Une image contenant Police, capture d’écran, Graphique, symbole

Description générée automatiquement

**Projet final**

**Gestion d’un casino**

Par Tarik Benakezouh, Carlos Eduardo Calle Guerrero,

Antoine Ho, Lucky Khounvongsa, & Kha Pham

Matricules : 20184524, 20220633, 20202790, 20172476 & 2018233

Travail présenté à Jihene Rezgui

dans le cadre du cours de

IFT2935-B – Base de données

**Remis le 10 avril 2024**

**Introduction**

Dans le monde dynamique et en constante évolution des jeux et divertissements, la gestion efficace d'un casino représente un défi majeur à entreprendre. Avec l'avancement technologique et la croissance des attentes des clients, il est impératif pour les casinos de moderniser leurs systèmes de gestion pour améliorer l'efficacité, garantir la sécurité et maximiser les profits. Ce projet vise à développer une solution de gestion de base de données sophistiquée pour un casino, couvrant tous les aspects cruciaux de son fonctionnement, y compris la gestion des installations, du personnel, des recettes des machines de jeu, et des interventions techniques. En utilisant SQL Server 2019, cette solution permettra au casino de gérer ses opérations de manière plus efficace et précise, en assurant une expérience de jeu fluide pour les clients et un environnement de travail optimisé pour le personnel.

**Description du projet**

**Analyse approfondie des besoins**

Le fonctionnement d'un casino moderne repose sur une gestion optimisée et automatisée de ses multiples composantes, allant des installations physiques aux ressources humaines, tout en passant par le suivi financier des activités de jeu. Une base de données conçue pour un casino doit, par conséquent, offrir une solution intégrée qui répond à ces exigences complexes. Ce projet vise à concevoir une base de données qui non seulement répond à ces besoins opérationnels, mais le fait de manière qui maximise l'efficacité, assure la conformité réglementaire, et optimise l'expérience des utilisateurs finaux, tant pour le personnel que pour les clients.

**Réalisme et adéquation**

Le réalisme de la conception de la base de données est assuré par l'identification précise et la modélisation des entités et des relations clés qui reflètent fidèlement les opérations d'un casino réel. Cela comprend :

* Les **installations**, qui sont diversifiées et comprennent des équipements à la fois statiques et dynamiques, nécessitant une gestion et un suivi détaillés pour assurer leur fonctionnement optimal et leur conformité aux normes de sécurité.
* La **gestion du personnel**, qui doit être adaptable pour couvrir divers rôles et compétences, tout en fournissant des informations essentielles pour la gestion des ressources humaines et la planification du personnel.
* Le **suivi des recettes**, essentiel pour la viabilité financière du casino, doit être précis et fiable, permettant une analyse détaillée des performances des différentes machines et installations.
* La **gestion des pannes** nécessitant une approche systématique pour minimiser les temps d'arrêt et maintenir une expérience client de haute qualité.

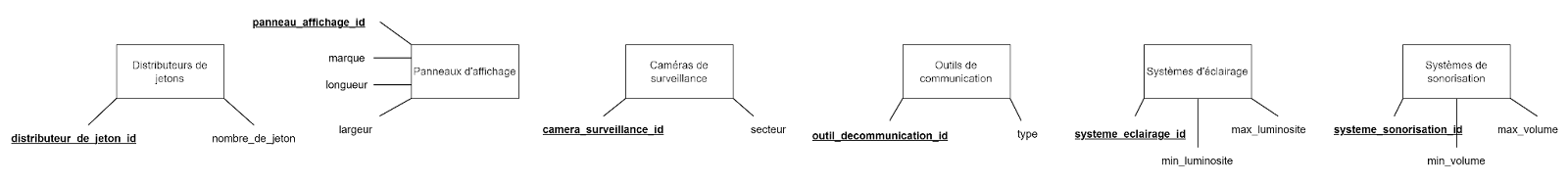
**Justification des choix de conception**

La conception du schéma Entité-Association (E-A) présentée repose sur une structuration minutieuse visant à optimiser la gestion et l'organisation des données au sein d'un casino. La décision de fusionner les trois catégories de personnel en une seule table "Personnel" tout en associant cette table à une autre dédiée aux rôles des personnels reflète une approche rationalisée. Cela simplifie la base de données en réduisant la redondance, tout en conservant la flexibilité nécessaire pour détailler les fonctions et responsabilités à travers la table des rôles, facilitant ainsi les requêtes et les mises à jour.

En ce qui concerne les installations, la décision de créer une table distincte pour chaque type d'installation (machines de jeu, distributeurs de jetons, panneaux d'affichage, etc.) est justifiée par la diversité et la complexité spécifique de chaque élément. Cette approche permet de gérer plus efficacement les caractéristiques uniques et les exigences de maintenance de chaque type d'installation, offrant une granularité fine dans le suivi et la gestion des ressources.

L'aspect des pannes souligne l'importance de la normalisation des données, en particulier le respect de la première forme normale (1FN). La création d'une table séparée "Type de panne" est une réponse stratégique à la nécessité de gérer des informations multiples et variées liées aux pannes, évitant ainsi les données non atomiques et assurant une meilleure intégrité des données. Cette séparation facilite les analyses détaillées des pannes et améliore la maintenance préventive et corrective.

L'inclusion d'une table "Pièce" sans référence explicite dans l'énoncé initial peut être vue comme une anticipation des besoins futurs ou une extension logique du modèle, permettant de suivre les pièces de rechange, leur état, et leur association avec les équipements et les interventions de maintenance. Bien qu'elle n'ait pas été spécifiée initialement, sa présence peut enrichir le schéma de données en offrant une vue complète sur la gestion des inventaires et des ressources matérielles, ce qui est crucial pour un fonctionnement optimal du casino.

Une image contenant capture d’écran, noir, Symétrie, noir et blanc

Description générée automatiquement**Schéma Entité-Association**

**Schéma relationnel**

MachineDeJeu (**machine\_jeu\_id**, date\_service, **#id\_jeu**)

Jeu(**id\_jeu**, nom\_jeu, min\_mise, max\_mise)

DistributeurDeJeton (**distributeur\_de\_jeton\_id**, nombre\_de\_jeton)

PanneauAffichage (**panneau\_affichage\_id**, marque, longueur, largeur)

CameraSurveillance (**camera\_surveillance\_id**, secteur)

OutilsDeCommunication (**outils\_de\_communication\_id**, type)

SystemeEclairage (**systeme\_eclairage\_id**, min\_luminosite, max\_luminosite)

SystemeSonorisation (**systeme\_sonorisation\_id**, min\_volume, max\_volume)

Adresse\_civique(**numero\_adresse, rue, ville**)

Role\_personnel(**id\_role**, nom, secteur, expertise, droit\_acces)

Personnel(**numero\_id**, nom, prenom, **#id\_role**, **#numero\_adresse**, **#rue**, **#ville**)

Recette(**recette\_id**, date, montant profit, coûts, **#machine\_jeu\_id**, rentable)

Panne(**panne\_id**, type, date, **#personnel\_id**, piece, **#id\_type\_panne**)

Type\_panne(**id\_type\_panne**, nom. catégorie)

Piece(**id\_piece**, nom, date\_service, etat, **#id\_fournisseur**, **#id\_machine\_jeu**, **#id\_panne**)

Fournisseur(**id\_fournisseur**, nom, ville)

**Normalisation**

Machine de jeu: (id) -> (date\_service, #id\_jeu)

Jeu: (id\_jeu) -> (nom\_jeu, min\_mise, max\_mise)

Distributeur de jeton: (id) -> (nombre de jeton, date\_service)

Panneau d'affichage: (id) -> (marque, date\_service, longueur, largeur)

Caméra de surveillance: (id) -> (secteur, date\_service)

Outil de communication: (id) -> (date\_service, type)

Système d'éclairage: (id) -> (date\_service, nom, min\_luminosite, max\_luminosite)

Système de sonorisation: (id) -> (date\_service, nom, min\_volume, max\_volume)

Adresse civique: (**numero\_adresse, rue, ville**) -> (numero\_adresse, rue, ville)

Rôle du personnel: (**id\_role**) -> (nom, secteur, expertise, droit\_acces)

Personnel: (**numero\_id**) -> (nom, prenom, #id\_role, #code\_postal, #numero\_adresse, rue, ville)

Recette: (**recette\_id**) -> (date, montant profit, coûts, #id\_machine\_jeu, rentable)

Panne: (**panne\_id**) -> (**#id\_type\_panne**, date, #id\_personel)

Type de panne: (**id\_type\_panne**) -> (nom, categorie)

Pièce: (**id\_piece**) -> (nom, date\_service, etat, #id\_fournisseur, #id\_machine\_jeu, #id\_panne)

Fournisseur: (**id\_fournisseur**) -> (nom, ville)

**Première forme normale (1FN)**

Passer à la première forme normale est pour nous une démarche assez intuitive puisqu’elle vise à garantir que chaque attribut contient des valeurs atomiques. Cette étape est essentielle pour éliminer les groupes de répétition et assurer que chaque champ stocke une valeur unique et spécifique. Cependant, nous avions initialement envisagé de regrouper des informations telles que le numéro d'adresse, la rue, et la ville en un seul attribut sous forme de concaténation. En plus du non-respect de la première forme normale, cela compliquait les requêtes nécessitant l'accès individuel à ces informations, comme pour trouver toutes les adresses dans une ville donnée. Nous avons ainsi résolu ce problème en séparant ces données en attributs distincts et atomiques, améliorant l'accessibilité et la manipulation des données.

**Deuxième forme normale (2FN)**

La deuxième forme normale se concentre sur l'élimination des dépendances partielles en veillant à ce que chaque attribut non-clé dépende entièrement de la clé primaire dans les situations où les clés composées sont présentes. Dans notre cas, l'application de la 2FN ne se manifeste pas de manière directement pertinente, principalement à cause de l'absence de clés composées dans la plupart des tables que nous avons définies. Chaque table semble avoir une clé primaire unique, indiquant que toutes les dépendances sont pleinement fonctionnelles par définition. Cette simplification de la structure nécessite cependant que nous soyons particulièrement attentifs lors de la définition de chaque table pour assurer l'unicité et l'adéquation de la clé primaire.

**Troisième forme normale (3FN)**

La troisième forme normale vise à éliminer les dépendances transitives, assurant que chaque attribut non-clé dépende directement de la clé primaire et non d'un autre attribut non-clé. Nous avons identifié que des tables telles qu’“Adresse civique”, “Rôle du personnel”, “Type de panne”, “Fournisseur”, et “Jeu” ne respectaient pas pleinement ce niveau de normalisation. Par exemple, les attributs d'un rôle peuvent ne pas dépendre directement de l'employé lui-même, mais sont liés par une clé étrangère. Afin d’y remédier, nous avons opté pour la création de nouvelles tables pour ces attributs interdépendants, avec une clé primaire distincte (souvent reprenant la clé étrangère de l'attribut dans la table originale), nous permettant de clarifier et de structurer ces relations de manière plus appropriée. Cette méthode facilite la maintenance des données, rend les mises à jour plus efficaces et améliore la cohérence de l'information en assurant que toutes les dépendances sont directement liées à la clé primaire appropriée.

**LDD (Create, Alter, Drop)**

--create dautabase projdb;

go

--use projdb;

go

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'recette' )

begin

alter table recette drop constraint fk\_recette\_machine;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'panne' )

begin

alter table panne drop constraint fk\_panne\_type;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'panne' )

begin

alter table panne drop constraint fk\_panne\_personnel;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'machine\_de\_jeu' )

begin

alter table machine\_de\_jeu drop constraint fk\_machine\_jeu;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'personnel' )

begin

alter table personnel drop constraint fk\_personnel\_role;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'personnel' )

begin

alter table personnel drop constraint fk\_personnel\_adresse;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'pieces' )

begin

alter table pieces drop constraint fk\_pieces\_fournisseur;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'pieces' )

begin

alter table pieces drop constraint fk\_pieces\_machine;

end;

if exists(SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

          WHERE TABLE\_NAME = 'pieces' )

begin

alter table pieces drop constraint fk\_pieces\_panne; end;

drop table if exists type\_panne;

drop table if exists jeu;

drop table if exists fournisseur;

drop table if exists pieces;

drop table if exists role\_personnel;

drop table if exists adresse\_civique;

drop table if exists recette;

drop table if exists panne;

drop table if exists machine\_de\_jeu;

drop table if exists distributeur\_de\_jeton;

drop table if exists panneau\_affichage;

drop table if exists camera\_surveillance;

drop table if exists outils\_de\_communication;

drop table if exists system\_eclairage;

drop table if exists systeme\_sonorisation;

drop table if exists personnel;

create table jeu(

id int primary key,

nom varchar(50),

min\_mise int,

max\_mise int

);

CREATE TABLE machine\_de\_jeu (

id int primary key,

date\_service date,

id\_jeu int

);

create table distributeur\_de\_jeton(

id int primary key,

nombre\_jeton int,

date\_service date

);

create table panneau\_affichage(

id int primary key,

marque varchar(50),

longueur int,

largeur int,

);

create table camera\_surveillance(

id int primary key,

secteur int,

);

create table outils\_de\_communication(

id int primary key,

type\_comm varchar(10),

);

create table system\_eclairage(

id int primary key,

nom varchar(50),

min\_luminosite int,

max\_luminosite int,

);

create table systeme\_sonorisation(

id int primary key,

nom varchar(50),

min\_volume int,

max\_volume int,

);

create table adresse\_civique(

numero\_adresse int,

rue varchar(50),

ville varchar(50),

primary key(numero\_adresse, rue, ville)

);

create table role\_personnel(

id int primary key,

nom varchar(20),

secteur int,

expertise varchar(20),

droit\_acces int

);

create table personnel(

id int primary key,

nom varchar(50) not null,

prenom varchar(50) not null,

id\_role int,

numero\_adresse int,

rue varchar(50),

ville varchar(50));

create table recette(

id int primary key,

date\_recette date,

profit decimal(19, 4),

cout decimal(19, 4),

information varchar(100),

id\_machine\_de\_jeu int,

rentable binary,

);

create table type\_panne(

id int primary key,

nom varchar(50),

categorie varchar(20)

);

create table panne(

id int primary key,

id\_type\_panne int,

date\_panne date,

id\_personnel int

);

create table fournisseur(

id int primary key,

nom varchar(50),

ville varchar(50)

);

create table pieces(

id int primary key,

nom varchar(50),

date\_service date,

etat varchar(50),

id\_fournisseur int,

id\_machine\_jeu int,

id\_panne int

);

alter table pieces add constraint fk\_pieces\_fournisseur foreign key (id\_fournisseur) references fournisseur(id) ON DELETE CASCADE;

alter table pieces add constraint fk\_pieces\_machine foreign key (id\_machine\_jeu) references machine\_de\_jeu(id) ON DELETE CASCADE;

alter table pieces add constraint fk\_pieces\_panne foreign key (id\_panne) references personnel(id) ON DELETE CASCADE;

alter table personnel add constraint fk\_personnel\_role foreign key (id\_role) references role\_personnel(id) ON DELETE CASCADE;

alter table personnel add constraint fk\_personnel\_adresse foreign key (numero\_adresse, rue, ville) references adresse\_civique(numero\_adresse, rue, ville) ON DELETE CASCADE;

alter table machine\_de\_jeu add constraint fk\_machine\_jeu foreign key (id\_jeu) references jeu(id) ON DELETE CASCADE;

alter table recette add constraint fk\_recette\_machine foreign key (id\_machine\_de\_jeu) references machine\_de\_jeu(id) ON DELETE CASCADE;

alter table panne add constraint fk\_panne\_type foreign key (id\_type\_panne) references type\_panne(id) ON DELETE CASCADE;

alter table panne add constraint fk\_panne\_personnel foreign key (id\_personnel) references personnel(id) ON DELETE CASCADE;

**LMD( INSERT , UPDATE,DELETE)**

INSERT INTO jeu (id, nom, min\_mise, max\_mise) VALUES

(1, 'Blackjack', 5, 500),

(2, 'Roulette', 2, 300),

(3, 'Poker', 10, 1000),

(4, 'Baccarat', 25, 1000),

(5, 'Slot Machine', 1, 100);

INSERT INTO machine\_de\_jeu (id, date\_service, id\_jeu) VALUES

(1, '2023-01-01', 1),

(2, '2023-02-01', 2),

(3, '2023-03-01', 3),

(4, '2023-04-01', 4),

(5, '2023-05-01', 5);

INSERT INTO distributeur\_de\_jeton (id, nombre\_jeton, date\_service) VALUES

(1, 500, '2023-01-15'),

(2, 600, '2023-02-15'),

(3, 700, '2023-03-15'),

(4, 800, '2023-04-15'),

(5, 900, '2023-05-15');

INSERT INTO panneau\_affichage (id, marque, longueur, largeur) VALUES

(1, 'Samsung', 100, 50),

(2, 'LG', 120, 60),

(3, 'Sony', 110, 55),

(4, 'Philips', 130, 65),

(5, 'Panasonic', 140, 70);

INSERT INTO camera\_surveillance (id, secteur) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

INSERT INTO outils\_de\_communication (id, type\_comm) VALUES

(1, 'Radio'),

(2, 'Phone'),

(3, 'Intercom'),

(4, 'Email'),

(5, 'Text');

INSERT INTO system\_eclairage (id, nom, min\_luminosite, max\_luminosite) VALUES

(1, 'Spotlight', 100, 1000),

(2, 'Ambient Light', 50, 500),

(3, 'Track Light', 75, 750),

(4, 'LED Panel', 60, 600),

(5, 'Halogen', 100, 1000);

INSERT INTO systeme\_sonorisation (id, nom, min\_volume, max\_volume) VALUES

(1, 'Basic PA', 10, 100),

(2, 'Surround Sound', 20, 200),

(3, 'Concert System', 30, 300),

(4, 'Studio Quality', 40, 400),

(5, 'Outdoor Event', 50, 500);

INSERT INTO adresse\_civique (numero\_adresse, rue, ville) VALUES

( 101, 'Main St', 'Montreal'),

( 202, 'Second St', 'Quebec City'),

( 303, 'Third St', 'Sherbrooke'),

( 404, 'Fourth St', 'Ottawa'),

( 505, 'Fifth St', 'Toronto');

INSERT INTO role\_personnel (id, nom, secteur, expertise, droit\_acces) VALUES

(1, 'Manager', 1, 'Administration', 5),

(2, 'Security', 2, 'Surveillance', 4),

(3, 'Dealer', 3, 'Games', 3),

(4, 'Technician', 4, 'Maintenance', 4),

(5, 'Croupier', 3, 'Card Games', 3);

INSERT INTO personnel (id, nom, prenom, id\_role, numero\_adresse, rue, ville) VALUES

(1, 'Doe', 'John', 1, 101, 'Main St', 'Montreal'),

(2, 'Smith', 'Jane', 2, 202, 'Second St', 'Quebec City'),

(3, 'Brown', 'Bob', 3, 303, 'Third St', 'Sherbrooke'),

(4, 'Green', 'Alice', 4, 404, 'Fourth St', 'Ottawa'),

(5, 'White', 'Tom', 5, 505, 'Fifth St', 'Toronto');

INSERT INTO recette (id, date\_recette, profit, cout, information, id\_machine\_de\_jeu, rentable) VALUES

(1, '2023-04-01', 1000.00, 500.00, 'Good day', 1, 1),

(2, '2023-04-02', 800.00, 300.00, 'Average day', 2, 1),

(3, '2023-04-03', 200.00, 400.00, 'Bad day', 3, 0),

(4, '2023-04-04', 500.00, 200.00, 'Steady day', 4, 1),

(5, '2023-04-05', 300.00, 150.00, 'Slow day', 5, 1);

INSERT INTO type\_panne (id, nom, categorie) VALUES

(1, 'Software Glitch', 'Software'),

(2, 'Mechanical Failure', 'Hardware'),

(3, 'Power Supply Issue', 'Electrical'),

(4, 'Screen Malfunction', 'Hardware'),

(5, 'Network Issue', 'Software');

INSERT INTO panne (id, id\_type\_panne, date\_panne, id\_personnel) VALUES

(1, 1, '2023-04-05', 1),

(2, 2, '2023-04-06', 2),

(3, 3, '2023-04-07', 3),

(4, 4, '2023-04-08', 4),

(5, 5, '2023-04-09', 5);

INSERT INTO fournisseur (id, nom, ville) VALUES

(1, 'Tech Supplies Inc.', 'Montreal'),

(2, 'Gaming Gear Ltd.', 'Quebec City'),

(3, 'Casino Equip Co.', 'Sherbrooke'),

(4, 'Entertainment Solutions', 'Montreal'),

(5, 'Professional Gaming Inc.', 'Toronto');

INSERT INTO pieces (id, nom, date\_service, etat, id\_fournisseur, id\_machine\_jeu, id\_panne) VALUES

(1, 'Chipset', '2023-05-01', 'New', 1, 1, 1),

(2, 'Motherboard', '2023-06-01', 'Used', 2, 2, 2),

(3, 'Power Supply', '2023-07-01', 'Refurbished', 3, 3, 3),

(4, 'Joystick', '2023-08-01', 'New', 4, 4, 4),

(5, 'Button Set', '2023-09-01', 'New', 5, 5, 5);

DELETE FROM jeu WHERE id =1;

DELETE FROM machine\_de\_jeu WHERE id =1;

DELETE FROM distributeur\_de\_jeton WHERE id =1;

DELETE FROM panneau\_affichage WHERE id =1;

DELETE FROM camera\_surveillance WHERE id =1;

DELETE FROM outils\_de\_communication WHERE id =1;

DELETE FROM system\_eclairage WHERE id =1;

DELETE FROM systeme\_sonorisation WHERE id =1;

DELETE FROM role\_personnel WHERE id =1;

DELETE FROM personnel WHERE id =1;

DELETE FROM recette WHERE id =1;

DELETE FROM type\_panne WHERE id =1;

DELETE FROM panne WHERE id =1;

DELETE FROM fournisseur WHERE id =1;

DELETE FROM pieces WHERE id =1;

UPDATE jeu

SET

    nom = 'New Name',

    min\_mise = 10,

    max\_mise = 1000

WHERE

    id = 1;

UPDATE machine\_de\_jeu

SET

    date\_service = '2024-04-09',

    id\_jeu = 1

WHERE

    id = 1;

UPDATE distributeur\_de\_jeton

SET

    nombre\_jeton = 100,

    date\_service = '2024-04-09'

WHERE

    id = 1;

UPDATE panneau\_affichage

SET

    marque = 'New Brand',

    longueur = 100,

    largeur = 50

WHERE

    id = 1;

UPDATE camera\_surveillance

SET

    secteur = 2

WHERE

    id = 1;

UPDATE outils\_de\_communication

SET

    type\_comm = 'New Type'

WHERE

    id = 1;

UPDATE system\_eclairage

SET

    nom = 'New Name',

    min\_luminosite = 50,

    max\_luminosite = 100

WHERE

    id = 1;

UPDATE systeme\_sonorisation

SET

    nom = 'New Name',

    min\_volume = 20,

    max\_volume = 80

WHERE

    id = 1;

UPDATE role\_personnel

SET

    nom = 'New Name',

    secteur = 2,

    expertise = 'New Expertise',

    droit\_acces = 1

WHERE

    id = 1;

UPDATE personnel

SET

    nom = 'New Name',

    prenom = 'New Firstname',

    id\_role = 2,

    numero\_adresse = 2,

    rue = 'New Street',

    ville = 'New City'

WHERE

    id = 1;

UPDATE recette

SET

    date\_recette = '2024-04-09',

    profit = 150.50,

    cout = 100.25,

    information = 'Updated information',

    id\_machine\_de\_jeu = 2,

    rentable = 1

WHERE

    id = 1;

UPDATE type\_panne

SET

    nom = 'New Name',

    categorie = 'New Category'

WHERE

    id = 1;

UPDATE panne

SET

    id\_type\_panne = 2,

    date\_panne = '2024-04-09',

    id\_personnel = 2

WHERE

    id = 1;

UPDATE fournisseur

SET

    nom = 'New Name',

    ville = 'New City'

WHERE

    id = 1;

UPDATE pieces

SET

    nom = 'New Name',

    date\_service = '2024-04-09',

    etat = 'New State',

    id\_fournisseur = 2,

    id\_machine\_jeu = 2,

    id\_panne = 2

WHERE

    id = 1;

**Ensemble des requêtes et l'application en SQL**

go

create or alter procedure rechercheProfitInterval (@date1 date, @date2 date)

as

Begin

    select \* from recette where date\_recette between @date1 and @date2;

end;

go

create or alter procedure rechercheProfitDate (@date1 date)

as

Begin

    select \* from recette where date\_recette = @date1;

end;

go

create or alter procedure recherchePieceParPanne (@enPanne binary)

as

Begin

if @enPanne = 0

    select \* from pieces where id\_panne is null;

else

select \* from pieces where id\_panne is not null;

end;

go

create or alter procedure NbrPieceParFournisseur

as

Begin

select f.id ,count(p.id) NbrPiece from pieces p join fournisseur f on p.id\_fournisseur=f.id group by f.id

;

end;

go

create or alter procedure PersonnelSpecifique(@id\_role int)

as

Begin

select \* from personnel p1 where id\_role = @id\_role

;

end;

go

create or alter procedure listePanne

as

Begin

select \* from machine\_de\_jeu ma

join (select pie.id\_machine\_jeu, pa.date\_panne,per.nom, per.prenom, tp.nom NomPanne, tp.categorie from panne pa

join pieces pie on pie.id\_panne = pa.id

join personnel per on per.id = pa.id\_personnel

join type\_panne tp on tp.id = pa.id\_type\_panne)

tmp on tmp.id\_machine\_jeu =ma.id

;

end;

go

create or alter procedure listeProfitJeuParSecteur

as

Begin

select rp.secteur, tmp2.id\_jeu, sum(tmp2.profit) ProfitParSecteur from role\_personnel rp

join personnel per

on per.id\_role =rp.id

join(select pa.id\_personnel, pie.id\_machine\_jeu from panne pa

join pieces pie

on pie.id\_panne = pa.id) tmp1

on tmp1.id\_personnel = per.id

join (select ma.id, ma.id\_jeu, j.nom, r.profit from machine\_de\_jeu ma

join recette r

on r.id\_machine\_de\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp2

on tmp2.id = tmp1.id\_machine\_jeu

group by rp.secteur, tmp2.id\_jeu

;

end;

go

create or alter procedure listeProfitMachineParSecteur

as

Begin

--profit machine

select rp.secteur, tmp2.id, sum(tmp2.profit) ProfitParSecteur from role\_personnel rp

join personnel per

on per.id\_role =rp.id

join(select pa.id\_personnel, pie.id\_machine\_jeu from panne pa

join pieces pie

on pie.id\_panne = pa.id) tmp1

on tmp1.id\_personnel = per.id

join (select ma.id, ma.id\_jeu, j.nom, r.profit from machine\_de\_jeu ma

join recette r

on r.id\_machine\_de\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp2

on tmp2.id = tmp1.id\_machine\_jeu

group by rp.secteur, tmp2.id

;

end;

go

create or alter procedure listeProfitMachineParPanne(@enPanne binary)

as

Begin

if @enPanne = 0

    --profit par machine

select tmp.id ,SUM(r.profit) Profit\_Total from recette r

join (select ma.id, pie.id\_panne, j.id JeuID from machine\_de\_jeu ma

join (select \* from pieces p where p.id\_panne is not null) pie

on pie.id\_machine\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp

on tmp.id = r.id\_machine\_de\_jeu

join panne pa

on pa.id = tmp.id\_panne

group by tmp.id

;

else

--profit par machine

select tmp.id ,SUM(r.profit) Profit\_Total from recette r

join (select ma.id, pie.id\_panne, j.id JeuID from machine\_de\_jeu ma

join (select \* from pieces p where p.id\_panne is null) pie

on pie.id\_machine\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp

on tmp.id = r.id\_machine\_de\_jeu

join panne pa

on pa.id = tmp.id\_panne

group by tmp.id

;

end;

go

create or alter procedure listeProfitJeuxParPanne(@enPanne binary)

as

Begin

if @enPanne = 0

--profit par jeux

select r.id, r.date\_recette, SUM(r.profit) AS Profit\_Total from recette r

join (select ma.id, pie.id\_panne, j.id JeuID from machine\_de\_jeu ma

join (select \* from pieces p where p.id\_panne is not null) pie

on pie.id\_machine\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp

on tmp.id = r.id\_machine\_de\_jeu

join panne pa

on pa.id = tmp.id\_panne

group by r.date\_recette, r.id

;

else

--profit par jeux

select r.id, r.date\_recette, SUM(r.profit) AS Profit\_Total from recette r

join (select ma.id, pie.id\_panne, j.id JeuID from machine\_de\_jeu ma

join (select \* from pieces p where p.id\_panne is null) pie

on pie.id\_machine\_jeu = ma.id

join jeu j

on j.id = ma.id\_jeu) tmp

on tmp.id = r.id\_machine\_de\_jeu

join panne pa

on pa.id = tmp.id\_panne

group by r.date\_recette, r.id

;

end;

Une image contenant texte, capture d’écran, conception

Description générée automatiquement  
**Captures d’écran de l’application**

La page présente une barre de navigation permettant d'accéder rapidement à différents tableaux disponibles sur la page principale. Chaque option de la barre de navigation correspond à un tableau spécifique présent sur la page principale. Lorsque l'utilisateur sélectionne l'une de ces options, il est dirigé directement vers la position du tableau désiré sur la page principale, facilitant ainsi la navigation et l'accès aux données pertinentes.

De plus, un bouton intitulé "Consulter les requêtes" est également inclus sur la barre de navigation. Lorsque l'utilisateur clique sur ce bouton, il est redirigé vers une autre page qui présente les différents types de requêtes possibles. Cette page peut inclure des informations sur les requêtes SQL ou les procédures stockées disponibles, ainsi qu'une brève description de leur fonctionnement et de leur utilisation.

En résumé, la barre de navigation offre une interface conviviale permettant aux utilisateurs de naviguer facilement entre les différents tableaux de la page principale et de consulter les types de requêtes disponibles, contribuant ainsi à une expérience utilisateur fluide et efficace.

**Une image contenant texte, capture d’écran, nombre, Rectangle

Description générée automatiquement**

La capture d’écran représente une interface d'application affichant un tableau SQL. En haut du tableau, il y a une barre de titre qui identifie probablement le contenu du tableau. Chaque ligne du tableau contient des données spécifiques de la base de données, et à côté de chaque ligne, il y a un bouton "Supprimer". Cette disposition suggère que les utilisateurs ont la possibilité de supprimer des entrées individuelles directement à partir du tableau en utilisant ces boutons.

L'interface semble offrir une manière conviviale pour visualiser et manipuler les données dans la table SQL, tout en facilitant la gestion en permettant aux utilisateurs de supprimer des éléments de manière sélective sans avoir à passer par des commandes plus complexes.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

La capture d’écran illustre *une page de rapport de casino* qui présente une variété d'options d'instructions SQL ou de procédures. Chaque option est représentée sous la forme d'un bouton lien (href) sur la page. Ces boutons permettent d'accéder à différentes requêtes ou procédures SQL préalablement définies pour générer des rapports spécifiques.

Parmi les options disponibles, on retrouve des rapports tels que :

1. Rapport de pièces par panne : Cette requête ou procédure SQL génère un rapport qui analyse les données des pièces par panne survenue dans le casino.

2. Liste de pannes : Cette option permet de générer un rapport qui répertorie toutes les pannes enregistrées dans une période donnée.

Chaque bouton lien conduit à une page HTML spécifique qui affiche le contenu du rapport correspondant une fois que la requête SQL associée est exécutée. Cette conception permet aux utilisateurs de sélectionner facilement le rapport qu'ils souhaitent consulter et d'accéder aux informations pertinentes de manière organisée et conviviale.

Ex : Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquementSur la droite de la page, un composant de type "slide" est présent, permettant aux utilisateurs d'insérer ou de mettre à jour des données dans trois tables spécifiques : JEU, Panneau d'affichage et Caméra de surveillance. Ce composant interactif offre une manière pratique et visuelle d'effectuer des opérations de modification ou d'insertion de données dans ces tables.

Lorsque l'utilisateur utilise cette diapositive, il peut sélectionner la table dans laquelle il souhaite effectuer des opérations en faisant glisser l'interface vers la gauche ou vers la droite. Une fois qu'il a sélectionné la table désirée, des champs de saisie ou d'autres éléments interactifs peuvent apparaître, permettant à l'utilisateur de saisir ou de modifier les données nécessaires.

Ce composant offre une approche conviviale pour interagir avec les données et facilite les opérations d'insertion et de mise à jour, en offrant une visibilité claire sur les différentes tables disponibles et en fournissant un moyen intuitif de travailler avec celles-ci.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementSur la droite de la page, un composant de type "slide" est présent, permettant aux utilisateurs d'insérer ou de mettre à jour des données dans trois tables spécifiques : JEU, Panneau d'affichage et Caméra de surveillance. Ce composant interactif offre une manière pratique et visuelle d'effectuer des opérations de modification ou d'insertion de données dans ces tables.

Lorsque l'utilisateur utilise cette diapositive, il peut sélectionner la table dans laquelle il souhaite effectuer des opérations en faisant glisser l'interface vers la gauche ou vers la droite. Une fois qu'il a sélectionné la table désirée, des champs de saisie ou d'autres éléments interactifs peuvent apparaître, permettant à l'utilisateur de saisir ou de modifier les données nécessaires.

Ce composant offre une approche conviviale pour interagir avec les données et facilite les opérations d'insertion et de mise à jour, en offrant une visibilité claire sur les différentes tables disponibles et en fournissant un moyen intuitif de travailler avec celles-ci.

Lien GitHub https://github.com/pomme2/Casino\_UdeM