Projet base de données

Base de préfab pour un Infinite Runner

Coutouly Julien

29/07/2022

To put in a nutshell…

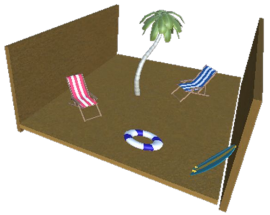
Sommaire

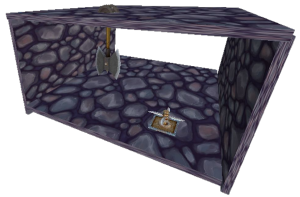
Cahier des charges

Contexte

- Infinite Runner sur Unity

- Prefabs générés aléatoirement

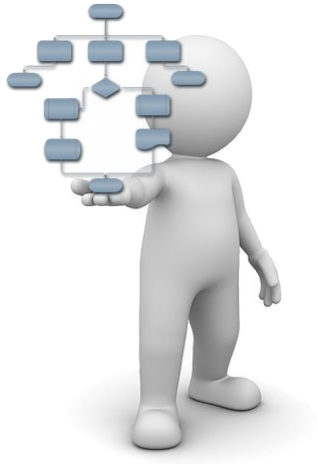




- BDD pour articuler les préfabs

Description fonctionnelles des besoins

- Tables de base : Prefabs, Obstacles, Author
- Table de liaison : Prefabs_Obstacles
- Triggers pour rendre la base autonome
- Vues pour simplifier l'utilisation
- Procédures stockées pour chercher des informations précises
- Gestion des permissions pour la sécurité

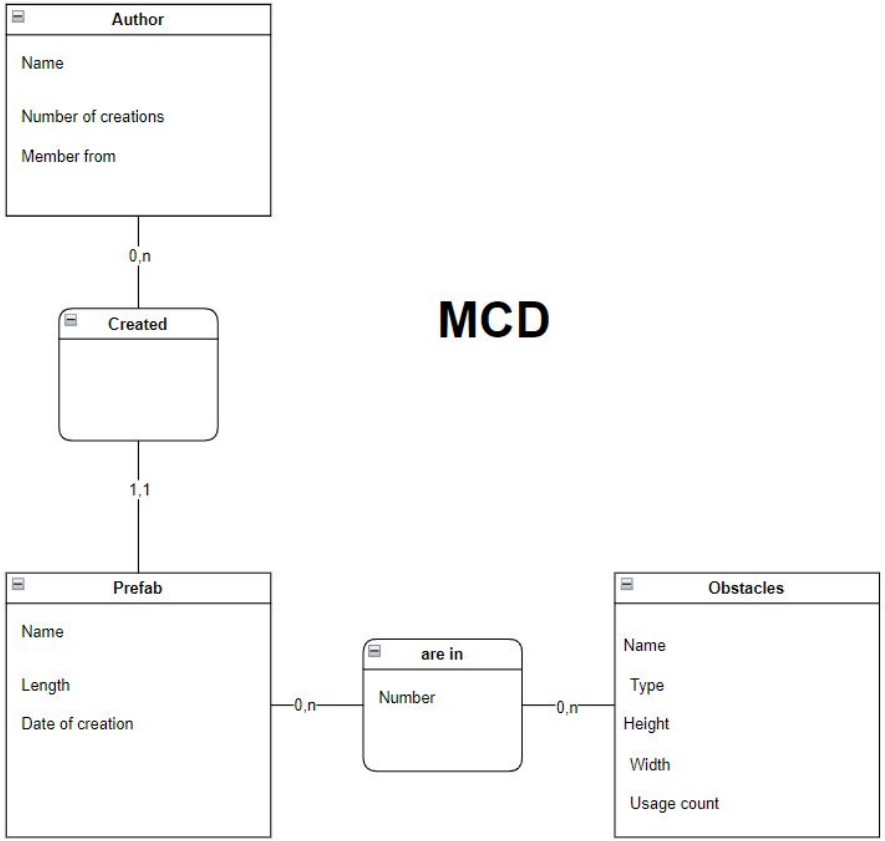


Modèles de la base de données

Modèle conceptuel des données

- Paramètres et relations des tables

- Cardinalités :
  - n,n ➜ table intermédiaire
  - 1,n ➜ clé étrangère

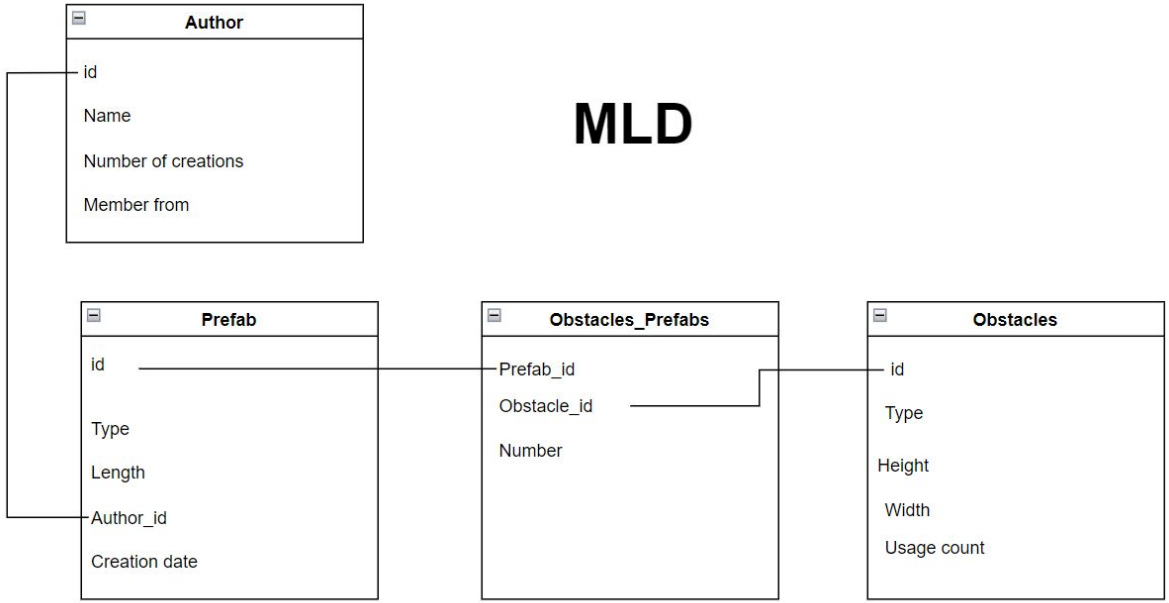


MCD

Modèle logique de données

Ajout des éléments suivants :

- Clés primaires id
- Clés étrangères
- Table Obstacles_Prefabs



Création de la base de données

Création de la base de données

Table Author

Chaque créateur sera représenté par :

- *Une clé primaire : id*
- *Un nom*
- *Un nombre de créations : triggered*
- *Une date d'arrivée*

Script :

```
CREATE TABLE Author(
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL UNIQUE,
        Number_of_creations INTEGER,
        Member_from DATETIME DEFAULT CURRENT_TIMESTAMP
        );
```

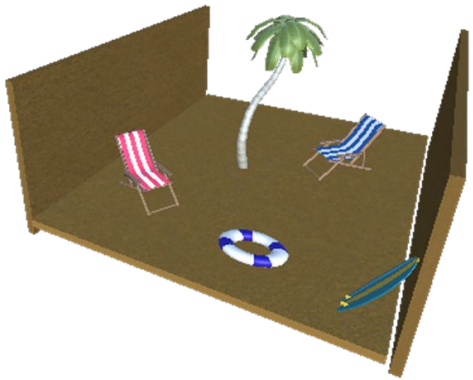
id	Name	Number_of_creations	Member_from
1	Benjamin	2	2022-07-17 21:00:08
2	xXKevinXx	1	2022-07-17 21:00:08
3	Beretdu73	1	2022-07-17 21:00:08

Création de la base de données

Table Prefabs

Chaque préfab sera représenté par :

- *Une clé primaire : id*
- *Un nom.*
- *Une longueur*
- *Une date de création.*
- *L'id du créateur : clé étrangère*



Script :

```
CREATE TABLE Prefabs(
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL UNIQUE,
        Length INTEGER NOT NULL,
        Creation_Date DATETIME DEFAULT CURRENT_TIMESTAMP,
        Author_id INTEGER NOT NULL UNIQUE,
        FOREIGN KEY(Author_id) REFERENCES Author(id) ON DELETE CASCADE
        );
```

id	Name	Length	Creation_Date	Author_id
1	Chemin enneigé	150	2022-07-18 20:26:39	2
2	Route de lave	120	2022-07-18 20:26:39	3
3	Forêt	120	2022-07-18 20:26:39	1
4	Plage	130	2022-07-18 20:26:39	1

Création de la base de données

Table Obstacles

Chaque obstacle sera représenté par :

- *Une clé primaire : id*
- *Un nom.*
- *Une hauteur*
- *Une largeur.*
- *Un compteur d'usage : triggered*

Script :

```
CREATE TABLE Obstacles(
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL UNIQUE,
        Height INTEGER NOT NULL,
        Width INTEGER NOT NULL,
        Usage_Count INTEGER NOT NULL DEFAULT 0
        );
```

id	Name	Height	Width	Type	Usage_Count
1	Arbre	50	10	Fixe	36
2	Piège à loup	15	15	Animé	5
3	Laser	5	200	Animé	4

Création de la base de données

Table Obstacles_Prefabs

La table permet d'ajouter l'entier *Number* d'obstacle, représenté par *Obstacle_id* à un préfab représenté par *Prefab_id.*

Script :

```
CREATE TABLE Obstacles_Prefabs(
        Number INTEGER,
        Prefab_id INTEGER,
        Obstacle_id INTEGER,
        FOREIGN KEY(Prefab_id) REFERENCES Prefabs(id) ON DELETE CASCADE,
        FOREIGN KEY(Obstacle_id) REFERENCES Obstacles(id) ON DELETE CASCADE
        );
```

Number	Prefab_id	Obstacle_id
5	1	2
12	1	1
4	2	3
6	2	1
18	3	1

Composants de la base de données

Composants de la base

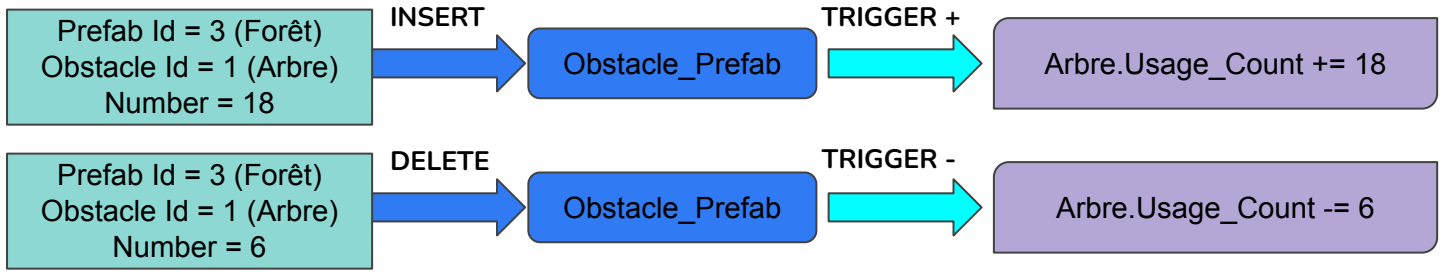
Trigger pour actualiser le nombre d'obstacle par préfab

Trigger +

```
DELIMITER &&
CREATE TRIGGER
Update_Usage_Count_On_Obstacle_Linked
AFTER INSERT ON Obstacles_Prefabs
FOR EACH ROW
BEGIN
        UPDATE Obstacles
        SET Usage_Count = Usage_Count +
NEW.Number
        WHERE id = NEW.Obstacle_id;
END&&
DELIMITER ;
```

Trigger -

```
DELIMITER &&
CREATE TRIGGER
Decrement_Usage_Count_On_Obstacle_Unlinked
AFTER DELETE ON Obstacles_Prefabs
FOR EACH ROW
BEGIN
        UPDATE Obstacles
        SET Usage_Count = Usage_Count -
OLD.Number
        WHERE id = OLD.Obstacle_id;
END&&
DELIMITER ;
```



Composants de la base

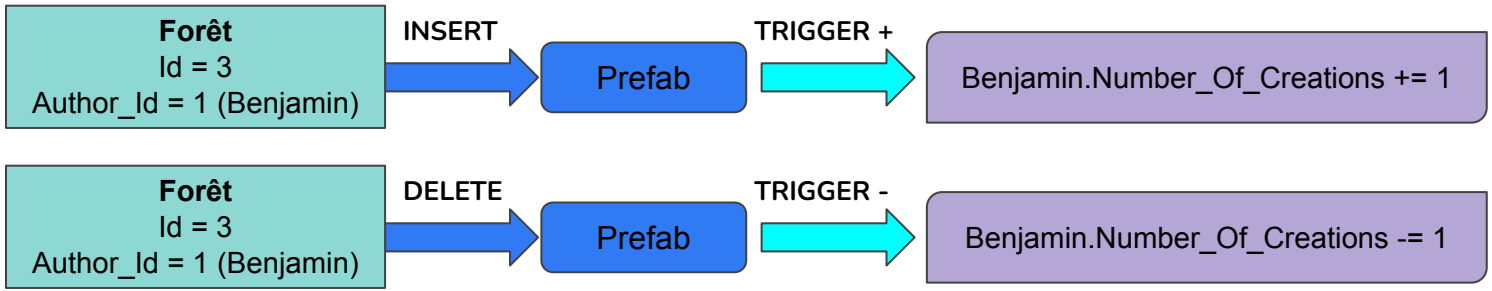
Trigger pour actualiser le nombre de créations d'un créateur

Trigger +

```
DELIMITER &&
CREATE TRIGGER
Update_Number_Of_Creations_On_Author_Linked
AFTER INSERT ON Prefabs
FOR EACH ROW
BEGIN
        UPDATE Author
        SET Number_of_creations = Number_of_creations + 1
        WHERE id = NEW.Author_id;
END&&
DELIMTER ;
```

Trigger -

```
DELIMITER &&
CREATE TRIGGER
Decrement_Number_Of_Creations_On_Author_Linked
        AFTER DELETE ON Prefabs
        FOR EACH ROW
BEGIN
        UPDATE Author
        SET Number_of_creations = Number_of_creations - 1
        WHERE id = OLD.Author_id;
END&&
DELIMITER ;
```



Composants de la base

Vue : Origine des préfab

Cette vue permet d'avoir toutes les informations importantes sur les préfabs.
Elle prend des informations dans les tables Préfabs et Authors.

id	Name	Length	Creation_Date	Author_Name	Number_of_creations	Member_from
1	Chemin enneigé	150	2022-07-18 20:26:39	xXKevinXx	1	2022-07-17 21:00:08
2	Route de lave	120	2022-07-18 20:26:39	Beretdu73	1	2022-07-17 21:00:08
3	Forêt	120	2022-07-18 20:26:39	Benjamin	2	2022-07-17 21:00:08
4	Plage	130	2022-07-18 20:26:39	Benjamin	2	2022-07-17 21:00:08

Script :

```
CREATE VIEW Prefabs_Origins AS
        SELECT  p.id,
         p.Name,
        p.Length,
        p.Creation_Date,
        a.Name as Author_Name,
        a.Number_of_creations,
        a.Member_from
        FROM Prefabs p
        JOIN Author a ON p.Author_id = a.id;
```

Composants de la base

Vue : Obstacles par préfab

Script :

```
CREATE VIEW Prefabs_with_Obstacles AS
        SELECT p.Name, GROUP_CONCAT(o.Name,', ') AS Obstacles
        FROM Prefabs p
        LEFT JOIN Obstacles_Prefabs op ON op.Prefab_id = p.id
        LEFT JOIN Obstacles o ON op.Obstacle_id  = o.id
        GROUP BY p.Name;
```

Name	Obstacles
Chemin enneigé	Piège à loup, ,Arbre,
Forêt	Arbre,
Plage	*NULL*
Route de lave	Arbre, ,Laser,

Cette vue permet de connaître facilement la composition en obstacles d'un préfab.

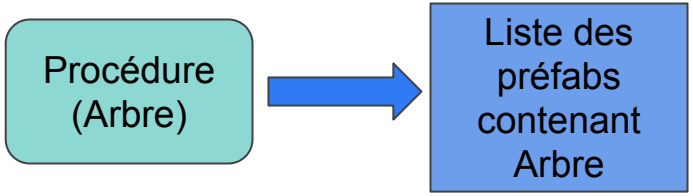
Composants de la base

Procédure : Trouver un préfab selon un obstacle

Exemple:

CALL getPrefab_By_Obstacle('Arbre');

Prefab_Name	Obstacle_Name	Number
Chemin enneigé	Arbre	12
Route de lave	Arbre	6
Forêt	Arbre	18



Script :

```
DELIMITER &&
CREATE OR REPLACE PROCEDURE getPrefab_By_Obstacle(
        IN obstacle VARCHAR(255)
)
BEGIN
        SELECT
        p.Name as Prefab_Name,
        o.Name as Obstacle_Name,
        op.Number
        FROM prefabs p
        INNER JOIN obstacles_prefabs op ON op.Prefab_id =
p.id
        INNER JOIN obstacles o ON op.Obstacle_id = o.id
        WHERE o.Name = obstacle;
END&&
DELIMITER ;
```

Cette procédure permet à l'aide de la requête CALL getPrefab('Nom_Obstacle') d'obtenir la liste des préfabs contenant l'obstacle recherché.

Composants de la base

Procédure : Trouver un préfab avec le plus d'obstacles

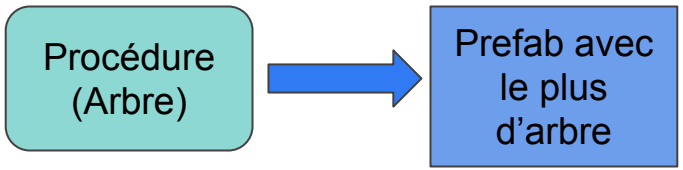
Exemple :

CALL getPrefab_More_Obstacle('Arbre',@Le_Plus_Abre);

Prefab_Name	Obstacle_Name	Number
Forêt	Arbre	18

SELECT @Le_Plus_Arbre

@Le_Plus_Arbre
Forêt



Cette procédure permet de trouver le préfab possédant la plus grande quantité de l'obstacle demandé.

Script :

```
DELIMITER &&
CREATE OR REPLACE PROCEDURE getPrefab_By_Obstacle(
        IN obstacle VARCHAR(255),
        OUT prefab_Name)
BEGIN
        SELECT p.Name INTO prefab_Name FROM(
                SELECT
                        p.Name as Prefab_Name,
                        o.Name as Obstacle_Name,
                        op.Number
                        FROM prefabs p
                        INNER JOIN obstacles_prefabs op ON op.Prefab_id = p.id
                        INNER JOIN obstacles o ON op.Obstacle_id = o.id
                        WHERE o.Name = obstacle
                        ORDER BY op.Number DESC
                        LIMIT 1;)
END&&
DELIMITER ;
```

Composants de la base

Gestion des utilisateurs

Script :

Readonly :
CREATE USER 'Readonly' IDENTIFIED BY '0000';
GRANT SELECT ON projet. TO 'Readonly';*

Administrateur:
CREATE USER 'Admin' IDENTIFIED BY 'AdminMDP';
GRANT ALL PERMISSIONS ON projet. TO 'Admin';*

Root :
CREATE USER 'root'@'localhost' IDENTIFIED BY 'rootMDP';
*GRANT ALL PERMISSIONS ON *.* TO 'root';*

Les utilisateurs **"Readonly"** n'ont accès à la base de données projet seulement en lecture.
Ils ne peuvent effectuer que des SELECT

Les utilisateurs de type **"Admin"** ont tous les droits sur notre base de données projet.

Le **root** a tous les droits sur toutes les bases.

Host	User	Password	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv
localhost	root	*EBC9CD04D27B65AEAAC12EBEFF7B06EFB0AFEF72	Y	Y	Y	Y	Y	Y
%	Readonly	*97E7471D816A37E38510728AEA47440F9C6E2585	Y	N	N	N	N	N
%	Admin	*428278163EF5B6A728D4E5F88E8E5C05324D9595	Y	Y	Y	Y	Y	Y

Conclusions et perspectives

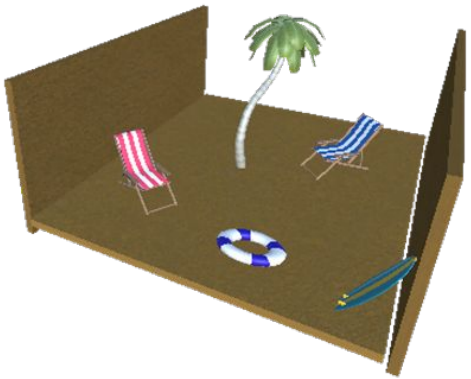
Situation de travail nécessitant une recherche

- DELIMITER sur le Trigger / Procédure
- Jointures dans les procédures
- Adaptation SQLite ➜ MySQL

Conclusions et perspectives

- Lien entre BDD et jeux vidéos
- Faire évoluer mon jeu
- Augmenter le nombre de préfabs disponibles

Nom du préfab : Mine
Créateur : Jold
Obstacle présents :
Hache et tourniquet
Longueur : 4m



Nom du préfab : Plage
Créateur : Jold
Obstacles présents : Palmier, chaise rouge, chaise bleu, bouée bleu et planche
Longueur : 4m

Merci de votre attention
Avez-vous des questions ?