Chapitre 2

Structure d'une BD relationnelle, interrogations

Chapitre 2 : Plan

Chapitre 2 : Plan

Relations : Clés primaires et clés étrangères

Relations

Relation ----> table, ensemble de tuples

Attribut ----> colonne de table

Tuple ----> ligne

Clé primaire ----> identifiant d'un tuple de la table

Clé étrangère ----> référence : identifiant d'un tuple d'une autre table

Clé primaire

Une **clé primaire** est un attribut (ou un ensemble d'attributs...) dont la valeur permet de retrouver un tuple de façon sûre.
(la fonction associée est totale et **injective** : par exemple, le « nom de login » d'un étudiant permet de retrouver les informations associées à cet étudiant : nom, prénom ...)
Une clé « **secondaire** » est un attribut qui pourrait servir de clé primaire. Mais la notion de clé secondaire n'est pas implantée directement dans les SGBD.
Contrainte d'intégrité d'une clé primaire: la valeur est **renseignée** (NOT NULL) et est **unique (UNIQUE)**.

Clé primaire

- En abrégé : **PK** (Primary Key)
- Pas de valeur manquante (**NOT NULL**)
- Unicité : la fonction **PK(tuple)** est injective, donc il existe une fonction réciproque **tuple(PK)** qui permet de retrouver un tuple
- Elle peut être **simple** (mono-attribut) ou **multi-attribut**

Clé étrangère

Une **clé étrangère** est un attribut **dont la valeur se retrouve dans une clé primaire.**
On dit qu'une clé étrangère **référence** une clé primaire.
Contrainte d'intégrité référentielle d'une clé étrangère :
si la valeur de la clé étrangère est renseignée, alors elle existe dans la table indiquée.

Une clé étrangère modélise une **dépendance fonctionnelle entre deux tables** (association de type * ---> 0..1 entre 2 tables)

Clé étrangère

- En abrégé : F**K** (Foreign Key)
- Peut avoir des valeurs manquantes (NULL)
- Pas d'unicité : la fonction F**K(tuple)** n'est pas en général injective
- La valeur d'une FK doit être une valeur de PK présente dans la table 'étrangère'
  (contrainte d'intégrité référentielle)
  table2 [FK] $\subseteq$ table1 [PK]

Exemples de clés primaires

L'entreprise E a plusieurs établissements. Nous supposons qu'il n'y a pas deux établissements dans la même ville, et que les villes ont toutes des noms différents ; le nom de la ville peut alors servir de **clé primaire** de la table **Etablissements** de la base de données de l'entreprise E.

Le nom et le prénom d'un employé peuvent servir de **clé primaire** multi-attributs de la table **Employes**, à condition que l'on connaisse le nom et le prénom de chaque employé, et que deux employés n'aient pas à la fois le même nom et le même prénom.

Exemple de clé étrangère

Supposons qu'un employé ne soit affecté qu'à un seul établissement : **le nom de la ville** où travaille l'employé sera alors une **clé étrangère** de la table **Employes.**

Pour connaître l'adresse ou le numéro de téléphone de l'établissement dans lequel travaille l'employé, nous irons chercher dans la table **Etablissements** les données associées à cette ville (clé primaire de la table).

Schéma relationnel

Etablissements ([Ville] (1), Telephone...)

Employes ([Nom , Prenom] (1) ,
LEtablissement = @Etablissements[Ville], ...)

Diagramme relationnel, tables entités, tables associations

Diagramme relationnel (non UML)

C'est le dessin associé au texte 'schéma relationnel'

Il met en évidence les deux niveaux de relations du modèle :

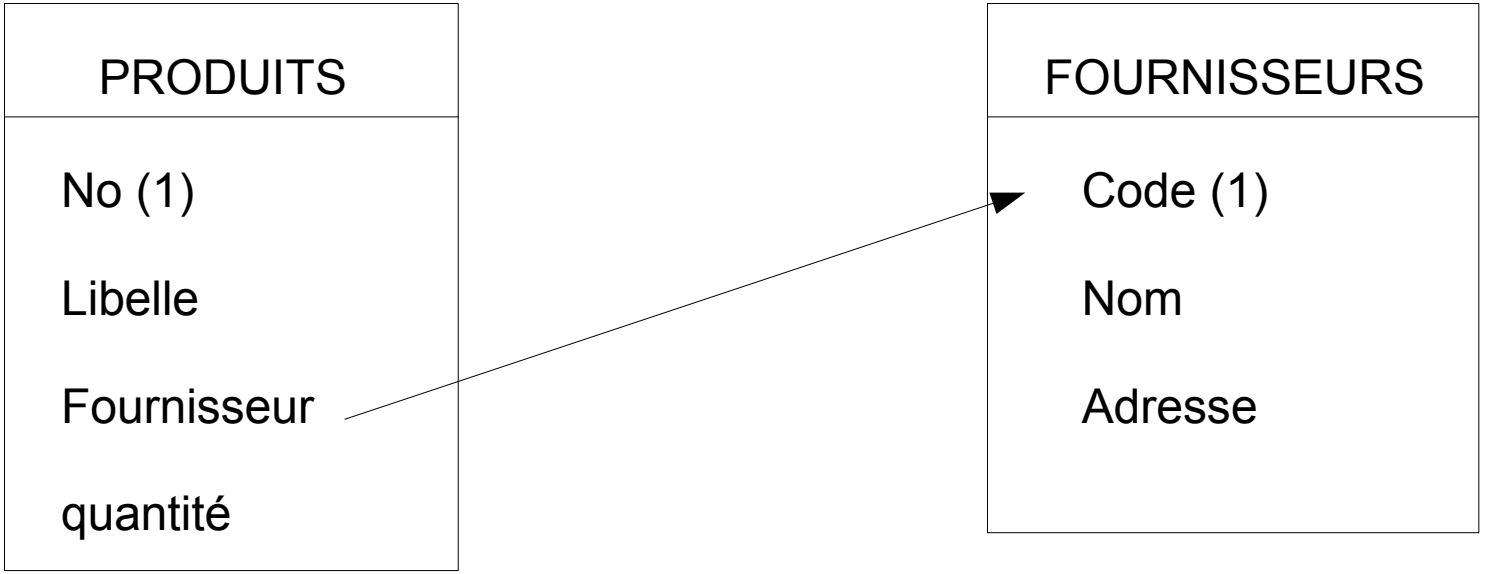
- Relation-table
- association entre tables par FK

Exemple de diagramme relationnel

'schéma relationnel' :

PRODUITS([No](1), libelle, fournisseur =@FOURNISSEURS[Code], quantite)

FOURNISSEURS ([Code](1), nom,adresse)



Construction de la base

Noter que :

•La table1 de la PK doit être construite avant la table2 de la FK associée

table1 (PK) puis table2 (FK)

•Le tuple (PK) doit être enregistré avant que cette valeur de PK puisse être utilisée dans le tuple (FK)

Types de tables

On distingue dans une base de données deux types de tables :

- les tables '**entités**'
- les tables '**associations**'

Tables entités

Une table 'entité' se distingue par le fait que sa clé primaire ne contient pas de références à une autre table.

Exemples :

Dans la base IUTens, la table **ETUDIANTS** et la table **MATIERES** sont des tables 'entités'.

Dans la base LITTORAL, la table **BATEAUX** et la table **PORTS** sont des tables 'entités'.

Dans la base USINE, la table **PRODUITS** et la table **COMMANDES** sont des tables 'entités'.

Table association

Une table 'association' se distingue par le fait que sa clé primaire est :

•multi - attributs
•et formée de clés étrangères

Une table association modélise l'existence d'une **dépendance non fonctionelle** entre deux tables (de type par exemple *----> *)

Exemple :

On veut enregistrer la note d'un **étudiant** dans chaque **matière** ;
ou le nombre de jours d'escale d'un **bateau** dans chaque **port**
ou la quantité de **produit** dans chaque **commande**

Exemple de table association

Exemple 1 (base IUTEns) :

tables 'entité' :

ETUDIANTS([Num](1), nom,prenom,dateNaissance)
MATIERES([Id](1), nom, responsable)

table 'association' :

CARNET(([UnEtudiant=@ETUDIANTS[Num],
UneMatiere=MATIERES[Id])(1), note)

Exemple de Table association

Exemple 2 (base LITTORAL) :

tables 'entité' :

BATEAUX([Imm](1), nom, type, longueur)

PORTS ([Code](1), nom, departement, capacite)

table 'association' :

ESCALES((UnBateau=@BATEAUX[Imm],
UnPort = PORTS[Code])(1), NbJours)

Règles de codage

Le programmeur passant beaucoup moins de temps à écrire du code nouveau qu'à le relire et le modifier, il faut que **le code soit facile à lire**.

Les règles suivantes sont impératives :

- mots du **SGBD en majuscules**
- autres mots en minuscules, sauf les initiales
- **alignement des parenthèses** et **indentation**
- **décomposition** systématique des instructions
- **commentaires** des passages délicats

Consultation du dictionnaire

Interrogation d'intention : structure des tables

•La base de données existe

•On cherche à écrire son schéma relationnel

Connaissant le nom des tables,

on utilise l'instruction :

DESCRIBE <table>

pour trouver les colonnes et leurs types

Intention

•Cette instruction DESCRIBE ne fait pas partie du langage SQL (Structured Query Language) de IBM, et n'exige pas de point-virgule final.

•Les clés primaires et les clés étrangères ne sont pas signalées ; l'obligation de présence (NOT NULL) permet de distinguer une clé primaire dans les cas simples.

Consultation des données

Interrogation d'extension (contenu des tables)

Le SQL comporte quatre instructions de LMD, dont une seule concerne les requêtes :

SELECT

Une requête s'exprime par un '**mapping**' contenant au moins un SELECT.
Un mapping peut être simple (un seul SELECT), ou contenir des sous-mappings.

Le maping simple

se décompose en général comme suit :

```
SELECT   DISTINCT  <colonne>   -- ligne de projection
FROM      <table>
WHERE <condition de restriction> -- ligne de restriction
;
```

Restriction simple

Sans projection donc :

SELECT *
FROM <table>
WHERE <condition de restriction>
;

Projection simple

Sans restriction donc :

SELECT DISTINCT <col1> [,<col2> ...]
FROM <table>
;

Mapping résultat de sous mappings

Déjà vu dans le chapitre précédent. Il s'agit de :

<Mapping> UNION <Mapping> ;

<Mapping> INTERSECT <Mapping> ;

<Mapping> MINUS <Mapping> ;

La jointure

Alg rel :

R1 * R2{C} ou **R1[[R1.A = R2.A]]R2{C}**

Une jointure « naturelle » se traduit toujours comme une theta-jointure :

```
SELECT *
FROM      r1, r2
WHERE     r1.a  =  r2.a   - - condition de jointure
    AND   c               - - condition de restriction
;
```

Exemple de jointure

Pb : donner toute information sur les bateaux de moins de 10 m qui sont sur des places de 10 m ou plus ;

soit en alg. rel. :

places*bateaux{longPlace >=10 et longBateau <10}

```
SELECT   *
FROM      places, bateaux
WHERE    places.NoBateau = bateaux.NoBateau
    AND   longPlace >= 10
    AND   longBateau < 10
;
```

Exemple d'auto jointure

Qui gagne plus que moi ?

```
SELECT   DISTINCT P2.Nom
FROM     Personnes P1 , Personnes P2
WHERE    P1.Nom    =  'moi'
     AND  P2.Nom  ! =  'moi'
     AND  P1.Salaire <  P2.Salaire
;
```

Mapping imbriqué

Un mapping peut en contenir un autre, par exemple :

```
SELECT  *
FROM Chevaux
WHERE Couleur = (  SELECT DISTINCT Couleur
                   FROM Chevaux
                   WHERE Jockey = 'Henri IV'
                 )
;
```

Attention !!

Il faut être sûr que les chevaux d'Henri IV ont tous la même couleur, sinon la requête échoue...

Pour éviter une erreur, on pourrait remplacer le symbole '=' par '**IN**'

mais est-ce bien ce que demande le client ?

Le renommage des colonnes

```
SELECT DISTINCT LibellePro AS NomProduit , Prix
FROM Produits
WHERE Prix > 18
;

-- AS est facultatif
-- on renomme les tables de la même façon
-- alg rel : PRODUITS[LibellePro AS NomProduit , Prix] {Prix > 18}
```

La division

N'existe pas de façon directe

On est obligé de procéder :

- soit **par comptage (COUNT)**
- soit **par différence (MINUS)**

Nous allons traiter la division par différence :

division normale : exemple 1

Voitures

imatriculation	marque	puissance	Dept d'immat
24ET7898	RENAULT	8	22
76YU9087	PEUGEOT	8	56
75GY6435	AUDI	8	35
67HR4321	PEUGEOT	7	35
46FC5687	RENAULT	7	56
55YT9462	PEUGEOT	9	22

Motos

Immat	marque	puiss	Date 1ere imat
34E87	Yamaha	8	17/06/2004
87Y54	Yamaha	7	08/05/2010
98I09	Honda	8	24/07/2009

Quelles sont les marques de voitures représentées par des voitures dont les puissances sont celles de toutes les motos ?

Voitures[marque,puissance]/Motos[puissance]

résultat : Peugeot et Renault

division normale : exemple 1

On cherche donc les marques de voitures telles que si on regarde les puissances associées dans la table voiture on retrouve toutes les puissances des motos.

En prenant l'ensemble des puissances de toutes les motos, et en faisant la différence avec les puissances associée aux marques cherchées dans la table voiture on doit retrouver l'ensemble vide.

C'est ce que nous allons mettre en oeuvre

division normale : exemple 1

L'ensemble des puissances des motos :

SELECT DISTINCT PUISS
FROM MOTOS;

La différence avec les puissances d'une marque de voiture :

MINUS
SELECT DISTINCT PUISSANCE
FROM VOITURES
WHERE Marque = *« une marque donnée »*

division normale : exemple 1

Il faut maintenant « recoller correctement les moceaux » :

```
SELECT DISTINCT Marques
FROM Voitures V1
WHERE NOT EXISTS ( SELECT DISTINCT Puiss
                   FROM Motos
                   MINUS
                   SELECT DISTINCT Puissance
                   FROM Voitures V2
                   WHERE V1.Marque = V2.Marque
                   )
;
```

Division exacte : exemple 1

On utilise les mêmes tables mais cette fois-ci la question posée est :

Quelles sont les marques de voitures représentées par des voitures dont les puissances sont celles de toutes les motos **et uniquement celles là** ?

Voitures[marque,puissance] // Motos[puissance]
résultat : Renault

Division exacte : exemple 1

On procède exactement de la même manière que pour la division normale.

Mais il faut tout simplement rajouter la condition que la différence entre les puissances des marques cherchées et les puissances des motos est vide :

ce qui donnera au final l'égalité des deux ensembles

Division exacte : exemple 1

Il suffit donc de rajouter au mapping précédent :

```
AND NOT EXISTS ( SELECT DISTINCT Puissance
                 FROM Voitures V2
                 WHERE V2.Marque = V1.Marque
                 MINUS
                 SELECT DISTINCT Puiss
                 FROM Motos
                 )
;
```

Division par différence : exemple 2

Prenons un autre exemple à partir des tables :

Table-entité Produits : **PRODUITS**([code](1),libelle)

Table-association Alivrer : **ALIVRER**([UnProduit@,UnClient@](1))

La question posée est :

Quelles sont les clés des clients auxquels on doit livrer tous les produits dont le libellé commence par 'A' ?

Division par différence : exemple 2

On peut interpréter cette question de deux façons en algèbre relationnelle :

Q1 : ALIVRER / (PRODUITS{libelle='A%'}[code])

ou

Q2 : ALIVRER // (PRODUITS{libelle='A%'}[code])

Q1 : division normale :

Quelles sont les clés des clients auxquels on doit livrer tous les produits dont le libellé commence par 'A' et peut-être d'autres ?

Q2 : division exacte :

Quelles sont les clés des clients auxquels on doit livrer tous les produits dont le libellé commence par 'A' et uniquement ceux-là ?

Division par différence : exemple 2

Division normale

On décompose la requête Q1 en deux :

Requête Q1a : Quels sont les codes des produits dont le libellé commence par 'A' ?

Requête Q1b=Q1 : Quelles sont les clés des clients auxquels on doit livrer un ensemble de codes de produits contenant Q1a ?

Division par différence : exemple 2

Division normale

Requête Q1a = ProduitsA% : Quels sont les codes des produits dont le libellé commence par 'A' ?

```
SELECT      Code  - -  clé
FROM        Produits
WHERE       Libelle   LIKE   'A%'
;
```

Division par différence : exemple 2

Division normale

Requête Q1b = Q1 : Quels sont les clients auxquels on doit livrer un ensemble de codes de produits contenant Q1a = ProduitsA% ?

La différence entre les ProduitsA% et les produits des clients cherchés doit etre vide !!

```
SELECT    DISTINCT  UnClient
FROM      ALivrer   L1
WHERE  NOT EXISTS (ProduitsA%
                   MINUS
                   ProduitsClientL1  - les produits
                  )                  - du client venant
;                                    - de la table L1
```

Division par différence : exemple 2

Division normale

```
SELECT   DISTINCT   UnClient
FROM     ALivrer L1
WHERE    NOT   EXISTS  (  SELECT          Code  - - clé
                          FROM            Produits
                          WHERE           Libelle  LIKE   'A%'
                          - -  Les produits A%
                          MINUS
                          - -  Les produits du client
                          SELECT          DISTINCT   UnProduit
                          FROM            ALivrer L2
                          WHERE           L2.client = L1.client
                        )
;
```

Division par différence : exemple 2

Division exacte

Requête Q2 : Quelles sont les clés des clients auxquels on doit livrer tous les produits dont le libellé commence par 'A' et peut-être d'autres ?

Requête Q2a : Quels sont les codes des produits dont le libellé commence par 'A' ?
Requête Q2b=Q2 : Quelles sont les clés des clients auxquels on doit livrer un ensemble de codes de produits exactement égal à Q1a ?

Division par différence : exemple 2

Division exacte

On refait la division normale en rajoutant le fait que la différence ProduitsClientL1 par Produits A% doit être vide : ce qui donne l'égalité de ces 2 ensembles.

```
....
AND   NOT   EXISTS  (      SELECT        DISTINCT  UnProduit
                           FROM          ALivrer L2
                           WHERE         L2.client = L1.client
                           - -  Les produits  du client
                           MINUS
                           - -  Les produits  A%
                           SELECT         Code  - - clé
                           FROM           Produits
                           WHERE          Libelle  LIKE   'A%'
                      )
;
```

Tri des T-uples

```
SELECT DISTINCT  LibellePro
FROM Produits
ORDER BY  LibellePro
;

-- ordre croissant (ASC) par défaut (sinon préciser DESC)
-- notation : PRODUITS[LibellePro](LibellePro>)
```

Tri multi-attribut

```
SELECT DISTINCT   Nom , Salaire
FROM   Employes
ORDER BY  Salaire DESC , Nom
;

-- notation :
EMPLOYES[Nom , Salaire](Salaire<) (Nom>)
```

Comptage interne

La 'pseudo-colonne' **RowNum** attribue à chaque tuple résultat de mapping un numéro de sortie.

Ce numéro change à chaque mapping (à chaque instruction SELECT) et est donné avant le tri (dans le cas où l'utilisateur demande un tri par ORDER BY)

Limitation du nombre de T-uples

```
SELECT LibellePro
FROM Produits
WHERE   ROWNUM  <=  10
;
```

'rownum' est une 'pseudo-colonne' qui compte les tuples d'un mapping

Tri + limitation

Quels sont les dix plus gros salaires ?

```
SELECT DISTINCT Salaire
FROM   Employes
WHERE ROWNUM <=10
ORDER BY  Salaire DESC
;
```

ne fonctionne pas !!!
(car la numérotation se fait avant le tri)

Tri + limitation

Quels sont les dix plus gros salaires ?

```
SELECT   *
FROM  ( SELECT DISTINCT Salaire
          FROM   Employes
          ORDER BY  Salaire DESC
        )
WHERE ROWNUM <=10
;
```

réponse correcte

Données manquantes

Quels sont les employés dont on n'a pas enregistré le prénom ?

```
SELECT   *
FROM     Employes
WHERE   Prenom   IS   NULL
;
```

(Prenom = NULL ne fontionne pas)

notation alg rel : Prenom = ^

Données non manquantes

Quels sont les employés dont on a enregistré le prénom ?

```
SELECT   *
FROM     Employes
WHERE    Prenom   IS   NOT  NULL
;
```

notation alg rel : Prenom != ^

Test avec une valeur retournée

Quels sont les employés qui font le même travail qu'Obélix ?

```
SELECT   DISTINCT    Nom
FROM     Employes
WHERE   Fonction  =    ( SELECT Fonction
           FROM    Employes
           WHERE UPPER(Nom) ='OBELIX'
                          )
;
```

Test avec plusieur valeurs (apartenance à une liste)

Quels sont les employés qui font un des travaux que fait Astérix ?

```
SELECT   DISTINCT   Nom
FROM     Employes
WHERE    Fonction  IN    ( SELECT Fonction
                            FROM     Employes
                            WHERE UPPER(Nom) ='ASTERIX'
                          )
;
```

notation alg rel : $\in$

Test avec plusieurs valeurs (non appartenance à une liste)

Quels sont les no des produits qui n'ont pas été livrés ?

```
SELECT   DISTINCT   Num
FROM     Produits
WHERE   Num  NOT  IN    ( SELECT NumProduit
                                  FROM      Livraisons
                          )
;
```

Test d'existence
(au moins un T-uple retourné)

Quels sont les depts d'où viennent au moins 2 employés ?

```
SELECT   DISTINCT   Dept
FROM     Employes E
WHERE   EXISTS    ( SELECT    *
                      FROM Employes
                      WHERE Dept = E.Dept
                       AND Nom != E.Nom
                    )
;
```

notation alg rel : mapping != ∅

Test de contenu vide
(aucun T-uple retourné)

Quels sont les no des produits qui n'ont pas été livrés ?

```
SELECT   DISTINCT   Num
FROM     Produits
WHERE   NOT EXISTS    ( SELECT   *
                          FROM Livraisons
                          WHERE Produits.Num = Livraisons.Num
                        )
;
```

-- notation alg rel : mapping = ∅

Recherche de motif

Quels sont les noms des employés dont l'adresse contient 'Vannes' ?

```
SELECT   DISTINCT   Nom
FROM     Employes
WHERE   UPPER (Adresse)  LIKE   '%VANNES%'
;
```

- - on peut aussi utiliser « NOT LIKE »

Date du système

Quelle est la date actuelle utilisée par le système du serveur Oracle ?

```
SELECT   SYSDATE
FROM     DUAL
;
```

Utilisation des dates

Format par défaut : '15-fev-2012'

Saisie sous un autre format :

TO_DATE ('15/11/2006','DD/MM/YYYY')

Emploi des dates

Q1 : Quels sont les noms des personnes enregistrées, triées par ordre croissant des âges ?

```
SELECT   DISTINCT   Nom
FROM      Personne
ORDER BY  DateNaissance DESC
;
```

ERREUR à la ligne 3 :
ORA-01791: cette expression n'a pas été SELECTionnée

Emploi des dates

Le tri se faisant **après** la projection, il est impossible de trier suivant une colonne non projetée ; il faut donc projeter au moins deux colonnes, mais le plus simple est de tout prendre.

Nous allons donc **décomposer** :

```
Q1a :=    SELECT   *
          FROM     Personne
          ORDER BY  DateNaissance DESC
```

Emploi des dates

Réponse finale :

```
SELECT   Nom    - - répétition éventuelle
FROM    ( SELECT   *
          FROM     Personne
          ORDER BY  DateNaissance DESC
          )
;
```

Emploi des dates

Quelles sont les personnes nées entre 1987 et 1989 ?

```
SELECT   DISTINCT   Nom
FROM      Personne
WHERE DateNaissance BETWEEN
    TO_DATE  ('1/1/1987','DD/MM/YYYY')
  AND
    TO_DATE  ('31/12/1989','DD/MM/YYYY')
;
```

Emploi des dates

Autre exemple avec toujours la table-entité Etudiants:

ETUDIANTS([nom,prenom](1),dateNaissance, villeResidence)

Requête Q2 : Quels sont les noms et prénoms des dix étudiants les plus jeunes rangés par ordre alphabétique ?

Emploi des dates

On décompose Q2 :

Requête Q2a : Quelle est la liste des étudiants par ordre croissant des âges ?

Requête Q2b : Quels sont les dix premiers étudiants de la liste Q2a ?

Requête Q2c = Q2 : Quels sont les noms et prénoms par ordre alphabétique des étudiants de Q2b ?

Emploi des dates

Requête Q2a : Quelle est la liste des étudiants par ordre croissant des âges ?

```
SELECT      *
FROM        Etudiants
ORDER BY    dateNaissance   DESC
```

Emploi des dates

Requête Q2b : Quels sont les dix premiers étudiants de la liste Q3a ?

```
SELECT   *
FROM     (Q3a)
WHERE    RowNum  <= 10
;
```

Emploi des dates

Requête Q2c = Q2 : Quels sont les noms et prénoms par ordre alphabétique des étudiants de Q2b ?

```
SELECT     Nom , Prenom   - - clé
FROM        (Q2a)
WHERE    RowNum  <= 10
ORDER BY  Nom , Prenom
;
```

Emploi des dates

Mapping final:

```
SELECT    Nom , Prenom    - - clé
FROM        ( SELECT       *
                FROM          Etudiants
                ORDER BY  dateNaissance   DESC
              )
WHERE    RowNum  <= 10
ORDER BY  Nom , Prenom
;
```