Qualité et au-delà du relationnel

S4 - R4.03 - 2022/2023

Objectif de cette séance

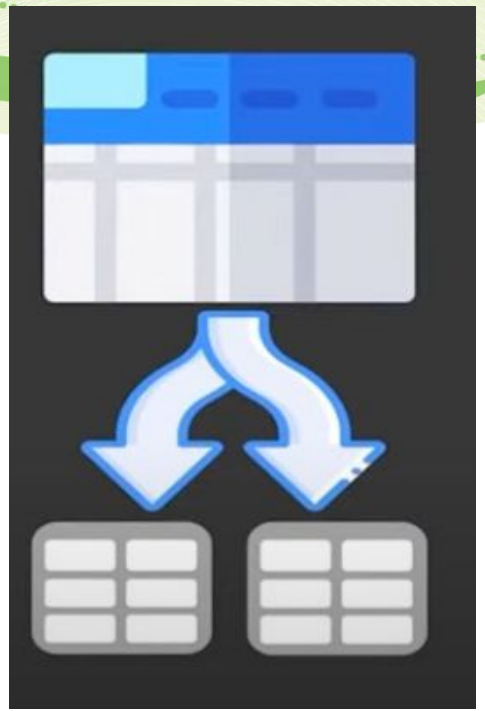
I. Qu'est-ce que la normalisation de base de données ?
II. Pourquoi normaliser une base de données ?
III. Quelles sont les formes normales ?
IV. Qu'est-ce que la première forme normale ?
V. Qu'est-ce que la deuxième forme normale ?
VI. Qu'est-ce que la troisième forme normale ?
VII. Forme normale de Boyce-Codd (BCNF)
VIII. Dénormalisation

Qu'est-ce que la normalisation de base de données ?

- La normalisation de la base de données, ou simplement la normalisation comme on l'appelle communément, est un processus utilisé pour la modélisation des données ou la création de la base de données, où vous organisez vos données et vos tables afin qu'elles puissent être ajoutées et mises à jour efficacement.
- C'est quelque chose qu'une personne fait manuellement, par opposition à un système ou un outil qui le fait. Il est généralement effectué par les développeurs de bases de données et les administrateurs de bases de données.
- Cela peut être fait sur n'importe quelle base de données relationnelle, où les données sont stockées dans des tables liées les unes aux autres. Cela signifie que la normalisation dans un SGBD (système de gestion de base de données) peut être effectuée dans Oracle, Microsoft SQL Server, MySQL, PostgreSQL et tout autre type de base de données.
- Pour effectuer le processus de normalisation, vous commencez avec une idée approximative des données que vous souhaitez stocker et vous leur appliquez certaines règles afin de les mettre sous une forme plus efficace.

Qu'est-ce que la normalisation de base de données ?

- La normalisation est un concept important dans la conception des bases de données, car elle permet de **minimiser la redondance** des données et d'**améliorer l'intégrité des données**.
- En normalisant une base de données, vous pouvez vous assurer que les données sont stockées de la manière la plus efficace et la plus logique possible.
- Améliorer les performances, rendre les données plus fiables et faciliter la maintenance de la base de données au fil du temps.
- Une bdd normalisée est plus facile à interroger, à mettre à jour et à mettre à l'échelle, ce qui peut être particulièrement important pour les bases de données volumineuses et complexes.



Pourquoi normaliser une base de données ?

- **Objectif :** construire un schéma relationnel évitant la redondance.
- La redondance implique des anomalies lors de :
  - L’ insertion : empêcher que les mêmes données soient stockées à plusieurs endroits
  - Mise à jour : empêcher les mises à jour de certaines données mais pas d'autres
  - Suppression : empêcher que les données ne soient pas supprimées alors qu'elles sont censées l'être, ou que des données ne soient perdues alors qu'elles ne sont pas censées l'être
- La normalisation d'une base de données peut faciliter la détection et la correction des erreurs dans les données.
- S’assurer que les requêtes sur une base de données s'exécutent aussi rapidement que possible.

Anomalies de données

Une anomalie est un problème dans les données qui n'est pas censé être là. Cela peut arriver si une base de données n'est pas normalisée.

Considérez la table suivante :

Student ID	Student Name	Fees Paid	Course Name	Class 1	Class 2	Class 3
1	John Smith	200	Economics	Economics 1	Biology 1	
2	Maria Griffin	500	Computer Science	Biology 1	Business Intro	Programming 2
3	Susan Johnson	400	Medicine	Biology 2		
4	Matt Long	850	Dentistry			

Anomalies de données

Anomalie d'insertion

Student ID	Student Name	Fees Paid	Course Name	Class 1	Class 2	Class 3
1	John Smith	200	Economics	Economics 1	Biology 1	
2	Maria Griffin	500	Computer Science	Biology 1	Business Intro	Programming 2
3	Susan Johnson	400	Medicine	Biology 2		
4	Matt Long	850	Dentistry			
5	**Jared Oldham**	**0**	**?**			

Une anomalie d'insertion se produit lorsque nous essayons d'insérer un enregistrement dans cette table sans connaître toutes les données dont nous avons besoin.

Par exemple, si nous voulions ajouter un nouvel étudiant mais ne connaissions pas le nom de son cours ⇒ données incomplètes.

Anomalies de données

Anomalie de mise à jour

Student ID	Student Name	Fees Paid	Course Name	Class 1	Class 2	Class 3
1	John Smith	200	Economics	Economics 1	**Intro to Biology**	
2	Maria Griffin	500	Computer Science	**Intro to Biology**	Business Intro	Programming 2
3	Susan Johnson	400	Medicine	Biology 2		
4	Matt Long	850	Dentistry			

Une anomalie de mise à jour se produit lorsque nous voulons mettre à jour des données, et nous mettons à jour certaines données mais pas d'autres données.

Par exemple, disons que la classe Biology 1 a été changée en "Intro to Biology". Nous devions interroger toutes les colonnes qui pourraient avoir ce champ Class et renommer chacune d'entre elles trouvées.

Anomalies de données

Anomalie de suppression

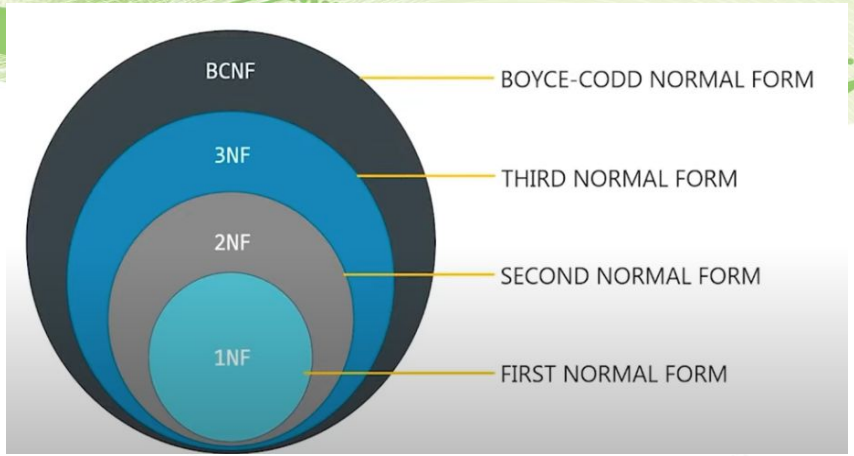
Student ID	Student Name	Fees Paid	Course Name	Class 1	Class 2	Class 3
1	John Smith	200	Economics	Economics 1	Biology 1	
2	Maria Griffin	500	Computer Science	Biology 1	Business Intro	Programming 2
3	**Susan Johnson**	**400**	**Medicine**	**Biology 2**		
4	Matt Long	850	Dentistry			

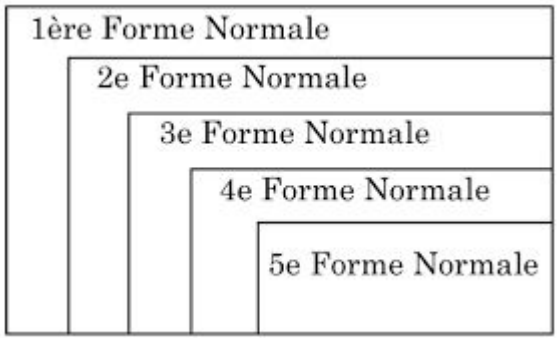
Une anomalie de suppression se produit lorsque nous voulons supprimer des données de la table, mais nous finissons par en supprimer plus que ce que nous avions prévu.

Par exemple, supposons que **Susan Johnson** démissionne et que son dossier doive être supprimé du système. Mais, si nous supprimons cette ligne, nous perdons l'enregistrement de la classe **Biology 2**, car il n'est stocké nulle part ailleurs.

Quelles sont les formes normales ?

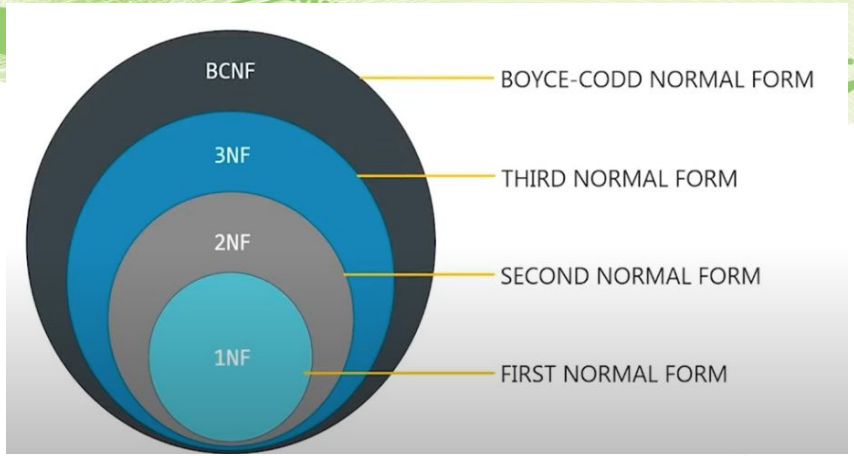
- La théorie de la normalisation s'inspire fortement de la théorie des dépendances fonctionnelles.
- La théorie de la normalisation définit six formes normales (NF).
- Chaque forme normale implique un ensemble de propriétés de dépendance qu'un schéma doit satisfaire et chaque forme normale donne des garanties sur la présence et/ou l'absence d'anomalies de mise à jour.
- Cela signifie que les formes normales supérieures ont moins de redondance et, par conséquent, moins de problèmes de mise à jour.

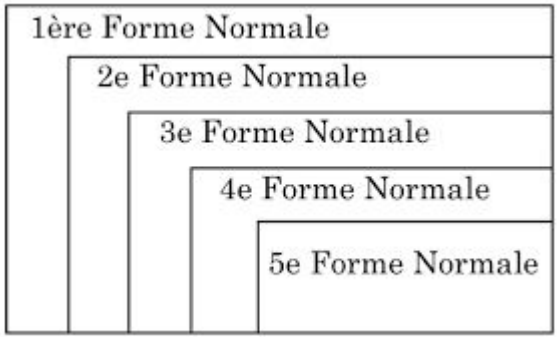




Quelles sont les formes normales ?

- 2NF est meilleur que 1NF ; 3NF est meilleur que 2NF.
- Pour la plupart des objectifs de conception de bases de données d'entreprise, 3NF est le plus élevé dont nous ayons besoin dans le processus de normalisation.
- Le niveau de normalisation le plus élevé n'est pas toujours le plus souhaitable (précision VS vitesse).





La première forme normale

Supprimer les groupes répétés de la table

Créer une table distincte pour chaque ensemble de données associées

Identifier chaque ensemble de données associées avec une clé primaire

La première forme normale

Une relation R est en première forme normale si :

- R respecte la définition du modèle relationnel.
- R ne possède pas d'attribut composés ou multivalués.

Pour appliquer la première forme normale à une base de données, nous examinons chaque table, une par une, et nous nous posons les questions suivantes :

- La combinaison de toutes les colonnes crée-t-elle une ligne unique à chaque fois ?
- Quel champ peut être utilisé pour identifier de manière unique la ligne ?

La première forme normale

Exemple 1

Department

DName	DNumber	DMgr	{DLocations}

Department

DName	DNumber	DMgr	{DLocations}
Research	5	333445555	{Bellaire,Sugarland,Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

⇓ 1NF Normalization

Department

DName	DNumber	DLocations	DMgr
Research	5	Bellaire	333445555
Research	5	Sugarland	333445555
Research	5	Houston	333445555
Administration	4	Stafford	987654321
Headquarters	1	Houston	888665555

La première forme normale

Exemple 2

Cette table viole la première forme normale à cause de la présence de groupes répétitifs.

Player_ID	Inventory
jdog21	2 amulets, 4 rings
gila19	18 copper coins
trev73	3 shields, 5 arrows, 30 copper coins, 7 rings

Groupe répétitif

La première forme normale

Exemple 2

Le stockage d'un groupe répétitif d'éléments de données sur une seule ligne viole la première forme normale.

Player_Inventory

Player_ID	Quantity_1	Item_Type_1	Quantity_2	Item_Type_2	Quantity_3	Item_Type_3	Quantity_4	Item_Type_4
jdog21	2	amulets	4	rings				
gila19	18	copper coins						
trev73	3	shields	5	arrows	30	copper coins	7	rings

La première forme normale

Exemple 2

- Pas de mélange de types de données dans la même colonne.
- Nous avons une table avec clé primaire.
- Pas de groupes répétitifs.

Player_Inventory

Player_ID	Item_Type	Item_Quantity
jdog21	amulets	2
jdog21	rings	4
gila19	copper coins	18
trev73	shields	3
trev73	arrows	5
trev73	copper coins	30
trev73	rings	7

La deuxième forme normale

La table doit être dans la première forme normale

La table ne doit pas contenir de dépendance partielle

Il n'y a pas d'attribut ne faisant pas partie d'une clé (non-prime attribute) qui dépend d'une partie d'une des clés de R

La deuxième forme normale

Exemple 1

Si la relation a une clé primaire composée, alors chaque attribut non clé doit être entièrement dépendant de la clé primaire entière et non d'un sous-ensemble de la clé primaire (c'est-à-dire qu'il ne doit y avoir aucune dépendance partielle).

Clé primaire composée

Cette colonne ne dépend que d'une partie de la clé primaire (Department Id)

Employee Id	Department Id	Office Location
1EDU001	ED-T1	Pune
1EDU002	ED-S2	Bengaluru
1EDU003	ED-M1	Delhi
1EDU004	ED-T3	Mumbai

La deuxième forme normale

Exemple 1

- Nous divisons la table en deux.
- "Office Location" dépend entièrement de la clé primaire ⇒ pas de dépendance partielle

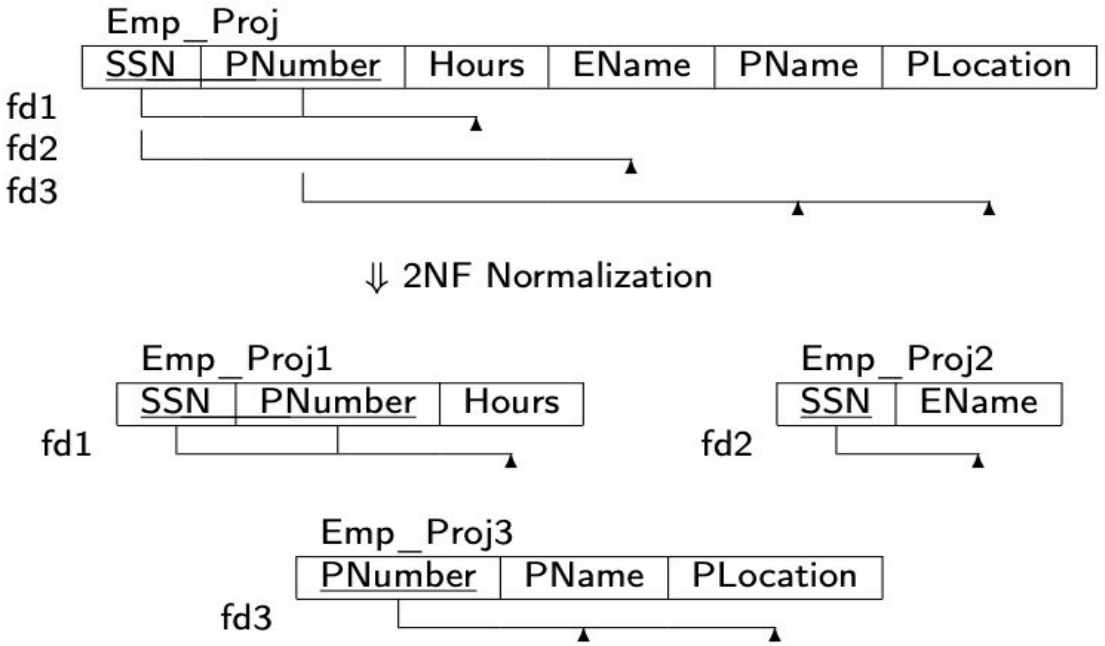
Department Id	Office Location
ED-T1	Pune
ED-S2	Bengaluru
ED-M1	Delhi
ED-T3	Mumbai

Employee Id	Department Id
1EDU001	ED-T1
1EDU002	ED-S2
1EDU003	ED-M1
1EDU004	ED-T3

Employee Id	Department Id	Office Location
1EDU001	ED-T1	Pune
1EDU002	ED-S2	Bengaluru
1EDU003	ED-M1	Delhi
1EDU004	ED-T3	Mumbai

La deuxième forme normale

Exemple 2



La troisième forme normale

La table doit être dans la deuxième forme normale

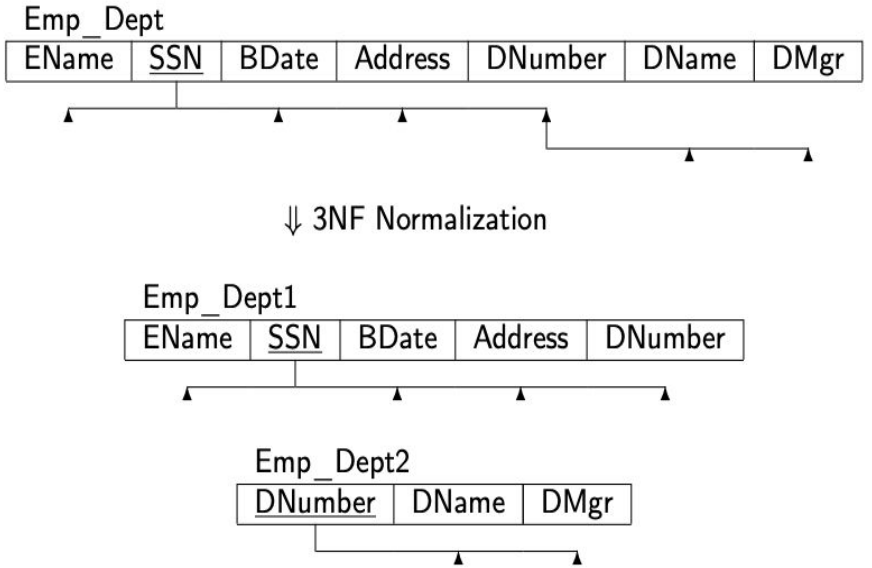
La table ne doit pas contenir de dépendance transitive pour les attributs non premiers (prime attributes)

L'objectif est de réduire la duplication des données et d'assurer l'intégrité référentielle

La troisième forme normale

Exemple 1

- DNumber ⇒ {Dname, DMgr}
- {Dname, DMgr} dépendent fonctionnellement d'un autre attribut non clé {DNumber}
  - ⇒ Dépendance transitive
- 3NF ⇒ Un attribut non clé ne peut pas être fonctionnellement dépendant d'un autre attribut non clé !



La troisième forme normale

Exemple 2

- Student Id ⇒ Subject Id
- Subject Id ⇒ Subject
- Student Id ⇒ Subject via Subject (transitivité)
  - ⇒ NON 3NF
- Nous devons supprimer cette dépendance transitive !

Student Id	Student Name	Subject Id	Subject	Address
1DT15ENG01	Alex	15CS11	SQL	Goa
1DT15ENG02	Barry	15CS13	JAVA	Bengaluru
1DT15ENG03	Clair	15CS12	C++	Delhi
1DT15ENG04	David	15CS13	JAVA	Kochi

La troisième forme normale

Exemple 2

Tous les attributs non clés ne dépendent entièrement fonctionnellement que de la clé primaire.

R1

Student Id	Student Name	Subject Id	Address
1DT15ENG01	Alex	15CS11	Goa
1DT15ENG02	Barry	15CS13	Bengaluru
1DT15ENG03	Clair	15CS12	Delhi
1DT15ENG04	David	15CS13	Kochi

R2

Subject Id	Subject
15CS11	SQL
15CS13	JAVA
15CS12	C++
15CS13	JAVA

Forme normale de Boyce-Codd (BCNF) - 3.5 NF

La table doit être dans la troisième forme normale

Version supérieure de 3NF et a été développée par Raymond F. Boyce et Edgar F. Codd

La partie gauche de chaque DF est une clé candidate entière (Super Key)

Forme normale de Boyce-Codd (BCNF)

Clé candidate entière (Super Key)

- Une clé candidate entière dans une dépendance fonctionnelle fait référence à un ensemble d'un ou plusieurs attributs dans une table qui identifie de manière unique chaque ligne de la table.
  - une clé candidate entière est un ensemble d'attributs permettant de distinguer une ligne de toutes les autres dans la table
- Par exemple, si nous avons une table appelée "Students" avec des attributs (student_id, name, major) , où student_id est la clé primaire, alors nous pouvons dire que "name" et "major" sont fonctionnellement dépendants de "student_id".
- Une clé candidate entière n'est pas toujours identique à une clé primaire, qui est un ensemble minimal d'attributs qui identifie de manière unique chaque ligne d'une table et ne peut pas être nulle. Une clé primaire est un type particulier de clé candidate entière.
- Clé candidate entière ⇒ clé candidate ou un sur-ensemble de clé candidate.

Forme normale de Boyce-Codd (BCNF)

Clé candidate entière (Super Key)

- Un exemple où la clé candidate entière est différente de la clé primaire est une table appelée "Courses", qui a les attributs suivants :
  - course_id (Primary Key)
  - course_name
  - instructor_name
  - department_name
  - semester
- Dans cette table, la clé primaire est "course_id" car elle est unique pour chaque cours et ne peut pas être nulle. Cependant, il existe d'autres combinaisons d'attributs qui pourraient également être utilisées pour identifier de manière unique chaque ligne de cette table. Par exemple, la combinaison de "course_name", "instructor_name" et "semester" pourrait également être utilisée pour identifier de manière unique chaque cours.
- Cette combinaison d'attributs forme une clé candidate entière, car elle identifie de manière unique chaque ligne, mais ce n'est pas un ensemble minimal d'attributs et elle ne peut pas être utilisée comme clé primaire, car elle ne suit pas la règle de ne pas avoir de valeurs nulles .

Forme normale de Boyce-Codd (BCNF)

Exemple 1

Une relation est en BCNF si elle est en 3NF et si tout attribut qui n'appartient pas à une clé n'est pas source d'une DF vers une partie d'une clé. Soit la relation **Personne** : Personne(#N°SS, #Pays, Nom, Région)

Soit les DF suivantes sur cette relation :

- DF1 : N°SS,Pays→Nom
- DF2 : N°SS,Pays→Région
- DF3 : Région→Pays

Il existe une DF qui n'est pas issue d'une clé et qui détermine un attribut appartenant à une clé (**DF3**). Cette relation est en 3NF, mais pas en BCNF (car en BCNF toutes les DF sont issues d'une clé).

Pour avoir un schéma relationnel en BCNF, il faut décomposer **Personne** :

- **Personne**(#N°SS, #Region=>**Region**, Nom)
- **Region**(#Region, Pays)

Remarquons que les DF n'ont pas été préservées par la décomposition puisque N°SS et Pays ne déterminent plus Région ==> Une décomposition en BCNF ne préserve pas toujours les DF.

Forme normale de Boyce-Codd (BCNF)

Clé candidate

Une clé candidate est un ensemble d'un ou plusieurs attributs dans une table de base de données **qui peuvent être utilisés comme clé primaire**, mais ce n'est pas nécessairement la clé primaire choisie. Une table peut avoir plus d'une clé candidate, et une clé candidate doit satisfaire les deux propriétés suivantes :

- **Unicité** : chaque clé candidate doit être unique pour chaque ligne de la table, ce qui signifie que deux lignes ne peuvent pas avoir les mêmes valeurs pour les attributs de la clé candidate.
- **Minimalité** : une clé candidate doit être un ensemble minimal d'attributs, ce qui signifie qu'aucun des attributs de la clé candidate ne peut être supprimé sans perdre la capacité d'identifier de manière unique chaque ligne de la table.

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

Quelles sont les clés candidates de cette relation ?

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

Les clés candidates :

- {Movie_Name}
- {Release_Year, Popularity_Ranking}
- {Release_Year_And_Month, Popularity_Ranking}

Forme normale de Boyce-Codd (BCNF)

Attribut principal/premier (Prime attribute)

- Un attribut principal/premier est un attribut qui appartient à une clé candidate. En d'autres termes, il s'agit d'un attribut faisant partie d'un ensemble minimal d'attributs pouvant être utilisé pour identifier de manière unique chaque ligne d'une table.
- Un attribut principal/premier est également appelé "prime" car il s'agit d'un attribut essentiel à la clé candidate et qui ne peut pas être supprimé sans perdre la capacité d'identifier de manière unique chaque ligne de la table.

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

Quels sont les attributs “prime” et non “prime” de cette relation ?

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

Les clés candidates :

- {Movie_Name}
- {Release_Year, Popularity_Ranking}
- {Release_Year_And_Month, Popularity_Ranking}
- ⇒ Chaque attribut est un attribut premier “prime” !

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

3NF Définition formelle : Chaque attribut non premier dans une table doit dépendre de chaque clé candidate ; il ne doit jamais dépendre d'une partie de la clé candidate ; et il ne devrait jamais dépendre d'autres attributs non premiers.

On remarque cette dépendance fonctionnelle : {Release_Year_And_Month} ⇒ {Release_Year}

Est-ce que cette table est dans la 3NF ?

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

- Si nous considérons Movie_Name comme clé primaire, la dépendance fonctionnelle {Release_Year_And_Month} ⇒ {Release_Year} est transitive.
- La table ne doit pas être en 3NF, mais **ELLE L'EST !!**
- La raison en est que la définition formelle stipule qu'un attribut non premier dans une table ne devrait jamais dépendre d'autres attributs non premiers.
  - Le problème ici est qu'il n'y a pas d'attribut non premier dans cette relation. Cela signifie qu'il y a une faille (LOOPHOLE) dans la définition formelle de 3NF.
- Pour résoudre ce problème, une forme plus stricte / plus forte de 3NF a été proposée, et c'est le BCNF ou 3.5NF.

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

BCNF : À l'exception des dépendances fonctionnelles triviales, chaque dépendance fonctionnelle dans une table doit être une dépendance sur une super clé (clé candidate entière).

Remarque:
dépendance fonctionnelle triviale ⇒ dépendance d'un attribut sur lui-même ou lui-même + autre attribut. Ex : {Popularity_Ranking} → {Popularity_Ranking, Movie_Name}

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Most_Popular_Movies_Of_The_Year

Release_Year	Popularity_Ranking	Movie_Name	Release_Year_And_Month
2008	1	The Dark Knight	2008-07
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	2008-05
2008	3	Kung Fu Panda	2008-06
2009	1	Avatar	2009-12
2009	2	Harry Potter and the Half_Blood Prince	2009-07
2009	3	Ice Age: Dawn of the Dinosaurs	2009-07

- Cette relation n'est pas dans la forme normale BCNF car **{Release_Year}** dépend de **{Release_Year_and_Month}** et **{Release_Year_and_Month}** n'est pas une super clé (clé candidate entière).
- Comment transformer cette table en forme normale BCNF ?

Forme normale de Boyce-Codd (BCNF)

Exemple 2

Release_Year	Popularity_Ranking	Movie_Name	Release_Month
2008	1	The Dark Knight	July
2008	2	Indiana Jones and the Kingdom of the Crystal Skull	May
2008	3	Kung Fu Panda	June
2009	1	Avatar	December
2009	2	Harry Potter and the Half_Blood Prince	July
2009	3	Ice Age: Dawn of the Dinosaurs	July

- Cela pourrait être fait en remplaçant **{Release_Year_and_Month}** par **{Release_Month}**. **{Release_Year}** ne dépend plus de **{Release_Month}**.
- BCNF surmonte une faille dans la troisième forme normale.
- Chaque attribut d'une table avec la forme normale BCNF doit dépendre de la clé, de la clé entière et de rien d'autre que la clé.

Dénormalisation

- La dénormalisation des données est un autre concept important, en particulier lorsqu'il s'agit d'interroger l'efficacité.
- Dans une entreprise, tout est question d'équilibre. La plupart des organisations qui traitent des données n'adhèrent pas complètement à la forme complexe de normalisation. La raison en est simple : la normalisation peut nuire à l'efficacité lorsque vous devez interroger une grande quantité de données.
- Par conséquent, alors que nous gagnons en termes d'intégrité des données lors de la normalisation des données, nous payons beaucoup en termes de puissance de traitement si nous devons interroger des données.

Dénormalisation

EmployeeId	Name	Designation	DepartmentID	AppID
67576	Jane Doe	Team Lead	DM	1
79247	Jim Doe	Team Lead	EUC	2
79254	John Doe	Network Specialist	NW	3

- Si un employé a deux adresses e-mail, auriez-vous une seule colonne et y ajouteriez-vous les deux adresses e-mail ? Vous pourriez, mais ce serait non conforme à la première forme normale de données. Comment travaillez-vous autour de ce problème?

Dénormalisation

EmployeeId	Name	Designation	DepartmentID	AppID
67576	Jane Doe	Team Lead	DM	1
79247	Jim Doe	Team Lead	EUC	2
79254	John Doe	Network Specialist	NW	3

- Vous pouvez créer une nouvelle table avec un ID d'employé et une adresse e-mail ? Mais alors, l'employé aurait deux adresses e-mail et ne pourrait pas être identifié de manière unique. Par conséquent, **vous devrez ajouter une clé primaire à cette table** et avec cela, la situation devient beaucoup plus complexe que nécessaire.

Dénormalisation

EmployeeId	Name	Designation	DepartmentID	AppID
67576	Jane Doe	Team Lead	DM	1
79247	Jim Doe	Team Lead	EUC	2
79254	John Doe	Network Specialist	NW	3

- Dans un tel cas, vous pouvez opter pour une dénormalisation mineure et ajouter quelques colonnes à la table **Employee** existante, par exemple, **PrimaryEmail** et **SecondaryEmail**. De cette façon, vos requêtes seraient simplifiées.

Dénormalisation

- La dénormalisation trouve la plupart de son utilisation dans les scénarios de **data warehousing**, c'est-à-dire les situations dans lesquelles vous interrogez un grand nombre de données en une seule fois.
- Dans une base de données normalisée, les données sont organisées en tables séparées avec des relations entre elles, afin de minimiser la redondance et d'assurer l'intégrité des données.
    - Cependant, cela peut entraîner des problèmes de performances, en particulier lors de l'interrogation de la base de données.
- La dénormalisation résout ce problème en introduisant une redondance afin d'accélérer certaines requêtes et de réduire la charge sur le serveur de base de données.
- La dénormalisation peut aussi avoir des inconvénients. Cela peut augmenter la taille de la base de données et la complexité de sa maintenance. Et cela peut également augmenter le risque d'incohérences des données et de problèmes d'intégrité. **Par conséquent, il est important de peser les coûts et les avantages de la dénormalisation avant d'apporter des modifications à un schéma de base de données.**

Conclusion

2NF Définition formelle : Nous ne pouvons pas avoir un attribut non premier qui dépend d'une partie d'une clé candidate (dépendance partielle).

3NF Définition formelle : Chaque attribut non premier dans une table doit dépendre de chaque clé candidate ; il ne doit jamais dépendre d'une partie de la clé candidate ; et il ne devrait jamais dépendre d'autres attributs non premiers.

BCNF : À l'exception des dépendances fonctionnelles triviales, chaque dépendance fonctionnelle dans une table doit être une dépendance sur une super clé (clé candidate entière).