Module 2 SQL Avancé et Normalisation

IGE 487 Modélisation de bases de données

Chargé de cours : Tudor Antohi

Plan

- **Explications sur le Devoir 1**
- **Explications sur le Devoir 2**
- **Connexions BD sur les equipes et les projets**
- **Exercices demontrés**
- **Recap quelques SQLs**
- **Normalisation**

Histoire SQL

Standard courant : SQL:2016

→ SQL:2016 → JSON, Fonctions tables polymorphes (*PostgreSQL couvres 170 sur 177 exigences de SQL:2016*)

→ SQL:2011 → Bases de données temporelles, Pipelined DML

→ SQL:2008 → Truncate, Sorting

→ SQL:2003 → XML, Windows, Séquences, Auto-Gen IDs.

→ SQL:1999 → Regex, Triggers, OO

Standard minimal demandé pour un système qui supporte SQL est SQL-92

SQL et autres langages

1. Kusto KQL – Azure Data Explorer
2. GraphQL , Rest API – Langages de requêtes APIs
3. Gremlin – Apache Tinker Pop, CosmosDB gremlin
4. Cypher - systèmes Graph , Neo4J
5. Lucene – Elastic Search
6. N1QL - Couchbase
7. Mongo
8. Autres …

SQL gagne par maturité mais les yeux ouverts.

Question vers vendeur : avez-vous l'infrastructure pour soutenir votre langage ?

Langages relationnels

DML – langage de manipulation de données (ex.: INSERT, SELECT, UPDATE, DELETE, MERGE)

DDL – langage de définition de données (création des objets comme tables, vues, contraintes ex.: CREATE TABLE)

DCL – langage de contrôle de données (sécurité, contrôle les accès ex.: GRANT, REVOKE)

SQL est basé sur les sacs (bags, duplications) pas les ensembles (pas de duplications)

Récap SQL et SQL Avancé

- Jointures complexes
- Agrégations et GROUP BY
- Opérations sur les chaines de caractères (STRING) et temporelles
- Contrôle de sortie + redirection
- Requêtes imbriquées
- Expressions sur les tables
- Fonctions Window

Pourquoi les NULLs sont si compliqués ?

- **NULL n’est pas la chaine vide ‘’.**
- **Toutes les valeurs sont nulles dans cette requête.**

```
SELECT NULL, 1+NULL, CONCAT('Invisible',NULL);
```

- **A = NULL est toujours Faux , A <> NULL est toujours FAUX**

Pour faciliter la gestion des valeurs NULL, on utilise les opérateurs IS NULL et IS NOT NULL et la fonction IFNULL().

```
SELECT * FROM my_table WHERE phone IS NULL;
SELECT * FROM my_table WHERE phone = '';
```

Jointures

- SEMI JOINTURE
- ANTI JOIN (SEMI-DIFFERENCE)
- THETA JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

SEMI JOIN = SEMI JOINTURE ⋉

Semi-Join compare les tuples de deux relations et affiche les lignes correspondantes de la relation dont le nom est mentionné à gauche de l'opérateur ⋉

```
SELECT DISTINCT s.id FROM students s
LEFT JOIN grades g ON g.student_id=s.id
WHERE g.student_id IS NOT NULL
```

ANTI-JOIN Semi-différence, ⋉, NOT MATCHING

Une **anti-join (semi-différence)** envoie une copie de chaque ligne de la première table pour laquelle aucune correspondance n'est trouvée.

$$R \ltimes S = R - (R \bowtie S)$$

```
SELECT l.id, l.value FROM t_left l
LEFT JOIN t_right r
ON r.value= l.value WHERE r.value IS NULL
```

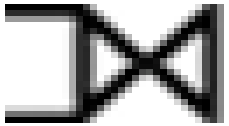
Theta Join θ et equi-join

La jointure theta est une généralisation de jointures.

La condition theta peut utiliser n’importe quelle condition.

$$A \bowtie_\theta B = \sigma_\theta(A \times B)$$

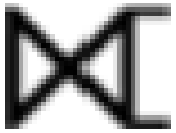
Lorsqu'une jointure thêta n'utilise que la condition d'équivalence, elle devient une **jointure equi-join.**

LEFT OUTER JOIN 

L'opération permet de conserver tous les tuple dans la relation située a gauche. Si aucun tuple correspondant n'est trouvé dans la relation a droite, les attributs de la relation de droite sont remplis de valeurs nulles.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID =
Orders.CustomerID
ORDER BY Customers.CustomerName;
```

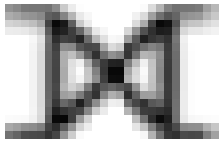
A $\bowtie_\theta$ B

RIGHT OUTER JOIN 

L'opération permet de conserver tous les tuple dans la relation situé a droite. Si aucun tuple correspondant n'est trouvé dans la relation de gauche, les attributs de la relation de gauche dans le résultat de la jointure sont remplis de valeurs nulles.

```
SELECT Orders.OrderID, Employees.LastName,
Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID =
Employees.EmployeeID
ORDER BY Orders.OrderID;
```

A $\bowtie_e$ B

FULL OUTER JOIN 

Tous les tuples des deux relations sont inclus dans le résultat, indépendamment de la condition de correspondance.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Order
s.CustomerID
ORDER BY Customers.CustomerName;
```

Base de données – exemple

Étudiant (Student)

sid	name	login	age	gpa
53666	Kanye	kayne@cs	39	4.0
53688	Bieber	jbieber@cs	26	3.9
53655	Tupac	shakur@cs	26	3.5

Cours (CourseIds)

cid	name
15-445	Database Systems
15-721	Advanced Database Systems
15-826	Data Mining
15-823	Advanced Topics in Databases

Inscription (Enroll)

sid	cid	grade
53666	15-445	C
53688	15-721	A
53688	15-826	B
53655	15-445	B
53666	15-721	C

Agrégations

→ **AVG(col)** → la valeur moyenne de la colonne

→ **MIN(col)** → la valeur minimale de la colonne.

→ **MAX(col)** → la valeur maximale de la colonne.

→ **SUM(col)** → la somme de la colonne.

→ **COUNT(col)** → le # de valeurs pour col.

Agrégations – GROUP BY

Le *# des étudiants avec "@TI" login:*

```
SELECT COUNT(login) AS cnt FROM student WHERE login LIKE'%@cs';
```

- `COUNT(login), COUNT(login), COUNT(login)`

```
SELECT AVG(gpa), COUNT(sid) FROM student WHERE login LIKE'%@cs';
```

```
SELECT AVG(s.gpa), e.cid FROM enrolled ASe, student AS s WHEREe.sid = s.sid GROUP BY e.cid;
```

```
SELECT AVG(s.gpa) AS avg_gpa, e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid AND avg_gpa > 3.9 FROM enrolled AS e, student AS s WHERE e.sid = s.sid AND avg_gpa > 3.9;
```

```
GROUP bY e.cid
```

e.sid	s.sid	s.gpa	e.cid
53435	53435	2.25	15-721
53439	53439	2.70	15-721
56023	56023	2.75	15-826
59439	59439	3.90	15-826
53961	53961	3.50	15-826
58345	58345	1.89	15-445

AVG(s.gpa)	e.cid
2.46	15-721
3.39	15-826
1.89	15-445

avg_gpa	e.cid
3.950000	15-721

Agrégations – clause HAVING

Les filtres s'applique après la calcule d'agrégation.

```
SELECT AVG(s.gpa) AS avg_gpa, e.cid FROM enrolled AS e, student AS s
WHERE e.sid = s.sid AND avg_gpa> 3.9 GROUP BY e.cid HAVING avg(s.gpa) > 3.9
```

e.sid	s.sid	s.gpa	e.cid
53435	53435	2.25	15-721
53439	53439	2.70	15-721
56023	56023	2.75	15-826
59439	59439	3.90	15-826
53961	53961	3.50	15-826
58345	58345	1.89	15-445

AVG(s.gpa)	e.cid
3.75	15-415
3.950000	15-721
3.900000	15-826

avg_gpa	e.cid
3.950000	15-721

Opérations sur les chaines de caractères

LIKE est utilisé pour la comparaison de chaînes.

'%' Correspond à n'importe quelle sous-chaîne (y compris chaînes vides). '_' Correspond à n'importe quel caractère

Étudiant (Student)

sid	name	login	age	gpa
53666	Kanye	kayne@cs	39	4.0
53688	Bieber	jbieber@cs	26	3.9
53655	Tupac	shakur@cs	26	3.5

Inscription (Enroll)

sid	cid	grade
53666	15-445	C
53688	15-721	A
53688	15-826	B
53655	15-445	B
53666	15-721	C

```
SELECT * FROM enrolled Ase WHERE e.cid LIKE'15-%'
SELECT * FROM student AS s WHERE s.login LIKE '%@c_'
SELECT SUBSTRING(name,1,5) AS abbrv_name FROM student WHERE sid= 53688
SELECT* FROM student AS s WHERE UPPER(s.name) LIKE 'KAN%'
SELECTname FROM student WHERE login = LOWER(name) || '@cs'
```

CONTROL de sortie

```
SELECT DISTINCT cid INTO CourseIds FROM enrolled;
CREATE TABLE CourseIds AS SELECT DISTINCT cid FROM enrolled;
INSERT INTO CourseIds (SELECT DISTINCT cid FROM enrolled);
SELECT sid, grade FROM enrolled WHERE cid = '15-721' ORDER By grade DESC
, SID ASC
SELECT sid, name FROM student WHERE login LIKE '%@cs' LIMIT 20 OFFSET10
```

Étudiant (Student)

sid	name	login	age	gpa
53666	Kanye	kayne@cs	39	4.0
53688	Bieber	jbieber@cs	26	3.9
53655	Tupac	shakur@cs	26	3.5

Inscription (Enroll)

sid	cid	grade
53666	15-445	C
53688	15-721	A
53688	15-826	B
53655	15-445	B
53666	15-721	C

Cours (CourseIds)

cid	name
15-445	Database Systems
15-721	Advanced Database Systems
15-826	Data Mining
15-823	Advanced Topics in Databases

REQUETES IMBRIQUÉES (Nested)

Requêtes contenant d'autres requêtes, souvent difficiles à optimiser. Les requêtes internes peuvent apparaître (presque) n'importe où dans la requête.

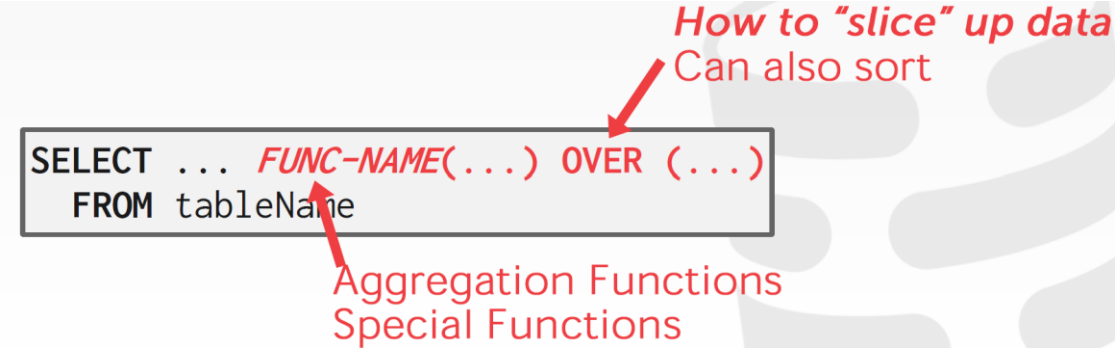
1. **ALL**→ Doit satisfaire l'expression pour toutes les lignes de la sous-requête.
2. **ANY**→ Doit satisfaire l'expression pour au moins une ligne dans la sous-requête.
3. **IN** → Équivalent à '=ANY()' .
4. **EXISTS**→ Au moins une ligne est renvoyée.

```
SELECT name FROM student WHERE sid = ANY(SELECT sid FROM enrolled WHERE cid= '15-445')
```

```
SELECT name FROM student WHERE sid IN (SELECT sid FROM enrolled WHERE cid= '15-445'
```

```
SELECT * FROM student WHERE EXISTS (select name from table where 1 = 2 );
```

Fonctions Fenêtre (WINDOW)




```
SELECT *,ROW_NUMBER() OVER () AS row_num FROM enrolled
```

```
SELECT cid, sid, ROW_NUMBER() OVER (PARTITION BY cid) FROM enrolled ORDER BY cid
```

sid	cid	grade	row_num
53666	15-445	C	1
53688	15-721	A	2
53688	15-826	B	3
53655	15-445	B	4
53666	15-721	C	5

cid	sid	row_number
15-445	53666	1
15-445	53655	2
15-721	53688	1
15-721	53666	2
15-826	53688	1

Expressions de type Table et Récursivité

Fournit un moyen d'écrire des instructions auxiliaires à utiliser dans une requête plus importante.
→ Pensez-y comme une table temporaire juste pour une seule requête.
Alternative aux requêtes imbriquées et aux vues.

```
WITH cteName(col1, col2) AS
(SELECT 1, 2 )
SELECT col1+ col2 FROM cteName

WITH RECURSIVE cteSource(counter) AS
(
(SELECT 1)
UNION
(SELECT counter + 1 FROM cteSource
WHERE counter < 10)
)
SELECT  *  FROM cteSource
```

- 6FN ⇒ 5FN ⇒ 4FN ⇒ FNBC ⇒ 3FN ⇒ 2FN ⇒ 1FN

Attributs – vocabulaire de base

Un **attribut** est **atomique** si et seulement si il est associé à une seule valeur de son type de définition.

Un attribut est **premier** si et seulement si il est élément d'une **clé candidate**.

Un ensemble d'attributs est une **sous-clé** relativement a une relation R, s'il est un sous-ensemble d'une **clé candidate** de R.

Un ensemble d'attributs est une **sur-clé** relativement a une relation R, s'il contient une **clé candidate** de **R**.

ANC – attributs non-clés

0CC – il existe zéro clés candidates

Dépendance fonctionnelle

Dépendance fonctionnelle. Dans une relation R, on dit qu'il y a dépendance fonctionnelle entre un ensemble d'attributs A et un ensemble d'attributs B, ou que l'ensemble A d'attributs détermine l'ensemble B d'attributs (et on écrit **DF[A→B])**

Une DF [A→B] est **triviale** si et seulement si B est un sous-ensemble de A.

Une DF [A→B] est **irréductible** si et seulement si aucun sous-ensemble propre de A ne détermine B.

S'il existe une **DF irréductible** entre {A1, ..., Ak} et tous les autres attributs Ak+1, ..., An d'une relation R, alors {A1, ..., Ak} est une **clé candidate** de R.

Une DF [A→B] est **applicable** a une relation R si et seulement si A et B sont sous-ensembles de l'entête de R.

Soit X et Y deux sous-ensembles d'attributs d'une relation R, une **DM X -≫ Y** représente le fait que ≪ *tout tuple ayant les mêmes valeurs en X est associe a un seul et même ensemble de valeurs en Y* ≫

Dépendance fonctionnelle - exemple

Les détails du **nom de l'employé, du salaire et de la ville** sont obtenus par la valeur du **numéro d'employé** (ou id d'un employé).

On peut donc dire que les attributs ville, salaire et nom sont fonctionnellement dépendants de l'attribut **Numéro d'employé.**

L’établissement de dependences fonctionnelles se fait avec les analystes fonctionnelles ou les analystes d’affaires ou directement avec le client.

Dépendance fonctionnelle – règles Amstrong

1. réflexivité : si $X \supseteq Y$, alors $X \rightarrow Y$
2. augmentation : si $X \rightarrow Y$, alors $XZ \rightarrow YZ$
3. transitivité : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$
4. décomposition : si $X \rightarrow YZ$, alors $X \rightarrow Y$ et $X \rightarrow Z$
5. union : si $X \rightarrow Y$ et $X \rightarrow Z$, alors $X \rightarrow YZ$
6. pseudo-transitivité : si $X \rightarrow Y$ et $WY \rightarrow Z$, alors $WX \rightarrow Z$

1FN – dépendances fonctionnelles

Si une relation contient un attribut composé ou à valeurs multiples, elle viole la première forme normale, ou la relation est dans la première forme normale si elle ne contient aucun attribut composé ou à valeurs multiples. Une relation est en première forme normale si chaque attribut de cette relation est un attribut à valeur unique.

Une table est en 1FN si :

- Il n'y a que des **attributs avec une valeur unique, pas des listes de valeurs**
- Le **domaine d'attribut** est fixe.
- Il existe un **nom unique** pour chaque attribut/colonne.
- **L'ordre** dans lequel les **données** sont **stockées** n'a **pas d'importance**.

1FN – exemple

ID	Name	Courses
1	A	c1, c2
2	E	c3
3	M	C2, c3

ID	Name	Course
1	A	c1
1	A	c2
2	E	c3
3	M	c2
3	M	c3

Sales Records:

Cust Name	Item	Shipping Address	Newsletter	Supplier	Supplier Phone	Price
Alan Smith	Xbox One	35 Palm St, Miami	Xbox News	Microsoft	(800) BUY-XBOX	250
Roger Banks	PlayStation 4	47 Campus Rd, Boston	PlayStation News	Sony	(800) BUY-SONY	300
Evan Wilson	Xbox One, PS Vita	28 Rock Av, Denver	Xbox News, PlayStation News	Wholesale	Toll Free	450
Alan Smith	PlayStation 4	47 Campus Rd, Boston	PlayStation News	Sony	(800) BUY-SONY	300

Primary Key

Cust ID	Cust Name	Item	Shipping Address	Newsletter	Supplier	Supplier Phone	Price
at_smith	Alan Smith	Xbox One	35 Palm St, Miami	Xbox News	Microsoft	(800) BUY-XBOX	250
roger25	Roger Banks	PlayStation 4	47 Campus Rd, Boston	PlayStation News	Sony	(800) BUY-SONY	300
wilson44	Evan Wilson	Xbox One	28 Rock Av, Denver	Xbox News	Microsoft	(800) BUY-XBOX	250
wilson44	Evan Wilson	PS Vita	28 Rock Av, Denver	PlayStation News	Sony	(800) BUY-SONY	200
am_smith	Alan Smith	PlayStation 4	47 Campus Rd, Boston	PlayStation News	Sony	(800) BUY-SONY	300

2FN – dépendances fonctionnelles

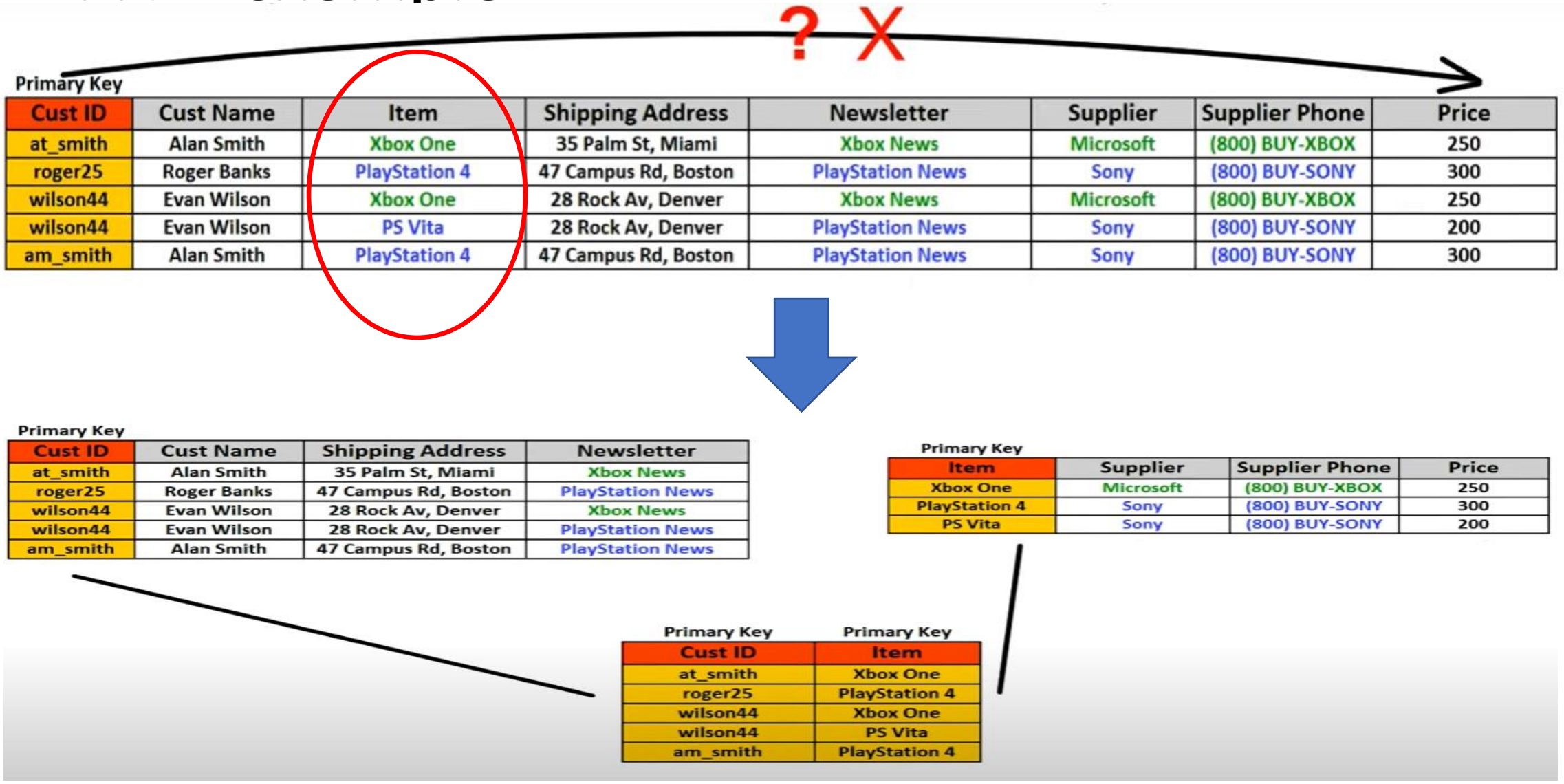
Une relation R est en **2FN** si et seulement si tous les attributs **non premiers** de R sont en *dépendance fonctionnelle irréductible* de chaque clé candidate de E.

Une relation R est en **2FN** si et seulement si, pour chaque **DF applicable non triviale [X→A]**, une des conditions suivantes est satisfaite :

- X est une **sur-clé,**
- A est **premier**,
- X n'est pas une **sous-clé**.

Tous les attributs non-clés sont dépendent de clés en 2FN.

2FN – exemple


? X
Primary Key
Cust ID | Cust Name | Item | Shipping Address | Newsletter | Supplier | Supplier Phone | Price
at_smith | Alan Smith | Xbox One | 35 Palm St, Miami | Xbox News | Microsoft | (800) BUY-XBOX | 250
roger25 | Roger Banks | PlayStation 4 | 47 Campus Rd, Boston | PlayStation News | Sony | (800) BUY-SONY | 300
wilson44 | Evan Wilson | Xbox One | 28 Rock Av, Denver | Xbox News | Microsoft | (800) BUY-XBOX | 250
wilson44 | Evan Wilson | PS Vita | 28 Rock Av, Denver | PlayStation News | Sony | (800) BUY-SONY | 200
am_smith | Alan Smith | PlayStation 4 | 47 Campus Rd, Boston | PlayStation News | Sony | (800) BUY-SONY | 300
Primary Key
Cust ID | Cust Name | Shipping Address | Newsletter
at_smith | Alan Smith | 35 Palm St, Miami | Xbox News
roger25 | Roger Banks | 47 Campus Rd, Boston | PlayStation News
wilson44 | Evan Wilson | 28 Rock Av, Denver | Xbox News
wilson44 | Evan Wilson | 28 Rock Av, Denver | PlayStation News
am_smith | Alan Smith | 47 Campus Rd, Boston | PlayStation News
Primary Key
Item | Supplier | Supplier Phone | Price
Xbox One | Microsoft | (800) BUY-XBOX | 250
PlayStation 4 | Sony | (800) BUY-SONY | 300
PS Vita | Sony | (800) BUY-SONY | 200
Primary Key
Primary Key
Cust ID | Item
at_smith | Xbox One
roger25 | PlayStation 4
wilson44 | Xbox One
wilson44 | PS Vita
am_smith | PlayStation 4

3FN – dépendances fonctionnelles

Une relation R est en 3FN si et seulement si, pour chaque **DF applicable non triviale [X→A],** une des conditions suivantes est satisfaite :

- X est une **sur-clé**,
- A est un **attribut premier**.

Tous les attributs non-clés sont dépendent **SEULEMENT** de clés.

3FN – exemple de 2FN vers 3FN

Cust ID	Cust Name	Shipping Address	Newsletter
at_smith	Alan Smith	35 Palm St, Miami	Xbox News
roger25	Roger Banks	47 Campus Rd, Boston	PlayStation News
wilson44	Evan Wilson	28 Rock Av, Denver	Xbox News
wilson44	Evan Wilson	28 Rock Av, Denver	PlayStation News
am_smith	Alan Smith	47 Campus Rd, Boston	PlayStation News

Primary Key

Item	Supplier	Supplier Phone	Price
Xbox One	Microsoft	(800) BUY-XBOX	250
PlayStation 4	Sony	(800) BUY-SONY	300
PS Vita	Sony	(800) BUY-SONY	200

Primary Key Primary Key

Cust ID	Item
at_smith	Xbox One
roger25	PlayStation 4
wilson44	Xbox One
wilson44	PS Vita
am_smith	PlayStation 4

Primary Key

Cust ID	Cust Name	Shipping Address	Newsletter
at_smith	Alan Smith	35 Palm St, Miami	Xbox News
roger25	Roger Banks	47 Campus Rd, Boston	PlayStation News
wilson44	Evan Wilson	28 Rock Av, Denver	Xbox News
wilson44	Evan Wilson	28 Rock Av, Denver	PlayStation News
am_smith	Alan Smith	47 Campus Rd, Boston	PlayStation News

Primary Key Foreign Key

Item	Supplier	Price
Xbox One	Microsoft	250
PlayStation 4	Sony	300
PS Vita	Sony	200

Primary Key Primary Key

Cust ID	Item
at_smith	Xbox One
roger25	PlayStation 4
wilson44	Xbox One
wilson44	PS Vita
am_smith	PlayStation 4

Primary Key

Supplier	Supplier Phone
Microsoft	(800) BUY-XBOX
Sony	(800) BUY-SONY

FNBC – dépendances fonctionnelles

Une relation R est en FNBC relativement une **DF non triviale [X→A]** applicable si et seulement si **X** est une **sur-clé** de R

Par définition, la table est considérée comme la forme normale de Boyce-Codd, si **elle est déjà dans la troisième forme normale** et pour **chaque dépendance fonctionnelle entre A et B**, A devrait être une **super clé.**

3FN permet aux attributs de faire partie d'une clé candidate qui n'est pas la clé primaire ; FNBC ne le permet pas.

FNBC - Théorème de Heath

Théorème

- Soit R, une relation dont l'entête est E,
- soit X, Y et Z des sous-ensembles de E tels que leur union est egale a E,
- si R est conforme a la DF : $X \rightarrow Y$

alors R est égal a la jointure de ses projections sur $X \cup Y$ et $X \cup Z$.

Corolaire

- . La FNBC est la forme normale ultime relativement aux DF.

Une relation R(X,Y,Z) qui satisfait une dépendance fonctionnelle $X \rightarrow Y$ peut toujours être décomposée sans perte en ses projections $R1=\pi_{XY}(R)$ et $R2=\pi_{XZ}(R)$.

FNBC – exemple de 3FN qui devient FNBC

ID	TutorID	TutorSIN
8245	1254	000 737 999
9995	1567	000 656 999
5643	1254	000 737 999
6179	1742	000 651 999

ID	**TutorID**
8245	1254
9995	1567
5643	1254
6179	1742

TutorID	TutorSIN
1254	000 737 999
1567	000 656 999
1742	000 651 999

3FN permet aux attributs de faire partie d'une clé candidate qui n'est pas la clé primaire ; BCNF ne le permet pas.

Les clés candidates sont : {ID, TutorID} et {ID, TutorSIN}

TutorID → TutorSIN et **TutorSIN → TutorID,** mais comme TutorID et TutorSIN sont tous deux des attributs premiers, ces FD ne violent pas la 3FN.

Ni TutorID ni TutorSIN seuls ne sont des super-clés, et donc la FNBC est violée.

La décomposition règle le problème.

Récap 2FN, 3FN, FNBC

Soit les conditions suivantes relativement a une DF non triviale [X→A] applicable a R :

1. X est une sur-clé,

2. A est premier,

3. X n'est pas une sous-clé.

Si *chacune* des DF non triviale [X→A] applicable a R répond :

- a la condition 1, alors R est en FNBC,
- a l'une des conditions 1 ou 2, alors R est en 3FN,
- a l'une des conditions 1, 2 ou 3, alors R est en 2FN

4FN – DM dépendances multi-valeurs

Soit :

- E l'entete de R et
- $Z = E-(X \cup Y)$

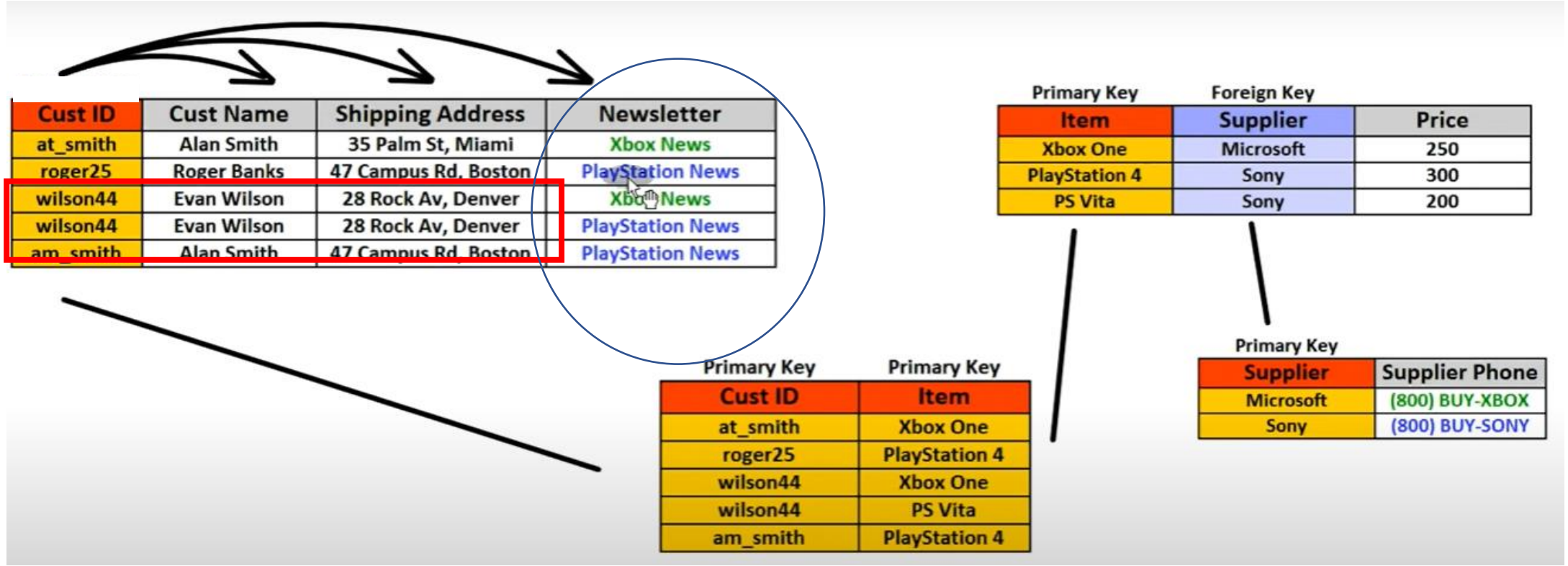
R est en 4FN relativement a la DM $X \twoheadrightarrow Y$ si et seulement si (X U Y) est une surclé de R

Autre définition :

- FNBC et
- Pas de dépendances multi-valeurs.

On dit qu'une dépendance multi-valeurs se produit lorsqu'il y a deux attributs dans une table qui dépendent d'un troisième attribut mais sont indépendants l'un de l'autre.

4FN – Exemple 3FN qui n'est pas 4FN



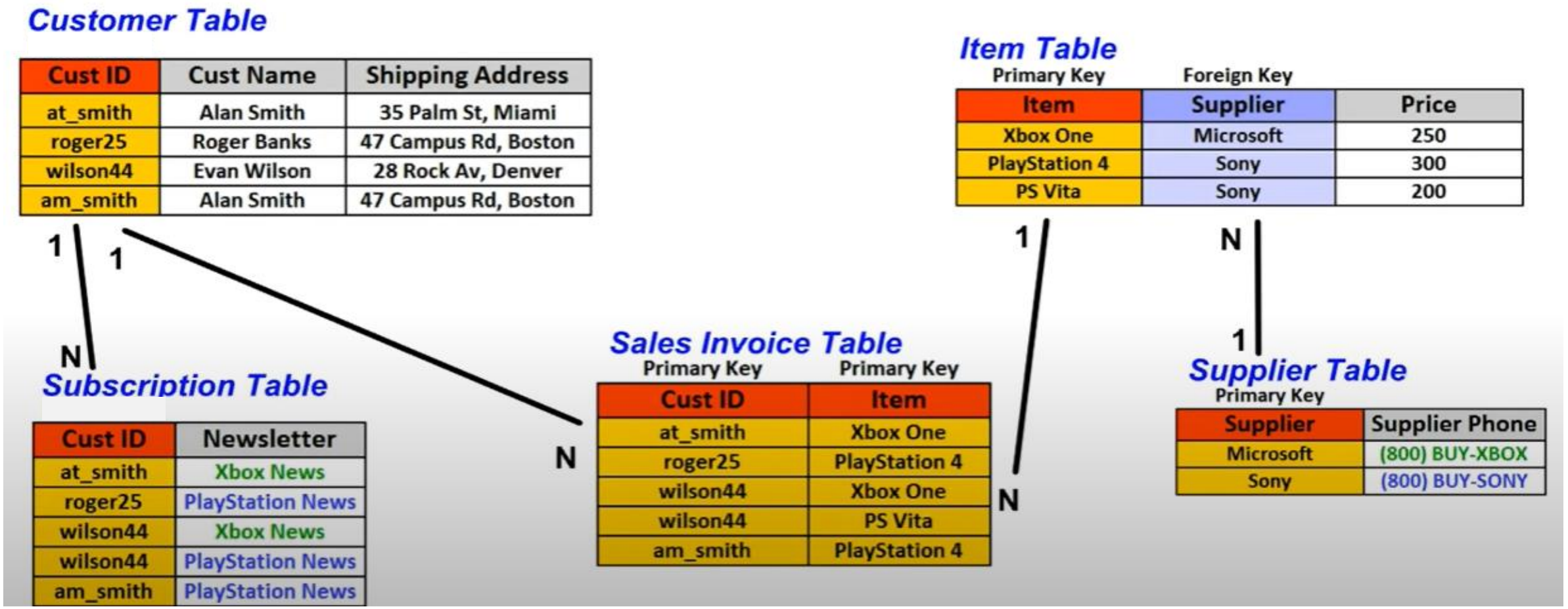
3FN vers 4FN

- Les attributs **newsletter** et (**customer name, customer address**) sont indépendantes.
- Les valeurs de **newsletter** et **customer address** dependent de valeurs de **cust_id**. On a donc 2 attributs qui dependent de valeurs d'un 3-éme et les deux attributs sont indépendants.

{cust_id} ↠ {customer_address}

{cust_id} ↠ {newsletter}

4FN – Exemple


Customer Table
Cust ID
Cust Name
Shipping Address
at_smith
Alan Smith
35 Palm St, Miami
roger25
Roger Banks
47 Campus Rd, Boston
wilson44
Evan Wilson
28 Rock Av, Denver
am_smith
Alan Smith
47 Campus Rd, Boston
Item Table
Primary Key
Foreign Key
Item
Supplier
Price
Xbox One
Microsoft
250
PlayStation 4
Sony
300
PS Vita
Sony
200
1
1
N
1
N
1
N
1
Subscription Table
Cust ID
Newsletter
at_smith
Xbox News
roger25
PlayStation News
wilson44
Xbox News
wilson44
PlayStation News
am_smith
PlayStation News
Sales Invoice Table
Primary Key
Primary Key
Cust ID
Item
at_smith
Xbox One
roger25
PlayStation 4
wilson44
Xbox One
wilson44
PS Vita
am_smith
PlayStation 4
Supplier Table
Primary Key
Supplier
Supplier Phone
Microsoft
(800) BUY-XBOX
Sony
(800) BUY-SONY

Théorème de Fagin

Théorème

- Soit R, une relation dont l'entête est E, et soit X, Y et Z des sous-ensembles de E tels que leur union est egale a E,

R est égal a la jointure de ses projections sur $X \cup Y$ et $X \cup Z$ si et seulement si $X \twoheadrightarrow Y$

Corolaire

- La 4FN est donc l'ultime forme normale pour les dépendances multi-valeurs

$(X \twoheadrightarrow Y) \Leftrightarrow (X \twoheadrightarrow Z)$

Dépendance de jointure (DJ)

⋈{d1, ... , dk} est une dépendance de jointure DJ sur E, si et seulement si :

d1 ⊆ E, ..., dk ⊆ E et E = d1 ∪ ... ∪ dk

Si une table T peut être crée en joignant plusieurs tables Ti et chacune de ces tables Ti contient un sous-ensemble de l’attributs de la table, cette table a une dépendance de jointure.

Dépendance de jointure (DJ) – exemples

Client = {numéro de commande, nom-client, nom-pizza, coursier}

1. {numéro de commande, nom-client} et {nom-pizza, coursier} n’est pas une dependence de jointure.

2. {numéro de commande, nom-client } et {nom-client, nom-pizza, coursier} est une dependence de jointure.

5FN

Une relation R est en cinquième forme normale (**5FN**) si et seulement si toutes les **DJ** qui lui sont applicables, sont induites par les clés de la relation R.

$$R = (\pi_{d1} R) \bowtie \ldots \bowtie (\pi_{dk} R)$$

Conclusions :

FNBC + ANC <=> 5FN

3FN + 0CC <=> 5FN

Rappel

ANC – attributs non-clés

0CC – il existe zéro clés candidates

5FN

Une table est dans la **5FN** si et seulement si chaque dépendance de jointure non triviale dans cette table est impliquée par les clés candidates.

Une table T est en cinquième forme normale (**5FN**) ou en forme normale de jointure par projection (PJ/NF) si elle ne peut pas avoir une décomposition sans perte en un nombre de tables plus petites.

Ce n'est que dans de rares situations qu'une table **4FN** n'est pas conforme à **5FN**. Par exemple, lorsque les tables décomposées sont cycliques.

4FN vers 5FN qui sont aussi de 6FN

Relation R

Département	Sujet	Nom
CSE	**C**	Jean
CSE	**C**	Jean
CSE	Java	Ali
IT	C	Isak

département ->-> sujet, département ->-> nom

La relation ci-dessus est en 4NF. La clé primaire est (**département, sujet, nom).** Parfois, la décomposition d'une relation en deux relations plus petites n'élimine pas la redondance. Dans de tels cas, il peut être possible de décomposer la relation en trois relations ou plus en utilisant 5NF. Par conséquent, la relation peut être décomposée en trois relations suivantes.

R1(département, sujet) R2(département, nom) et R3(sujet, nom)

Revue 2FN, 3FN, FNBC

Soit les conditions suivantes relativement a une DF non triviale [X→A] applicable a R:

1. X est une sur-clé,

2. A est un attribut premier, fait partie d'une clé candidate.

3. X n'est pas une sous-clé.

Si *chacune* des DF non triviale [X→A] applicable a R répond :

- a la condition 1, alors R est en FNBC,
- a l'une des conditions 1 ou 2, alors R est en 3FN,
- a l'une des conditions 1, 2 ou 3, alors R est en 2FN

Revue FNBC – 4FN - 5FN

Une relvar R est en FNBC si et seulement si chaque DF à laquelle elle satisfait est impliquée par les clés candidates de R.

Une relvar R est en 4FN si et seulement si chaque DMV à laquelle elle satisfait est impliquée par les clés candidates de R.

Une relvar R est en 5FN si et seulement si chaque DJ à laquelle elle satisfait est impliquée par les clés candidates de R. En général les 4FN sont en 5FN – les cas rares sont les relations cycliques.

6FN

Trois définitions :

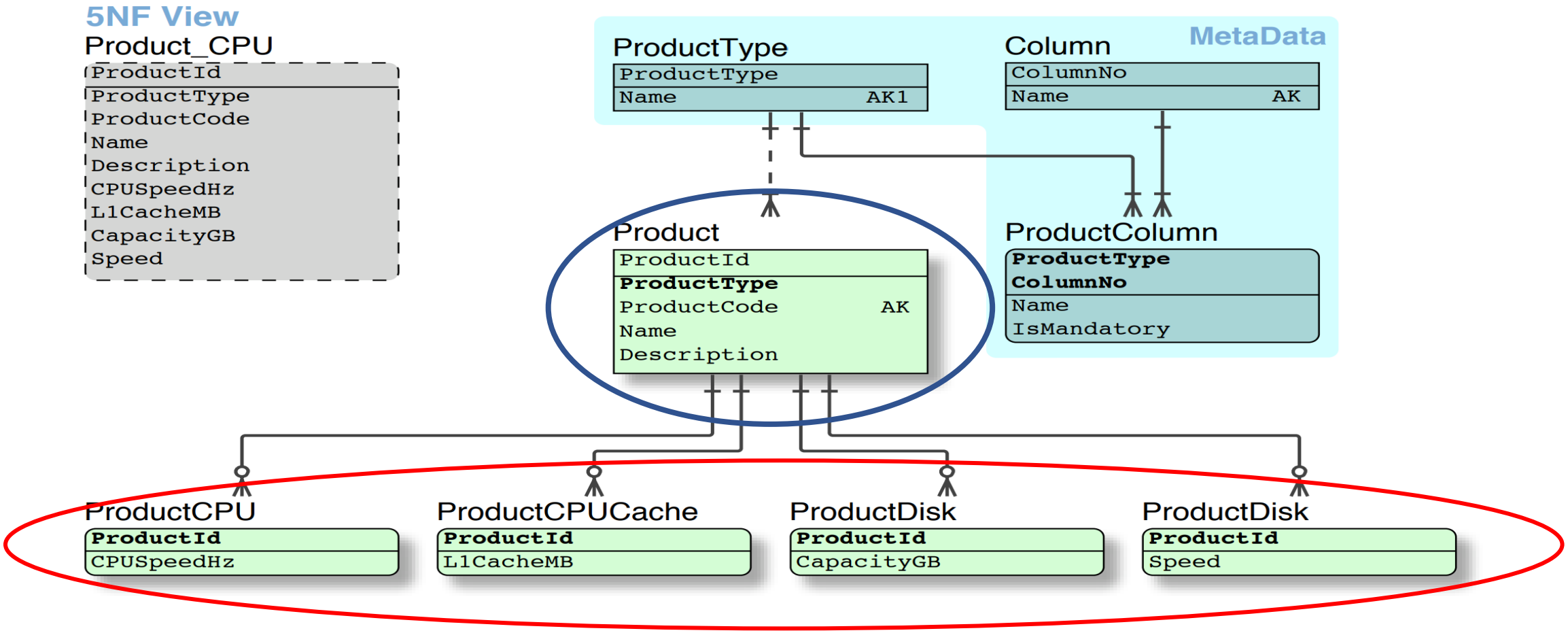
- Une relvar est en sixième forme normale (6FN) si et seulement si, quelle que soit la **dépendance de jointure** à laquelle elle satisfait, cette dépendance est **triviale**.
- Une relvar est en 6FN si et seulement si elle est en 5FN, elle est de degré *n*, et n'a aucune clé de degré inferieur à *n* - 1.
- Une relvar est en 6FN si et seulement si elle ne peut être **PJ** décomposée en relations de degré inferieur.

Une table en 6FN contient la clé primaire et maximum un attribut.

5 FN vers 6FN – faire attention

Product 6NF

No Null Columns



Bibliographie

1. Luc Lavoie et Christina Khnaisser

http://info.usherbrooke.ca/llavoie/enseignement/IGE487/index.php

BD0025_PRE : Normalisation

Aussi

https://fsmrel.developpez.com/basesrelationnelles/normalisation/

Bibliographie

1. Elmasri et Navathe

Database Systems 7th edition

Chapitre 14 - Basics of Functional Dependencies and Normalization for Relational Databases