MODÉLISER la **BASE de DONNÉES**

...

PLAN

01 Faire le modèle CONCEPTUEL des données

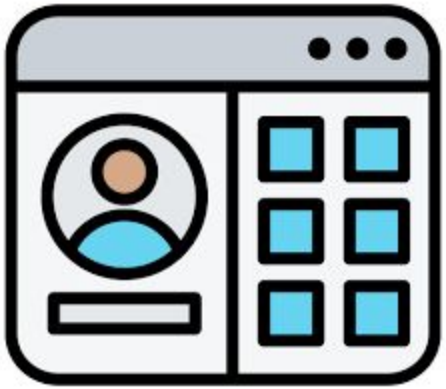
02 Faire le modèle LOGIQUE des données

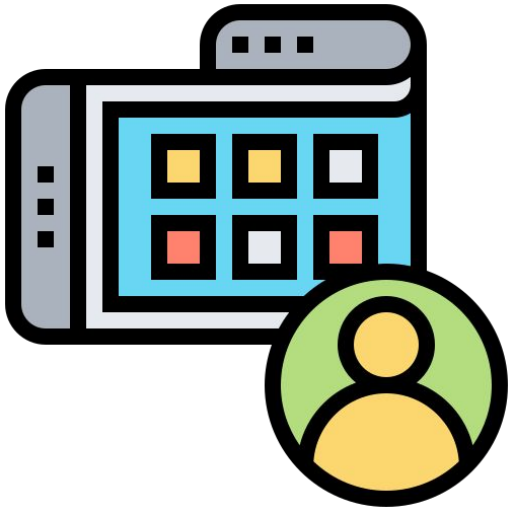
03 Faire le modèle PHYSIQUE des données

Créer les tables

Clarifier les pages de l'app

ANALYSE FONCTIONNELLE





nombre de pages
layout des pages

nature et la complexité des données

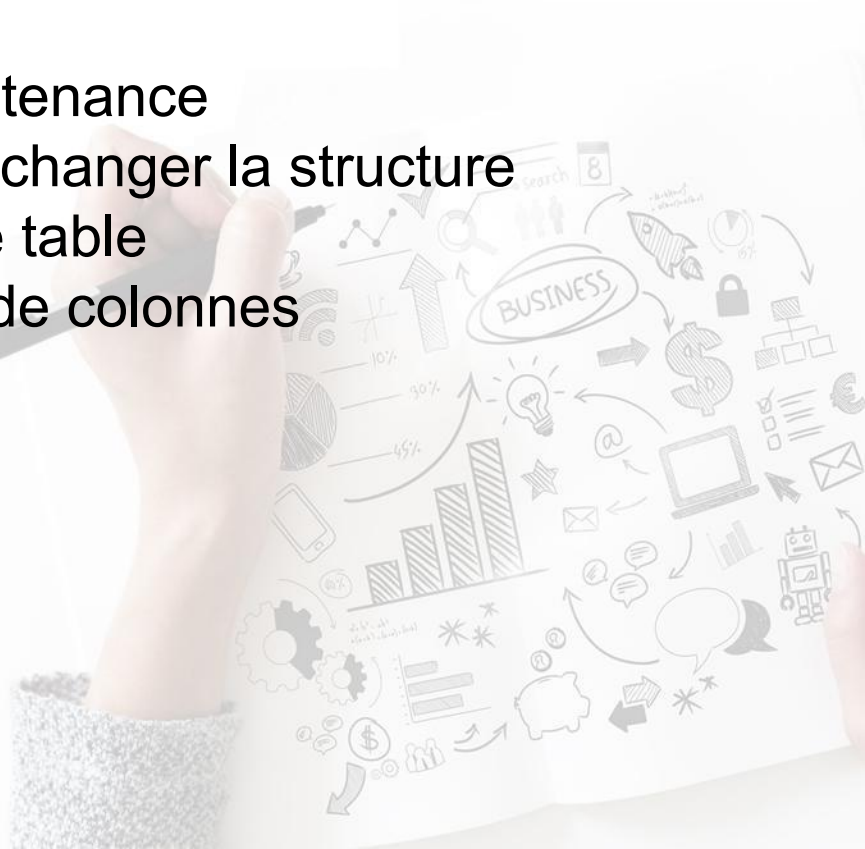
Créer les tables

- *Le concepteur (vous) AVANT que le logiciel ne s'opère*
  - Peut ajouter des colonnes (champs)
  - Peut ajouter des tables
  - Les données qu'il ajoute sont des données tests

- *Le logiciel en opération*
  - Peut faire des écritures de vraies données
    - INSERT, UPDATE, DELETE
  - Peut faire des lectures de vraies données
    - SELECT
  - Ne peut PAS faire de changements à la structure des tables
    - pas sécuritaire
    - donneés inaccessibles
      - les colonnes inconnues ne peuvent être utilisées dans le code des SELECT

Changer le logiciel

- *Maintenance - on répète les deux étapes*
  - Dev
    - nouveau create table
    - surtout des alter table (ajout de champs, rename de champs, etc.)
  - Mise en opération de la maintenance
    - elle ne peut toujours pas changer la structure
      - elle peut pas créer de table
      - elle peut pas ajouter de colonnes



01

Modèle CONCEPTUEL des données

1. Recueillir les besoins
    - Faire parler le client !
2. Identifier les concepts
    - Mindstorming et mindmap
3. Mettre les concepts en relation
    - Identifier les multiplicités (one-to-many ou many-to-many)

Recueillir les besoins du client

On doit faire parler le client, sinon le système ne répondra pas à son besoin.
De plus, des conséquences financières s'ensuivent si le système produit ne répond pas au besoin réel du client, peu importe ce qu'il a dit ou écrit.

- Le client ou chargé de projet n'a pas les connaissances informatiques pour s'exprimer clairement concernant son besoin
  - Il peut écrire un MOT ERRONÉ et utiliser des termes inadéquats : dire blog à la place de chat
  - Il peut OUBLIER des parties du système : par exemple pour lui c'est évident que tous les tableaux sont triables par colonne en cliquant dessus ! il ne mentionne pas le panneau d'administration des données pensant que c'est automatique, etc.
    - Il peut donner des spécifications INCOMPLÈTES en pensant qu'on va les compléter.
  - Il peut CHANGER d'idée ce qui a plus de chance de se produire sur un développement long.
    - Si il obtient plus d'information technique ou sur les coût
    - Si un de ses compétiteur évolue entre-temps.

Comment recueillir les idées

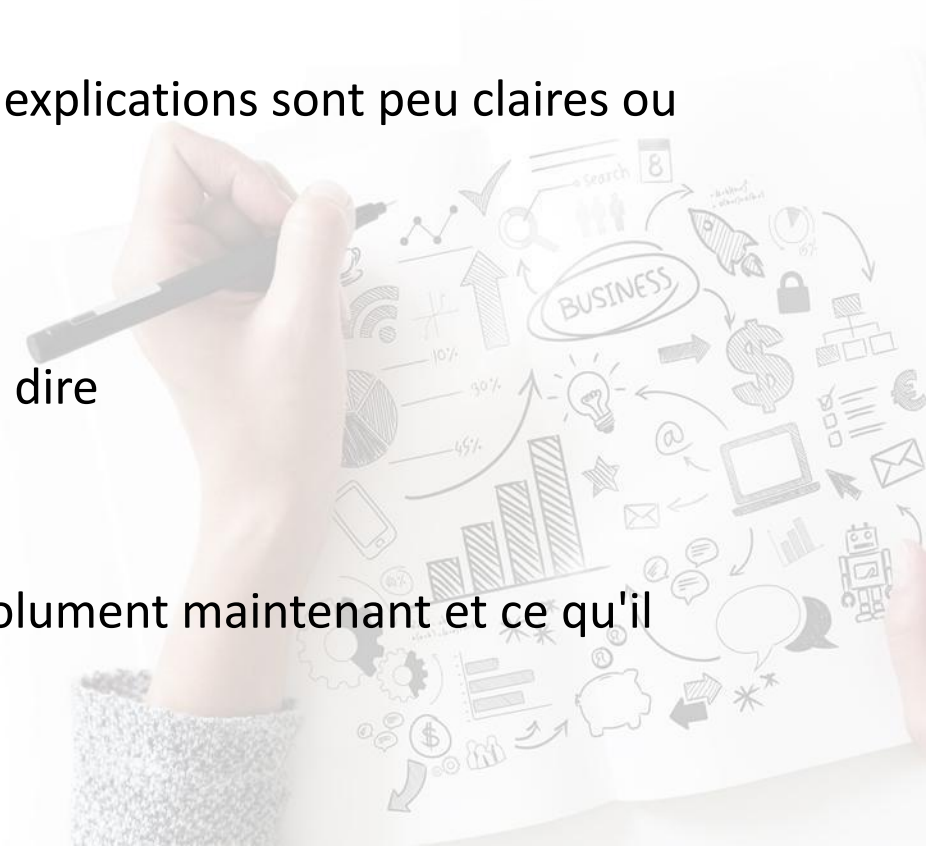
- ★ *maquettes*
- ★ *diagramme de concept avec le client*
- ★ *se poser des questions*
- ★ *aller voir les pages des logiciels*
- ★ *aller voir les pages des competiteurs*
- ★ *etc.*

Comment faire parler le client ?

Non ! on ne va pas torturer notre client ! Mais on va le faire parler avec des techniques de communication lui permettant de constater notre compréhension de son système.

Il va falloir trouver des TECHNIQUES pour :

- Lui faire PRÉCISER les items dont les explications sont peu claires ou qu'on a comprises autrement
  - Vos idées ?
- Lui faire réaliser ce qu'il a OUBLIÉ de dire
  - Vos idées ?
- Lui faire PRIORISER ce qu'il veut absolument maintenant et ce qu'il aimerait peut-être plus tard
  - Vos idées ?



Si on écoute juste ce qu'il dit...

La communication est rarement claire et complète !

Un dialogue avec des validations

On va devoir créer un dialogue avec des validations.

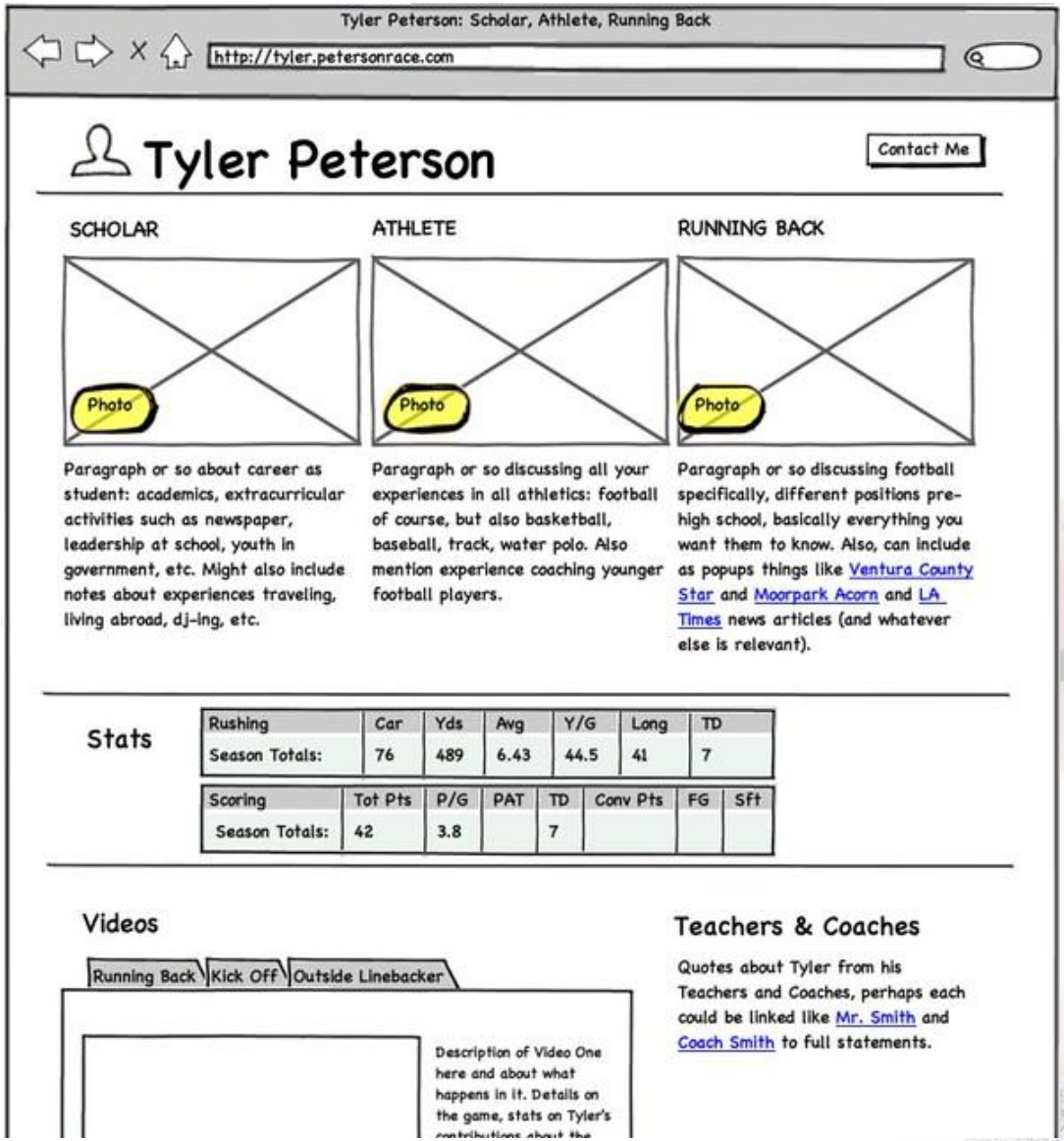
- *La communication va devoir aller dans les deux sens pour que le client VALIDE ce qu'on a compris*

- *On va devoir apprendre à parler le LANGAGE DU CLIENT afin qu'il comprenne ce que nous on veut dire*

Une technique : le brainstorming de concepts

C'est une technique qu'on utilise souvent lorsque l'idée du projet est encore peu claire. On lance des mots sur un tableau, souvent avec des post-it ou encore dans un diagramme de concepts.

Une technique : les maquettes et wireframe

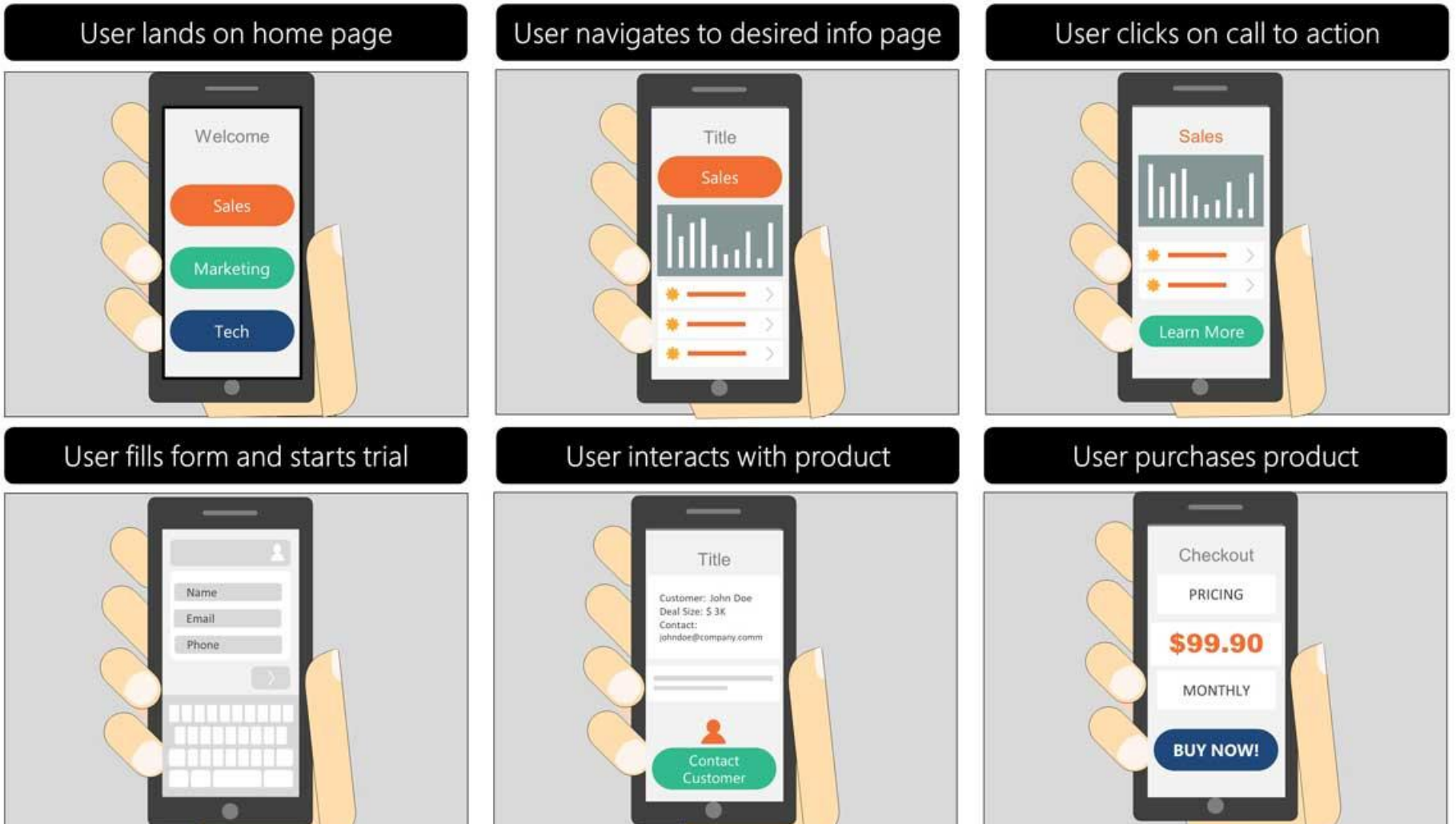


Les maquettes sont un outil de communication idéal avec le client puisqu'il est CAPABLE de réaliser lui-même le dessin ou la validation des écrans.

On peut aussi utiliser les wireframe pour aider à clarifier les liens entre les écrans ce qui indique souvent les liens entre les données.

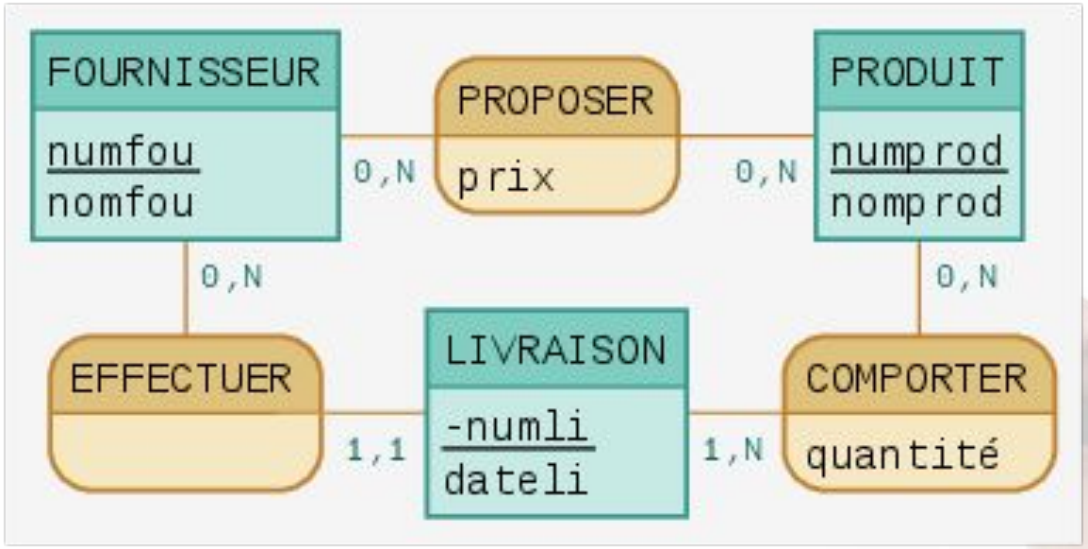
Une technique : le storyboard

Pour lui faire prioriser, faites-lui dessiner ou valider la bande dessinée de ce qu'il pourra faire avec son logiciel après la première itération. Cela vous permettra d'identifier les données nécessaires pour cette première itération.



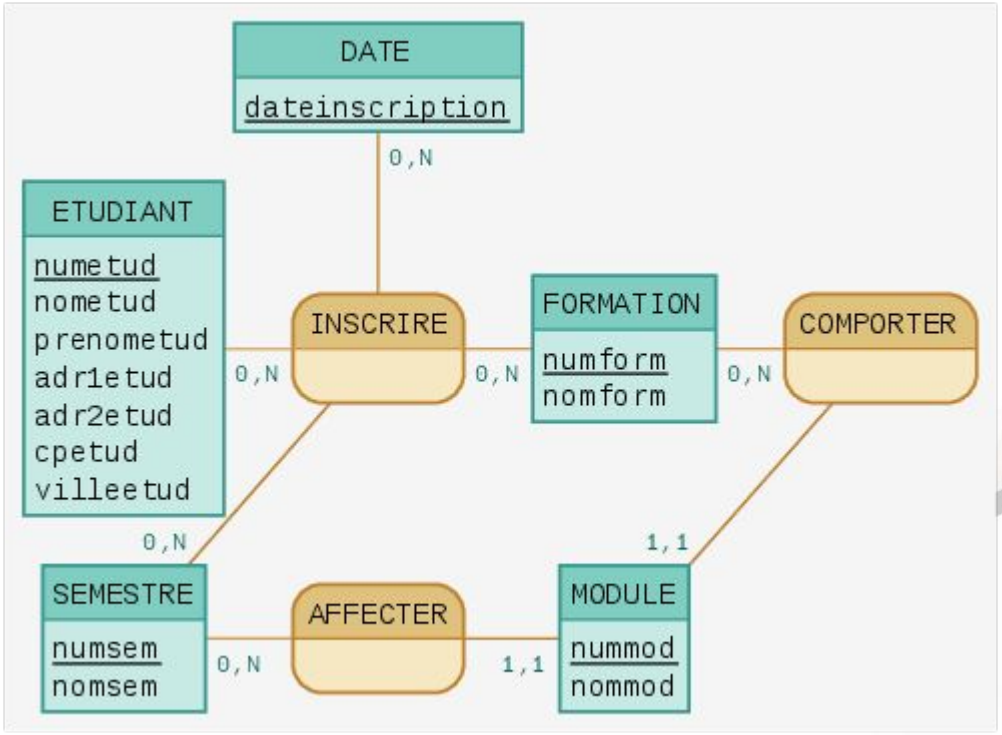
Graphique MCD

Des fournisseurs proposent des produits qui font ensuite partie de livraisons

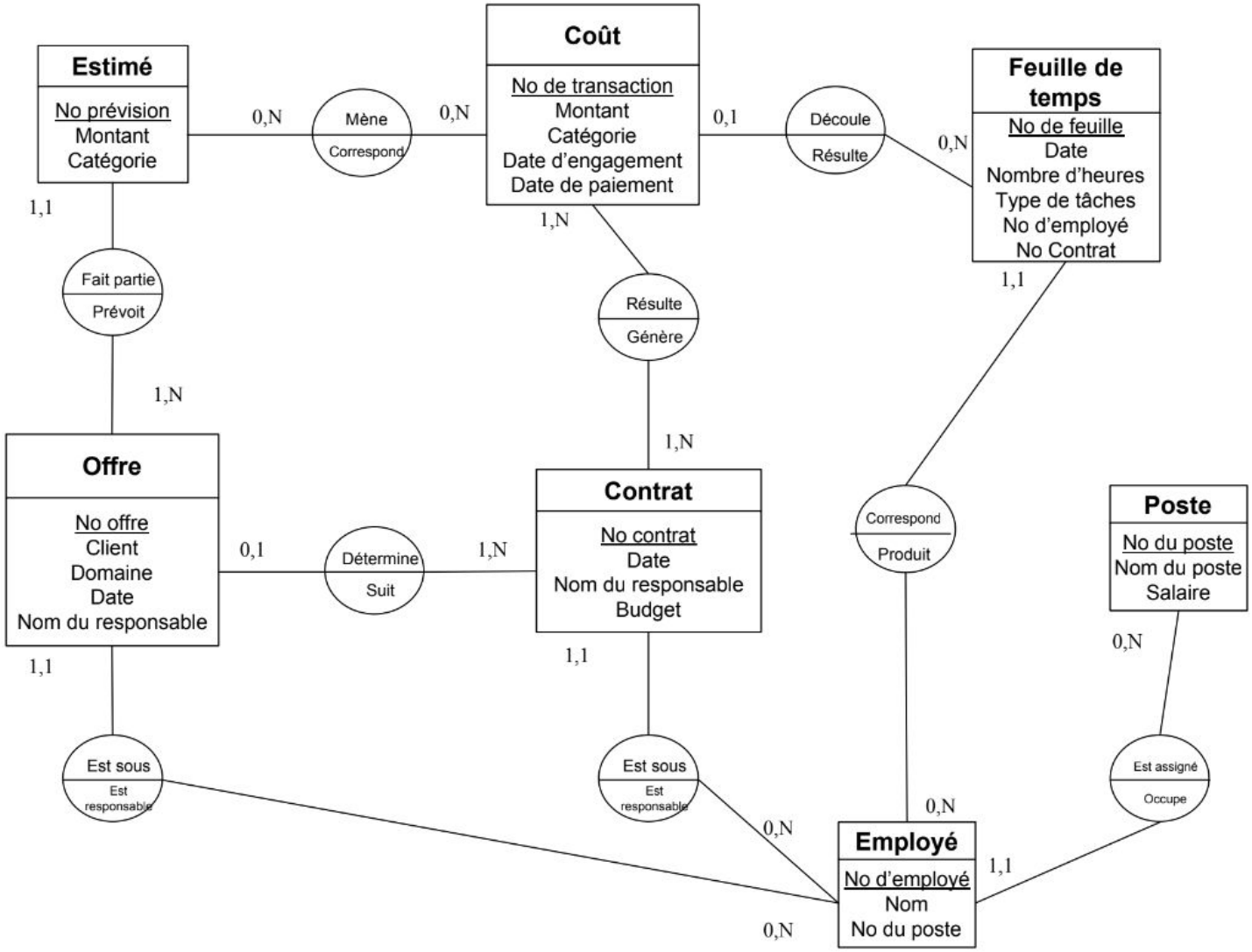


Graphique MCD

Des fournisseurs proposent des produits qui font ensuite partie de livraisons

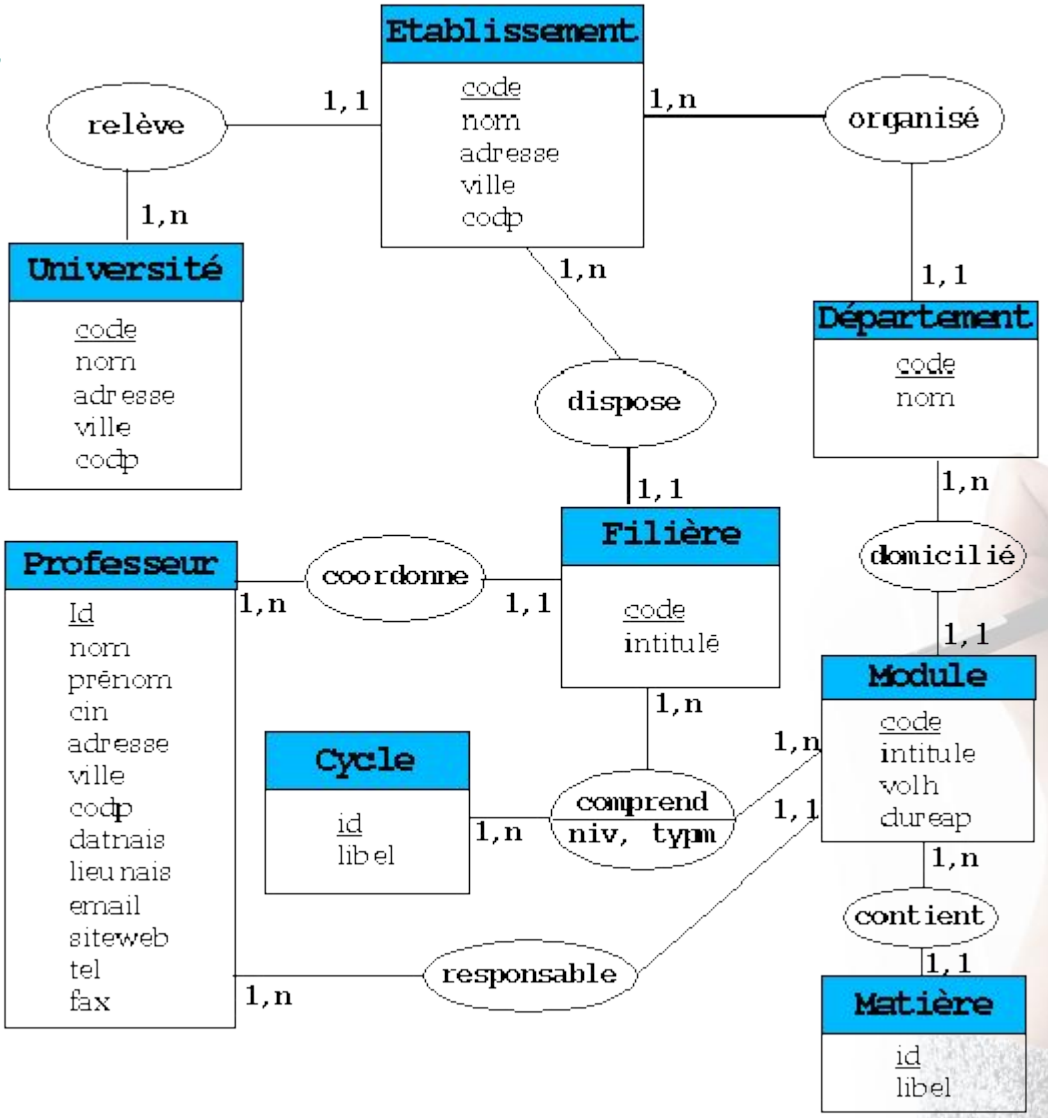


Graphique MCD

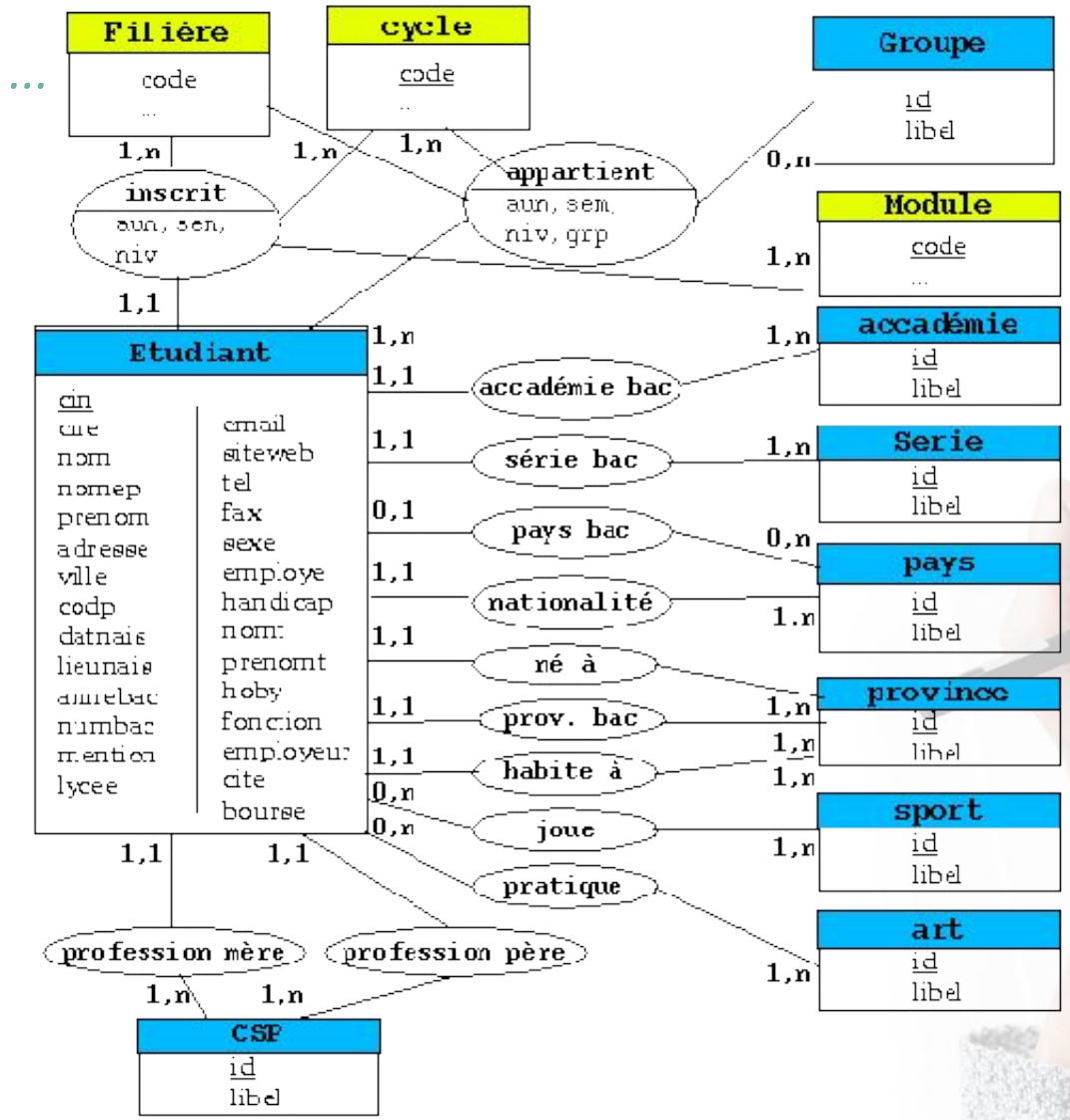


Graphique MCD

...



Graphique MCD

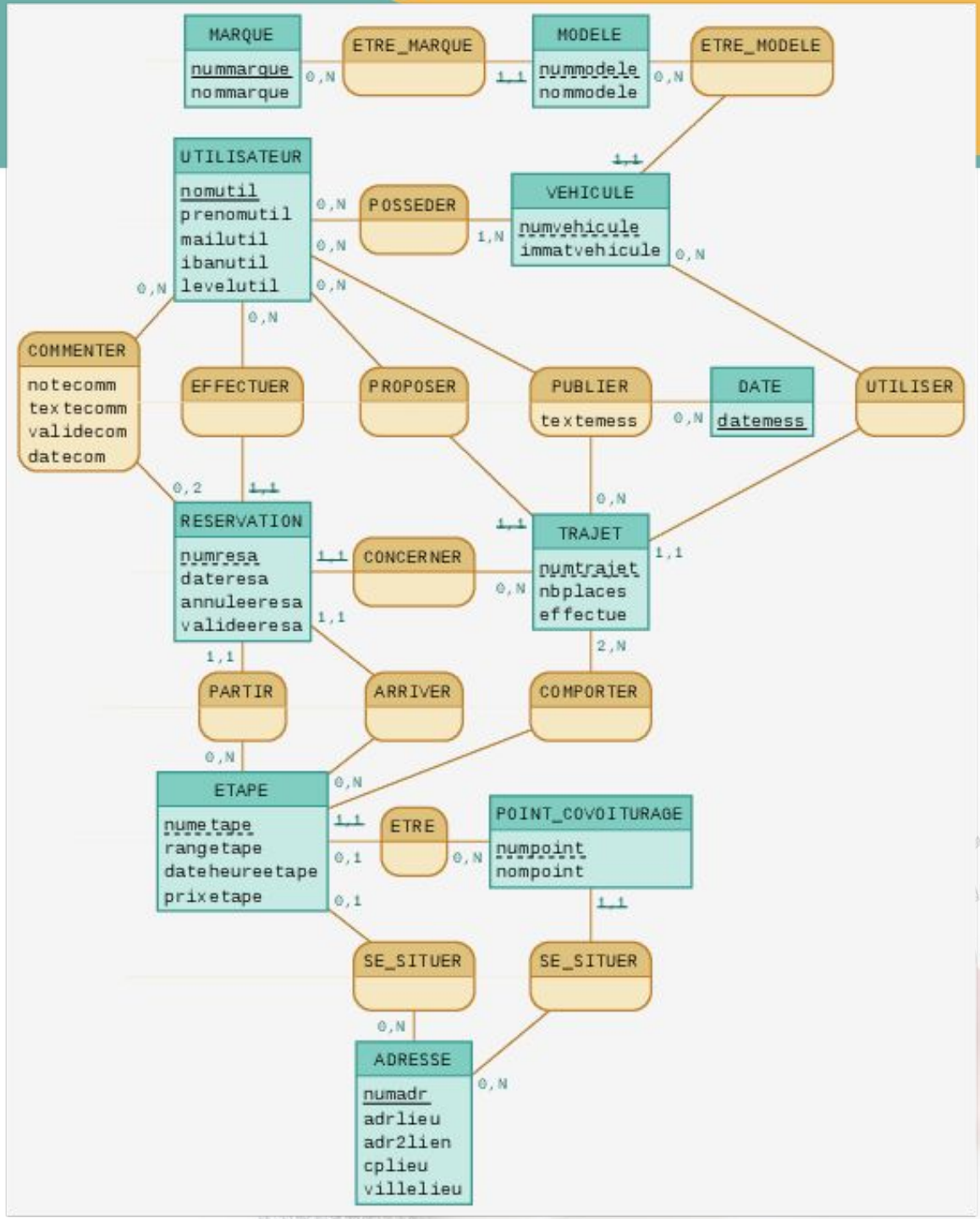


NB. '...' dans les propriétés d'une entité signifie que celle-ci est définie dans un autre MCD

Graphique MCD

Pour un logiciel de covoiturage.

Autre diagrammes ici :
https://enseignement.alexandre-mesle.com/analyse/analyse017.html#sec120



02

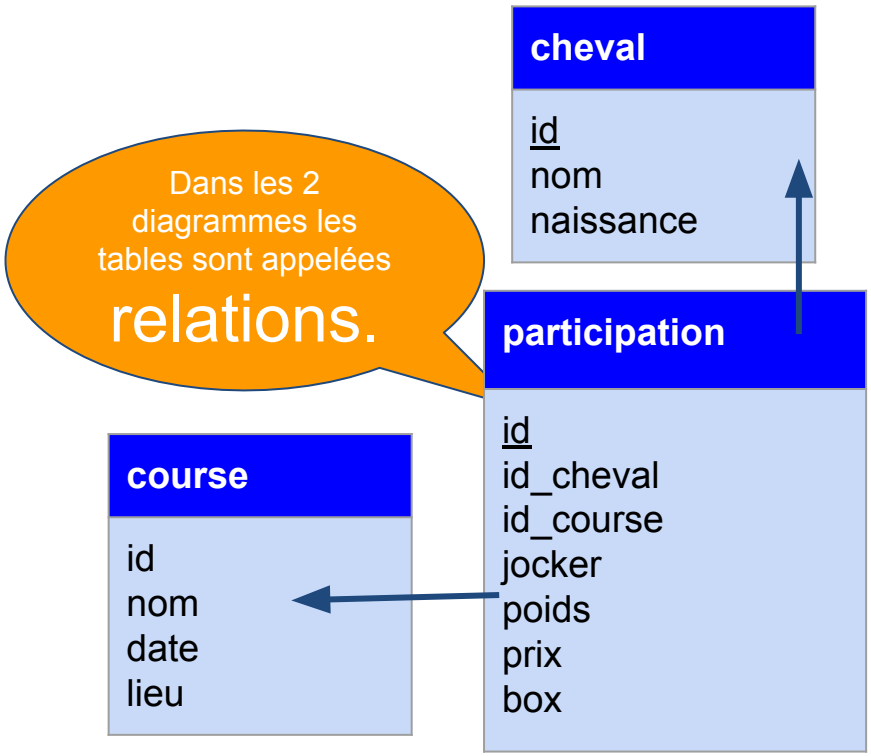
Modèle LOGIQUE des données

1. Créer les tables
    - Identifier les champs
    - Identifier les clés primaires
2. Créer les relations one-to-many
    - Ajouter une clé étrangère
3. Créer les relations many-to-many
    - Créer une table de relations qui a deux clés étrangères

Formats traditionnels d'un MLD

L'étape de représenter le MLD dans un format traditionnel est souvent sautée dans la pratique et la plupart des programmeurs et architectes vont modéliser directement dans un outil de création de tables en sql.Cependant, dans certains contextes formels d'ingénierie (et académique à cette école) nous devons les représenter dans le format officiel traditionnel. Il y en a deux.

MLD version graphique



MLD version texte

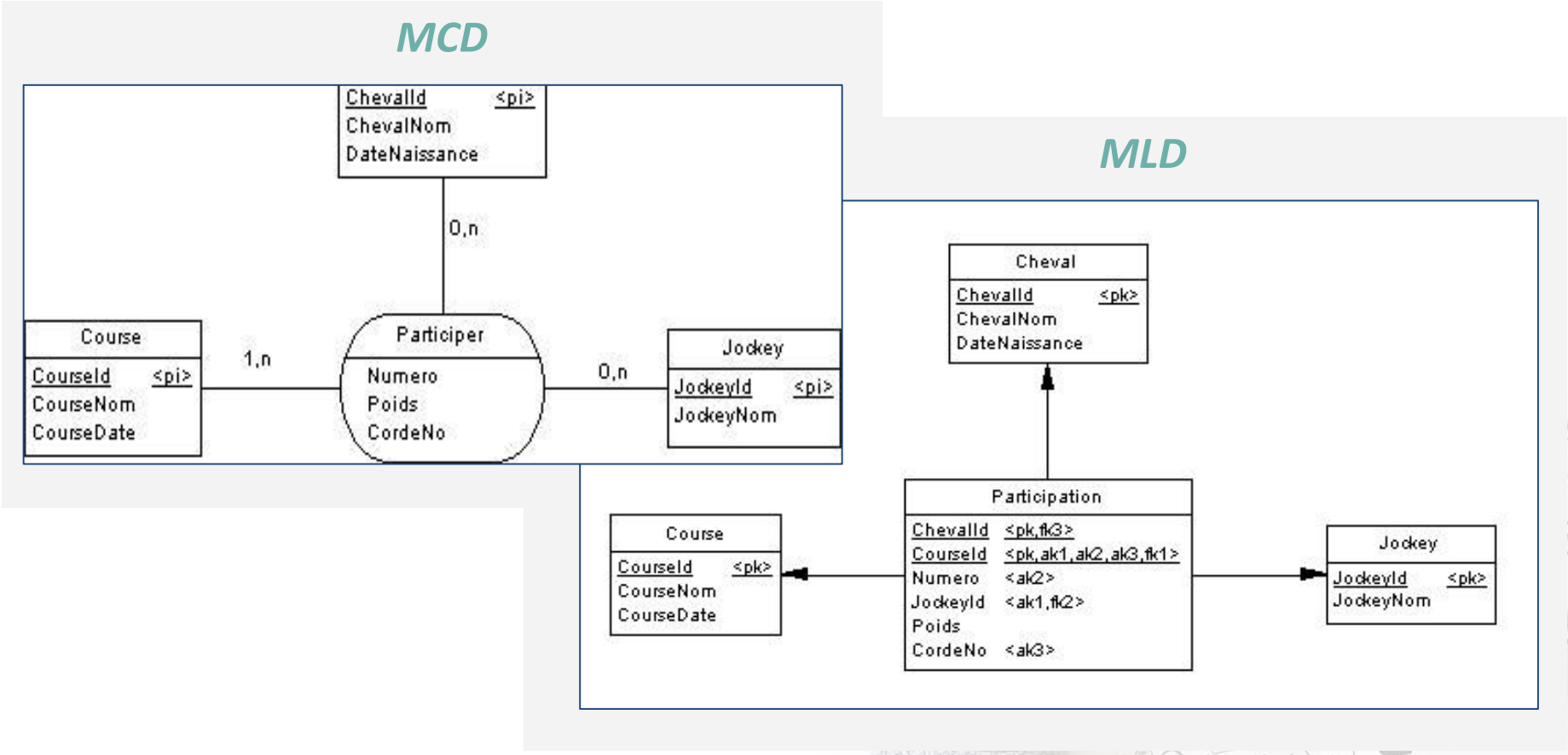
*On peut aussi exprimer un MLD traditionnellement en texte sous forme de **relations**.*

cheval(<u>id</u>, nom, naissance)

participation(<u>id</u>, id_cheval, id_course, jocker, poids, prix, box)

course(<u>id</u>, nom, date, lieu)

Différence entre un MCD et un MLD



Étapes pour transformer un mcd en mld

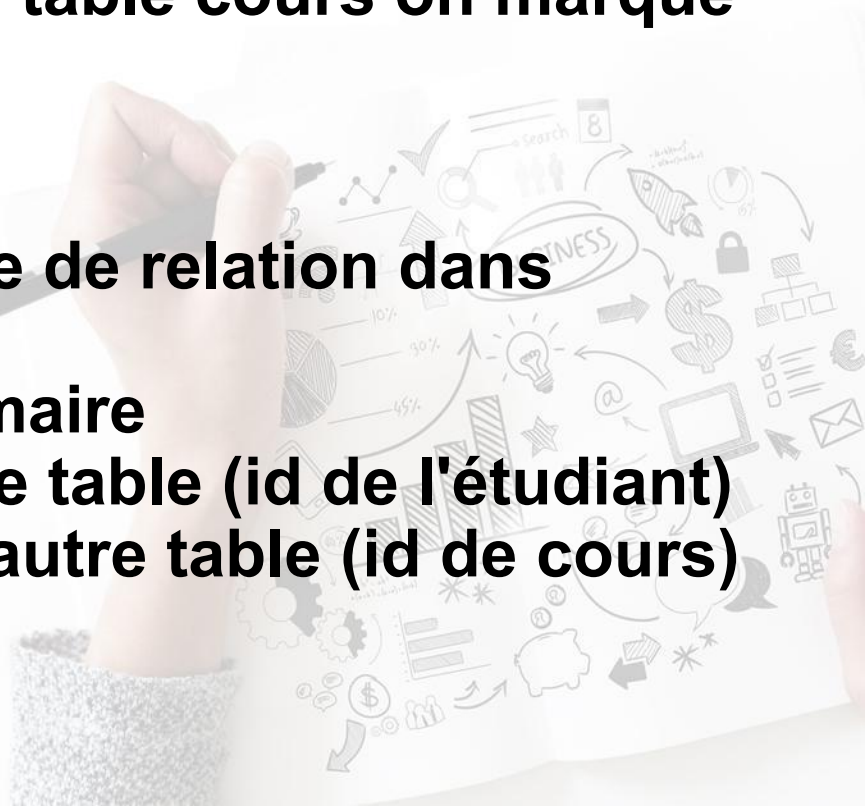
- *Tranformer les concepts en tables*
  - lister les champs
  - Identifier la clé primaire
    - Le champs souligné est marqué comme clé primaire.
    - Utiliser une clé primaire technique autogénérée
- *Transformer les associations*
  - Si les deux cotés de l'association sont N,
    - l'association devient une table de relations
      - (avec 2 clés étrangères)
  - Si un seul coté de l'association est N
    - L'association devient une clé étrangère du coté multiple
  - Si aucun coté de l'association est N
    - Les deux tables peuvent être fusionnées ou non
- *Si l'association a des attributs*
  - Il y a de fortes chance que ce soit un N-N et les attributs vont dans la table de relation
  - Dans le cas contraire, l'attribut va du coté multiple

Créez les relations de multiplicité

On va créer les relations

- *ONE-TO-MANY*
  - **on insère une clé étrangère dans la table multiple**
    - **par exemple, dans la table cours on marque l'id du programme**

- *MANY-TO-MANY*
  - **on crée la nouvelle table de relation dans laquelle il y a**
    - **une nouvelle clé primaire**
    - **la clé étrangère d'une table (id de l'étudiant)**
    - **la clé étrangère de l'autre table (id de cours)**



03

Modèle PHYSIQUE des données

1. Créer les tables et leurs clés primaires
   soit avec un wizard ou DDL
   - choisir le type de chaque champs
2. Ajouter les clés étrangères

Le modèle physique est le DDL

C'est donc le SQL qui sert à créer les tables et les clés étrangère

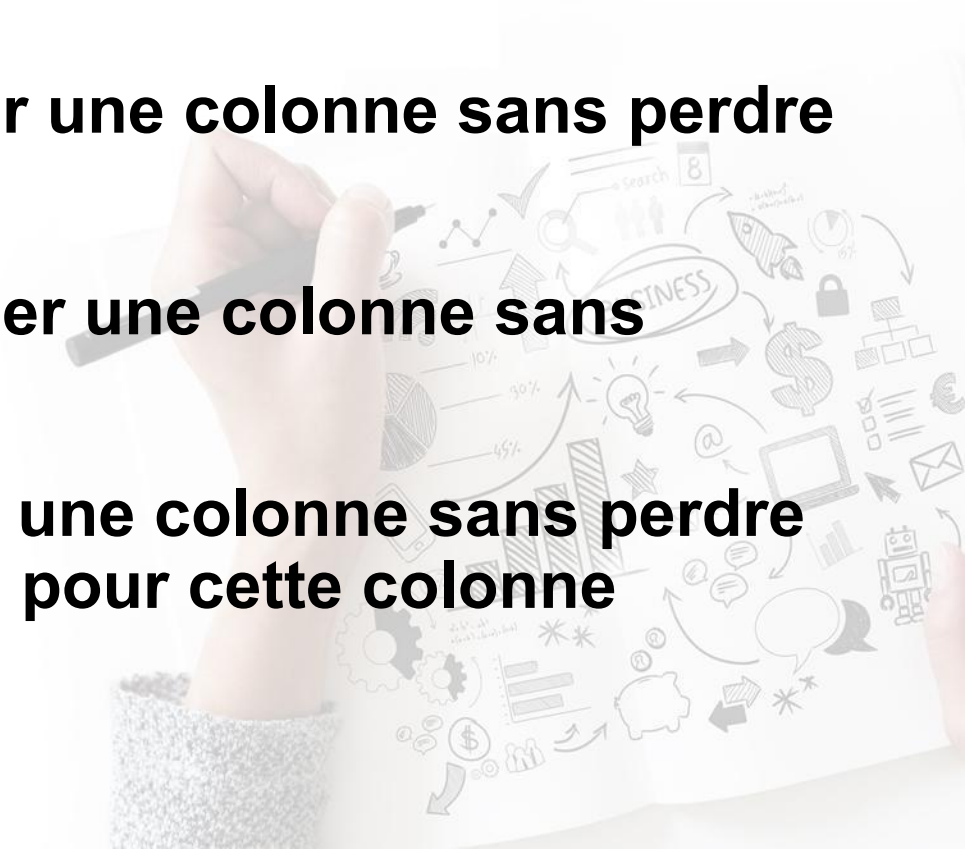
- *CREATE TABLE*
  - **create table cheval**
    (
    id int,
    nom varchar(255),
    naissance timestamp
)

- *PAS de INSERT*
  - **le DDL = Data Definition Language**

le DDL = Data Definition Language

- *CREATE TABLE*
  - code sql pour créer la table

- *ALTER TABLE ADD COLUMN*
  - code sql pour ajouter une colonne sans perdre les données
- *ALTER TABLE rename COLUMN*
  - code sql pour changer une colonne sans perdre les données
- *ALTER TABLE DROP COLUMN*
  - code sql pour retirer une colonne sans perdre les données excepté pour cette colonne



DDL

le DDL = Data Definition Language

- *CREATE TABLE*
  - code sql pour créer la table

MODÉLISATION INITIALE des données

- *ALTER TABLE ADD COLUMN*
  - code sql pour ajouter une colonne sans perdre les données
- *ALTER TABLE rename COLUMN*
  - code sql pour changer une colonne sans perdre les données
- *ALTER TABLE DROP COLUMN*
  - code sql pour retirer une colonne sans perdre les données excepté pour cette colonne

MAINTENANCE

MODE AGILE

Table itération 1

Comment mettre à jour cette table ?

- *TABLE ORIGINALE*
  - **create table touriste**
    (
    id int,
    nom varchar(255),
    courriel varchar(255)
)

Suite à l'analyse des maquettes et papiers d'assurance on doit ajouter le numéro assurance maladie

Mise à jour dans l'itération 2

Comment mettre à jour cette table ?

- *TABLE ORIGINALE*
  - **create table touriste**
    (
    id int,
    nom varchar(255),
    courriel varchar(255),
    assuranceMaladie varchar(20)
)

oui pour recréer le système mais efface toutes les données

Suite à l'analyse des maquettes et papiers d'assurance on doit ajouter le numéro assurance maladie

Mise à jour dans l'itération 2

Comment mettre à jour cette table ?

- *TABLE ORIGINALE*
  - **alter table touriste**
    add column assuranceMaladie varchar(20)

Suite à l'analyse des maquettes et papiers d'assurance on doit ajouter le numéro assurance maladie

Créer les tables

On va tenter de démêler toutes les informations du client et identifier les tables qui sont obligatoires pour réaliser l'app requise pour l'itération 1

- *Toutes les colonnes nécessaires à la première version du logiciel (itération 1) doivent être prêtes*
- *On va décider des types de champs*
  - Attention de ne pas être trop restrictif.
    - Par exemple, si vous décidez de faire un code postal de 5 lettres pour une application Usa elle ne fonctionnera pas au Canada
    - Par exemple, si vous décidez de restreindre le téléphone à 7 chiffres, il ne marchera pas en France ! dans le futur
- *Implémenter les clés*
  - Implémenter la clé primaire (serial, auto_increment, série)
  - Implémenter la clé étrangère (add foreign key)

Types des champs dans PostgreSQL

https://www.postgresql.org/docs/9.1/datatype-datetime.html
https://www.postgresql.org/docs/9.5/datatype.html
https://www.educba.com/postgresql-data-types/

- *uen commande liste les types de données actuels*
  - **postgres=# \d pg_type ;**

- *ces types sont*
  - **bool, bytea,**
  - **char, text, name**
  - **int8, int2, int4**
  - **int2vector, oidvector**
  - **json, xml, _xml**
  - **date, time, timstamp, timestamptz**

- *les types internes sont*
  - **regproc**
  - **oid, tid, xid, cid**

Types des champs dans PostgreSQL

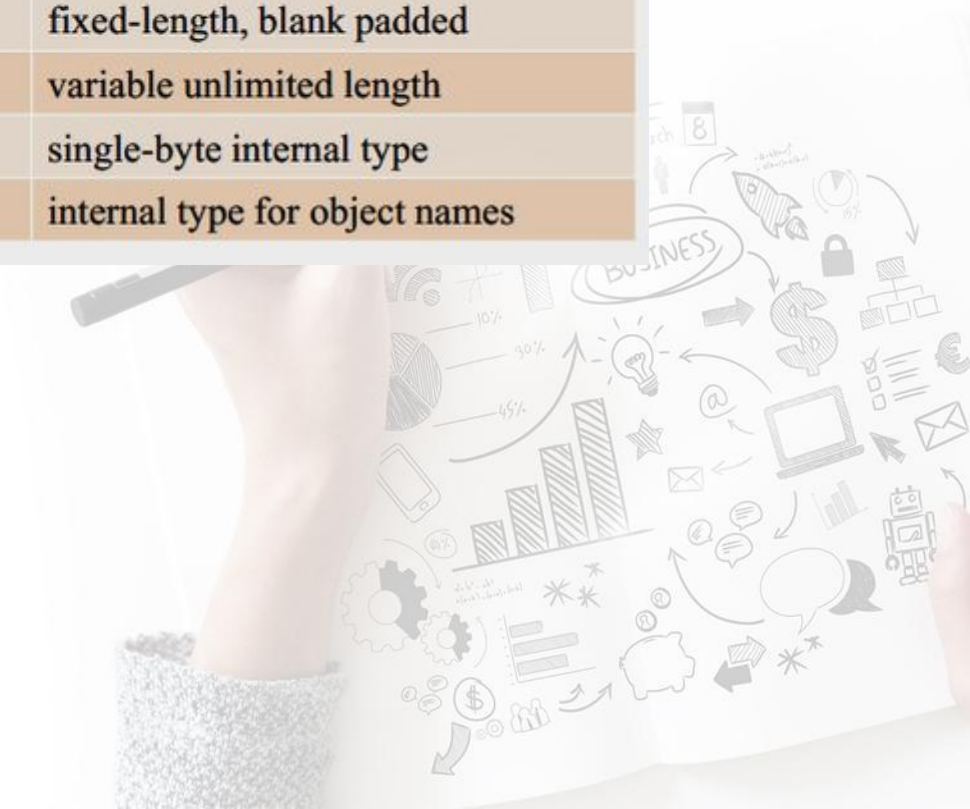
Les types nombres

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
decimal / numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Types des champs dans PostgreSQL

Les types texte

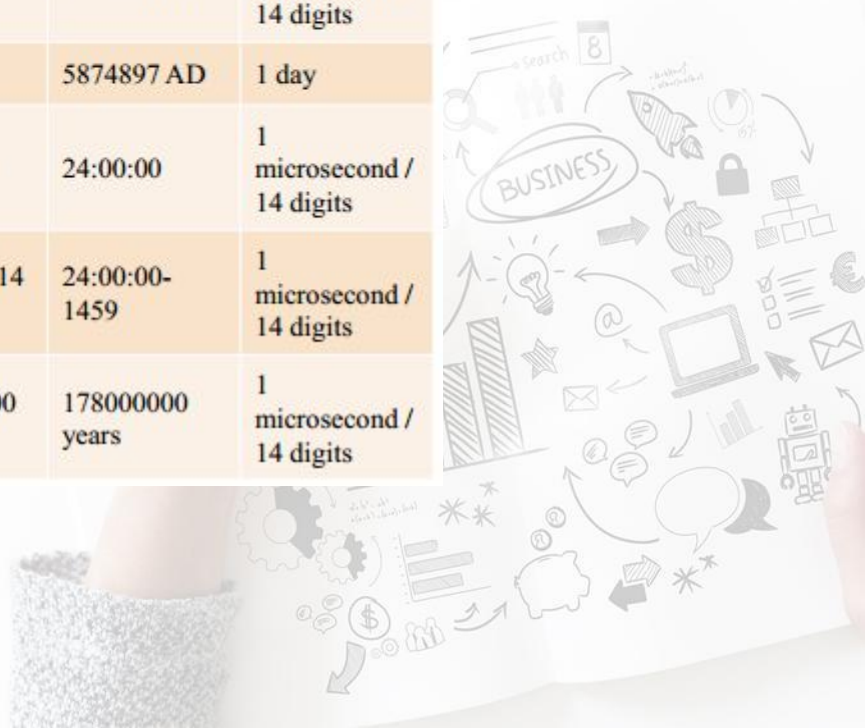
Name	Storage Size	Description
character varying(n), varchar(n)	variable(can store n chars)	variable-length with limit
character(n), char(n)	n chars	fixed-length, blank padded
text	variable	variable unlimited length
"char"	1 byte	single-byte internal type
name	64 bytes	internal type for object names



Types des champs dans PostgreSQL

Les types date

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond / 14 digits
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond / 14 digits
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond / 14 digits
time [(p)] with time zone	12 bytes	times of day only, with time zone	00:00:00+1459	24:00:00-1459	1 microsecond / 14 digits
interval [fields] [(p)]	12 bytes	time interval	-178000000 years	178000000 years	1 microsecond / 14 digits



BONNES

modélisations