

Projet base de données

Chronicles Runner : *un infinite runner basé sur les données persistantes*

Présenté par Simon Foucher – 02/09/2023



About this project...

- **Developed Unity3D game:** a database-centered project.
- **Unique concept:** Merge of infinite runner mechanics with collaborative treasure hunt.
- **Central role of persistent data:** Players discover unique fragments, claim ownership; managed via SQLite - tracks fragments, owners, dates, and enables features like player registration and level access.

Sommaire

- **About this project...**
- **Contexte et cahier des charges**
- **Conception de la base de données**
- **Mise en place de la base de données**
- **Développement des composants**
- **Situations de recherches**
- **Conclusion**

Contexte et cahiers des charges

The background of the slide is a solid orange color with a faint, low-poly geometric pattern. In the lower right quadrant, there is a faint, semi-transparent illustration of a person walking away from the viewer on a path that leads towards a large, glowing globe. The path is composed of small, light-colored stones or pebbles.

Contexte

- **Contexte de développement** : Infinite runner réalisé en un mois, dans le cadre d'une formation de Développeur de jeux vidéo.
- **Contraintes** : Utilisation d'Unity3D en C# et SQLite.
- **Une volonté d'intégrer les BDD au gameplay** : une trame narrative incitant les joueurs à interagir avec la base de données pour découvrir son contenu.



Objectifs et spécifications

Spécifications fonctionnelles de la base de données :

- > **Gérer les utilisateurs** : inscription sécurisée avec pseudonyme, mot de passe hashé et salt.
- > **Stocker et gérer les fragments d'histoire** : titre, contenu, époque, associés à un joueur et une date.
- > **Stocker et gérer les époques historiques** : informations sur les fragments disponibles, trouvés, etc.
- > **Suivi de progression** : enregistrer l'accès aux époques et les meilleurs scores des joueurs.
- > **Consultation des fragments** : récupérer les fragments par joueur ou par époque.
- > **Pas de procédures stockées et pas de gestion de droits** : en raison de l'utilisation de SQLite.

Conception de la base de données

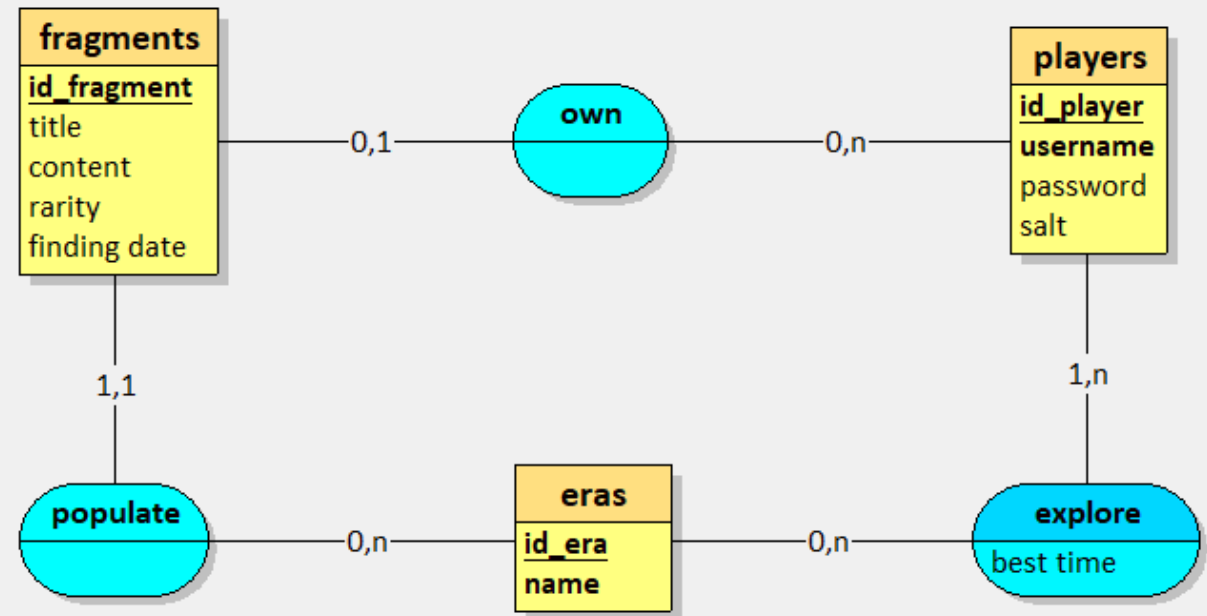


Modèle conceptuel des données

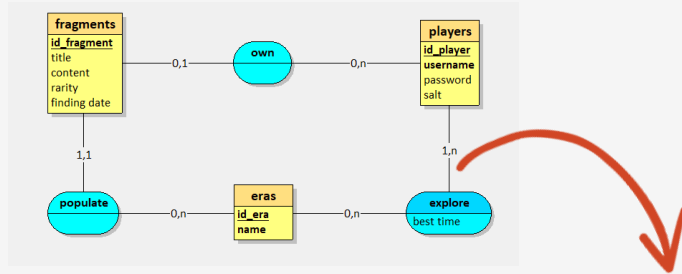
Nom	Signification	Type	Taille
id_player	Identifiant du joueur	N	
username	Pseudonyme du joueur	AN	32
password	Mot de passe hashé du joueur	AN	500
salt	Salt attribué au joueur	AN	64
id_era	Identifiant de la période historique	N	
name	Nom de la période historique	A	25
id_fragment	Identifiant du fragment d'histoire	N	
title	Titre du fragment d'histoire	AN	128
content	Contenu du fragment d'histoire	AN	1000
rarity	Rareté du fragment d'histoire (1 banal, 10 légendaire)	N	2
finding date	Date où le fragment a été trouvé en timestamp Unix	N	10
best time	Meilleur temps du joueur en millisecondes	N	10

À partir d'un dictionnaire des données :

- > Établir les associations entre les entités.
- > Définir les cardinalités minimales et maximales.
- > Définir les identifiants.
- > Définir les attributs.

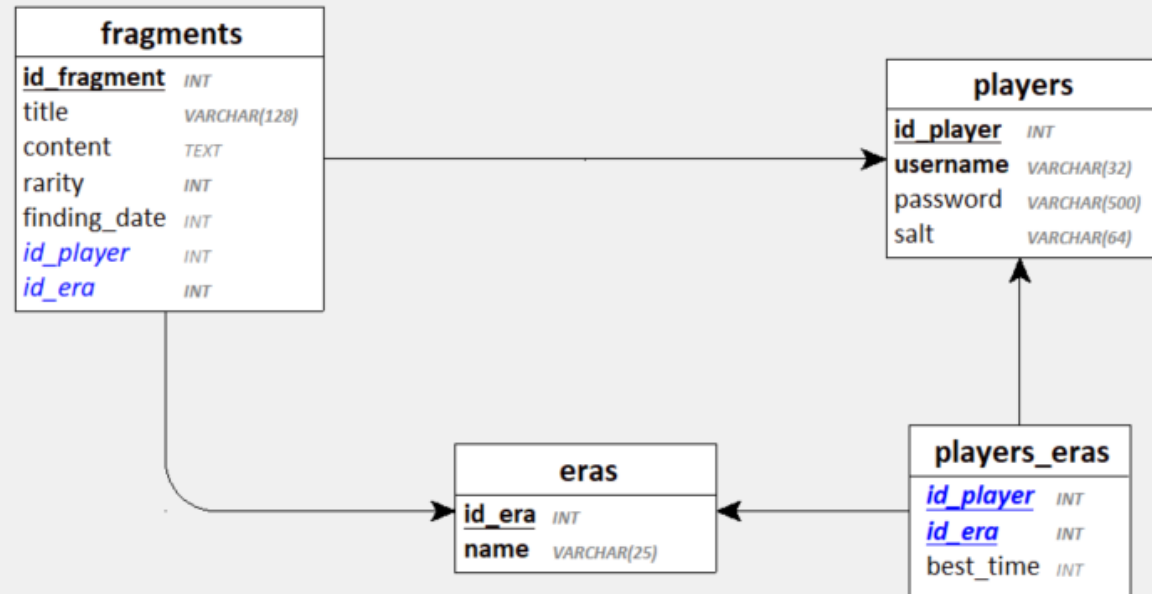


Modèle logique de données

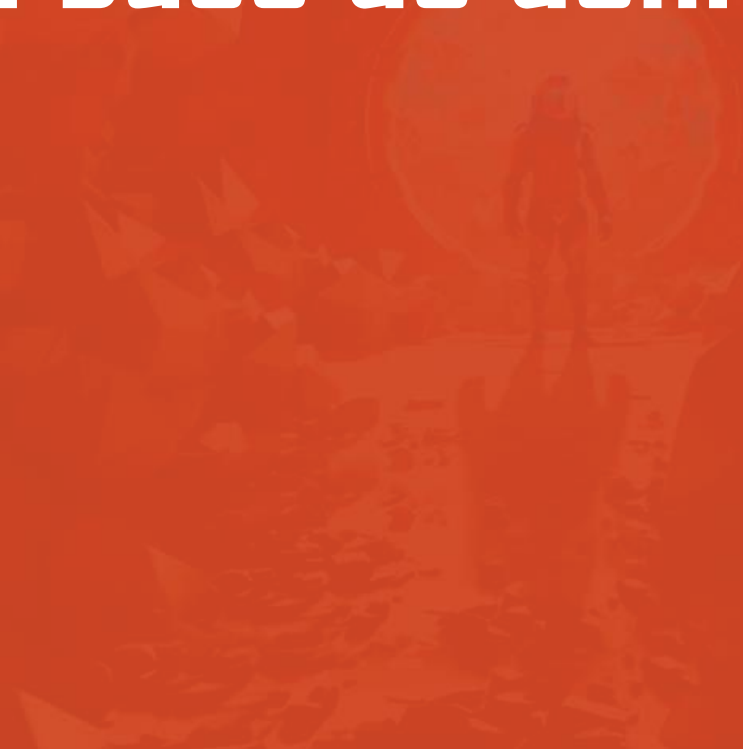


Définition des tables et des colonnes à partir du MCD :

- > **Association 1:n** : clé étrangère côté 1 référençant la clé primaire côté n.
- > **Association n:m** : création d'une table d'association avec deux clés étrangères formant la clé primaire, attribut comme colonne.
- > Les identifiants de players (username) et eras (name) deviennent des index.



Mise en place de la base de données



Création de la BDD

Table players :

```
CREATE TABLE players (  
  id_player INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,  
  username VARCHAR(32) NOT NULL UNIQUE,  
  password VARCHAR(500) NOT NULL,  
  salt VARCHAR(64) NOT NULL  
);
```

	id_player	username	password	salt
	Filtre	Filtre	Filtre	Filtre
1	3	Simon	B3im+OzgYWSpcymYEs7EUmzvy2yxhmqL2ch59...	NUHUiKpauXvJjxo4ICY6B1m4IuQfjSoz72xdOUvwx...
2	4	Tom	SITmaHnImT89Q4d/...	fjEks9ql5zDpjITE8XYOhZd+TM9TgVvF98pvScaR...
3	5	GamingCampus	ccn9jqulxyZicunLTbxCY/QmmB/...	b+Sq+/...
4	6	Mimi	hKSWq0BdrpGnHht0IymxwtzEseECPLrvdHD7DFS...	imSPMenClBupiOF5I6eWgJXZE4o2IUVvtyEdC6ib...
5	7	Meyriu	IYzJxYJTkAO17NN0JzQXhJrt+e+IRUhcX7SIBoMv9I...	CVpx+TQhVXNoIZTIZUBbextcOVHtjspleug3BlvN4w...
6	8	Oariskiller	tSFH71FGVwMRteDfFnY1VaO9YfdvXqbCaPhRTAg...	yiv6ruKglLicSKZMTythOWKE6kJDfyVQMfGOJxDd5...

- Identifiant id_player devient clé primaire auto-incrémentée.
- Username, password et salt : VARCHAR NOT NULL pour ne pas avoir de comptes fantômes.
- Username UNIQUE : pas de doublons possibles pour les comptes, il devient un index.

Table eras :

```
CREATE TABLE eras (  
  id_era INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,  
  name VARCHAR(25) NOT NULL UNIQUE  
);
```

	id_era	name
	Filtre	Filtre
1	1	préhistoire
2	2	far west
3	3	médiéval...

- Identifiant id_era devient clé primaire auto-incrémentée.
- Name : VARCHAR(25), NOT NULL UNIQUE, une époque ne peut pas être sans nom et deux époques ne peuvent avoir le même nom. Il devient un index.

Création de la BDD

Table fragments :

```
CREATE TABLE fragments (  
  id_fragment INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,  
  title VARCHAR(128) NOT NULL,  
  content TEXT NOT NULL,  
  rarity INTEGER NOT NULL DEFAULT 1,  
  finding_date INTEGER,  
  id_player INTEGER,  
  id_era INTEGER NOT NULL,  
  FOREIGN KEY(id_era) REFERENCES eras(id_era),  
  FOREIGN KEY(id_player) REFERENCES players(id_player)  
);
```

	id_fragment	title	content	rarity	finding_date	id_player	id_era
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
28	28	Légendes Ancestrales	"Dans un tournant inattendu, Jackson fit la ...	1	NULL	NULL	2
29	29	Histoires dans la Pierre	"C'est grâce à ces Indiens que Jackson découvrit ...	1	1692657671	3	2
30	30	Mosaïque du Passé	"Inspiré par cette nouvelle compréhension, Jacks...	1	NULL	NULL	2
31	31	Harmonie des Étoiles	"Les cristaux devinrent alors une mosaïque de ...	1	NULL	NULL	2
32	32	Murmures du Vent	"Alors que les étoiles brillaient au-dessus du Far ...	1	NULL	NULL	2
33	33	Flamme de Vérité	"Les vents du Far West murmuraient les histoire...	1	1692657671	3	2
34	34	Vérité Insolite	"La trahison du shérif Hayes fut un rappel amer ...	1	NULL	NULL	2

- Identifiant id_fragment devient clé primaire auto-incrémentée.
- Title et content NOT NULL : un fragment ne peut exister s'il n'y a pas d'histoire.
- Rarity INTEGER NOT NULL à DEFAULT 1 : ordinaire.
- Finding_date INTEGER : timestamp unix.
- Finding_date et id_player : peuvent être NULL si fragment pas encore découvert.
- Clés étrangères id_era et id_player référençant les clés primaires des tables eras et players.

















Table de liaison players_eras :

```
CREATE TABLE players_eras (  
  id_player INTEGER NOT NULL,  
  id_era INTEGER NOT NULL,  
  best_time INTEGER,  
  CONSTRAINT id PRIMARY KEY(id_player,id_era),  
  FOREIGN KEY(id_player) REFERENCES players(id_player),  
  FOREIGN KEY(id_era) REFERENCES eras(id_era)  
);
```

	id_player	id_era	best_time
	Filtre	Filtre	Filtre
1	4	9	NULL
2	3	8	235
3	6	2	NULL
4	5	6	147
5	5	1	53

- Clé primaire composée des clés étrangères id_player et id_era référençant les clés primaires des tables players et eras.
- Champ best_time INTEGER pouvant être NULL si pas encore de course.

Création de la BDD

Nom	Type	Schéma
▼  Tables (5)		
▼  eras		CREATE TABLE "eras" ("id_era" INTEGER NOT NULL UNIQUE, "name" VARCHAR(25) NOT NULL UNIQUE, PRIMARY KEY("id_era" AUTOINCREMENT))
 id_era	INTEGER	"id_era" INTEGER NOT NULL UNIQUE
 name	VARCHAR(25)	"name" VARCHAR(25) NOT NULL UNIQUE
▼  fragments		CREATE TABLE "fragments" ("id_fragment" INTEGER NOT NULL UNIQUE, "title" VARCHAR(128) NOT NULL, "content" TEXT NOT NULL, "rarity" INTEGER NOT NULL, "finding_date" INTEGER, "id_player" INTEGER, "id_era" INTEGER)
 id_fragment	INTEGER	"id_fragment" INTEGER NOT NULL UNIQUE
 title	VARCHAR(128)	"title" VARCHAR(128) NOT NULL
 content	TEXT	"content" TEXT NOT NULL
 rarity	INTEGER	"rarity" INTEGER NOT NULL DEFAULT 1
 finding_date	INTEGER	"finding_date" INTEGER
 id_player	INTEGER	"id_player" INTEGER
 id_era	INTEGER	"id_era" INTEGER NOT NULL
▼  players		CREATE TABLE "players" ("id_player" integer NOT NULL UNIQUE, "username" varchar(32) NOT NULL UNIQUE, "password" varchar(500) NOT NULL, "salt" varchar(64) NOT NULL)
 id_player	integer	"id_player" integer NOT NULL UNIQUE
 username	varchar(32)	"username" varchar(32) NOT NULL UNIQUE
 password	varchar(500)	"password" varchar(500) NOT NULL
 salt	varchar(64)	"salt" varchar(64) NOT NULL
▼  players_eras		CREATE TABLE "players_eras" ("id_player" INTEGER NOT NULL, "id_era" INTEGER NOT NULL, "best_time" INTEGER, FOREIGN KEY("id_era") REFERENCES "eras" ("id_era"))
 id_player	INTEGER	"id_player" INTEGER NOT NULL
 id_era	INTEGER	"id_era" INTEGER NOT NULL
 best_time	INTEGER	"best_time" INTEGER

Développement des composants



Trigger 1

Assigner une époque à explorer à l'inscription du joueur

```
CREATE TRIGGER t_assign_era_to_new_player AFTER INSERT ON players
✓ BEGIN
    INSERT INTO players_eras (id_player, id_era, best_time)
    VALUES (NEW.id_player, (SELECT id_era FROM eras ORDER BY RANDOM() LIMIT 1), NULL);
END;
```

players

19	22	Arthur	rL35iAV8xMGMHTDjjd/...	eKGvHKS8566kcOHY0Glln99BceGiDzh0ZNogZh3th...
20	23	Harfang	4PVJ+aVxu3reQamPhAzPuOkR69oJ4aHAYsl6Qnb...	F+aO5i6jTEZfdoCNihrFI5i6mKnV2UwzflQrzuEZIY=
21	24	Maxime	sHaREpdMNVwvu06HGmsZWPTqNScpl/...	mU1U19grPukWH+kxycezeKEVHNAF+VTW7nKUfa...

players_eras

28	22	3	1684766600
29	22	8	NULL
30	23	4	NULL

INSERT INTO players

19	22	Arthur	rL35iAV8xMGMHTDjjd/...	eKGvHKS8566kcOHY0Glln99BceGiDzh0ZNogZh3th...
20	23	Harfang	4PVJ+aVxu3reQamPhAzPuOkR69oJ4aHAYsl6Qnb...	F+aO5i6jTEZfdoCNihrFI5i6mKnV2UwzflQrzuEZIY=
21	24	Maxime	sHaREpdMNVwvu06HGmsZWPTqNScpl/...	mU1U19grPukWH+kxycezeKEVHNAF+VTW7nKUfa...
22	25	Eric	+WYpaCjRHXNg8Q+DbLzbgZQ4CVZxxpebT+ChC...	o/4ku8Lu0ePrEjYxa8Mj7GKikp58ul8SBPGL+IkLBXQ=

28	22	3	1684766600
29	22	8	NULL
30	23	4	NULL
31	25	1	NULL

Trigger 2

Libérer les fragments trouvés par un joueur si ce joueur est supprimé

```
CREATE TRIGGER t_delete_player BEFORE DELETE ON players
BEGIN
    DELETE FROM players_eras WHERE id_player = OLD.id_player;
    UPDATE fragments SET id_player = NULL, finding_date = NULL WHERE id_player = OLD.id_player;
END;
```

players

19	22	Arthur	rL35iAV8xMGMHTDjjd/...	eKGvHKS8566kcOHY0Glln998ceGiDzh0ZNogZh3tH...
20	23	Harfang	4PVJ+aVxu3reQamPhAzPuOkR69oJ4aHAYsl6Qnb...	F+aO5i6jTEZfdoCNiihrFI5i6mKnV2UwzflQrzuEZIY=
21	24	Maxime	sHaREpdMNVvvu06HGmsZWPTqNScpl/...	mU1U19grPukWH+kxvycezKEVHNAF+VTW7nKUfa...
22	25	Eric	+WYpaCjRHXNg8Q+DbLzbqZQ4CVZxXpebT+ChC...	o/4ku8Lu0ePrEjYxa8Mj7Gkikp58ul8SBPGL+IkLBXQ=

fragments

148	148	Toile Cachée d'un Monde Oublié	Sous la lueur des étoiles, Takeshi se plonge da...	1	NULL	NULL	8
149	149	Secrets Cachés dans les Pierres du Temps	Le temps s'écoulait comme les pétales d'une fleu...	1	1692657333	25	8
150	150	Les Toiles Magiques d'Hiroshi	Parmi les histoires entrelacées, celle d'un peintre...	1	1693261542	23	8
151	151	Émotions Vivantes dans les Reflets	Dans les reflets énigmatiques de ses tableaux, le...	1	NULL	NULL	8
152	152	Encre des Créatures Scellées	Une légende racontait que l'encre utilisée pour c...	1	1692657490	25	8
153	153	Lame Gravée de Symboles Ésotériques	Convaincu de la valeur de cette encre envoûtant...	1	NULL	NULL	8

DELETE FROM players

players

19	22	Arthur	rL35iAV8xMGMHTDjjd/...	eKGvHKS8566kcOHY0Glln998ceGiDzh0ZNogZh3tH...
20	23	Harfang	4PVJ+aVxu3reQamPhAzPuOkR69oJ4aHAYsl6Qnb...	F+aO5i6jTEZfdoCNiihrFI5i6mKnV2UwzflQrzuEZIY=
21	24	Maxime	sHaREpdMNVvvu06HGmsZWPTqNScpl/...	mU1U19grPukWH+kxvycezKEVHNAF+VTW7nKUfa...

148	148	Toile Cachée d'un Monde Oublié	Sous la lueur des étoiles, Takeshi se plonge da...	1	NULL	NULL	8
149	149	Secrets Cachés dans les Pierres du Temps	Le temps s'écoulait comme les pétales d'une fleu...	1	NULL	NULL	8
150	150	Les Toiles Magiques d'Hiroshi	Parmi les histoires entrelacées, celle d'un peintre...	1	1693261542	23	8
151	151	Émotions Vivantes dans les Reflets	Dans les reflets énigmatiques de ses tableaux, le...	1	NULL	NULL	8
152	152	Encre des Créatures Scellées	Une légende racontait que l'encre utilisée pour c...	1	NULL	NULL	8
153	153	Lame Gravée de Symboles Ésotériques	Convaincu de la valeur de cette encre envoûtant...	1	NULL	NULL	8

Vue 1

Vue permettant de lister tous les fragments d'un joueur

```
CREATE VIEW v_fragments_by_player AS
✓ SELECT players.id_player, players.username,
      GROUP_CONCAT(fragments.id_fragment) AS list_fragments
FROM players
LEFT JOIN fragments ON players.id_player = fragments.id_player
GROUP BY players.id_player;
```

	id_player	username	list_fragments
1	3	Simon	29,33,40,114,118,164,165
2	4	Tom	NULL
3	5	GamingCampus	113,121
4	6	Mimi	NULL

Vue 2

Vue permettant d'afficher toutes les informations nécessaires à l'affichage d'un fragment in game

```
CREATE VIEW v_fragment_info AS
SELECT fragments.id_fragment, fragments.title, fragments.content, fragments.rarity, fragments.finding_date,
       players.id_player, players.username, eras.id_era, eras.name
FROM fragments
JOIN players ON fragments.id_player = players.id_player
JOIN eras ON fragments.id_era = eras.id_era;
```

	id_fragment	title	content	rarity	finding_date	id_player	username	id_era	name
1	29	Histoires dans la Pierre	"C'est grâce à ces Indiens que Jackson découvrit ...	1	1692657671	3	Simon	2	far west
2	33	Flamme de Vérité	"Les vents du Far West murmuraient les histoire...	1	1692657671	3	Simon	2	far west
3	40	Héritage Envolé	"Et ainsi, les cristaux du Far West furent dispers...	1	1692657666	3	Simon	2	far west
4	42	L'Énigme de la Lame d'Héritage	"Les gravures énigmatiques ornant une antique ...	1	1693261111	23	Harfang	3	médiéval...
5	52	Éclats d'une Symphonie Inachevée	"Des fragments cristallins captaient la tourmente ...	1	1693261111	23	Harfang	3	médiéval...

Procédure stockée 1

Pas de procédures stockées en SQLite ! Cependant, si nous étions sur MySQL nous aurions pu imaginer...

Afficher les fragments pour une époque et un joueur donnés dans un ordre voulu : croissant, décroissant ou aléatoire

```
DELIMITER //
```

```
CREATE PROCEDURE GetFragmentsByEraAndUser(  
    IN eraID INT,  
    IN userID INT,  
    IN orderBy VARCHAR(10)  
)  
BEGIN  
    IF orderBy = "asc" THEN  
        SELECT * FROM fragments  
        WHERE id_era = eraID AND id_player = userID  
        ORDER BY finding_date ASC;  
    ELSEIF orderBy = "desc" THEN  
        SELECT * FROM fragments  
        WHERE id_era = eraID AND id_player = userID  
        ORDER BY finding_date DESC;  
    ELSEIF orderBy = "random" THEN  
        SELECT * FROM fragments  
        WHERE id_era = eraID AND id_player = userID  
        ORDER BY RAND();  
    END IF;  
END //
```

```
DELIMITER ;
```

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

CALL GetFragmentsByEraAndUser(6, 26, "desc");

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_fragment	title	content	rarity	finding_date	id_player	id_era
11	Les Toiles Magiques d'Hiroshi	Parmi les histoires entrelacées, celle d'un peintre...	1	1693261542	26	6
12	Les Destins Entremêlés	Les époques s'entrelaçaient, les histoires fusionn...	1	1693261401	26	6
10	Lueur Énigmatique	"Le souffle du passé était palpable. Les images an...	1	1692655963	26	6

Showing rows 0 - 2 (3 total, Query took 0.0011 seconds.)

CALL GetFragmentsByEraAndUser(6, 26, "random");

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_fragment	title	content	rarity	finding_date	id_player	id_era
10	Lueur Énigmatique	"Le souffle du passé était palpable. Les images an...	1	1692655963	26	6
12	Les Destins Entremêlés	Les époques s'entrelaçaient, les histoires fusionn...	1	1693261401	26	6
11	Les Toiles Magiques d'Hiroshi	Parmi les histoires entrelacées, celle d'un peintre...	1	1693261542	26	6

Procédure stockée 2

Pas de procédures stockées en SQLite ! Cependant, si nous étions sur MySQL nous aurions pu imaginer...

Afficher le meilleur joueur de chaque époque via un calcul de score incluant nombre de fragments et meilleur temps

```
DELIMITER //
CREATE PROCEDURE GetTopPlayersByEra(
    IN eraID INT,
    IN topNB INT
)
BEGIN
    SELECT players.id_player, players.username, players_eras.best_time,
    COUNT(fragments.id_fragment) AS num_fragments,
    (COUNT(fragments.id_fragment) / GREATEST(players_eras.best_time, 1)) AS score
    FROM players
    JOIN players_eras ON players.id_player = players_eras.id_player AND players_eras.id_era = eraID
    LEFT JOIN fragments ON players.id_player = fragments.id_player AND fragments.id_era = eraID
    GROUP BY players.id_player, players.username, players_eras.best_time
    ORDER BY score DESC
    LIMIT topNB;
END //
DELIMITER ;
```

Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

CALL GetTopPlayersByEra(6,3);

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_player	username	best_time	num_fragments	score
26	Jack	45	3	0.0667
22	Caro	134	4	0.0299
21	Arnaud	32	0	0.0000

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

CALL GetTopPlayersByEra(6,3);

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_player	username	best_time	num_fragments	score
22	Caro	134	4	0.0299
26	Jack	105	3	0.0286
21	Arnaud	32	0	0.0000

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Aspect sécurité

- > Lecture des recommandations de l'ANSSI et CNIL :
- > Hashage via **PBKDF2**.
- > Salage de plus de 128 bits (256).
- > Mots de passe complexes via regex.
- > Sessions expirables avec token **JWT**.
- > Requêtes préparées contre injections SQL.
- > Ajout côté jeu d'un backup de la BDD.

```
1 référence
public string GenerateToken(int id, string username)
{
    int date = (int)DateTimeOffset.UtcNow.AddHours(2).ToUnixTimeSeconds();

    var payload = new Dictionary<string, object>
    {
        { "id", id },
        { "username", username },
        { "exp", date }
    };

    // Création de l'encodeur JWT
    IJwtAlgorithm algorithm = new HMACSHA256Algorithm();
    IJsonSerializer serializer = new JsonSerializer();
    IBase64UrlEncoder urlEncoder = new JwtBase64UrlEncoder();
    IJwtEncoder encoder = new JwtEncoder(algorithm, serializer, urlEncoder);

    // Génération du token
    string token = encoder.Encode(payload, DBConstant.JWTSecret);

    return token;
}
```

```
4 références
public class PasswordEncrypter : IPasswordSecurity
{
    /// <summary> Modify for the project by Simon F. to use RNGCryptoServiceProvider ...
    3 références
    public string CreateSalt()
    {
        byte[] salt = new byte[32];

        using (RNGCryptoServiceProvider rngCsp = new RNGCryptoServiceProvider())
        {
            rngCsp.GetBytes(salt);
        }

        return Convert.ToBase64String(salt);
    }

    /// <summary> Hashes password using 1000 iterations
    3 références
    public string HashPassword(string password, string salt)
    {
        var hashWithSalt = string.Format("{0}:{1}", password, salt);
        var saltBytes = Encoding.UTF8.GetBytes(hashWithSalt);
        using (var rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, saltBytes, 1000))
        {
            return Convert.ToBase64String(rfc2898DeriveBytes.GetBytes(256));
        }
    }

    /// <summary> Compares two passwords using a compare in length-constant time.
    1 référence
    public bool Compare(string hashedPassword1, string hashedPassword2)
    {
        var hash1 = Convert.FromBase64String(hashedPassword1);
        var hash2 = Convert.FromBase64String(hashedPassword2);
        return SlowEquals(hash1, hash2);
    }

    /// <summary> Compares two byte arrays in length-constant time. This comparison ...
    1 référence
    public static bool SlowEquals(byte[] a, byte[] b)
    {
        var diff = (uint)a.Length ^ (uint)b.Length;
        for (var i = 0; i < a.Length && i < b.Length; i++)
            diff |= (uint)(a[i] ^ b[i]);
        return diff == 0;
    }
}
```

Nom

20230829003309_backup.db
20230829003314_backup.db
20230829003323_backup.db

```
@ Script Unity (1 référence de ressource) | 0 références
public class DatabaseBackup : MonoBehaviour
{
    private string databasePath;
    private string backupFolderPath;
    private int maxBackupCount = 3;

    @ Message Unity | 0 références
    private void Start()
    {
        databasePath = Application.streamingAssetsPath + "/Database/database.db";
        backupFolderPath = Application.persistentDataPath + "/Database_saves/";
        BackupDatabase();
    }

    1 référence
    private void BackupDatabase()
    {
        try
        {
            if (!Directory.Exists(backupFolderPath))
            {
                Directory.CreateDirectory(backupFolderPath);
            }

            string backupFileName = DateTime.Now.ToString("yyyyMMddHHmmss") + "_backup.db";
            string backupFilePath = Path.Combine(backupFolderPath, backupFileName);

            File.Copy(databasePath, backupFilePath);

            CleanupBackups();

            Debug.Log("Backup successful: " + backupFileName);
        }
        catch (Exception e)
        {
            Debug.LogError("Backup failed: " + e.Message);
        }
    }
}
```

Gestion des droits

Pas de gestion de droits en SQLite ! Cependant, si nous étions sur MySQL nous aurions pu imaginer...

> **User "readOnly"** : Accès en lecture seule à certaines informations (fragments, époques, utilisateurs). Pas d'accès aux mots de passe/salts.

> **User "game"** : Correspondant à Unity3D. Contrôle sur les actions liées aux utilisateurs. Restrictions sur les autres tables pour éviter les erreurs.

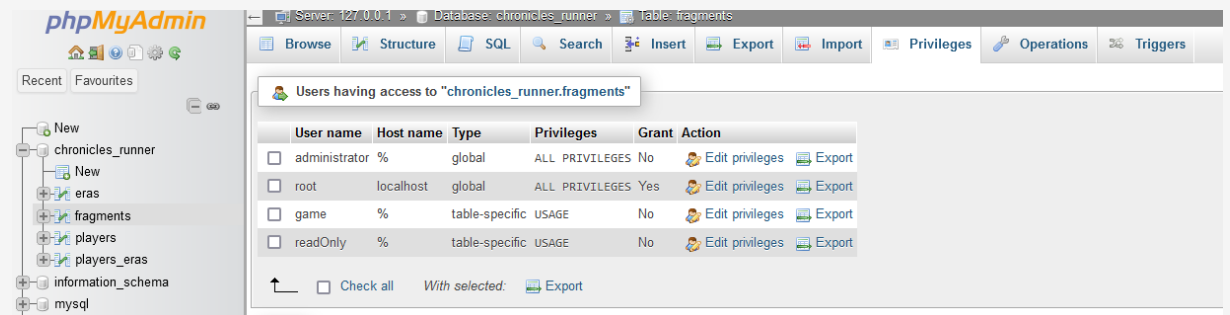
> **User "administrator"** : Privilèges étendus pour la gestion à distance. Utilisation sensible avec mot de passe fort.

> **User "root"** : pour l'administration en local.

```
CREATE USER 'readOnly'@'%' IDENTIFIED BY '0000';
GRANT SELECT ON fragments TO 'readOnly'@'%';
GRANT SELECT ON eras TO 'readOnly'@'%';
GRANT SELECT (id_player, username) ON players TO 'readOnly'@'%';
GRANT SELECT ON players_eras TO 'readOnly'@'%';
```

```
CREATE USER 'game'@'%' IDENTIFIED BY 'gamePassword';
GRANT INSERT, UPDATE, SELECT, DROP ON players TO 'game'@'%';
GRANT SELECT ON fragments TO 'game'@'%';
GRANT INSERT, UPDATE (id_player, finding_date) ON fragments TO 'game'@'%';
GRANT SELECT ON eras TO 'game'@'%';
```

```
CREATE USER 'administrator'@'%' IDENTIFIED BY 'administratorStrongPassword';
GRANT ALL PRIVILEGES ON *.* TO 'administrator'@'%';
```



The screenshot shows the phpMyAdmin interface with the 'Privileges' tab selected for the 'chronicles_runner' database. The table 'Users having access to "chronicles_runner.fragments"' displays the following data:

User name	Host name	Type	Privileges	Grant	Action
<input type="checkbox"/> administrator	%	global	ALL PRIVILEGES	No	Edit privileges Export
<input type="checkbox"/> root	localhost	global	ALL PRIVILEGES	Yes	Edit privileges Export
<input type="checkbox"/> game	%	table-specific	USAGE	No	Edit privileges Export
<input type="checkbox"/> readOnly	%	table-specific	USAGE	No	Edit privileges Export

At the bottom, there are checkboxes for 'Check all' and 'With selected:' followed by an 'Export' button.

Recherches



Situations de recherche

- > **Sécurisation des mots de passe** : choisir l'algorithme approprié, implémenter PBKDF2 dans Unity3D.
- > **Création du système de sessions** : découverte de JWT.
- > **Placement exact des sous-requêtes**
- > **Adaptation SQLite > MySQL pour le futur**

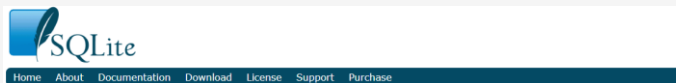



Table Of Contents

1. Syntax

create-trigger-stmt:

```
CREATE [TEMP | TEMPORARY] TRIGGER IF NOT EXISTS
    schema-name trigger-name BEFORE | AFTER
    (DELETE | INSERT | UPDATE) OF column-name
    ON table-name
```

The diagram illustrates the syntax for creating a trigger in SQLite. It shows a flowchart starting with 'CREATE' and 'TRIGGER' keywords, followed by optional 'TEMP' or 'TEMPORARY' keywords. The trigger is defined by a 'schema-name', a 'trigger-name', and a timing ('BEFORE' or 'AFTER'). The trigger is associated with an event ('DELETE', 'INSERT', or 'UPDATE') and an optional 'OF' clause followed by a 'column-name'. Finally, the trigger is attached to a 'table-name' using the 'ON' keyword.



SQLite Subquery

If this SQLite tutorial saves you hours of work, please whitelist it in your ad blocker or donate now to support us in paying for web hosting to keep the website running.

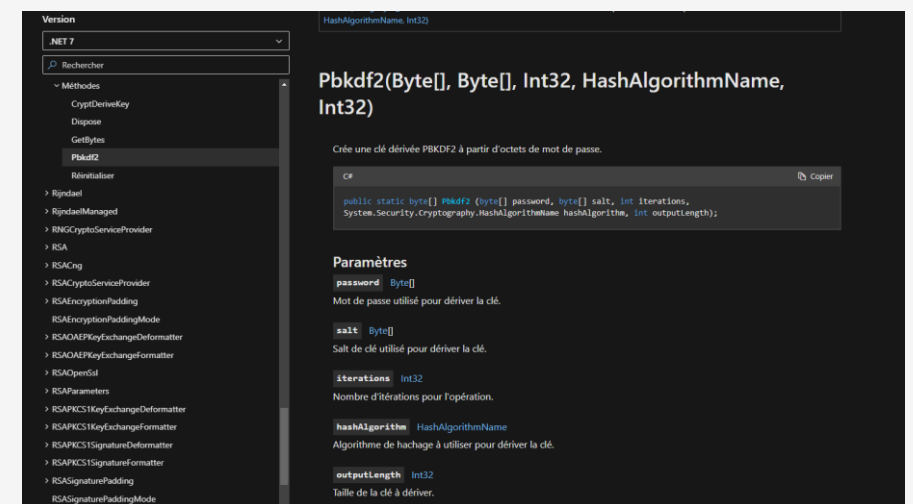
Summary: In this tutorial, you will learn about the SQLite subquery to construct more readable and complex queries.

Introduction to SQLite subquery

A subquery is a **SELECT** statement nested in another statement. See the following statement.

```
SELECT column_1
FROM table_1
WHERE column_1 = (
    SELECT column_1
    FROM table_2
);
```

The following query is the **outer query**.



Visual Studio Code interface showing the **Pbkdf2** method in the **System.Security.Cryptography** namespace. The method signature is **Pbkdf2(Byte[], Byte[], Int32, HashAlgorithmName, Int32)**. The description states: "Crée une clé dérivée PBKDF2 à partir d'octets de mot de passe." The parameters are:

- password** (Byte[]): Mot de passe utilisé pour dériver la clé.
- salt** (Byte[]): Salt de clé utilisé pour dériver la clé.
- iterations** (Int32): Nombre d'itérations pour l'opération.
- hashAlgorithm** (HashAlgorithmName): Algorithme de hachage à utiliser pour dériver la clé.
- outputLength** (Int32): Taille de la clé à dériver.

Exemple : PBKDF2

S'informer



Découvrir

PBKDF2

PBKDF2 is a simple cryptographic key derivation function, which is resistant to **dictionary attacks** and **rainbow table attacks**. It is based on iteratively deriving HMAC many times with some padding. The PBKDF2 algorithm is described in the Internet standard [RFC 2898 \(PKCS #5\)](#).

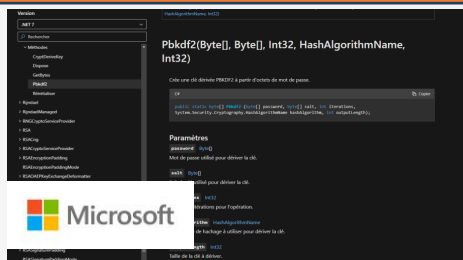
PBKDF2 takes several **input parameters** and produces the **derived key** as output:

```
key = pbkdf2(password, salt, iterations-count, hash-function, derived-key-len)
```

Technically, the **input data** for PBKDF2 consists of:

- password -- array of bytes / string, e.g. "p@ssw0rd-3" (8-10 chars minimal length is recommended)
- salt -- security-generated random bytes, e.g. "d1f2d3d4d77acde6fc5dc3d8f921b4" (minimum 64 bits, 128 bits is recommended)
- iterations-count, e.g. 1024 iterations
- hash-function for calculating HMAC, e.g. SHA256
- derived-key-len for the output, e.g. 32 bytes (256 bits)

The **output data** is the **derived key** of requested length (e.g. 256 bits).



Comprendre

The right way to implement password hashing using PBKDF2 and C#

Of [May 2014](#) posted security, [exploits](#), or [Following from my previous post about building using Bcrypt](#) and in response to some comments I received on [Google+](#) - I decide to provide an alternative hashing implementation using PBKDF2.

As you will notice, the implementation is somewhat bigger than the one provided for Bcrypt but in effect, both code segments perform the same task. First we create a hash from the plain text password and then we validate a password against the stored hash.

NOTE: The constants, like the iterations, can be changed to meet the hash strength.

```
using System;
using System.Text;
using System.Security.Cryptography;

namespace PasswordHashing
{
    public class PasswordHash
    {
        public static byte[] HashPassword(string password)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash;
        }

        public static bool ValidatePassword(string password, byte[] storedHash)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash.SequenceEqual(storedHash);
        }
    }
}
```



Hashing and Salting Passwords in C# With PBKDF2

PBKDF2 is a key derivation function that we use to generate password-based keys. It is based on a pseudo-random function (PRF) applied iteratively to a password and a salt.

We will use the `SHA256` function (SHA-256) as our PRF. The algorithm takes several parameters including a key to generate cryptographically secure keys from, a salt, a pseudo-random function (PRF), and a number of iterations.

```
using System;
using System.Text;
using System.Security.Cryptography;

namespace PasswordHashing
{
    public class PasswordHash
    {
        public static byte[] HashPassword(string password)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash;
        }

        public static bool ValidatePassword(string password, byte[] storedHash)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash.SequenceEqual(storedHash);
        }
    }
}
```

Hashing and Salting Passwords in C# With PBKDF2

PBKDF2 is a key derivation function that we use to generate password-based keys. It is based on a pseudo-random function (PRF) applied iteratively to a password and a salt.

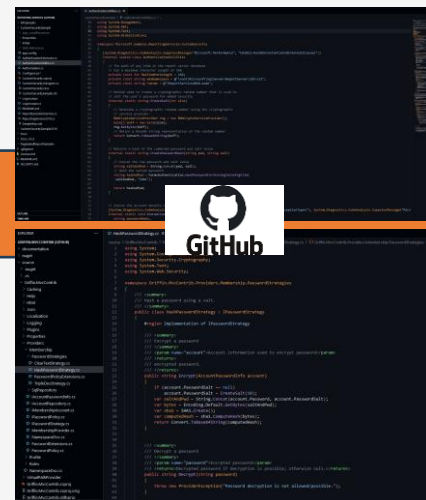
We will use the `SHA256` function (SHA-256) as our PRF. The algorithm takes several parameters including a key to generate cryptographically secure keys from, a salt, a pseudo-random function (PRF), and a number of iterations.

```
using System;
using System.Text;
using System.Security.Cryptography;

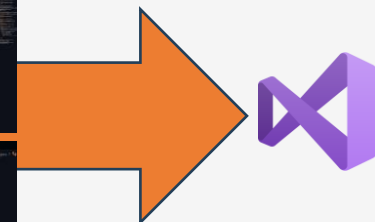
namespace PasswordHashing
{
    public class PasswordHash
    {
        public static byte[] HashPassword(string password)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash;
        }

        public static bool ValidatePassword(string password, byte[] storedHash)
        {
            var cryptographic = new HMACSHA256(password);
            byte[] salt = new byte[16];
            RandomNumberGenerator.GetBytes(salt);
            byte[] hash = cryptographic.ComputeHash(salt);
            return hash.SequenceEqual(storedHash);
        }
    }
}
```

Détailer



Implémenter



Conclusion



Vers MySQL ?

- > **Beaucoup de plaisir** à développer *Chronicles Runner* et à voir les fragments d'histoire prendre forme.
- > **SQLite** : rapide, flexible et embarqué, idéal pour du prototypage, mais pas adapté à un jeu collaboratif.
- > **Envie d'aller vers MySQL pour franchir un cap supplémentaire :**
 1. Audit à effectuer sur les spécificités de MySQL.
 2. Création de la base de données sur MySQL via PhpMyAdmin en local.
 3. Migration des données via dump ou export/import CSV.
 4. Transposition des composants avec ajout des procédures stockées.
 5. Migration des requêtes vers des fichiers PHP.
 6. Adaptation de la syntaxe côté Unity3D pour prendre en compte PHP.
 7. Tests.
 8. Dump de la BDD locale vers BDD hébergée.

Merci.

Avez-vous des questions ?