\_USER)

);

/\*==============================================================\*/

/\* Table : GAMES \*/

/\*==============================================================\*/

create table GAMES (

ID\_GAME INTEGER not null,

DATE\_GAME DATE not null,

NBR\_TURNS INTEGER not null,

RESULT\_GAME INTEGER not null,

ID\_USER INTEGER not null,

constraint PK\_GAMES primary key (ID\_GAME)

);

/\*==============================================================\*/

/\* Index : PLAY\_FK \*/

/\*==============================================================\*/

create index PLAY\_FK on GAMES (

ID\_USER ASC

);

/\*==============================================================\*/

/\* Table : TURNS \*/

/\*==============================================================\*/

create table TURNS (

ID\_TURN INTEGER not null,

WIN INTEGER not null,

CROUPIER\_SCORE INTEGER not null,

USER\_SCORE INTEGER not null,

BET INTEGER not null,

ID\_GAME INTEGER not null,

constraint PK\_TURNS primary key (ID\_TURN),

constraint CHK\_WIN check (WIN = 1 or WIN = 0)

);

/\*==============================================================\*/

/\* Index : INCLUDE\_FK \*/

/\*==============================================================\*/

create index INCLUDE\_FK on TURNS (

ID\_GAME ASC

);

/\*==============================================================\*/

/\* Table : TRANSACTIONS \*/

/\*==============================================================\*/

create table TRANSACTIONS (

ID\_TRANSACTION INTEGER not null,

DATE\_TRANSACTION DATE not null,

AMOUNT INTEGER not null,

ID\_USER INTEGER not null,

constraint PK\_TRANSACTIONS primary key (ID\_TRANSACTION)

);

/\*==============================================================\*/

/\* Index : PERFORM\_FK \*/

/\*==============================================================\*/

create index PERFORM\_FK on TRANSACTIONS (

ID\_USER ASC

);

/\*==============================================================\*/

/\* CONSTRAINTS FOREIGN KEY \*/

/\*==============================================================\*/

alter table GAMES

add constraint FK\_GAMES\_PLAY\_USERS foreign key (ID\_USER)

references USERS (ID\_USER) on delete cascade;

alter table TURNS

add constraint FK\_TURNS\_INCLUDE\_GAMES foreign key (ID\_GAME)

references GAMES (ID\_GAME) on delete cascade;

alter table TRANSACTIONS

add constraint FK\_TRANSACTIONS\_PLAY\_USERS foreign key (ID\_USER)

references USERS (ID\_USER) on delete cascade;

## 

## Gestion de l’auto incrémentation pour les id

/\*==============================================================\*/

/\* AUTO INCREMENT \*/

/\*==============================================================\*/

create sequence USERS\_SEQ start with 1;

create or replace trigger USERS\_BIR

before insert on USERS

for each row

begin

select USERS\_SEQ.nextval

into :new.ID\_USER

from dual;

end;

/

create sequence GAMES\_SEQ start with 1;

create or replace trigger GAMES\_BIR

before insert on GAMES

for each row

begin

select GAMES\_SEQ.nextval

into :new.ID\_GAME

from dual;

end;

/

create sequence TURNS\_SEQ START with 1;

create or replace trigger TURNS\_BIR

before insert on TURNS

for each row

begin

select TURNS\_SEQ.nextval

into :new.ID\_TURN

from dual;

end;

/

create sequence TRANSACTIONS\_SEQ START with 1;

create or replace trigger TRANSACTIONS\_BIR

before insert on TRANSACTIONS

for each row

begin

select TRANSACTIONS\_SEQ.nextval

into :new.ID\_TRANSACTION

from dual;

end;

/

# Scripts PL/SQL

Les scripts ont été séparé en 2 packages :

* Package **GetData** : contient toutes les procédures de récupération de données.
  + *findGame*: cherche une partie dans la table Games.
  + *findTransactionByUser* : cherche une transaction en function d’un utilisateur.
  + *findTurnByGame*: cherche un tour de jeu en function d’une partie.
  + *findUser* : cherche un utilisateur.
  + *getIdGame* : récupère l’Id d’un partie.

Ces procédures ont un paramètre de type OUT qui renvoi un curseur. Ce curseur servira à lire les données dans le code Java.

* Package **AddData** : contient toutes les procédures d’ajout de données.
  + *addGame*: ajoute une partie.
  + *addTransaction*: ajoute une transaction.
  + *addTurn*: ajoute un tour de jeu pour une partie.
  + *addUser*: ajoute un utilisateur.
  + *updateGame* : met à jour la table Games.
  + *updateUser* : met à jour la table Users.

En ce qui concerne les exceptions, chaque package contient la déclaration suivante :

fk\_violationException EXCEPTION;

PRAGMA EXCEPTION\_INIT(fk\_violationException,-2291);

S’il y a une violation des contraintes, pl/sql renvoie automatiquement une exception qui sera intercepté comme une SQLException en Java. Le message contenu dans l’exception contiendra donc « fk\_violationException » et c’est donc le code Java qui gére l’exception.

## Script des packages et des procédures :

CREATE OR REPLACE PACKAGE AddData --Crée le package

AS

--Définitions des procédures

procedure addGame (dateGame number, nbrTurns number, resultGame number, idUser number);

procedure addTransaction (dateTransaction number,var\_amount number,idUser number);

procedure addTurn (var\_win number,var\_croupierScore number,var\_useScore number,var\_bet number,var\_idGame number);

procedure addUser (login varchar,passwd varchar,name varchar,first\_name varchar,birth\_date number,mail varchar,capital number);

procedure updateGame(dateGame number,nbrTurns number,resultGame number,idUser number,idGame number);

procedure updateUser (var\_capital number,idUser number);

--Exception

fk\_violationException EXCEPTION;

PRAGMA EXCEPTION\_INIT(fk\_violationException,-2291);

END AddData;

/

--Corp du package et des procedures

CREATE OR REPLACE PACKAGE BODY AddData

IS

fk\_violationException EXCEPTION;

PRAGMA EXCEPTION\_INIT(fk\_violationException,-2291);

procedure addGame(

dateGame number,

nbrTurns number,

resultGame number,

idUser number

)

as

begin

INSERT INTO GAMES (date\_game, nbr\_turns, result\_game, id\_user)

VALUES (dateGame, nbrTurns, resultGame, idUser);

end addGame;

procedure addTransaction(

dateTransaction number,

var\_amount number,

idUser number

)

as

begin

INSERT INTO transactions (date\_transaction, amount, id\_user)

VALUES (dateTransaction, var\_amount, idUser);

end addTransaction;

procedure addTurn(

var\_win number,

var\_croupierScore number,

var\_useScore number,

var\_bet number,

var\_idGame number

)

as

begin

INSERT INTO TURNS (win, croupier\_score, user\_score, bet, id\_game)

VALUES (var\_win, var\_croupierScore, var\_useScore, var\_bet, var\_idGame);

end addTurn;

procedure addUser(

login varchar,

passwd varchar,

name varchar,

first\_name varchar,

birth\_date number,

mail varchar,

capital number

)

as

begin

INSERT INTO USERS (login, password, name, first\_name, birth\_date, mail, capital)

VALUES (login,passwd, name, first\_name, birth\_date, mail, capital);

end addUser;

procedure updateGame(

dateGame number,

nbrTurns number,

resultGame number,

idUser number,

idGame number

)

as

begin

UPDATE GAMES

SET date\_game = dateGame,

nbr\_turns = nbrTurns,

result\_game = resultGame,

id\_user = idUser

WHERE id\_game = idGame;

end updateGame;

procedure updateUser(

var\_capital number,

idUser number

)

as

begin

UPDATE USERS

SET capital = var\_capital

WHERE id\_user = idUser;

end updateUser;

END AddData;

/

CREATE OR REPLACE PACKAGE GetData --Création du package

AS

--Définitions des procédures

procedure findGame (id number, recordset OUT SYS\_REFCURSOR);

procedure findGameByUser (id number, recordset OUT SYS\_REFCURSOR);

procedure findTransactionByUser (id number, recordset OUT SYS\_REFCURSOR);

procedure findTurnByGame (id number, recordset OUT SYS\_REFCURSOR);

procedure findUser (var\_mail varchar, var\_passwd varchar, recordset OUT SYS\_REFCURSOR);

procedure getIdGame (var\_date number, var\_nbrTurns number, var\_resultGame number, var\_idUser number, id out number);

--Exception

fk\_violationException EXCEPTION;

PRAGMA EXCEPTION\_INIT(fk\_violationException,-2291);

END GetData;

/

--Corp du package et des procedures

CREATE OR REPLACE PACKAGE BODY GetData

IS

fk\_violationException EXCEPTION;

PRAGMA EXCEPTION\_INIT(fk\_violationException,-2291);

procedure findGame(id number, recordset OUT SYS\_REFCURSOR)

as

begin

OPEN recordset FOR

SELECT \* FROM GAMEs WHERE id\_game = id;

end findGame;

procedure findGameByUser(id number, recordset OUT SYS\_REFCURSOR)

as

begin

OPEN recordset FOR

SELECT \* FROM GAMEs WHERE id\_user = id;

end findGameByUser;

procedure findTransactionByUser(id number, recordset OUT SYS\_REFCURSOR)

as

begin

OPEN recordset FOR

SELECT \* FROM TRANSACTIONS WHERE id\_user = id;

end findTransactionByUser;

procedure findTurnByGame(id number, recordset OUT SYS\_REFCURSOR)

as

begin

OPEN recordset FOR

SELECT \* FROM TURNS WHERE id\_game = id;

end findTurnByGame;

procedure findUser(var\_mail varchar, var\_passwd varchar, recordset OUT SYS\_REFCURSOR)

as

begin

OPEN recordset FOR

SELECT \* FROM USERS WHERE MAIL = var\_mail and PASSWORD = var\_passwd;

end findUser;

procedure getIdGame(var\_date number, var\_nbrTurns number, var\_resultGame number, var\_idUser number, id out number)

as

begin

--On récupère uniquement la dernière game.

SELECT id\_game into id

FROM (

--Renvois tout les id possibles (peut en avoir plusieurs et créer un erreur)

SELECT id\_game FROM GAMES

WHERE

date\_game = var\_date and

nbr\_turns = var\_nbrTurns and

result\_game = var\_resultGame and

id\_user = var\_idUser

ORDER BY id\_game DESC

)

WHERE rownum <= 1;

end getIdGame;

END GetData;

/