

---

# Compte rendu

## SID - Projet

Thomas Belaid

Pierre Gavrel

Baptiste Tivrier

---

— 2022 —

# SOMMAIRE

- CONTEXTE
- MODÉLISATION RELATIONNELLE
- EXPLICATION DES CHOIX
- DESCRIPTION DES TABLES
- CRÉATION DU CONTENUE
- GRANULARITÉ , GESTION DE CHANGEMENT ET DIMENSION
- MESURE ET REQUÊTE

# CONTEXTE

Le projet en cours vise à mettre en place un système d'analyse approfondie du comportement des utilisateurs. Cette initiative ambitionne de fournir à Nil des indicateurs pertinents concernant la navigation des clients sur la plateforme, les pages et catégories les plus consultées, ainsi que les taux de conversion. Ce faisant, Nil aspire à optimiser l'expérience client et à prendre des décisions éclairées pour soutenir sa croissance continue sur le marché du e-commerce des livres.

Pour cela il nous a été demander de produire plusieurs livrables qui comporte :-

- Un Code R visant à la création de la notre datawarehouse et l'insertion des données afin de pouvoir par la suite via des requêtes répondre à certaines interrogations.
- Une maquette de reporting qu'on pourrait utiliser de façon hebdomadaire afin d'avoir une suivi et un historique de tous les KPI clés importants au développement du site
- Un rapport de nos travaux

# MODÉLISATION RELATIONNELLE

La modélisation relationnelle est une méthode pour structurer les données en utilisant des tables et des liens entre elles, simplifiant ainsi la gestion et l'accès aux informations dans une base de données.

MCD :

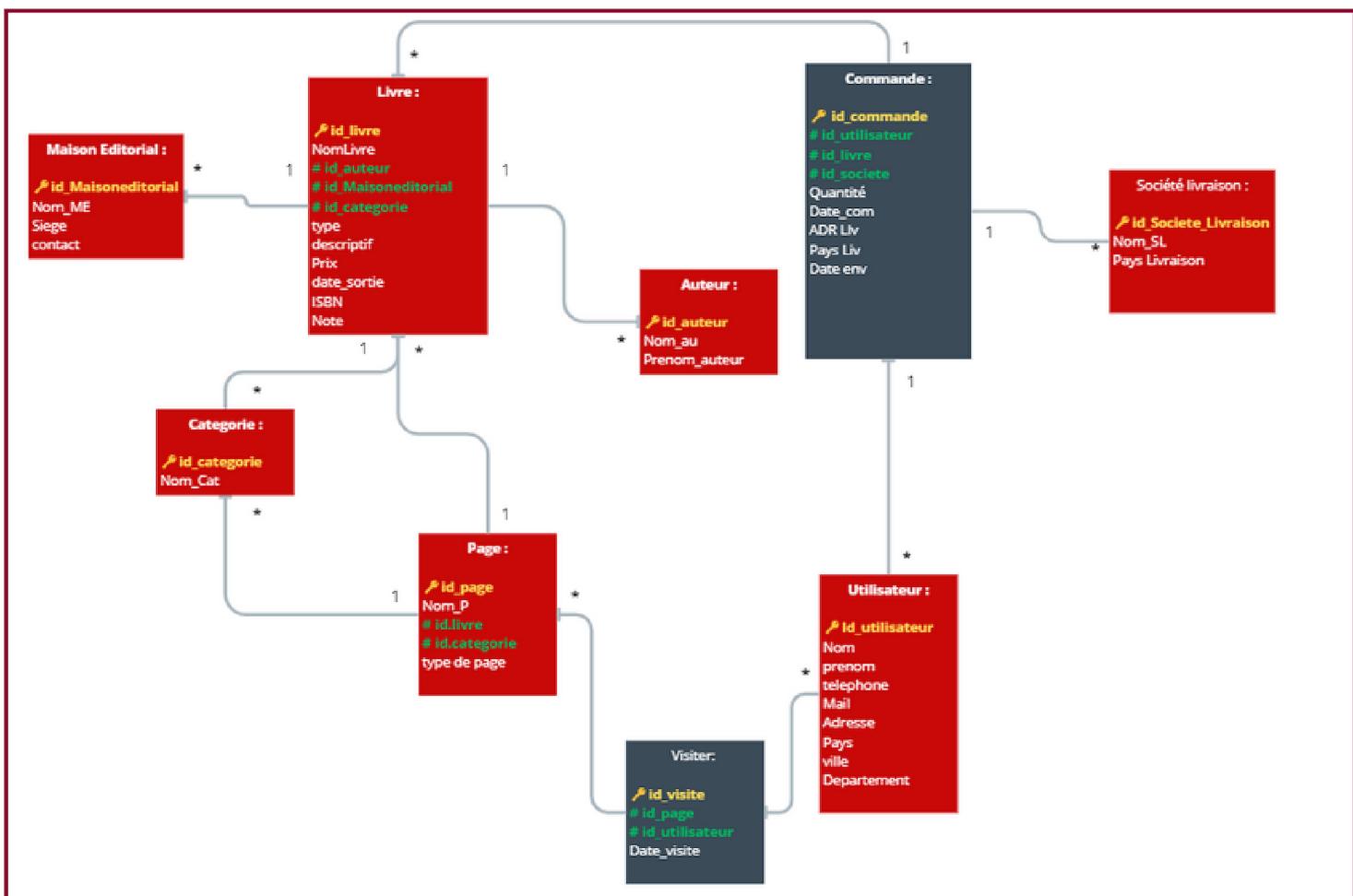


Table de faits

Table de dimension

# EXPLICATION DES CHOIX

Dans la création de notre entrepôt de données (datawarehouse), nous avons créé plusieurs tables, et ces choix ont été motivés par certaines raisons spécifiques :

## Pour la table "Page" :

Nous avons identifié deux types de pages, à savoir les pages "accueil" et les pages "détails". Afin d'éviter d'avoir deux tables distinctes, nous les avons regroupées dans une table unique "Page" avec un attribut "type de page" permettant de les différencier.

## Pour la table "Livre" :

Nous avons cherché à créer une table aussi précise que possible. Dans notre table, le prix du livre a été inclus dans un but précis, celui de ne pas introduire une table supplémentaire. Par exemple, nous avons évité la création d'une table "détail de commande" dans le but, une fois de plus, d'optimiser au maximum notre entrepôt de données.

(NB : Moins de tables signifient des traitements plus rapides, que ce soit pour l'extraction ou tous les traitements de type ETL.)

## Pour la table "Commande" :

En suivant le même principe que pour la table "Livre", nous avons cherché à être aussi précis que possible. Nous disposons d'une colonne "quantité" qui indique la quantité commandée. Il est important de noter qu'une commande représente l'achat d'un livre spécifique par un utilisateur. Par exemple, si un client A achète plusieurs livres, il apparaîtra plusieurs fois dans notre table. Cette modélisation facilite le comptage des commandes par livre ou par maison d'édition.

Nous avons également inclus la date de commande et la date de livraison. La date de commande peut être utile lorsqu'elle est liée à la date de visite, permettant d'analyser la relation entre "visite et commande". Elle est également utilisée dans le contexte du délai de livraison, représenté par la différence entre la date de livraison et la date de commande.

# EXPLICATION DES CHOIX

## Pour la table "Visite" :

Dans cette table, nous avons ajouté un attribut "date de visite" afin de pouvoir extraire des informations importantes. Par exemple, cela nous permet d'analyser les variations du nombre de visites en fonction de périodes ou de dates spécifiques, ce qui est utile pour élaborer des stratégies marketing ou autres. Un exemple concret serait d'étudier l'impact sur les visites pendant une période de soldes.

## Pour la table "Utilisateur":

Nous avons choisi d'inclure le plus grand nombre d'attributs possible dans le but de limiter les "doubles comptes", c'est-à-dire les clients qui utilisent plusieurs comptes pour effectuer des achats. Nous avons intégré plusieurs caractéristiques permettant d'obtenir des informations sur l'utilisateur. Ces informations sont généralement récupérées lors de l'inscription sur le site. Ces données sont considérées comme des DCP (Données de Compte Personnel : dans certains contextes administratifs, "DCP" peut se référer à des données liées à un compte personnel, telles que des informations financières, des relevés bancaires ou d'autres données personnelles).

Il est à noter que dans la table, il existe trois types d'utilisateurs possibles :

- 1.Les utilisateurs qui n'ont jamais effectué de commande ni de visite : ils sont présents dans la base de données car ils sont inscrits mais n'ont entrepris aucune action.
- 2.Les utilisateurs qui ont effectué une ou plusieurs visites mais qui n'ont pas commandé. On les retrouve également dans la table "Visite" avec leur propre ID\_utilisateur.
- 3.Les utilisateurs qui ont visité une ou plusieurs pages et qui ont passé une ou plusieurs commandes : ils apparaissent donc dans les trois tables - "Commander", "Utilisateur" et "Visiter".

## Pour la table Maison éditorial :

Nous avons choisi de mettre tous les éléments dans une table afin d'éviter de surcharger la Base. Nous aurions pu créer plusieurs tables pour "Siege Social" et Maison Editorial mais cela aurait impacté la taille de la Base.

# DESCRIPTION DES TABLES

Dans votre base de données, vous avez neuf tables au total. De ces neuf tables, deux sont des tables de faits (contenant des mesures ou des événements) et sept sont des tables de dimensions (décrivant des entités avec des attributs détaillés). Les tables de faits contiennent généralement des informations numériques et les tables de dimensions fournissent un contexte pour les données.

## Pour les tables de faits :

### Visiter :

Cette table permet de suivre les visites des utilisateurs sur différentes pages du site web, en enregistrant la date de la visite, la page visitée et l'utilisateur associé à chaque visite.

### Nous avons comme attribut :

- **id\_visite (BIGINT)** : Identifiant unique pour chaque visite.
- **id\_page (BIGINT)** : Clé étrangère faisant référence à la table "Page", représentant la page visitée.
- **id\_utilisateur (BIGINT)** : Clé étrangère faisant référence à la table "Utilisateur", identifiant l'utilisateur qui a effectué la visite.
- **Date\_visite (TIMESTAMP)** : Enregistre la date de la visite.

### Commande :

La table "Commander" enregistre les détails des commandes d'utilisateurs, incluant le livre commandé, la quantité, la date et l'adresse de livraison. Les clés étrangères établissent des liens avec d'autres tables pour des relations entre les données.

### Nous avons comme attribut :

- **id\_commande (BIGINT)** : Identifiant unique pour chaque commande.
- **id\_utilisateur (BIGINT)** : Clé étrangère faisant référence à la table "Utilisateur", identifiant l'utilisateur qui a passé la commande.
- **id\_livre (BIGINT)** : Clé étrangère faisant référence à la table "Livre", représentant le livre commandé.
- **id\_societe (BIGINT)** : Clé étrangère faisant référence à la table "Societe", identifiant la société associée à la commande.
- **Quantite (INTEGER)** : Nombre d'exemplaires du livre commandés.
- **Date\_com (TIMESTAMP)** : Date et heure de la commande.
- **ADR\_LIVR (VARCHAR(100))** : Adresse de livraison pour la commande.
- **Pays\_Liv (VARCHAR(50))** : Pays de livraison pour la commande.

# DESCRIPTION DES TABLES

**Pour les tables de dimension :**

**Maison Editorial :**

La table "MaisonEditorial" enregistre les détails des maisons éditoriales, comprenant leur nom, leur siège social, leur adresse et un contact. L'identifiant unique "id\_Maisoneditorial" est utilisé comme clé primaire pour chaque enregistrement.

Nos maisons d'éditoriaux sont "Livraria", "Bookhouse", "Papyrus Press", "Novelty Editions", "Storybook Publishers"

**Nous avons ces attributs :**

- **id\_Maisoneditorial BIGINT** : Clé primaire, chaque Maison Editorial possède un id différent ce qui nous permet de la reconnaître.
- **Nom\_ME VARCHAR(50)** : Nom de la Maison Editorial
- **Siege\_social VARCHAR(100)** : Désigne le Pays où se situe le siege social de la Maison
- **adresse VARCHAR(100)** : Adresse du Siege Social
- 

**Catégorie :**

La table "Categorie" stocke des catégories, chaque catégorie étant représentée par un identifiant unique (id\_categorie) et un nom (Nom\_Cat). L'identifiant "id\_categorie" agit comme clé primaire pour garantir l'unicité de chaque catégorie enregistrée.

"Fiction", "Non-fiction", "Thriller", "Science-fiction", "Romance"

**Société de livraison :**

La table "SocieteLivraison" conserve des informations sur les sociétés de livraison. Chaque société possède un identifiant unique (id\_Societe\_Livraison) ainsi qu'un nom (Nom\_SL).

"ExpressShip", "SwiftDeliver", "RapidTransit", "SpeedyCourier", "FastTrackLogistics"

**Auteur :**

La table auteur concerne les informations des différents auteurs. Chaque auteur a un nom et un id\_auteur.\*

# DESCRIPTION DES TABLES

## Livre :

La table "Livre" stocke des détails clés sur chaque livre, y compris son titre, son auteur, sa maison d'édition, sa catégorie, sa description, son genre, son prix, sa date de sortie, son numéro ISBN et une note. Ces informations sont essentielles pour une gestion complète des ouvrages.

## Page :

La table "Page" enregistre des données liées aux pages des livres. Elle inclut un identifiant unique pour chaque page, son nom, ainsi que des liens avec la table "Livre" via l'identifiant du livre et avec la table "Categorie" via l'identifiant de catégorie. Ces informations permettent de structurer et d'organiser les pages des différents livres selon leurs catégories et leurs types spécifiques.

## Utilisateur :

La table "Utilisateur" stocke les informations personnelles des utilisateurs, telles que leur nom, prénom, coordonnées (téléphone, e-mail, adresse), et des détails géographiques (ville, pays, département). Cette table est utilisée pour gérer les profils des utilisateurs, faciliter leur identification.

- Id\_utilisateur : Identifiant unique de l'utilisateur (BIGINT).
- Nom : Nom de l'utilisateur (VARCHAR(50)).
- Prenom : Prénom de l'utilisateur (VARCHAR(100)).
- telephone : Numéro de téléphone de l'utilisateur (VARCHAR(50)).
- Mail : Adresse e-mail de l'utilisateur (VARCHAR(50)).
- Adresse : Adresse postale de l'utilisateur (VARCHAR(50)).
- Ville : Ville de résidence de l'utilisateur (VARCHAR(50)).
- Pays : Pays de résidence de l'utilisateur (VARCHAR(50)).
- Departement : Numéro du département de résidence de l'utilisateur (INT).
- Contrainte pk\_hid : Clé primaire pour garantir l'unicité de l'Id\_utilisateur.

Pour la création de toutes les tables nous avons d'abord crée les tables via les fonctions de la librairie BDI et SQLite.

```
#### creation de la table Utilisateur ####

dbExecute(bd, "
CREATE TABLE Utilisateur (
    Id_utilisateur BIGINT,
    Nom VARCHAR(50),
    Prenom VARCHAR(100),
    telephone VARCHAR(50),
    Mail VARCHAR(50),
    Adresse VARCHAR(50),
    Ville VARCHAR(50),
    Pays VARCHAR(50),
    Departement INT,
    CONSTRAINT pk_hid PRIMARY KEY (Id_utilisateur)
);")
```

Afin de crée une table nous utilisons la fonction dbExcute() qui nous permet d'utiliser des requêtes SQL sur R.

Nous crées les colonnes de la table en spécifiant le type de valeur.

La fonction Constraint nous permet de mettre les clé primaires et étrangères

# CRÉATION DU CONTENUE

## Première Méthode :

Dans le langage de programmation R, nous avons créé des vecteurs distincts pour chaque colonne, permettant ainsi l'insertion des valeurs.

Dans cet exemple, nous avons constitué une data frame appelée "societes\_livraison", comprenant un vecteur dédié à chaque colonne de la table associée. Une fois la data frame créée, nous avons pu l'implémenter dans la table "SocieteLivraison" à l'aide de la fonction DbWriteTable.

```
societes_livraison <- data.frame(
  id_Societe_Livraison = 1:5,
  Nom_SL = c("ExpressShip", "SwiftDeliver", "RapidTransit", "SpeedyCourier", "FastTrackLogistics"),
  Pays_de_livraison = c("France", "USA", "UK", "Japan", "Germany")
)
dbwriteTable(bd, "SocieteLivraison", societes_livraison, append = TRUE)

dbGetQuery(bd,
  "SELECT *
   FROM SocieteLivraison;
  ")
```

## Deuxième Méthode :

Nous avons opté pour la création d'une base de données sur Excel. Postérieurement à cette création, nous avons procédé à son importation dans R, facilitant ainsi son insertion dans une table.

Prenons l'exemple de la création de la base de données "Livre" :

A	B	C	D	E	F	G	H	I	J	K
1	2	3	4	5	6	7	8	9	10	11
NomLivre	id_livre	id_auteur	id_Maisoneditorial	id_categorie	Note	type	descriptif	Prix	date_sortie	ISBN
Ça	1	1	1	1	4,2	Science-fiction	Un groupe d'amis affronte un mal ancien et terrifiant dans leur ville natale	21,50	15/09/1986	383940414
Shining	2	2	2	2	3,75	Science-fiction	Un homme avec des pouvoirs psychiques lutte contre ses démons intérieurs dans un hôtel isolé	12,99	28/01/1977	123456785
Silencier	3	3	3	3	4,26	Science-fiction	Une famille découvre un cimetière mystérieux avec des pouvoirs sinistres de résurrection	14,99	14/11/1983	987654321
Carrie	4	1	4	1	2	Roman	Une adolescente tourmentée découvre ses pouvoirs psychiques alors qu'elle est victime de harcèlement	11,99	05/04/1974	456789123
Le Fléau	5	4	5	4	3,12	Roman	Une épopee post-apocalyptique dépeignant la lutte entre le bien et le mal après une pandémie dévastatrice	13,50	15/09/1978	111222332
Ça ne peut pas être vrai	6	5	1	5	4,8	Roman	Une épopee post-apocalyptique dépeignant la lutte entre le bien et le mal après une pandémie dévastatrice	20,00	01/08/1978	444556666
Les Animaux fantastiques	7	6	2	1	5	Fantasy	Une exploration de l'inconscient et de l'inexplicable dans des événements du quotidien	9,99	02/06/2001	777888999
La Coupe de feu	8	3	3	2	4	Fantasy	Les aventures d'un magizoologiste à travers un monde de créatures magiques	15,00	08/07/2000	101112133
L'Ordre du Phénix	9	7	4	3	3,5	Fantasy	Un tournoi dangereux met en péril la vie d'un jeune sorcier et ses amis	18,99	21/06/2003	141516177
Les Reliques de la Mort	10	8	5	4	2,75	Fantasy	Une résistance se forme contre un sorcier maléfique, tandis que des secrets sont révélés	11,49	21/07/2007	181920212

Une fois la base de données créée dans Excel, nous avons utilisé la bibliothèque (library) readxl pour l'importer dans R, prête à être insérée dans la table correspondante.



```
library(readxl)
livres = read_excel("C:/users/thomas.belaïd/Desktop/BD.xlsx", sheet="Livre")
dbwriteTable(bd, "Livre", livres, append = TRUE)
```

# GRANULARITÉ , GESTION DE CHANGEMENT ET DIMENSION

## Gestion des changements :

Nous avons opté pour l'utilisation de la méthode SCD (Slowly Changing Dimension) afin de suivre les évolutions au sein des dimensions telles que Livres, Utilisateurs, Maisons d'Édition, etc.

Dans cette approche, la table Commandes pourrait intégrer des dimensions dégénérées pour traiter des attributs spécifiques à une commande.

**Pour ce qui est des tables de faits, nous avons défini les granularités suivantes :**

### 1. Commander

- Granularité : Niveau de la commande.

### 1. Visiter

- Granularité : Niveau de la visite de la page détaillée.

Concernant la dimension douteuse, on la caractérise lorsque des doublons peuvent survenir dans notre contexte. Un exemple concret serait la possibilité pour un livre d'avoir plusieurs Maisons d'Édition, entraînant ainsi plusieurs occurrences dans la base de données. De même, si une personne utilise plusieurs comptes pour ses visites sur le site, cela génère des doublons.

# MESURE ET REQUÊTE

Dans notre contexte nous pouvons avoir de nombreuse mesure, ces mesures peuvent de plusieurs additivités :

## Non additif :

- Moyenne des délais de livraisons ( Délai entre la date de commande et la date de livraison)

```
# Quelles est le délai moyen entre la commande et la livraison

dbGetQuery(bd,
           "SELECT avg(Date_Liv - Date_com )
            FROM Commander;
          ")
```

Dans notre base de donnée, nous avons une moyenne de 2.4 jours.

Cette mesure est non additif car un taux ne peut pas être additionner ni en temps, ni en lien et ni en produit.

Exemple :

Si aujourd'hui nous avons une moyenne de 2.4 jours et demain 2.6 jours la moyenne ne peut pas être additionner.

- Taux d'utilisateurs qui ont visiter en moins une page : C'est le nombre d'utilisateur qui ont visiter une page sur le nombre d'utilisateur totale.

```
# taux d'utilisateur qui ont visiter en moins une page

dbGetQuery(bd,
           "SELECT DISTINCT count(id_utilisateur)
            FROM utilisateur
            WHERE id_utilisateur NOT IN (SELECT DISTINCT id_utilisateur FROM visiter);
          ")
ratio = (4/18)*100
ratio
```

Dans notre de donnée le taux est de 22% soit (4/18).

Cette mesure est non additif car un taux ne peut pas être additionner ni en temps, ni en lien et ni en produit.

## Additif :

- Chiffre d'affaire par Maison Editorial : c'est le chiffre d'affaire c'est à dire le nombre de livre vendu multiplier par le prix ce qui nous donne une valeur totale.

# MESURE ET REQUÊTE

```
# Quels est la maison qui a le plus gros chiffres  
dbGetQuery(bd, "  
    SELECT (a.Prix * b.Quantite),c.id_maisoneditorial,c.Nom_ME  
    FROM Livre as a  
    LEFT JOIN Commander as b ON a.id_livre = b.id_livre  
    LEFT JOIN MaisonEditorial as c ON a.id_maisoneditorial = c.id_maisoneditorial  
    GROUP BY c.id_maisoneditorial  
    ORDER BY 1;  
)
```

Pour notre base de donnée la maison d'Edition avec le plus haut chiffre d'affaire est "Papyrus Press". Cette mesure est additif car elle peut être additionnée en produit, en temps et en lieu.

Exemple :

Nous pouvons additionner le chiffre d'affaire de plusieurs jours ou mois pour avoir une chiffre d'affaire annuel.

- Nombre de livre commandé : Comptage du nombre de commande.

```
# Nombre de livre commandé/Nombre de vente  
dbGetQuery(bd,  
    "SELECT sum(quantite)  
    FROM Commander;  
)
```

Pour notre base de donnée 50 livres ont été commandés. Cette mesure est additif car elle peut être additionnée en temps, en produit et en lieu. Effectivement nous pouvons additionner par livre, par pays et par jour.

Semi Additif :

- Nombre de client : Comptage du nombre de personnes qui ont commandé c'est à dire ce qui ont visité une page et qui ont procédé à une commande. Cette mesure est semi additif car en produit et en temps nous pouvons pas les additionner mais en lieu il est possible de les additionner .

Pour le temps : Si nous avons à un jour A 5 clients et à un jour B 6 clients, Parmis les 6 clients il peut y avoir des clients du jour A alors nous ne pouvons pas additionner le nombre de client.

Même principe pour produit.

# MESURE ET REQUÊTE

```
# Nombre de client ( qui ont commander )  
  
dbGetQuery(bd,  
           "SELECT count(distinct(id_utilisateur))  
FROM Commander  
WHERE id_utilisateur IN (SELECT DISTINCT id_utilisateur FROM Utilisateur);")
```

Dans notre base de donnée nous avons 16 clients c'est à dire 16 personnes qui ont passé une commande.

- Nombre de page différente visité : Comptage du nombre de page qui ont été visiter. Cette mesure n'est pas additif en temps mais additif en produit et lieu.

Exemple :

Si a un jour nous avons un nombre de page et à un autre jour un autre nombre nous ne pouvons pas les additionner. alors que si dans un pays A et un pays nous avons 5 pages visité et 6 pages visités nous pouvons les additionner.

```
# Nombre de page différente visité  
dbGetQuery(bd,  
           "SELECT count(distinct(id_page))  
FROM Page;  
")
```