



Rapport Final

Plateforme d'apprentissage

(MOOC)

MIASHS DCISS M2 RS

Conception de bases de données relationnelles

Chi ZHANG (Numéro d'étudiant : 10468200)

Trinh Minh Phuong TA (Numéro d'étudiant : 12312527)

1. Introduction

Ce rapport présente la conception d'une base de données pour Les MOOC (Massive Open Online Courses), en détaillant les choix de conception, la justification des relations entre les entités et l'adéquation du schéma aux besoins du système.

Espaces de travail:

Trello : <https://trello.com/b/JcYxEt8o/projet-bdd>

Github : https://github.com/tamipu/Projet-de-conception-BDD2_TA_ZHANG

Vertabelo : <https://my.vertabelo.com/model/sxs0hqosXnAXOyeHeSea8Y5xHRMUncR8>

2. Installation

Après avoir téléchargé trois fichiers, dont un script SQL pour créer une structure de base de données (`database_create.sql`), un script SQL pour insérer des données (`database_data.sql`) et un script SQL pour les contraintes (`database_constraints.sql`). Voici comment vous pouvez les utiliser dans DBeaver :

1. Création de la structure de la base

- Connectez-vous à votre base de données PostgreSQL.
- Ouvrez le fichier `database_create.sql` avec votre éditeur SQL.
- Exécutez le script pour créer la structure de votre base de données (tables, index, etc.).

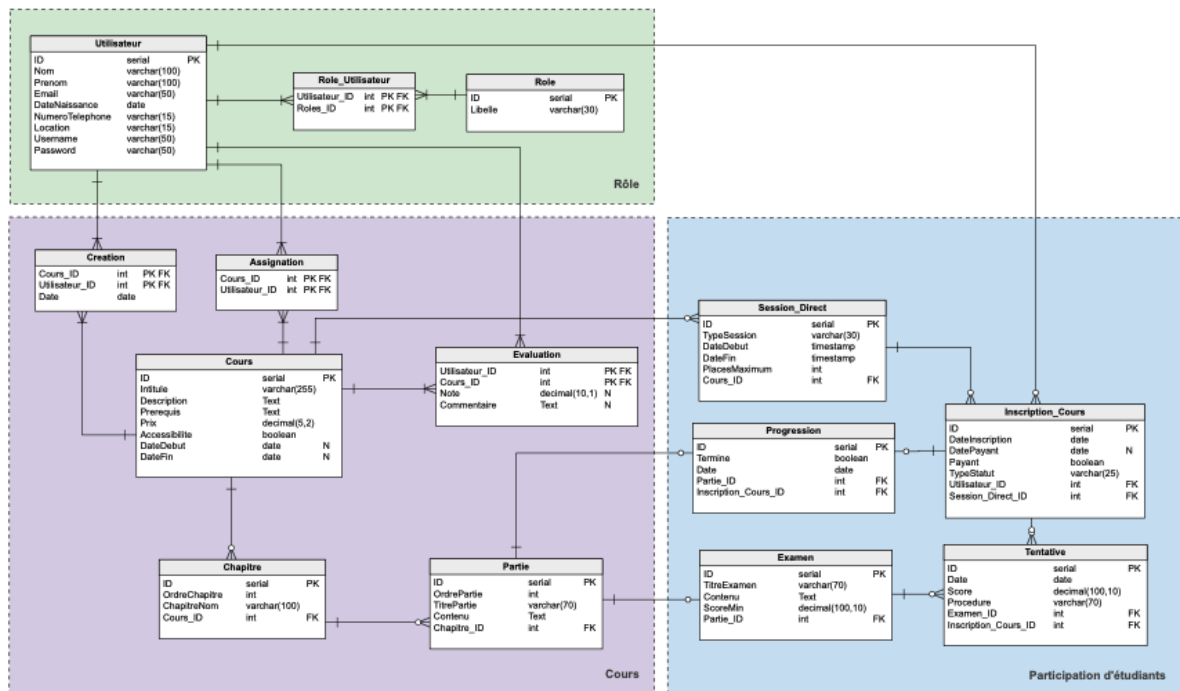
2. Exécuter le script d'insertion de données (`.sql`)

- De manière similaire, ouvrez le fichier `database_data.sql` dans DBeaver.
- Assurez-vous que la bonne base de données est sélectionnée et exécutez le script. Ceci va insérer les données dans les tables que vous avez créées précédemment.

! S'il y a des erreurs, merci d'essayer de créer les tables et insérer les données dans votre interface de commande SQL (comme psql). Connectez-vous à votre serveur PostgreSQL et sélectionnez la base de données appropriée en utilisant `c nom_base de données`.

3. Conception et Diagrammes

Les choix de conception ont été orientés par les besoins fonctionnels du système et les standards reconnus en matière de modélisation de données. Nous avons opté pour une division de notre architecture en trois segments : les rôles, les cours, et la participation des étudiants.



Les rôles :

Au sein de la catégorie des utilisateurs, nous distinguons plusieurs rôles. Pour ce modèle, nous avons choisi de spécifier trois rôles principaux. Les autres rôles, n'étant pas directement associés aux entités restantes, ne seront pas représentés dans ce schéma.

Les cours :

Le cours est structuré en chapitres et en parties. Nous proposons que l'accès au cours soit payant. Une fois le paiement effectué, les étudiants auront la possibilité d'accéder à l'ensemble des chapitres et des parties.

La participation des étudiants :

Nous avons centralisé toutes les activités des étudiants au sein d'un même ensemble. Ainsi, il est possible de suivre la progression dans chaque partie, de visualiser le statut des examens et les tentatives, ainsi que le statut d'inscription, et les sessions en direct.

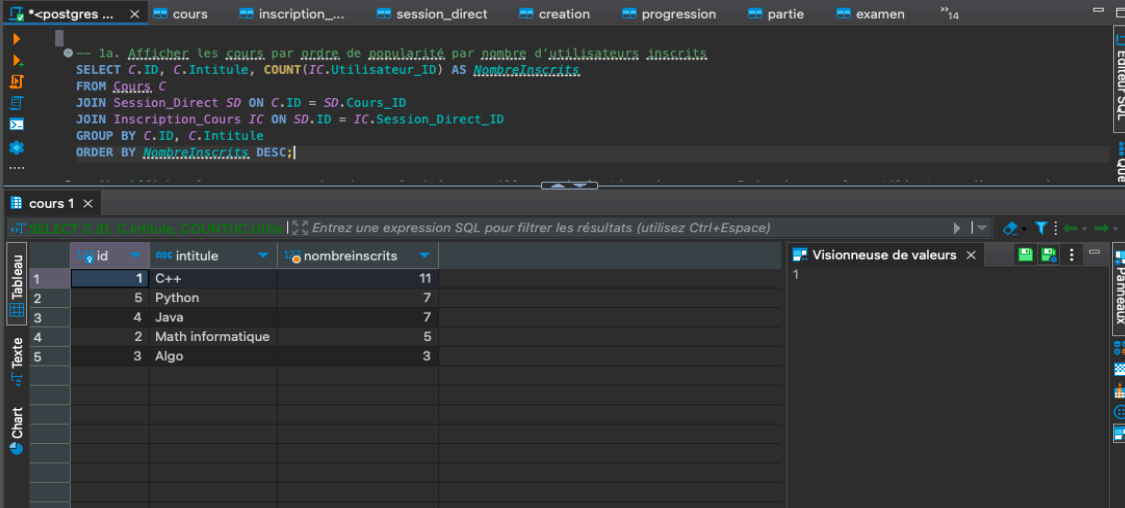
4. Implémentation et Résultat

Les principales fonctionnalités implémentées comprennent la gestion des utilisateurs, l'inscription aux cours, la progression des étudiants à travers les cours et les examens, ainsi que les sessions en direct. Pour chaque fonctionnalité, des contraintes et des triggers ont été soigneusement définis pour assurer l'intégrité des données.

<Tableaux créés dans DBeaver>

>	assigination	24K
>	chapitre	24K
>	cours	64K
>	creation	24K
>	evaluation	32K
>	examen	32K
>	inscription_cours	24K
>	partie	32K
>	progression	24K
>	role	24K
>	role_utilisateur	24K
>	session_direct	88K
>	tentative	24K
>	utilisateur	24K

Les résultats des requêtes exécutés dans DBeaver (on a inclut aussi dans le zip):



The screenshot shows the DBeaver interface with a SQL query executed in the 'cours' table. The query is:
-- 1a. Afficher les cours par ordre de popularité par nombre d'utilisateurs inscrits
SELECT C.ID, C.Intitule, COUNT(IC.Utilisateur_ID) AS NombreInscrits
FROM Cours, C
JOIN Session_Direct SD ON C.ID = SD.Cours_ID
JOIN Inscription_Cours IC ON SD.ID = IC.Session_Direct_ID
GROUP BY C.ID, C.Intitule
ORDER BY NombreInscrits DESC;
The results are displayed in a table with 5 rows:

	id	intitule	nombreinscrits
1	1	C++	11
2	5	Python	7
3	4	Java	7
4	2	Math informatique	5
5	3	Algo	3

2a. Afficher les utilisateurs qui ont terminé toutes les parties du cours donné

```

SELECT DISTINCT U.ID, U.Nom, U.Prenom
FROM Utilisateur U
JOIN Inscription_Cours IC ON U.ID = IC.Utilisateur_ID
JOIN Progression P ON IC.ID = P.Inscription_Cours_ID
JOIN Partie Pa ON P.Partie_ID = Pa.ID
JOIN Chapitre Ch ON Pa.Chapitre_ID = Ch.ID
WHERE Ch.Cours_ID = 6 AND P.Terminé = TRUE
GROUP BY U.ID, U.Nom, U.Prenom
HAVING COUNT(DISTINCT Pa.ID) = (
    SELECT COUNT(*)
    FROM Partie
    JOIN Chapitre ON Partie.Chapitre_ID = Chapitre.ID
    WHERE Chapitre.Cours_ID = 6
);

```

utilisateur 1 x

SELECT DISTINCT U.ID, U.Nom, U.Prenom FROM Utilisateur U

id	nom	prenom
4	Nadeau	Christophe
28	Admin	Jean

Visionneuse de valeurs x

2b. Afficher les utilisateurs qui ont tenté au moins une fois tous les examens du cours donné

```

SELECT U.ID, U.Nom, U.Prenom
FROM Utilisateur U
JOIN Inscription_Cours IC ON U.ID = IC.Utilisateur_ID
JOIN Tentative T ON IC.ID = T.Inscription_Cours_ID
JOIN Examen E ON T.Examen_ID = E.ID
JOIN Partie Pa ON E.Partie_ID = Pa.ID
JOIN Chapitre Ch ON Pa.Chapitre_ID = Ch.ID
WHERE Ch.Cours_ID = 6
GROUP BY U.ID, U.Nom, U.Prenom
HAVING COUNT(DISTINCT E.ID) = (
    SELECT COUNT(*)
    FROM Examen
    JOIN Partie ON Examen.Partie_ID = Partie.ID
    JOIN Chapitre ON Partie.Chapitre_ID = Chapitre.ID
    WHERE Chapitre.Cours_ID = 6
);

```

utilisateur 1 x

SELECT U.ID, U.Nom, U.Prenom FROM Utilisateur U

id	nom	prenom
17	Gracia	Julie
24	Belanger	Julie

Visionneuse de valeurs x

2c. Afficher les utilisateurs qui ont réussi tous les examens du cours donné

```

SELECT U.ID, U.Nom, U.Prenom
FROM Utilisateur U
JOIN Inscription_Cours IC ON U.ID = IC.Utilisateur_ID
JOIN Tentative T ON IC.ID = T.Inscription_Cours_ID
JOIN Examen E ON T.Examen_ID = E.ID
JOIN Partie Pa ON E.Partie_ID = Pa.ID
JOIN Chapitre Ch ON Pa.Chapitre_ID = Ch.ID
WHERE Ch.Cours_ID = 6 AND T.Score >= E.ScoreMin
GROUP BY U.ID, U.Nom, U.Prenom
HAVING COUNT(DISTINCT E.ID) = (
    SELECT COUNT(*)
    FROM Examen
    JOIN Partie ON Examen.Partie_ID = Partie.ID
    JOIN Chapitre ON Partie.Chapitre_ID = Chapitre.ID
    WHERE Chapitre.Cours_ID = 6
);

```

utilisateur 1 x

SELECT U.ID, U.Nom, U.Prenom FROM Utilisateur U

id	nom	prenom
17	Gracia	Julie

Visionneuse de valeurs x

• 3. Afficher la liste des utilisateurs par ordre de dépenses (les utilisateurs qui ont dépensé le plus d'argent en achetant des cours). On calcule la dépense totale de chaque utilisateur sur les cours payants et on les trie par dépense décroissante.

```

SELECT
  U.ID,
  U.Nom,
  U.Prenom,
  SUM(C.Prix) AS TotalDepense
FROM
  Utilisateur U
JOIN
  Inscription_Cours IC ON U.ID = IC.Utilisateur_ID
JOIN
  Session_Direct SD ON IC.Session_Direct_ID = SD.ID
JOIN
  Cours C ON SD.Cours_ID = C.ID
WHERE
  IC.Payant = TRUE
GROUP BY
  U.ID, U.Nom, U.Prenom
ORDER BY
  TotalDepense DESC;

```

utilisateur 1 x

SELECT U.ID, U.Nom, U.Prenom, SUM(C.Prix) AS TotalDepense

	id	asc nom	asc prenom	totaldepense
1	20	Delacroix	Nathalie	1893,48
2	1	Giroux	Richard	1095,46
3	29	Intelligent	Phuong	1084,4
4	14	Bonnet	Julien	946,74
5	2	Desjardins	Alexandre	946,74
6	25	Forest	Jessica	946,74
7	30	Sage	Chi	946,74
8	10	Fournier	Nicolas	946,74
9	21	Matisse	Pauline	902,19
10	17	Gracia	Julie	275,32
11	8	Andre	Sophie	193,27
12	22	Magnan	Anne	193,27
13	28	Admin	Jean	193,27
14	18	Lalonde	Annie	193,27
15	4	Nadeau	Christophe	137,66
16	15	Simon	Charlotte	0

Régénérer Save Cancel Exporter les résultats ... 200 19

• 4. Afficher les parties d'un cours, ordonnées par chapitres et ordre dans les chapitres. Liste toutes les parties d'un cours, classées par ordre des chapitres et des parties au sein des chapitres.

```

SELECT
  Ch.Cours_ID,
  Ch.OrdreChapitre,
  Ch.ChapitreNom,
  Pa.OrdrePartie,
  Pa.TitrePartie,
  Pa.Contenu
FROM
  Chapitre Ch
JOIN
  Partie Pa ON Ch.ID = Pa.Chapitre_ID
WHERE
  Ch.Cours_ID = 1
ORDER BY
  Ch.OrdreChapitre, Pa.OrdrePartie;

```

chapitre(+) 1 x

SELECT Ch.Cours_ID, Ch.OrdreChapitre, Ch.ChapitreNom, Pa.OrdrePartie, Pa.TitrePartie, Pa.Contenu

	cours_id	ordrechapitre	asc chapitrenom	ordrepartie	asc titrepartie	asc contenu
1	1	11	lazy dogThe quick	1	Part 1	Intr
2	1	12	the	2	Part 2	Ob

Régénérer Save Cancel Exporter les résultats ... 200 2

5. Afficher tous les cours ainsi que les créateurs de cours et formateurs qui y sont rattachés. Joint les tables pour afficher l'information sur les créateurs et les formateurs liés à chaque cours.

```

SELECT
  C.ID AS ID_Cours,
  C.Intitule AS Titre_Cours,
  U.Nom AS Nom_Utilisateur,
  U.Prenom AS Prenom_Utilisateur,
  'Créateur' AS Role
FROM
  Cours C
JOIN
  Creation Cr ON C.ID = Cr.Cours_ID
JOIN
  Utilisateur U ON Cr.Utilisateur_ID = U.ID
UNION ALL
SELECT
  C.ID AS ID_Cours,
  C.Intitule AS Titre_Cours,
  U.Nom AS Nom_Utilisateur,
  U.Prenom AS Prenom_Utilisateur,
  'Assigné' AS Role
FROM
  Cours C
JOIN
  Assignation A ON C.ID = A.Cours_ID
JOIN
  Utilisateur U ON A.Utilisateur_ID = U.ID
ORDER BY
  ID_Cours, Role;

```

Résultats 1 x

Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espace)

	id_cours	titre_cours	nom_utilisateur	prenom_utilisateur	role
1	1	C++	Forest	Jessica	Assigné
2	1	C++	Belanger	Julie	Assigné
3	1	C++	Morel	Lucie	Assigné
4	1	C++	Boucher	Kate	Créateur
5	1	C++	Andre	Sophie	Créateur
6	1	C++	Leroy	Vincent	Créateur
7	2	Math informatique	Bonnet	Julien	Assigné
8	2	Math informatique	Admin	Jean	Assigné

Régénérer Save Cancel Exporter les résultats ... 200 21

6. Pour un utilisateur donné, afficher les cours auxquels il est inscrit, ainsi que son pourcentage de progression de chaque cours (nombre de parties terminées sur le nombre total de parties).

```

SELECT
  IC.Utilisateur_ID AS ID_Utilisateur,
  C.ID AS ID_Cours,
  C.Intitule AS Titre_Cours,
  COALESCE(CAST(COUNT(DISTINCT CASE WHEN P.Termine THEN P.Partie_ID END) AS FLOAT) / NULLIF(COUNT(DISTINCT PartieTotale.ID), 0) * 100, 0) AS P
FROM
  Inscription_Cours IC
JOIN
  Session_Direct SD ON IC.Session_Direct_ID = SD.ID
JOIN
  Cours C ON SD.Cours_ID = C.ID
LEFT JOIN
  Chapitre Ch ON C.ID = Ch.Cours_ID
LEFT JOIN
  Partie PartieTotale ON Ch.ID = PartieTotale.Chapitre_ID -- Jointure pour toutes les parties, terminées ou non
LEFT JOIN
  Progression P ON IC.ID = P.Inscription_Cours_ID AND P.Termine = TRUE
LEFT JOIN
  Partie Pa ON P.Partie_ID = Pa.ID -- Seulement pour les parties terminées
WHERE
  IC.Utilisateur_ID = 2
GROUP BY
  IC.Utilisateur_ID, C.ID, C.Intitule
ORDER BY
  C.ID;

```

inscription_cours(+) 1 x

Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espace)

	id_utilisateur	id_cours	titre_cours	pourcentage_progression
1	2	1	C++	50
2	2	5	Python	0

Régénérer Save Cancel Exporter les résultats ... 200 2

5. Difficultés rencontrées

Gestion des rôles d'accès:

La gestion des rôles d'accès a représenté un véritable défi pour nous. En effet, notre plateforme MOOC nécessite une distinction claire entre les différents utilisateurs (étudiants, formateurs, administrateurs, etc.) pour assurer à chacun un accès approprié aux fonctionnalités et aux données. La complexité résidait dans la conception d'un système de permissions flexible mais sécurisé, qui permette une gestion fine des droits d'accès sans compromettre la sécurité des informations.

Contraintes:

La mise en place des contraintes dans la base de données pour garantir l'intégrité et la validité des données a également été une épreuve. Nous avons été confrontés à la difficulté de définir des règles qui soient à la fois suffisamment strictes pour prévenir les erreurs de données et assez souples pour ne pas entraver l'utilisation de la plateforme.

Mock data:

Enfin, la création de données fictives (mock data) pour tester notre base de données a été plus ardue que prévu. Il était essentiel pour nous de générer un volume important de données réalistes pour simuler l'utilisation de la plateforme dans des conditions variées et tester la performance de nos requêtes SQL. Cependant, concevoir ces ensembles de données tout en s'assurant qu'ils reflètent fidèlement les scénarios d'utilisation réels sans violer les contraintes de la base a été complexe. Nous avons utilisé des outils de génération de données (DBeaver "tool - generate mock data") et écrit des scripts personnalisés, ce qui a été un excellent exercice pour comprendre les implications des données sur la performance et la fiabilité du système.

Utilisation de DBeaver:

DBeaver est un logiciel très détaillé dans ses réglages, et certains paramètres par défaut peuvent dérouter les utilisateurs novices. Par exemple, l'affichage des bases de données est configuré par défaut sur « ne pas afficher toutes les bases de données », ce qui peut être effrayant car après un redémarrage de DBeaver, il semble que les bases de données travaillées la veille aient disparu. Lors de l'exportation des données, l'option binaire est automatiquement cochée, ce qui rend le fichier .sql exporté illisible et peut également causer des problèmes.

6. Travail en groupe

Nous collaborons ensemble sur toutes les tâches. Après avoir effectué nos travaux respectifs, nous examinons et discutons ensemble des problèmes rencontrés.

Pour faciliter notre collaboration et optimiser notre organisation, nous avons adopté un ensemble d'outils spécifiques: Trello, Google Drive et WhatsApp. Trello occupe une place centrale dans notre système de gestion des tâches et de suivi des projets. Google Drive joue un rôle indispensable dans notre processus de collaboration. Il nous permet de partager et de modifier en temps réel nos documents. WhatsApp joue un rôle crucial dans notre communication quotidienne.

Pour assurer la cohérence et l'efficacité de notre travail en équipe, nous avons instauré une réunion hebdomadaire. Ce rendez-vous régulier est l'occasion pour nous de mettre en commun notre travail et nos conclusions. C'est également un moment privilégié pour tester toutes les requêtes et s'assurer qu'elles répondent aux exigences fixées. Cette méthode de travail nous permet de progresser de manière significative dans nos projets, en garantissant une qualité et une précision maximales.

7. Conclusion et Améliorations

Ce projet a représenté un défi stimulant et enrichissant, nous permettant de concrétiser une plateforme d'apprentissage en ligne robuste et fonctionnelle. Grâce à une étude approfondie des besoins et des exigences des utilisateurs, nous avons réussi à concevoir et implémenter une base de données capable de soutenir une expérience d'apprentissage flexible et interactive. Les contraintes et triggers que nous avons mis en place garantissent l'intégrité et la sécurité des données, tout en facilitant une gestion efficace des utilisateurs, des cours, des sessions, et des examens. Notre travail sur la normalisation des données a réduit la redondance, améliorant ainsi la performance et la maintenabilité de la plateforme.

Pour adresser ces limites, plusieurs améliorations sont envisagées. D'abord, l'intégration de technologies de streaming en temps réel améliorera les sessions en direct, permettant une interaction plus fluide et un meilleur engagement des étudiants. Aussi l'exploration de l'intelligence artificielle pour la personnalisation des parcours d'apprentissage représente une opportunité passionnante pour rendre l'éducation en ligne encore plus réactive et adaptée aux besoins individuels.

Database model documentation

Table of contents

1. Model details	4
2. Tables	5
1.1. Table Cours	5
1.2. Table Partie	5
1.3. Table Session_Direct	6
1.4. Table Examen	6
1.5. Table Progression	7
1.6. Table Utilisateur	7
1.7. Table Chapitre	8
1.8. Table Tentative	8
1.9. Table Inscription_Cours	9
1.10. Table Role	10
1.11. Table Evaluation	10
1.12. Table Assignment	10
1.13. Table Creation	11
1.14. Table Role_Utilisateur	11
3. References	12
2.1. Reference Chapitres_Cours	12
2.2. Reference Tentative_Examen	12
2.3. Reference Session_Cours	12
2.4. Reference Partie_Chapitre	12
2.5. Reference Inscription_Tentative	13
2.6. Reference Inscrire	13
2.7. Reference De	13
2.8. Reference Examen_Partie	13
2.9. Reference Progression_Partie	13
2.10. Reference Avoir	14
2.11. Reference Cours_association_2	14
2.12. Reference Cours_association_1	14
2.13. Reference Utilisateur_Creation	14
2.14. Reference Utilisateur_Assignation	14
2.15. Reference Utilisateur_Evaluation	14
2.16. Reference Utilisateur_Role_Utilisateur	14
2.17. Reference Roles_Role_Utilisateur	15
2.18. Reference Evaluation_Cours	15

4. Sequences **16**

5. Subject areas **17**

1. Model details

Model name:

MOOC BDD logique V4 Physical Export

Version:

2.4

Database engine:

PostgreSQL

Description:

2. Tables

2.1. Table Cours

Description:

MOOC a des cours.

2.1.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
Intitule	varchar(255)		Les cours ont un intitulé.
Description	Text		Les cours ont une description.
Prerequis	Text		Les cours ont des pré-requis.
Prix	decimal(5,2)		Les cours peuvent être payants (>0€) ou gratuits (=0€). Le prix ne peut être négatif
Accessibilite	boolean		gratuit/payant + certification/non
DateDebut	date	null	Les cours peuvent avoir des dates de début et de fin. La date début doit être inférieur à la date de fin
DateFin	date	null	Les cours peuvent avoir des dates de début et de fin. Les cours avec une date de fin passée ne devraient plus être accessibles ou visibles.

2.2. Table Partie

Description:

Les parties peuvent être regroupées et ordonnées par chapitres

2.2.1. Columns

Column name	Type	Properties	Description
ID	int	PK	

OrdrePartie	int		
TitrePartie	varchar(70)		Chaque partie a un titre.
Contenu	Text		Chaque partie a du contenu.
Chapitre_ID	int		

2.3. Table Session_Direct

Description:

Les cours peuvent avoir des sessions en direct.

Les cours peuvent être dispensés en autonomie.

2.3.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
TypeSession	varchar(30)		En présentiel ou en distanciel.
DateDebut	timestamp		Les sessions ont des dates de début et de fin. Les dates de début doivent être inférieures aux dates de fin.
DateFin	timestamp		Les sessions ont des dates de début et de fin. Date de début doit être inférieur à la date de fin.
PlacesMaximum	int		Les sessions peuvent préciser un nombre de places maximum. Le nombre d'étudiants inscrits à une session en direct ne peut pas dépasser le nombre de places maximales.
Cours_ID	int		

2.4. Table Examen

Description:

Les cours peuvent proposer des examens.

Les examens sont liés à des parties de cours.

2.4.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
TitreExamen	varchar(70)		Les examens ont un titre.
Contenu	Text		Les examens ont un contenu textuel.
ScoreMin	decimal(100,10)		Le score minimum à atteindre pour valider (entre 40 et 100).
Partie_ID	int		

2.5. Table Progression

Description:

Les cours peuvent être dispensés en autonomie (les étudiants progressent par eux-même).

2.5.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
Termine	boolean		Un étudiant inscrit à un cours peut marquer une partie comme étant terminée.
Date	date		
Partie_ID	int		
Inscription_Cours_ID	int		

2.6. Table Utilisateur

Description:

Les utilisateurs de la plateforme peuvent avoir différents rôles : Administrateur, Personnel administratif, Créateur de cours, Formateur, Étudiant, etc.

2.6.1. Columns

Column name	Type	Properties	Description
ID	int	PK	Clé primaire

Nom	varchar(100)		Les utilisateurs ont différents rôles : Administrateur, Personnel administratif, Créateur de cours, Formateur, Étudiant, etc.
Prenom	varchar(100)		
Email	varchar(50)		Colonne email doit vérifier le format nom@domaine.ext
DateNaissance	date		
NumeroTelephone	varchar(15)		
Location	varchar(15)		Code postal
Username	varchar(50)		Le rôle des utilisateurs : un étudiant ne peut pas accéder à l'édition ou à la suppression de cours s'il n'est pas administrateur, formateur ou créateur de cours
Password	varchar(50)		

2.7. Table Chapitre

Description:

Les chapitres dans un cours

2.7.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
OrdreChapitre	int		
ChapitreNom	varchar(100)		
Cours_ID	int		

2.8. Table Tentative

Description:

Les étudiants peuvent tenter de passer les examens autant de fois que nécessaire pour valider

2.8.1. Columns

Column name	Type	Properties	Description
ID	int	PK	

Date	date		
Score	decimal(100,10)		Les tentatives d'examen enregistrent le score obtenu (entre 0 et 100).
Procedure	varchar(70)		Une procédure permet de marquer les tentatives comme réussies (si le score obtenu est supérieur au score minimum entre 40 et 100).
Examen_ID	int		
Inscription_Cours_ID	int		
statut_de_reussite	boolean		

2.9. Table Inscription_Cours

Description:

Les étudiants peuvent s'inscrire aux cours.

2.9.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
DateInscription	date		La date d'inscription à un cours est enregistrée. La date d'inscription doit être inférieure à la date de début du cours
DatePayant	date	null	Uniquement s'ils ont préalablement payé le cours pour les cours payants -> DatePayant doit être inférieur à la date d'inscrire.
Payant	boolean		Les étudiants doivent avoir payé pour s'inscrire aux cours.
TypeStatut	varchar(25)		?Réussi, Défaillant, Ajourné
Utilisateur_ID	int		Clé primaire
Session_Direct_ID	int		

2.10. Table Role

2.10.1. Columns

Column name	Type	Properties	Description
ID	int	PK	
Libelle	varchar(30)		

2.11. Table Evaluation

Description:

Etudiant - Cours : Un étudiant peut évaluer plusieurs cours (la contrainte est les étudiants peuvent seulement évaluer aux cours ils suivent), un cours peut être évalué par plusieurs étudiants (1-N) - (1-N)

=> Association Evaluation (N-N)

2.11.1. Columns

Column name	Type	Properties	Description
Utilisateur_ID	int	PK	Clé primaire
Cours_ID	int	PK	
Note	decimal(10,1)	null	Les cours peuvent être notés de 1 à 5. Un étudiant ne peut pas noter et commenter le même cours plus d'une fois
Commentaire	Text	null	Les cours peuvent avoir un commentaire optionnel. Un étudiant ne peut pas noter et commenter le même cours plus d'une fois. Un étudiant ne peut pas commenter un cours s'il n'y est pas inscrit.

2.12. Table Assignment

Description:

Les cours sont assignés à un ou plusieurs formateurs.

La relation entre Formateur et Cours est N-N, donc ici on a besoin une nouvelle table qui contient les clés de ces entités.

2.12.1. Columns

Column name	Type	Properties	Description
Cours_ID	int	PK	
Utilisateur_ID	int	PK	Clé primaire

2.13. Table Creation

Description:

Association entre créateur et cours

2.13.1. Columns

Column name	Type	Properties	Description
Cours_ID	int	PK	
Utilisateur_ID	int	PK	Clé primaire
Date	date		Date de création

2.14. Table Role_Utilisateur

2.14.1. Columns

Column name	Type	Properties	Description
Utilisateur_ID	int	PK	Clé primaire
Roles_ID	int	PK	

3. References

3.1. Reference Chapitres_Cours

Description:

Les cours sont segmentés en parties.

Cours - Chapitre : Un cours peut contenir plusieurs chapitres, un chapitre est contenu dans un cours (1-N) - (1-1)

=> Association "contenir" (1-N)

Cours	0..*	Chapitre
ID	<->	Cours_ID

3.2. Reference Tentative_Examen

Description:

Tentatives d'Examens - Examens : Chaque tentative d'examen est liée à un examen spécifique, un examen peut avoir plusieurs tentatives (1,1) - (1,N)

=> Association "lier" (1-N)

Examen	0..*	Tentative
ID	<->	Examen_ID

3.3. Reference Session_Cours

Description:

Les sessions sont assignées aux cours.

Sessions en Direct - Cours : Une session en direct est associée à un cours, un cours peut avoir plusieurs sessions en direct (1,1) - (1,N)

=> Association "assigner" (1-N)

Cours	0..*	Session_Direct
ID	<->	Cours_ID

3.4. Reference Partie_Chapitre

Description:

Les parties peuvent être regroupées et ordonnées par chapitres.

Un chapitre peut diviser en plusieurs parties, une partie est divisée par un chapitre (1,N) - (1,1)

=> association "diviser" (1-N)

Chapitre	0..*	Partie
ID	<->	Chapitre_ID

3.5. Reference Inscription_Tentative

Description:

Une inscription d'un étudiant d'un cours peut avoir plusieurs tentatives d'examens, une tentative d'examen peut lier juste une inscription (1,N) - (1,1)

=> Association "contenir" (1-N)

Inscription_Cours	0..*	Tentative
ID	<->	Inscription_Cours_ID

3.6. Reference Inscrire

Utilisateur	0..*	Inscription_Cours
ID	<->	Utilisateur_ID

3.7. Reference De

Description:

Session en Direct - Inscriptions aux Cours : Une Session peut lier à 0 ou plusieurs inscriptions, et un inscription peut être inscrit pour une Session (0,N) - (1,1)

=> Association "de" (1-N)

Session_Direct	0..*	Inscription_Cours
ID	<->	Session_Direct_ID

3.8. Reference Examen_Partie

Description:

Examen - Partie : Un examen est lié à un partie, uExamens - Partie : Un examen est proposé à un partie, un partie peut proposer un examen (0,1) - (1,1)

=> Association "proposer" (1-1)

Partie	0..1	Examen
ID	<->	Partie_ID

3.9. Reference Progression_Partie

Description:

Une partie peut avoir une progression, une progression fait partie d'une Partie (1,1) - (1-1)

=> Association "lier" (1-1)

Partie	0..1	Progression
ID	<->	Partie_ID

3.10. Reference Avoir

Description:

Une progression fait partie d'une inscription, une inscription du cours peut avoir une progression
(1,1) - (1,1)

=> Association "contenir" (1-1)

Inscription_Cours	0..1	Progression
ID	<->	Inscription_Cours_ID

3.11. Reference Cours_association_2

Cours	1..*	Assigation
ID	<->	Cours_ID

3.12. Reference Cours_association_1

Cours	1..*	Creation
ID	<->	Cours_ID

3.13. Reference Utilisateur_Creation

Utilisateur	1..*	Creation
ID	<->	Utilisateur_ID

3.14. Reference Utilisateur_Assignation

Utilisateur	1..*	Assigation
ID	<->	Utilisateur_ID

3.15. Reference Utilisateur_Evaluation

Utilisateur	1..*	Evaluation
ID	<->	Utilisateur_ID

3.16. Reference Utilisateur_Role_Utilisateur

Utilisateur	1..*	Role_Utilisateur
ID	<->	Utilisateur_ID

3.17. Reference Roles_Role_Utilisateur

Role	1..*	Role_Utilisateur
ID	<->	Roles_ID

3.18. Reference Evaluation_Cours

Cours	1..*	Evaluation
ID	<->	Cours_ID

4. Sequences

Sequence name	Starts with	Description
Cours_seq	1	
Partie_seq	1	
Session_Direct_seq	1	
Examen_seq	1	
Progression_seq	1	
Utilisateur_seq	1	
Chapitre_seq	1	
Tentative_seq	1	
Inscription_Cours_seq	1	
Roles_seq	1	

5. Areas

5.1. Participation d'étudiants subject area

5.1.1. Tables

- Session_Direct
- Examen
- Progression
- Tentative
- Inscription_Cours

5.1.2. References

- Chapitres_Cours
- Tentative_Examen
- Session_Cours
- Partie_Chapitre
- Inscription_Tentative
- Inscrire
- De
- Examen_Partie
- Progression_Partie
- Avoir
- Cours_association_2
- Cours_association_1
- Utilisateur_Creation
- Utilisateur_Assignation
- Utilisateur_Evaluation
- Utilisateur_Role_Utilisateur
- Roles_Role_Utilisateur
- Evaluation_Cours

5.2. Cours subject area

5.2.1. Tables

- Cours
- Partie
- Chapitre
- Evaluation
- Assignation
- Creation

5.2.2. References

- Chapitres_Cours
- Tentative_Examen
- Session_Cours
- Partie_Chapitre
- Inscription_Tentative
- Inscrire
- De
- Examen_Partie
- Progression_Partie
- Avoir
- Cours_association_2
- Cours_association_1
- Utilisateur_Creation
- Utilisateur_Assignation
- Utilisateur_Evaluation
- Utilisateur_Role_Utilisateur
- Roles_Role_Utilisateur
- Evaluation_Cours

5.3. Rôle subject area

5.3.1. Tables

- Utilisateur
- Role
- Role_Utilisateur

5.3.2. References

- Chapitres_Cours
- Tentative_Examen
- Session_Cours
- Partie_Chapitre
- Inscription_Tentative
- Inscrire
- De
- Examen_Partie
- Progression_Partie
- Avoir
- Cours_association_2
- Cours_association_1
- Utilisateur_Creation
- Utilisateur_Assignation

- Utilisateur_Evaluation
- Utilisateur_Role_Utilisateur
- Roles_Role_Utilisateur
- Evaluation_Cours