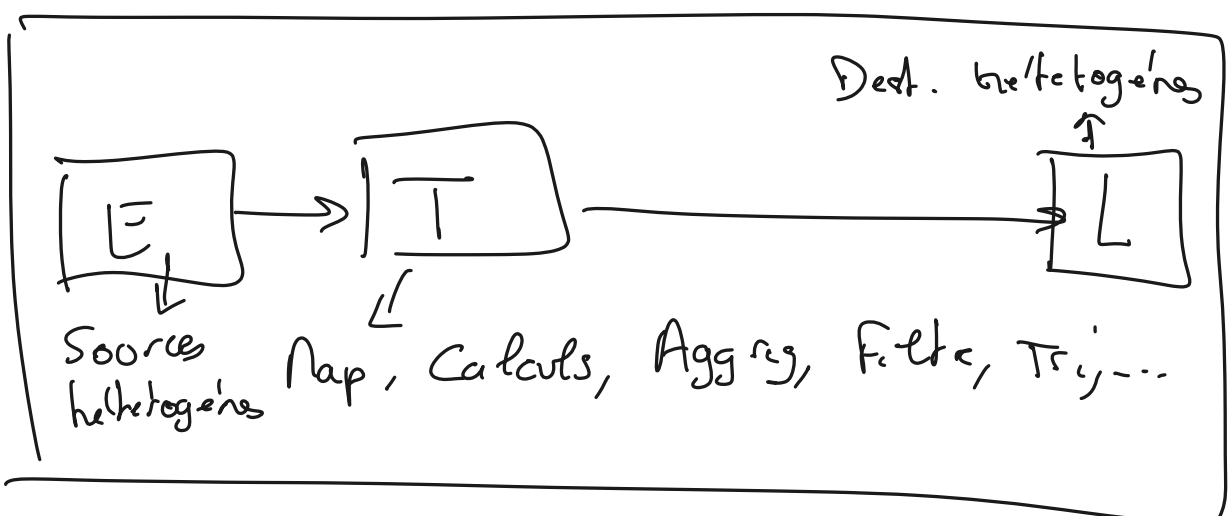
Talend for Data Integration = ETL

" " Enterprise Service BUS = ETL + ESB

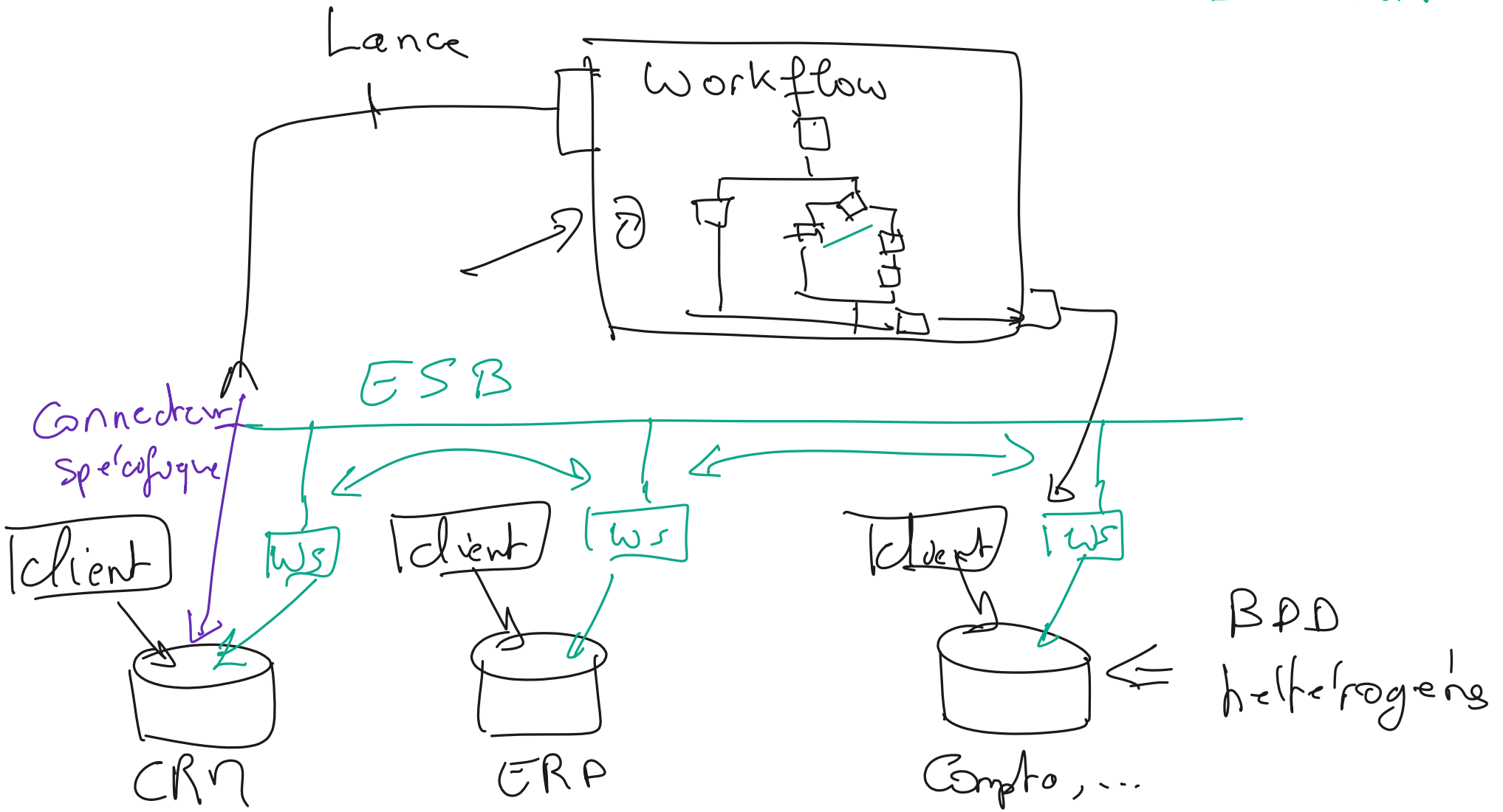
" " Big Data = ETL + ESB + Hadoop

Extract Transform Load



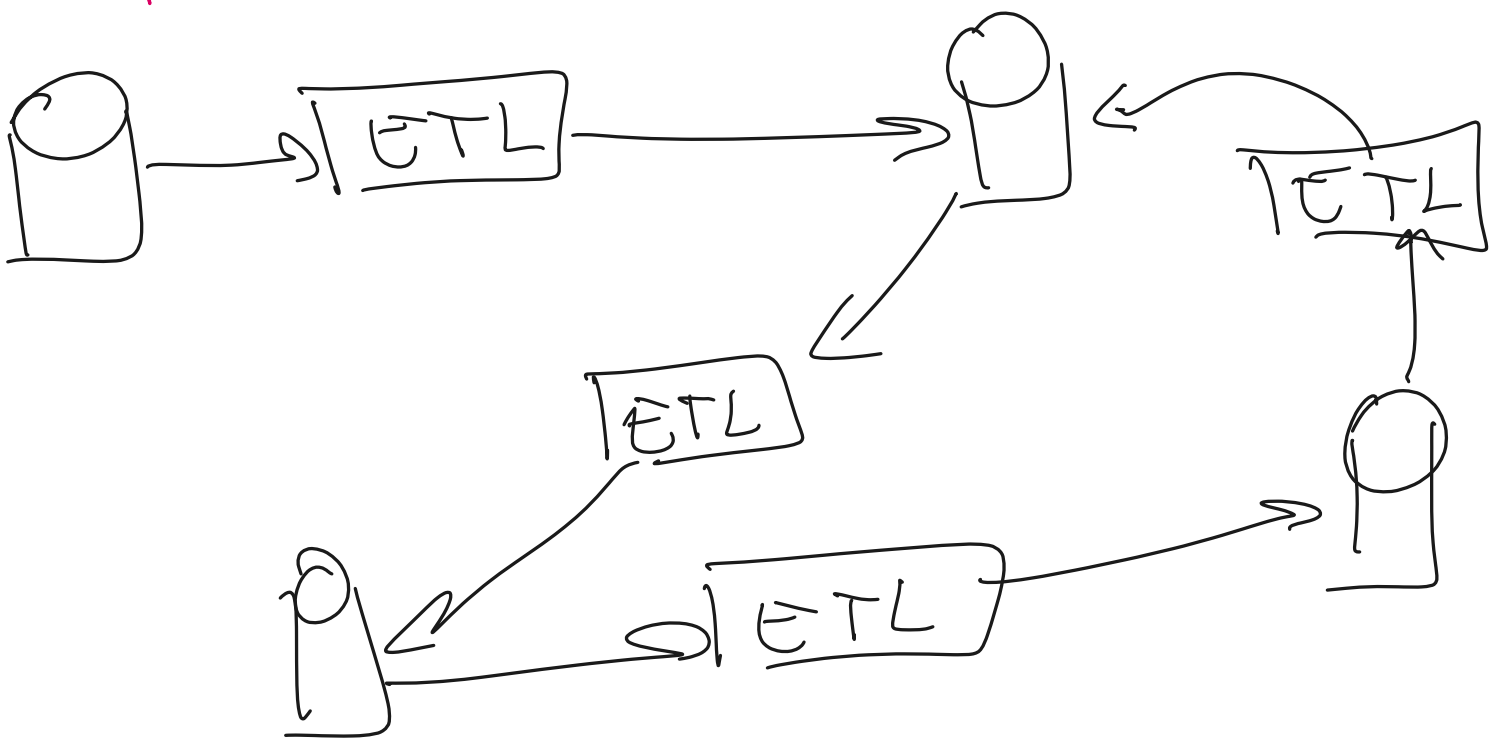
ESB

WS = Web Service

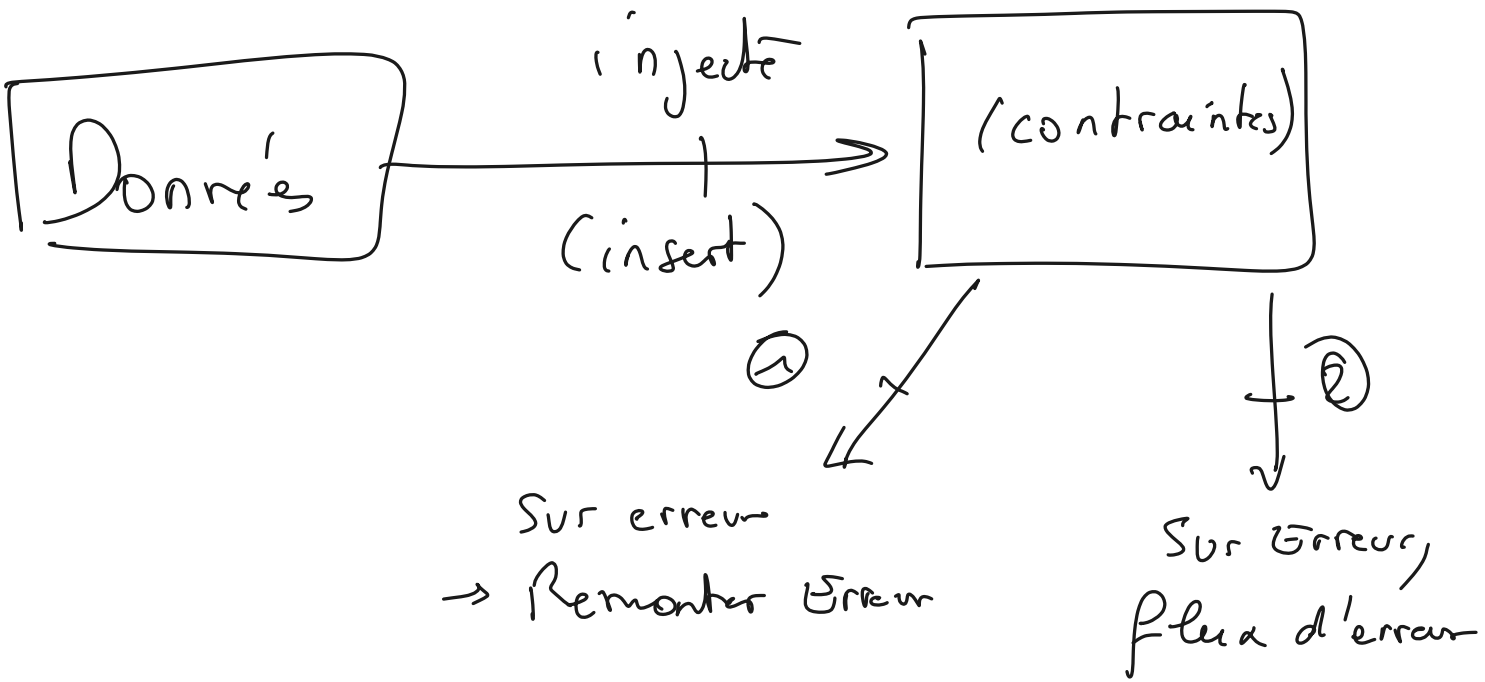


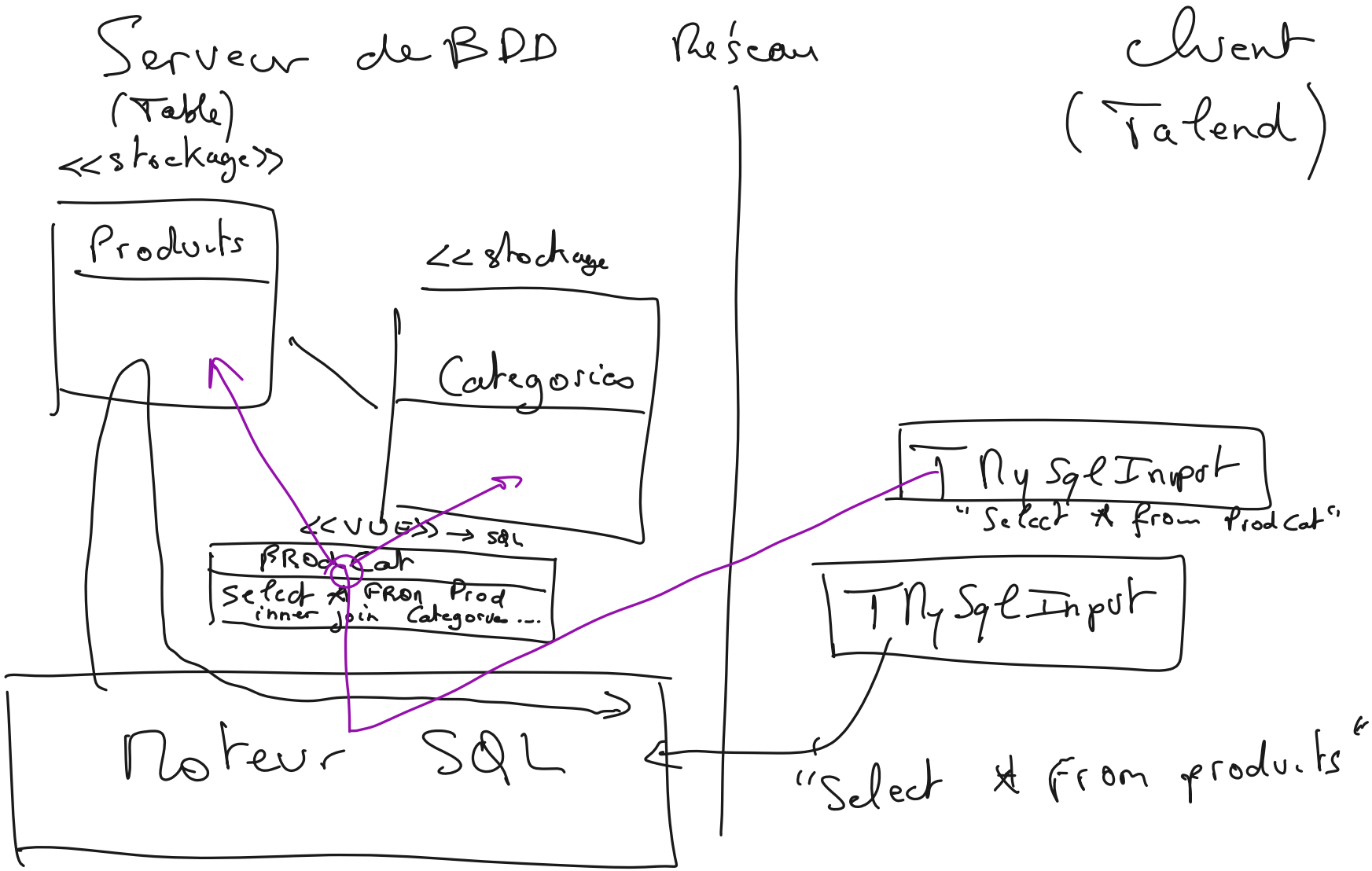
Limite de l'ETL

Defauts: plein de process dans tous les sens



Qualités: Simple et rapide


Données
injecté
(insert)
(contraintes)
a
b
Sur erreur
→ Remonter Erreur
Sur Erreur,
flux d'erreur

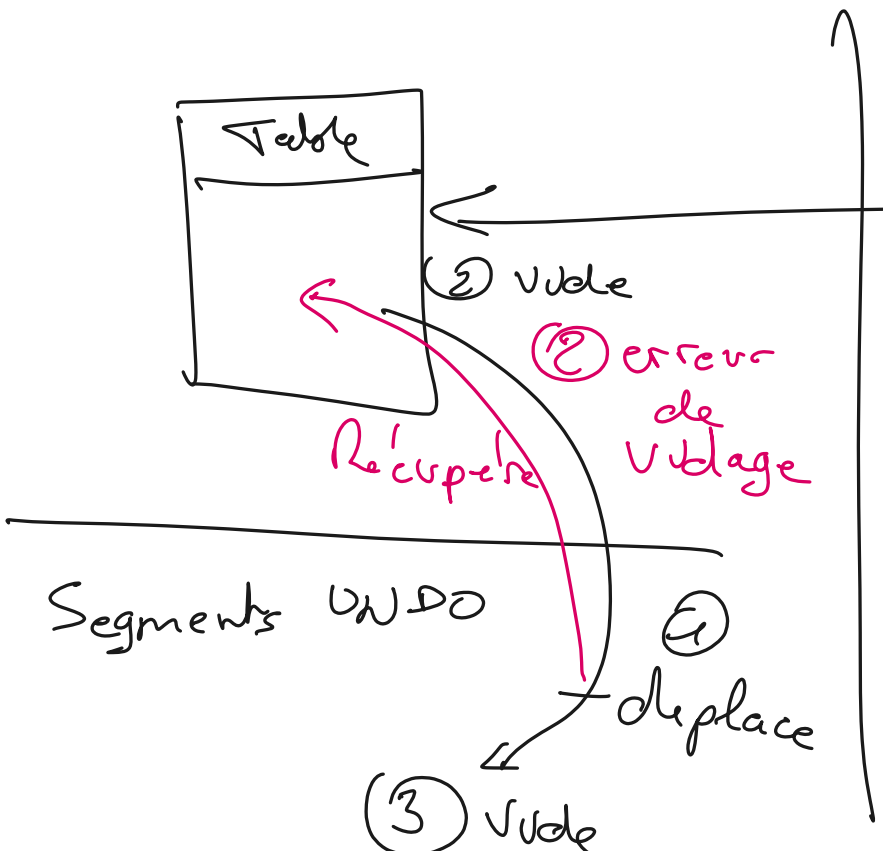

Serveur de BDD
(Table)
<<stockage>>
Réseau
client
(Talend)
Products
<<stockage
Categories
<<VUE>> → SQL
PRODCAT
Select * FROM Prod
inner join Categories ...
TMySqlInput
"Select * from ProdCat"
TMySqlInput
Moteur SQL
"Select * From products"

SQL :

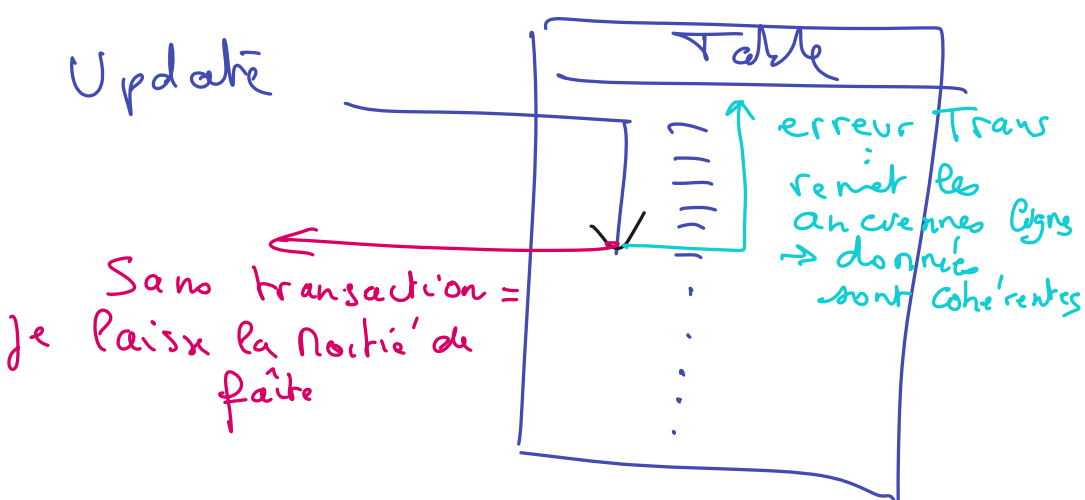
- Creer une Copie de products
- y injecter une Copie de données -
  - (a) - récupérer les lignes en erreur et les afficher sur la console
  - (b) chercher le paramètre pour "planter" le job si au moins 1 ligne est en erreur -

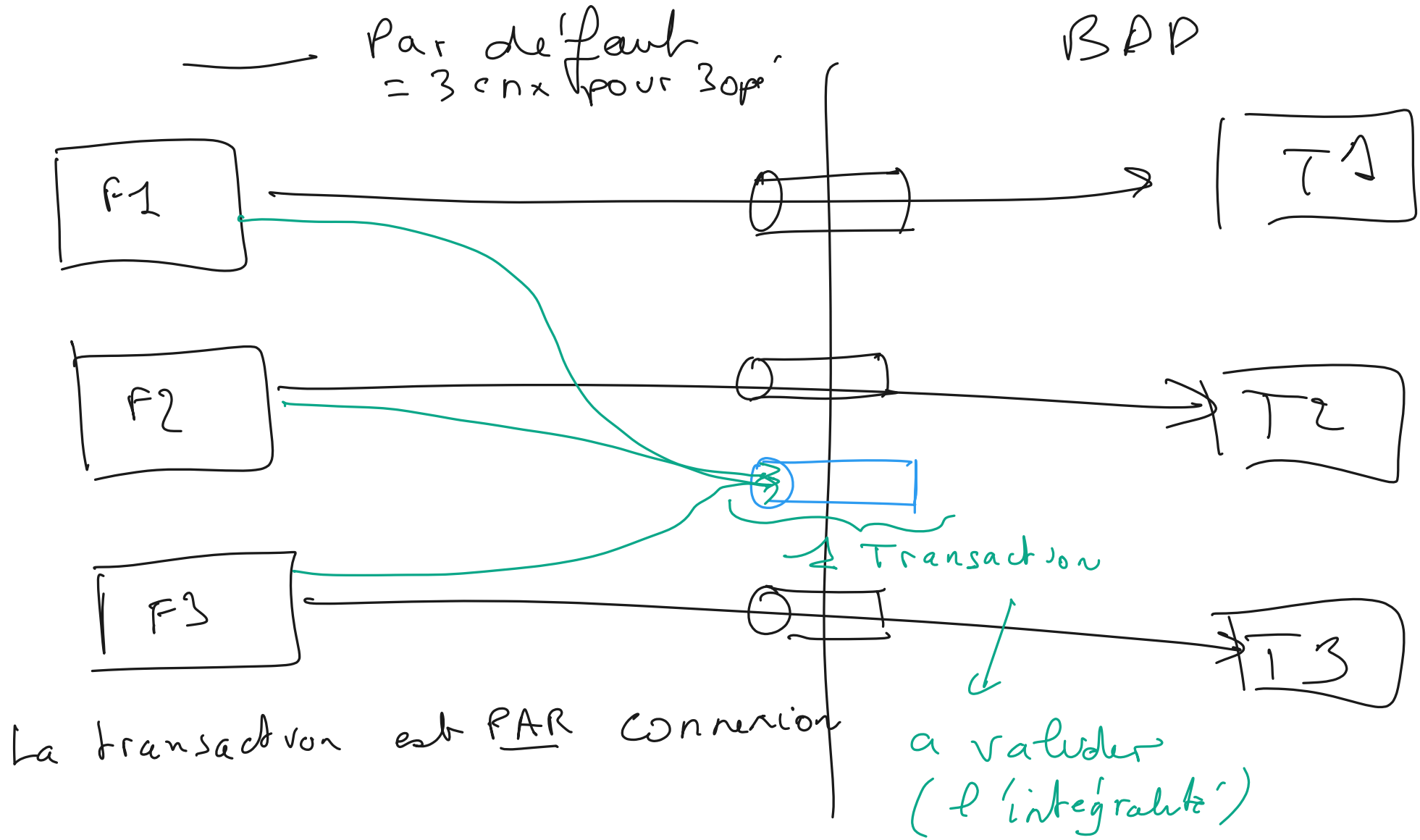
Notion de Transaction

BDD



Modification non Transactionnelle :




Par défaut
= 3 cnx pour 3 opé
BDD
F1
F2
F3
T1
T2
T3
Transaction
a valider
(l'intégralité)
La transaction est PAR connexion

Faire une Copie de 3 table

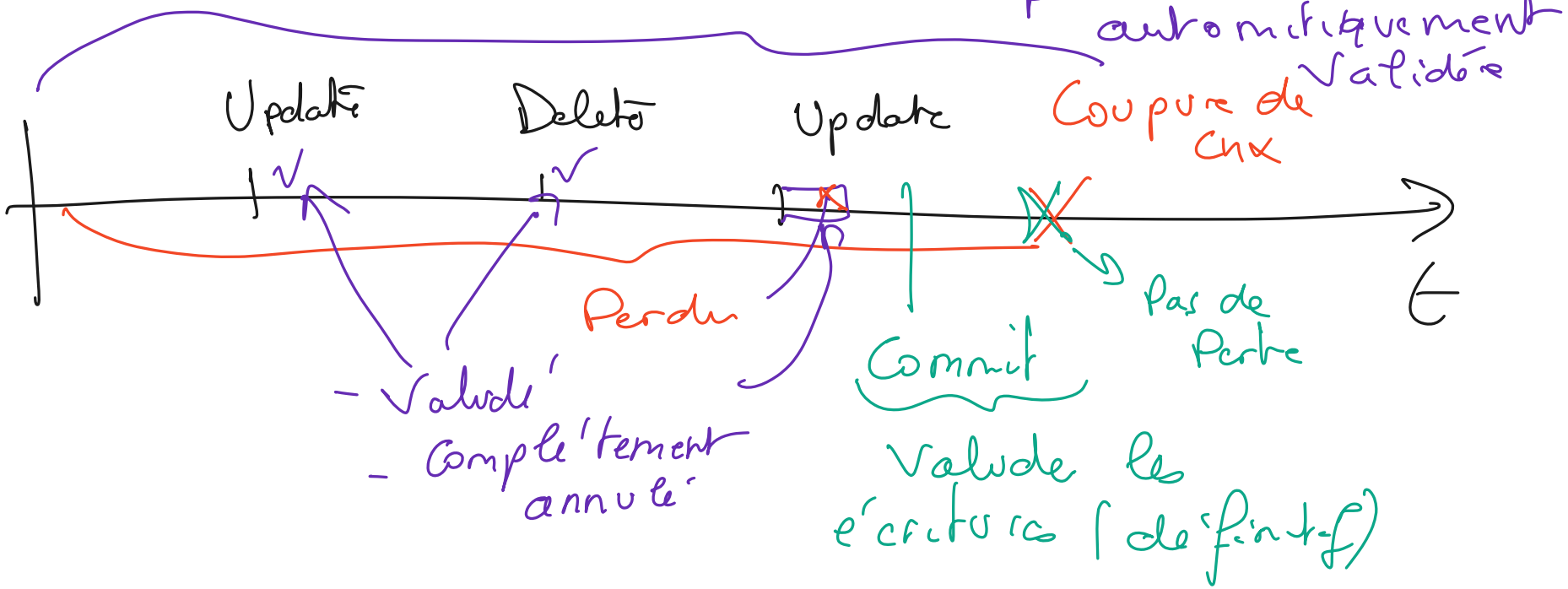
Mais en cas d'erreur, rien ne

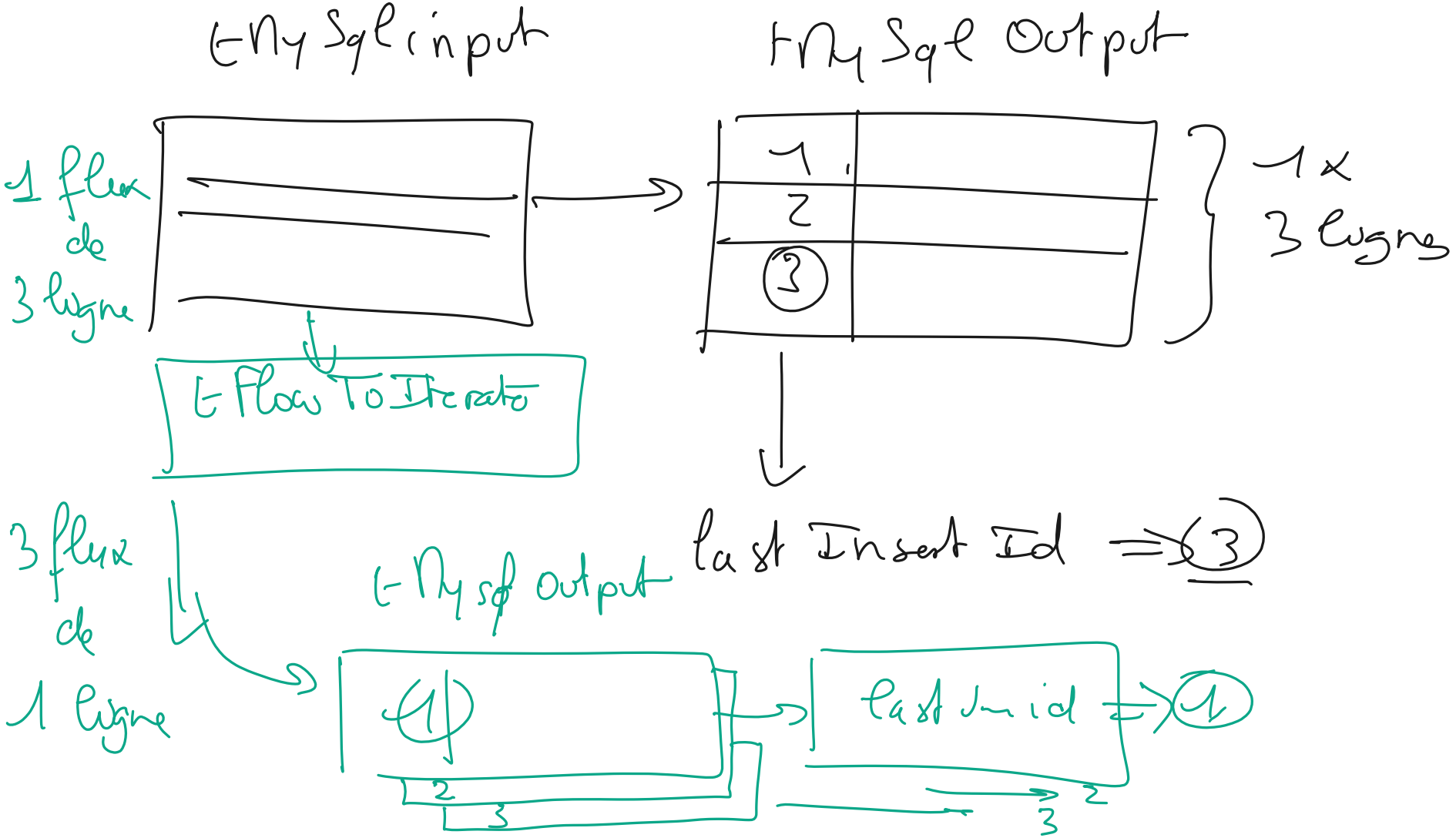
doit être modifié,

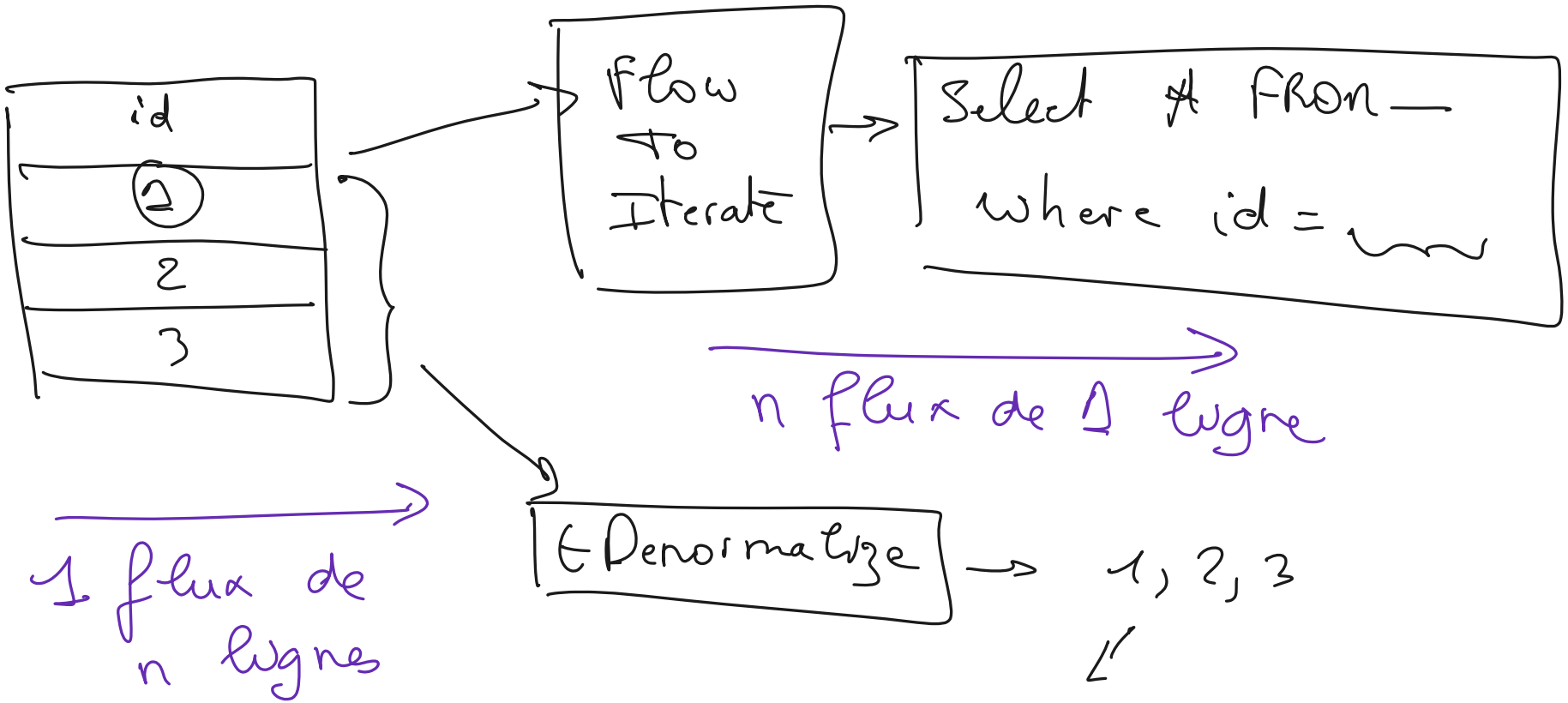
soit Tout est ok, soit Rien

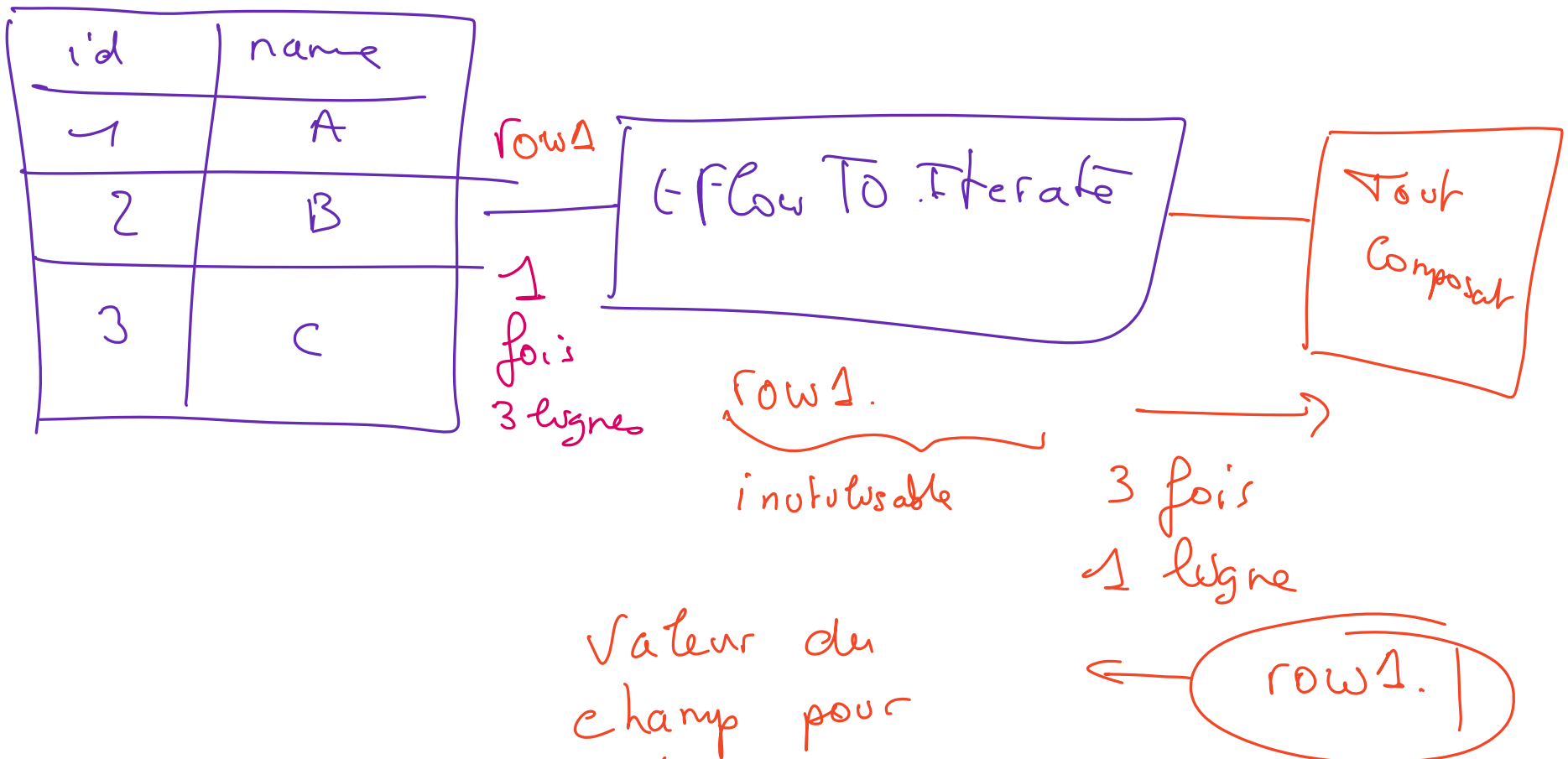
Transaction

Si autocommit, alors chaque opération est automatiquement validée

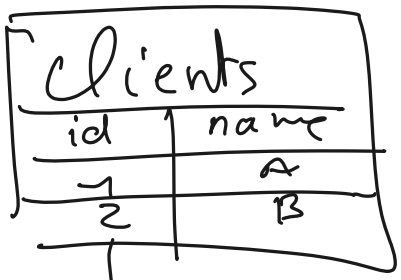


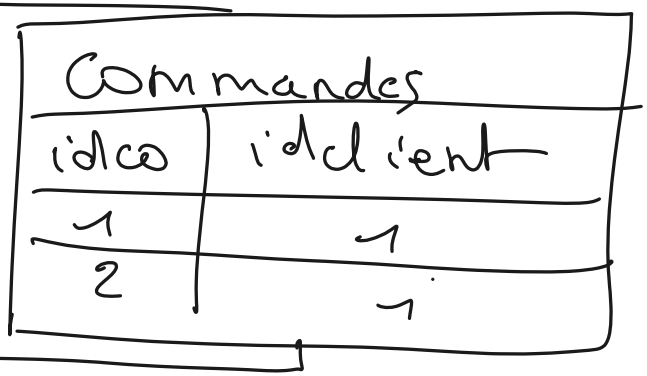

tMySql input
tMySql Output
1 flux de 3 lignes
1
2
3
1 x 3 lignes
tFlowToIterate
3 flux de 1 ligne
tMysql output
last Insert Id => 3
1
last_insert_id => 1
2
3
2
3


id
2
2
3
Flow
To
Iterate
Select * FROM —
where id = ...
n flux de 1 ligne
1 flux de
n lignes
Denormalize
1, 2, 3
Selct * FROM — id IN (1,2,3)


id
name
1
A
2
B
3
C
row1
1 fois
3 lignes
FLOW TO Iterate
Tout
Composer
row1.
inutilisable
3 fois
1 ligne
row1.
Valeur du
champ pour
l'itération en cours

Jointures





de quel côté on garde les lignes sans correspondances.

Externe

à gauche

Complète

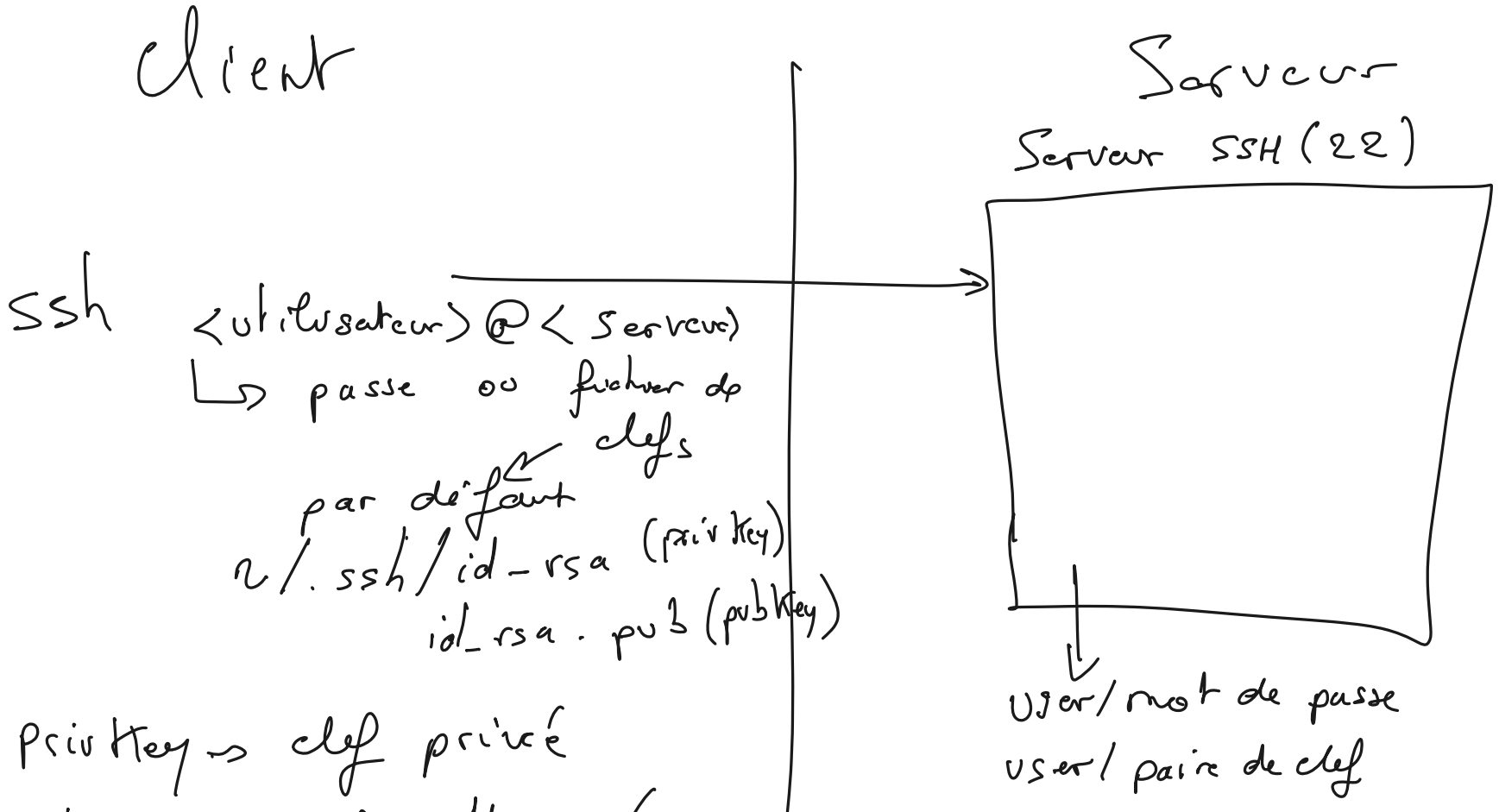
à droite

idclient	name	idco
1	A	1
1	A	2
2	B	null

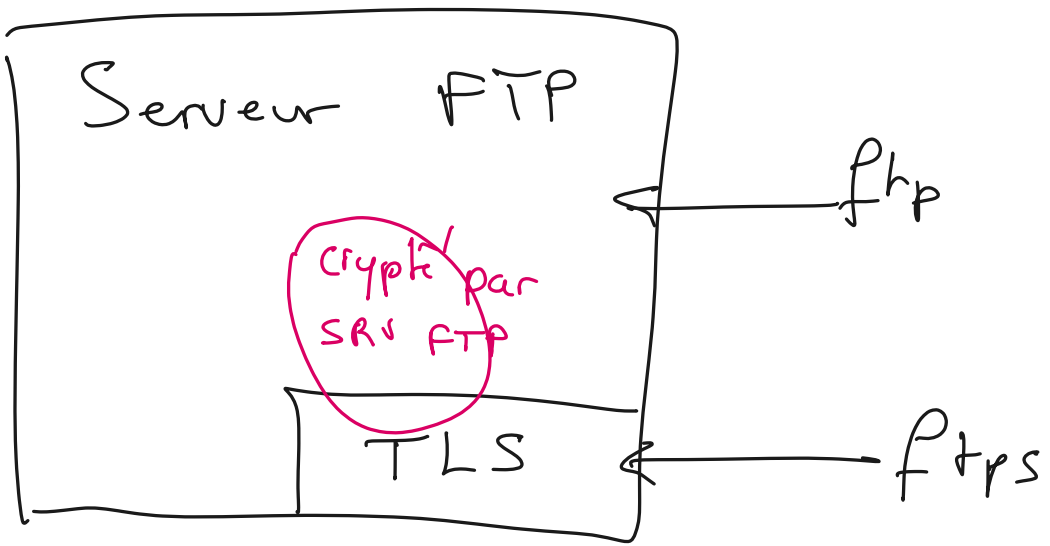
interne

→ uniquement les correspondances

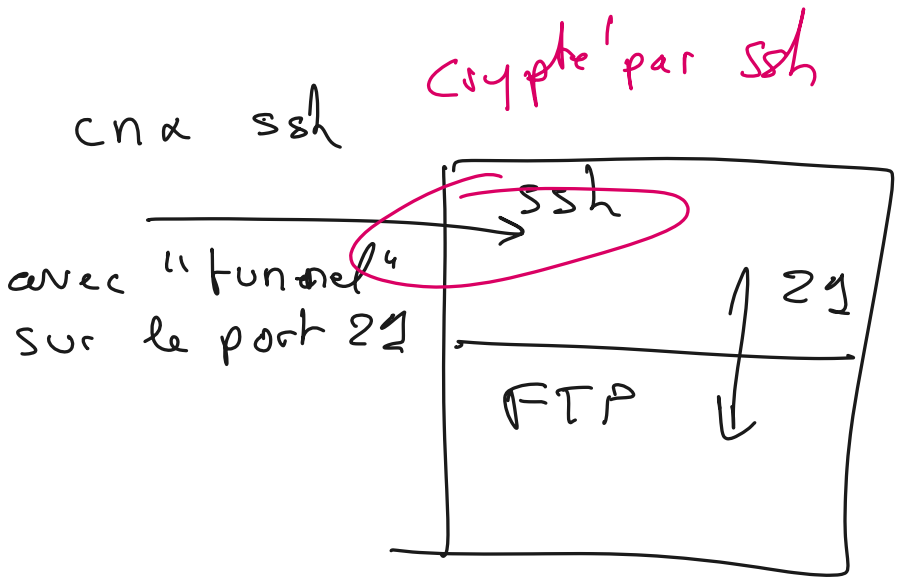
idcli	name	idco
1	A	1
1	A	2

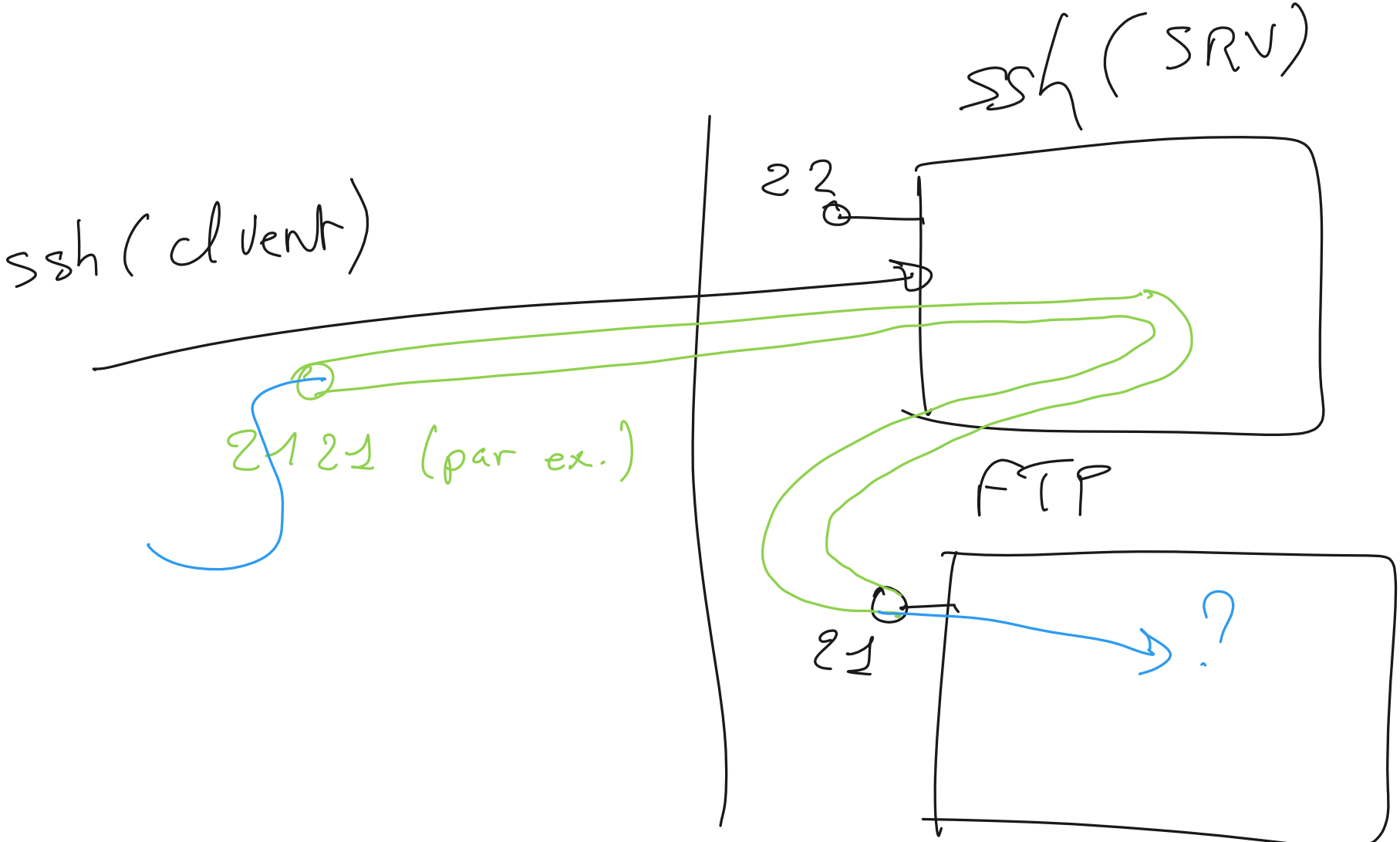

client
Serveur
Serveur SSH (22)
ssh <utilisateur>@<serveur>
↳ passe ou fichier de clefs
par défaut
~/.ssh/id_rsa (privKey)
id_rsa.pub (pubKey)
privKey → clef privé
pubKey → clef publique (a envoyer ou obtenir du serveur)
User/mot de passe
user/paire de clef

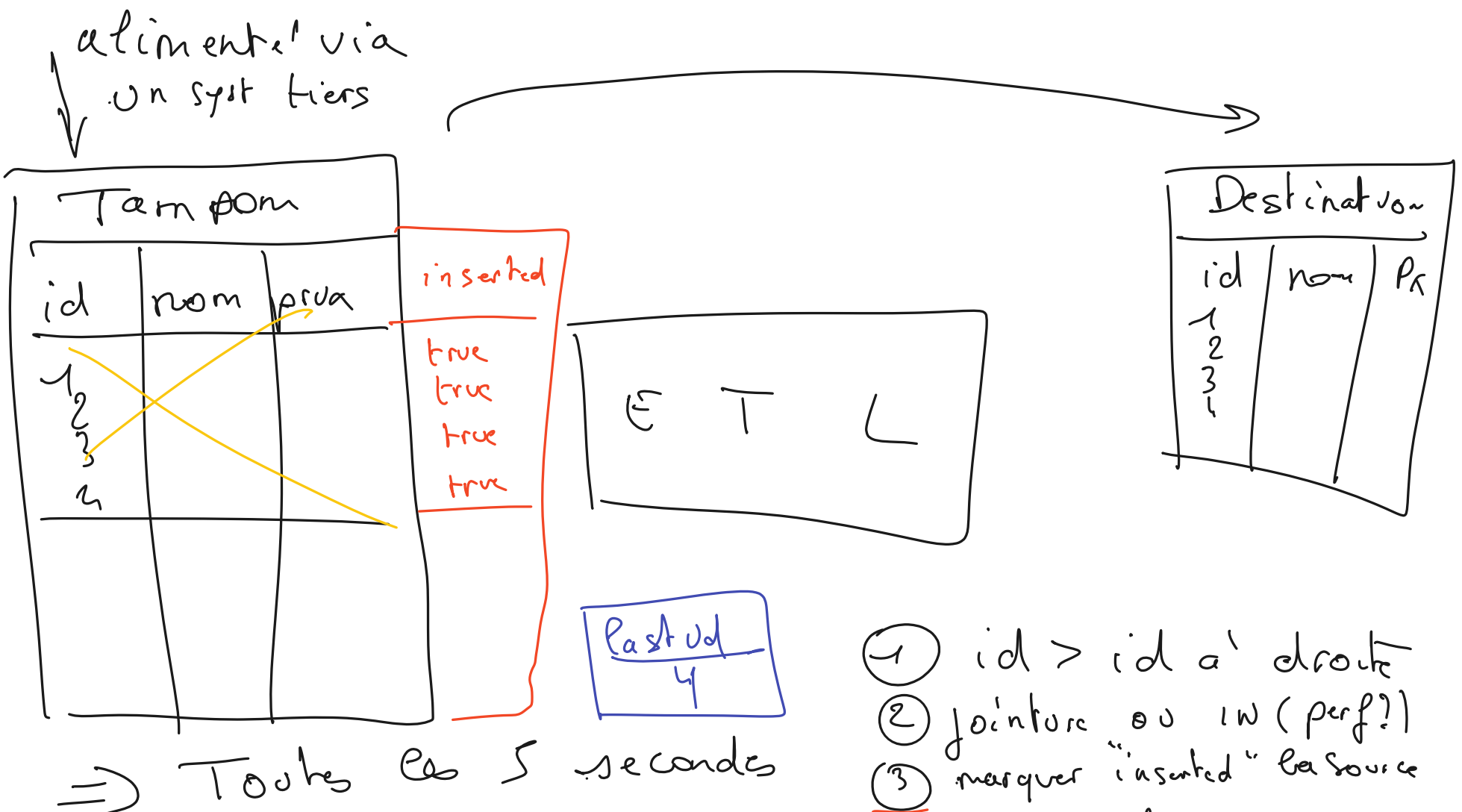
FTPS


Serveur FTP
ftp
Crypté par SRV FTP
TLS
ftps

SFTP


Crypté par ssh
cnx ssh
ssh
avec "tunnel" sur le port 22
22
FTP


ssh (SRV)
22
ssh (client)
2121 (par ex.)
FTP
21
?

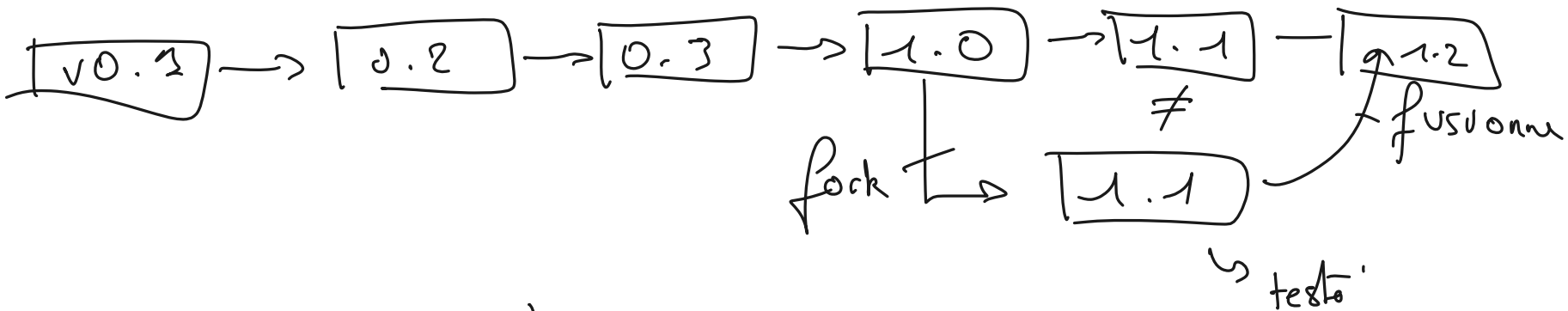


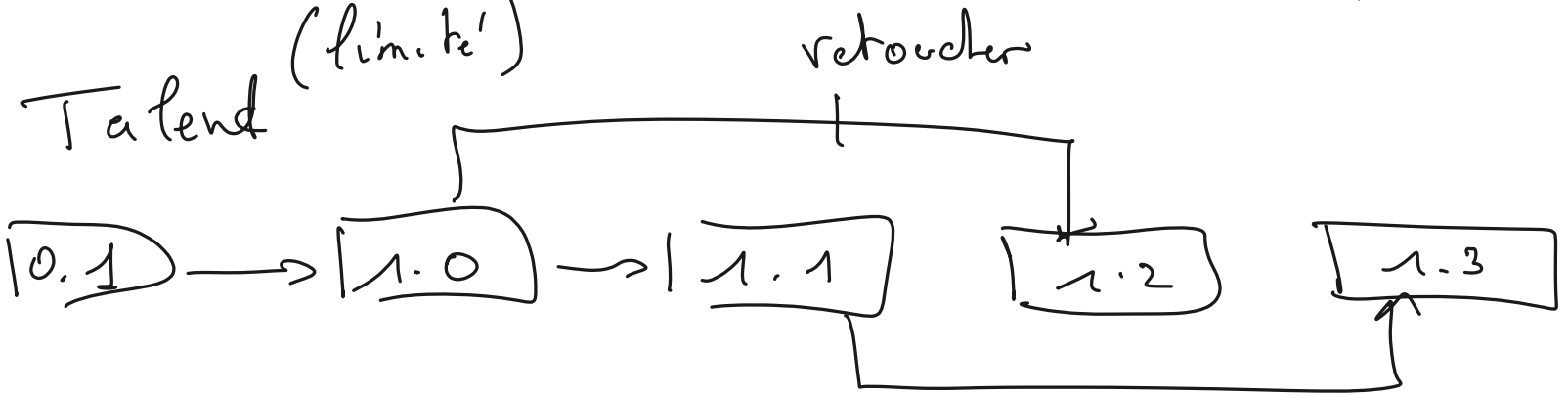
1. id > id à droite
2. jointure ou IN (perf?)
3. marquer "inserted" la source
4. suppr les lignes
5. stocker le lastId dans une table

- Créer un job, "le tester"
  - le sauvegarder (ne pas pouvoir le perdre)
  - retoucher ce job, le tester
  - il est ok → l'enregistrer pour mise en production

- Retoucher ce job en prod pour le corriger, enregistrer.
  - on doit pouvoir revenir en arrière en cas d'erreur -

Git


v0.1
0.2
0.3
1.0
1.1
1.2
≠
fusionner
fork
1.1
testé


Talend
(limité)
retoucher
0.1
1.0
1.1
1.2
1.3

Variables

① locales à un script : ex : int a = 0;
→ uniquement la portée en cours

② valable pour l'execution du job (pas les autres jobs) → globalMap (Talend globale)
→ il y a des var. intrinsèques (fournies par Talend)
→ Possibilité d'y mettre nos var. de tout type

Map :

clé	Valeur
chaine	Tout type

globalMap.getOrDefault

containsValue → on a une valeur pour la clef ?

put(clef, valeur)

get(clef) : valeur

Avec le Composant pre job,
appelez un script java d'initialisation
Fixez dans une variable, un message
("Hello...")

Votre Job contiendra un Msgbox
Ce qui sera affiché sera la variable