

MU5IN852
Bases de Données Large Echelle

Analyse multidimensionnelle en SQL

Hubert.Naacke@lip6.fr

Objectifs

- Notions de schéma multidimensionnel
- Requêtes d'analyse multidimensionnelle
- Efficacité des requêtes multidimensionnelles à large échelle

Motivations dans le contexte big data

- Besoin d'analyse multidimensionnelle
 - Analytics, tableau de bord,
 - Analyse en ligne (online) « instantanée »
- Besoin de solution unifiée
 - Un même environnement pour la préparation, l'analyse, la prédiction, ...
- Solution évolutive
 - Analyse exprimée dans un langage déclaratif
 - bénéficie des progrès en optimisation de requêtes
 - Transformation «requête vers programme» toujours plus efficace
 - bénéficie des progrès technologiques sous-jacents
 - calcul parallèle SIMD dans un processeur, ...

Rappels sur les modèles de données

- Modèle Entité/Association
 - Une entité a des attributs
 - Une association entre plusieurs entités
 - Une association peut avoir des attributs d'association
- Modèle relationnel (SQL)
 - Relation : une table
 - Identifiant : clé d'une table
 - Référence à un identifiant : lien entre relations
- Modèle multidimensionnel
 - Fait, mesure
 - Dimension, hiérarchie

Fait et mesure

- Une relation principale représente un fait
 - Un type d'événement à analyser
 - Exple: Achat(numA, client, date, magasin, produit, prix)
- Identifiant d'un fait
 - Identifiant propre au fait. Exple: numA
 - Ou identifiant composé
- Mesure du fait
 - Un attribut décrivant le fait
 - Exple : le prix d'achat d'un produit

Dimensions et hiérarchie

- Dimensions
 - Les entités associées au fait
 - Un fait peut référencer plusieurs dimensions
 - Exple: date, client, magasin
- Dimension **hiérarchique**
 - Objectif : offrir de multiples analyses
 - Analyse globale très synthétique → ... → analyse fine
 - Une dimension peut être organisée en plusieurs niveaux
 - Niveau supérieur global → ... → Niveau inférieur détaillé
 - Une table par niveau

Exemples de dimensions hiérarchiques

- Dimension temporelle
 - Date(horodatage, idJour)
 - Jour(idJour, nom, idMois)
 - Mois (idMois, nom, idAnnée)
 - ...
- Dimension géographique
 - Magasin(gps, idVille)
 - Ville(idVille, nom, idPays)
 - Pays(idPays, nom, continent)
- Catégorie d'un produit en 3 niveaux
 - Famille → rayon → catégorie détaillée
- Classification biologique
 - famille → genre → espèce

Forme des schémas

- Etoile simple
 - La table de faits est directement reliée à chaque dimension
 - dimension sans hiérarchie
- Etoile avec des branches « longues »
 - Le fait est relié au niveau inférieur de chaque dimension
 - dimensions hiérarchiques
- Flocon
 - une dimension peut être décrite par plusieurs hiérarchies
- Formes réelles
 - Schéma formé de plusieurs étoiles ou flocons
 - Tenir compte des associations N-N.
 - Exemple : vente de médicaments. Un médicament peut contenir plusieurs molécules. Nb de ventes par molécule.

Exemple de faits

Les visites

photoID ▲	personID ▲	date ▲	lon ▲	lat ▲	note ▲
p1	Bob	2020-09-03	41.5	12.8	3
p2	Alice	2020-09-01	41.6	12.8	5
p3	Bob	2020-09-04	41.6	13.2	1
p4	Bob	2020-09-04	40.1	12	2
p5	Alice	2020-09-04	40.1	12	2
p6	Alice	2020-09-05	41.6	12.7	4
p7	Alice	2020-10-05	41.8	12.8	4
p8	Carole	2019-12-25	30.1	10.1	2
p9	David	2019-12-25	30.1	10.1	1
p10	Eva	2019-12-25	30.1	10.1	5
p11	Eva	2019-12-26	31.1	10.1	3
p12	Alice	2020-02-01	32.1	12.1	3
p13	Bob	2020-02-01	32.1	12.1	5
p14	Carole	2020-02-01	32.1	12.1	2
p15	Carole	2019-11-11	49.1	10.1	1
p16	Alice	2019-12-25	30.1	10.1	4
p17	Alice	2020-02-02	49.1	10.1	5

Exemple de dimensions

Date

date ▲	jour ▲	mois ▲	annee ▲
2019-11-11	11	11	2019
2019-12-25	25	12	2019
2019-12-26	26	12	2019
2020-02-01	1	2	2020
2020-02-02	2	2	2020
2020-09-01	1	9	2020
2020-09-03	3	9	2020
2020-09-04	4	9	2020

Place

lon ▲	lat ▲	ville ▲	pays ▲
41.5	12.8	Aix	France
41.6	12.8	Aix	France
41.6	12.7	Nice	France
41.8	12.8	Marseille	France
41.6	13.2	Rome	Italie
40.1	12	Oslo	Norvege
30.1	10.1	Paris	France
31.1	10.1	StDenis	France

Profession

personID ▲	profession ▲
Alice	Architecte
Bob	Data science
Carole	Data science
David	Architecte
Eva	Vendeuse
Franck	Vendeur
Greta	Economiste

Exemple de schéma en étoile

Date

date ▲	jour ▲	mois ▲	annee ▲
2019-11-11	11	11	2019
2019-12-25	25	12	2019
2019-12-26	26	12	2019

Visites

photoID ▲	personID ▲	date ▲	lon ▲	lat ▲	note ▲
p1	Bob	2020-09-03	41.5	12.8	3
p2	Alice	2020-09-01	41.6	12.8	5
p3	Bob	2020-09-04	41.6	13.2	1

Profession

personID ▲	profession ▲
Alice	Architecte
Bob	Data science
Carole	Data science

Place

lon ▲	lat ▲	ville ▲	pays ▲
41.5	12.8	Aix	France
41.6	12.8	Aix	France

Exemple : le benchmark TPC-H

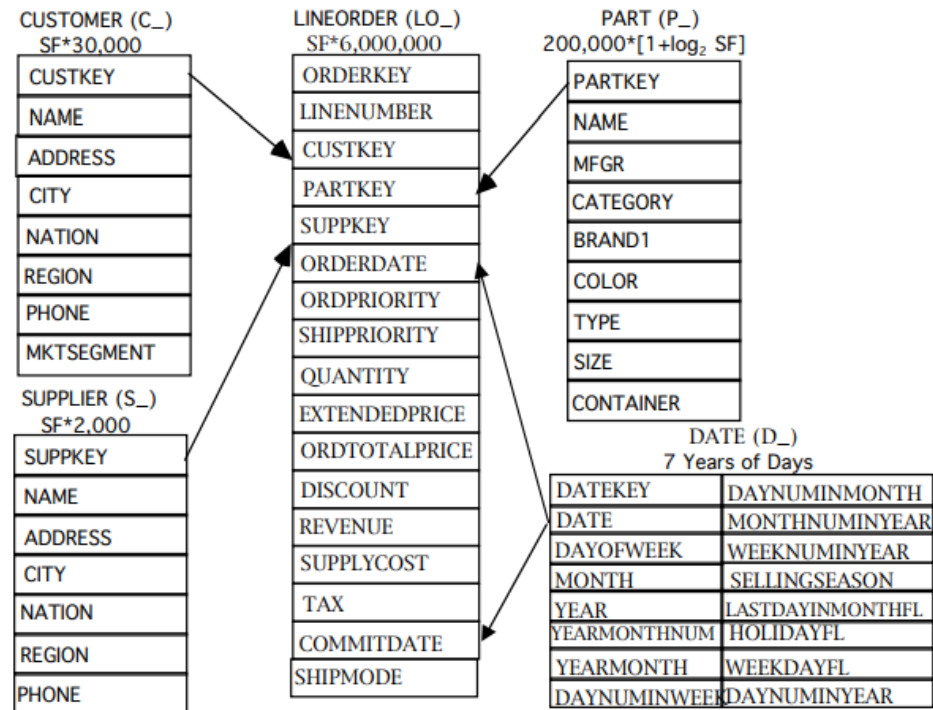
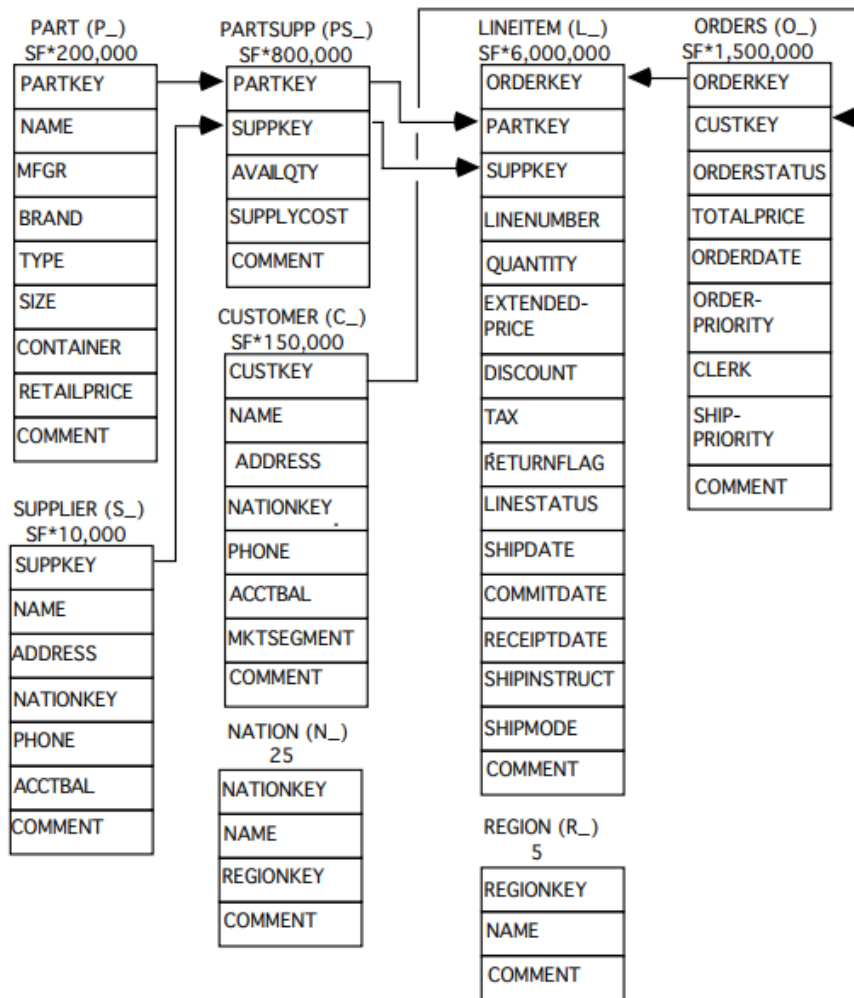


Schéma en étoile pour TPC-H

<https://www.cs.umb.edu/~xuedchen/research/publications/StarSchemaB.PDF>

Principaux types d'analyse

Agrégation

- Dimensions
- Fonctions d'agrégation
- Cube = multi agrégation

Fenêtres

- Fonctions sur fenêtre
- Fenêtre partitionnée
- Fenêtre glissante

Agrégation :
group by ROLLUP
et
group by CUBE

Agrégations : problème


- Explosion du nombre de regroupements potentiels à calculer
 - Nombreuses dimensions
 - Dimensions avec nombreux niveaux
- Problème lié à l'exécution
 - Calculer efficacement un ensemble d'agrégations ?
 - Factoriser les calculs communs entre deux agrégations
- Problème lié au langage
 - Exprimer de manière déclarative un ensemble d'agrégations ?

Factoriser le calcul d'agrégats

Exemple

- Foot (numMatch, date, buts)
(m1, 25/12/2019, 3) (m2, 16/3/2020,2) (m3, 31/8/2020, 1)
(m4, 14/7/2019, 5) (m5, 4/7/2019, 2)
- Mois (jour, numMois) numMois de M1 à M24
- Année (numMois, année) de 2019 à 2020
- Nombre total de buts par année
 - 2019: 3+5+2=10 2020: 2+1=3
- Nombre total de buts: deux méthodes possibles
 - Partir des données initiales: 3+2+1+5+2=13
 - Partir de l'agrégat déjà calculé: 10+3=13
 - correct car la fonction d'agrégation **est associative**
 - $(v_{11} + v_{12} + v_{13} + v_{21} + v_{22}) = (v_{11} + v_{12} + v_{13}) + (v_{21} + v_{22})$
 - moins d'opérations donc plus efficace
- Rmq : factorisation possible des agrégations **non associatives** pouvant s'exprimer avec des fonctions associatives:
 - Exple exprimer avg() avec des sum()

Exprimer un ensemble d'agrégations : ROLLUP

- Rollup sur une seule dimension hiérarchique
- tables **Fait**(n1, mesure) et **Dimension** (n1,n2, ...)
- Requête

```
Select n1, n2, n3, ..., agrégation(...), ...  
From Fait f, Dimension d  
Where f.n1 = d.n1  
Group by ROLLUP (n1, n2, n3, ...)
```
- Le résultat contient plusieurs niveaux d'agrégation
 - Niveau général
 - Niveau n1
 - Niveau n1, n2
 - Niveau n1, n2, n3
 - ...

Exemple de ROLLUP (1)

```
select annee, mois, jour, count(*) as nb_visite  
from Visite v, DateDimension d  
where v.date = d.date  
group by rollup (annee, mois, jour)  
order by annee, mois, jour
```

Agrégation
sans group by

Group by **annee**

Group by **annee, mois**

annee	mois	jour	nb_visite
null	null	null	17
2019	null	null	6
2019	11	null	1
2019	11	11	1
2019	12	null	5
2019	12	25	4
2019	12	26	1
2020	null	null	11
2020	2	null	4
2020	2	1	3
2020	2	2	1
2020	9	null	6
2020	9	1	1
2020	9	3	1
2020	9	4	3

Exemple de ROLLUP (2)

Faits : VisiteGPS

photoID	personID	date	gps	note
p1	Bob	2020-09-03	▶{"lon": 41.5, "lat": 12.8}	3
p2	Alice	2020-09-01	▶{"lon": 41.6, "lat": 12.8}	5

Dimension : PlaceGPS

gps	ville	pays
▶{"lon": 41.5, "lat": 12.8}	Aix	France
▶{"lon": 41.6, "lat": 12.8}	Aix	France

select p.pays, p.ville, v.gps, count(*) as nb_visite
from VisiteGPS v, PlaceGPS p
where v.gps = p.gps
group by **rollup** (p.pays, p.ville, v.gps)
order by pays, ville, gps

pays	ville	gps	nb_visite
null	null	null	17
France	null	null	14
France	Aix	null	2
France	Aix	▶{"lon": 41.5, "lat": 12.8}	1
France	Aix	▶{"lon": 41.6, "lat": 12.8}	1
France	Lille	null	3

Toutes les visites

Visites par pays

Visites par villes d'un pays

Exprimer un ensemble d'agrégations : CUBE

- Group by **CUBE** (d1,d2,d3, ...)
 - table Fait(d1,d2,d3,m). Cubes sur des dimensions différentes.
 - Agrégation totale
 - Agrégats sur chaque dimension : d1 et d2 et d3
 - Agrégats sur chaque paire de dimensions :
(d1, d2) et (d1, d3) et (d2, d3)
 - Agrégat sur chaque triplet de dimensions : (d1, d2, d3)
 - Total : $2^3 = 8$ agrégats
- 2^n regroupements pour n dimensions ...

Exemple de CUBE

Select
annee, pays, profession,
round(avg(note), 1) as noteMoyenne,
count(*) as nbVisite
From VisitesDetail
Group by cube (annee, pays, profession)
Order by annee, pays, profession

Visites par pays

Visites par profession

Visites par ...

...

annee	pays	profession	noteMoyenne	nbVisite
null	null	null	3.1	17
null	null	Architecte	3.5	8
null	null	Data science	2.3	7
null	null	Vendeuse	4	2
null	France	null	3.4	14
null	France	Architecte	3.7	7
null	France	Data science	2.6	5
null	France	Vendeuse	4	2
null	Italie	null	1	1
null	Italie	Data science	1	1
null	Norvege	null	2	2
null	Norvege	Architecte	2	1
null	Norvege	Data science	2	1
2019	null	null	2.7	6
2019	null	Architecte	2.5	2
2019	null	Data science	1.5	2
2019	null	Vendeuse	4	2
2019	France	null	2.7	6
2019	France	Architecte	2.5	2
2019	France	Data science	1.5	2
2019	France	Vendeuse	4	2

Rollup et Cube : valeurs nulles

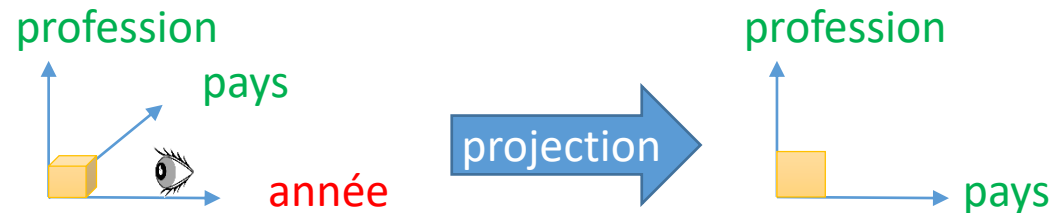
- Comment réunir plusieurs agrégations dans une même relation ?
- Présence de valeurs nulles
 - Seules les dimensions qui définissent le regroupement sont renseignées
 - Les dimensions « inutilisées » ont une valeur nulle
- Requête : R(année, mois, nb) =
Select année, mois, sum(buts) as nb
From Foot
Group by **ROLLUP** (année, mois)
- Résultat : R contient
(2019, M7, 7) (2019, M12, 3) (2020, M15, 2) (2020, M20,1)
(2019, NULL, 10) (2020, NULL, 3) (NULL, NULL, 13)
- Réciproquement : Sélectionner une agrégation parmi celles réunies dans le résultat d'une requête CUBE ou ROLLUP ?
Exple: select * from R where année IS NOT NULL and mois IS NULL
Ou where grouping(année) = 0 and grouping(mois) = 1

Opérations sur le cube

- Manipulation *multi-dimensionnelle* des agrégats
- Approche algébrique
 - Opération : cube \rightarrow cube
 - Composition d'opérations
- Principales opérations
 - Projection agrégative
 - Slice
 - Dice
 - Drill down
 - Rollup
- Ces opérations sont abstraites
 - doivent être traduites en SQL

Opération : Projection agrégative

- Sélectionner certaines dimensions du cube



- Exemple

select profession, pays, notemoyenne, nbVisite
from Cube1 c

where **profession is not null**

and **pays is not null**

and **annee is null**

order by profession, pays

profession ▲	pays ▲	notemoyenne ▲	nbVisite ▲
Architecte	France	3.7	7
Architecte	Norvege	2	1
Data science	France	2.6	5
Data science	Italie	1	1
Data science	Norvege	2	1
Vendeuse	France	4	2

Opération Dice

- Critère de sélection sur les mesures

- Exemple

select *

from Cube1

where noteMoyenne >= 3.5 and nbVisite < 8

annee ▲	pays ▲	profession ▲	noteMoyenne ▲	nbVisite ▲
null	null	Vendeuse	4	2
null	France	Architecte	3.7	7
null	France	Vendeuse	4	2
2019	null	Vendeuse	4	2
2019	France	Vendeuse	4	2
2020	null	Architecte	3.8	6
2020	France	Architecte	4.2	5

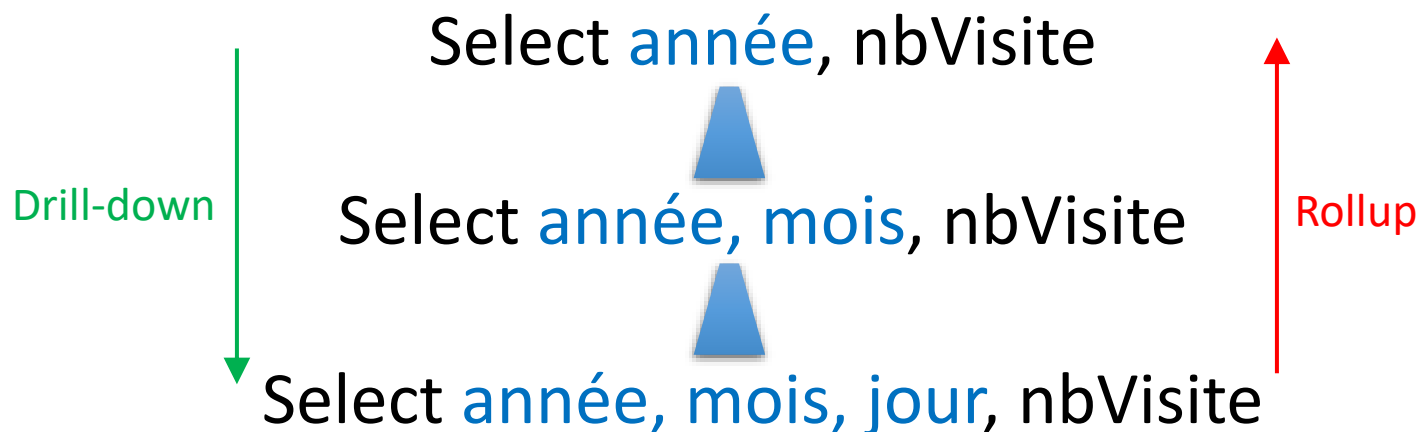
Opération Slice

- Critère de sélection sur des **dimensions**
- Exemple
select *
from Cube1
where **pays** in ('Norvege', 'France')
and **annee** > 2018

annee ▲	pays ▲	profession ▲	noteMoyenne ▲	nbVisite
2019	France	null	2.7	6
2019	France	Architecte	2.5	2
2019	France	Data science	1.5	2
2019	France	Vendeuse	4	2
2020	France	null	3.9	8
2020	France	Architecte	4.2	5
2020	France	Data science	3.3	3
2020	Norvege	null	2	2

Opérations drill-down / rollup

- Drill-down
 - Zoom **in** : descend dans les niveaux des dimensions
- Rollup
 - Zoom **out** : monte dans les niveaux des dimensions
- Exemple pour la dimension Date



Requêtes sur les agrégats

- Situation
 - Un ensemble d'agrégats pré-calculés est stocké : vues matérialisées
 - Une requête d'agrégation est posée
 - Les agrégats exprimés dans la requête sont différents des agrégats pré-calculés
- Problème
 - Quels agrégats matérialisés faut-il accéder
 - pour produire le résultat de la requête plus rapidement qu'en recalculant tous les agrégats exprimés dans la requête ?
- Solution
 - Déterminer les agrégats qui contiennent une partie de la requête
 - Exple « Vente par an » dans « vente par mois »
 - Reformuler la requête
 - requête équivalente posée sur des agrégats et sur des tables si nécessaire
 - Plusieurs alternatives --> optimisation

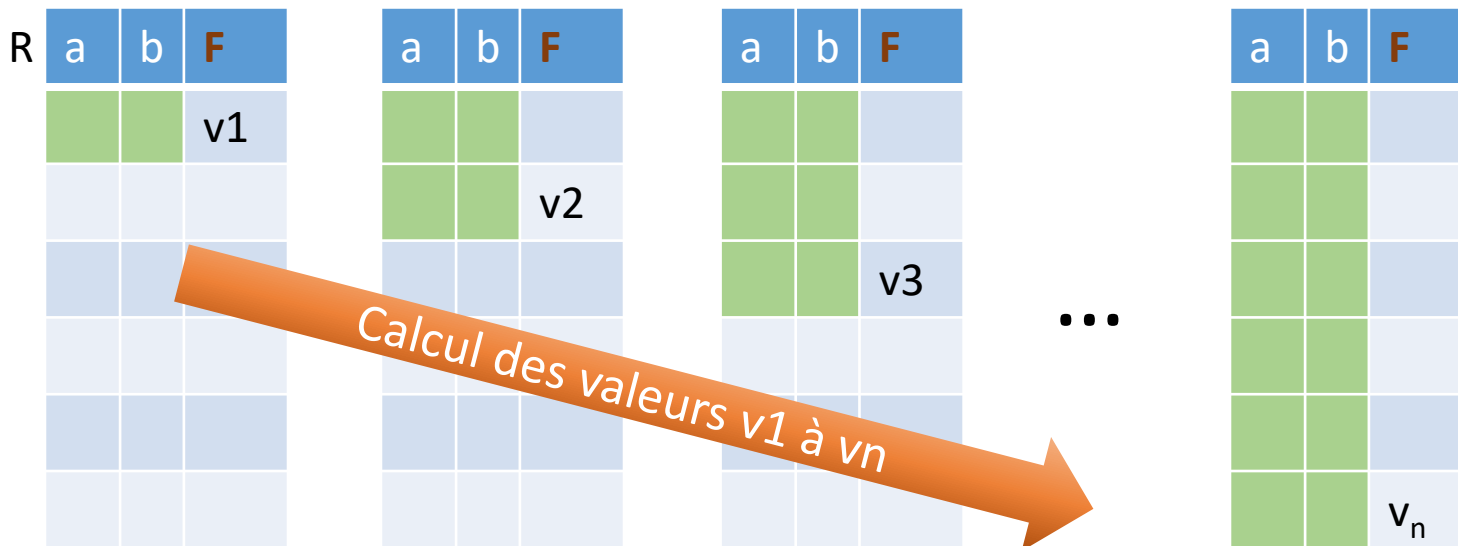
Maintenir les agrégats

- Contexte: dynamicité
 - Des nouveaux faits sont insérés
- Problèmes
 - Quels agrégats sont impactés par les nouvelles données
 - Comment recalculer efficacement ces agrégats ?
- Contrôle : régler le compromis efficacité/précision
 - Quand recalculer les agrégats
 - Niveau de tolérance sur l'obsolescence des agrégats
- Tenir compte de la charge : workload aware
 - Requêtes fréquentes/régulières : exple dashboard
 - Adapter l'ensemble d'agrégats maintenus en fonction des requêtes demandées

Fenêtrage: OVER

Fenêtre

- Fenêtre = contexte d'un tuple = **OVER**
 - On considère une liste ordonnées de tuples : $R(a,b)$
 - Le **contexte** d'un tuple contient les tuples qui le précèdent (ou le succèdent)



Fonctions de fenêtrage : OVER

- Fonction de fenêtrage
 - F: (tuple avec sa fenêtre) --> agrégat
- Fonctions usuelles : min, max, sum, count, rank, lag,
- Exemple
 - Table **Dépense** (date, montant)
 - Somme **cumulée** des dépenses ?

```
SELECT sum(montant) OVER (order by date) as cumul  
FROM Dépense
```


Numérotation : row_number() OVER

- Attribuer un numéro 1,2,... aux tuples
- Exemple

Select v.personId, v.photoId, v.note,

row_number() **over** (order by personId, photoId) as numeroPhoto

From Visite v

personId ▲	photoId ▲	note ▲	numeroPhoto ▲
Alice	p12	3	1
Alice	p16	4	2
Alice	p17	5	3
Alice	p2	5	4
Alice	p5	2	5
Alice	p6	4	6
Alice	p7	4	7
Bob	p1	3	8
Bob	p13	5	9

Classement : rank() OVER

- Attribuer un rang à des données triées. Exemple :
select v.personId, v.photoId, v.note,
rank() **over** (order by note desc) as rang
from Visite v
- Le rang contient des ex aequos
-> Valeurs **non consécutives** du rang

personId	photoId	note	rang
Alice	p2	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p17	5	1
Alice	p6	4	5
Alice	p7	4	5
Alice	p16	4	5
Bob	p1	3	8

« pas de rang=2 »

Classement et top-K

- Attribuer un rang + **sélection rang $\leq k$**
- Exemple : top 5 des visites

With Classement as (

Select v.personId, v.photoId, v.note,
rank() over (order by note desc) as rang

From Visite v

)

Select * from Classement

where **rang ≤ 5**

order by rang

personId	photoId	note	rang
Alice	p2	5	1
Alice	p17	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p6	4	5
Alice	p7	4	5
Alice	p16	4	5

Le top5 contient **7** tuples

Classement normalisé

- `percent_rank()`
 - Valeur décimale dans $]0, 1]$

Classement : dense_rank() OVER

- Le rang dense prend des valeurs consécutives
 - malgré les ex aequos

- Exemple

```
select v.personId, v.photoId, v.note,  
       dense_rank() over (order by note desc) as rang  
from Visite v
```

personId ▲	photoId ▲	note ▲	rang_dense ▲
Alice	p2	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p17	5	1
Alice	p6	4	2
Alice	p7	4	2
Alice	p16	4	2
Bob	p1	3	3

Fenêtrage et partitionnement : PARTITION BY

- Objectif : traitement **multi-fenêtres**
 - Partitionner le résultat d'une requête
 - Une partition contient tous les tuples ayant la même valeur pour att_1, \dots, att_n
 - Une fenêtre par partition
 - Deux fenêtres dans deux partitions sont donc disjointes
- Syntaxe

```
SELECT ... OVER ( PARTITION BY att1, ..., attn )  
FROM ...
```

Exemple: partition by

VisitesDetail (photoID, personID, ville, note)

- Le classement **par ville** des visites les mieux notées

```
SELECT v.photoID, v.personID, v.ville,  
       rank() OVER( PARTITION BY ville order by note) as rang  
FROM VisitesDetail v  
order by ville, rang
```

photoID	personID	ville	rang
p1	Bob	Aix	1
p2	Alice	Aix	2
p14	Carole	Lille	1
p12	Alice	Lille	2
p13	Bob	Lille	3
p7	Alice	Marseille	1
p6	Alice	Nice	1
p4	Bob	Oslo	1
p5	Alice	Oslo	1
p9	David	Paris	1

Exemple : Partition by dans une requête d'agrégation

- Le classement **par pays** des villes ayant le plus de visites
 - Calculer le nombre de visites dans chaque ville
 - **group by pays, ville** pour obtenir des tuples (pays, ville, nbVisite)
 - Définir une fenêtre pour chaque pays:
 - **partition by pays** : le tri est local à un pays

Select **pays, ville, count(*) as nbVisite,**
rank() over (**partition by pays** order by count(*) desc) as rang

From VisitesDetail v

Group by pays, ville

Order by pays, rang

pays	ville	nbVisite	rang
France	Paris	4	1
France	Lille	3	2
France	Pau	2	3
France	Aix	2	3
France	StDenis	1	5
France	Marseille	1	5
France	Nice	1	5
Italie	Rome	1	1
Norvege	Oslo	2	1

Fenêtre croissante

- taille allant du 1^{er} tuple au tuple courant
 - 1^{er} tuple : UNBOUNDED PRECEDING
 - tuple courant : CURRENT ROW
- Exemple
Select sum(montant) OVER(ORDER BY date ROWS
BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW) as somme
From Visite

Fenêtre glissante : ROWS

- Définie par un **nombre de tuples** avant/après le tuple courant
 - ROWS between **n** preceding and **m** following
 - ROWS between **n** preceding and current row (pour $m=0$)
 - ROWS between current row and **m** following (pour $n=0$)
- Taille d'une fenêtre glissante ROWS
 - Taille fixe = $n+m+1$, sauf pour
 - les n premiers tuples
 - les m derniers tuples

Fenetre glissante: exemple

```
SELECT points,  
       SUM(points) OVER (  
         ROWS BETWEEN 1 PRECEDING  
                     AND 1 FOLLOWING) we  
FROM results
```

This query computes the `SUM` of each point and the points on either side of it:

POINTS	WE
10	18
8	30
12	29
9	30
9	27
9	23
5	21
7	12

POINTS	WE
10	18
8	30
12	29
9	30
9	27
9	23
5	21
7	12

POINTS	WE
10	18
8	30
12	29
9	30
9	27
9	23
5	21
7	12

POINTS	WE
10	18
8	30
12	29
9	30
9	27
9	23
5	21
7	12

Notice that at the edge of the partition, there are only two values added together. This is because frames are cropped to the edge of the partition.

Fenêtre glissante : RANGE

- Définie par un **écart de valeur** avant/après la valeur courante de l'attribut de tri
 - RANGE between **n** preceding and **m** following
- La taille d'une fenêtre range peut varier
 - On considère le tri sur l'attribut att : order by att
 - Taille = nombre de tuples dans l'intervalle [att-n, att+m]

Efficacité des analyses

- Agrégation
 - Group by : calcul indépendant dans chaque groupe
 - Traité en //
 - Cube et Rollup permettent de factoriser les agrégats
- Fenêtre :
 - Partition by : calcul indépendant dans chaque partition
 - Traité en //
 - Fenêtre glissante : calcul incrémental
 - réutiliser les résultats sur la partie commune entre deux fenêtres successives

Perspectives

Perspectives académiques

- Article et prototype LMFAO
 - Layered Multiple Functional Aggregate Optimization
- Article DIFF
- Article EDBT 2020
 - Sharing Computations for User-Defined Aggregate Functions
 - Chao Zhang (prix thèse BDA 2020), Farouk Toumani
 - https://openproceedings.org/2020/conf/edbt/paper_120.pdf
 - Regarder la video de présentation

Perspective industrielle

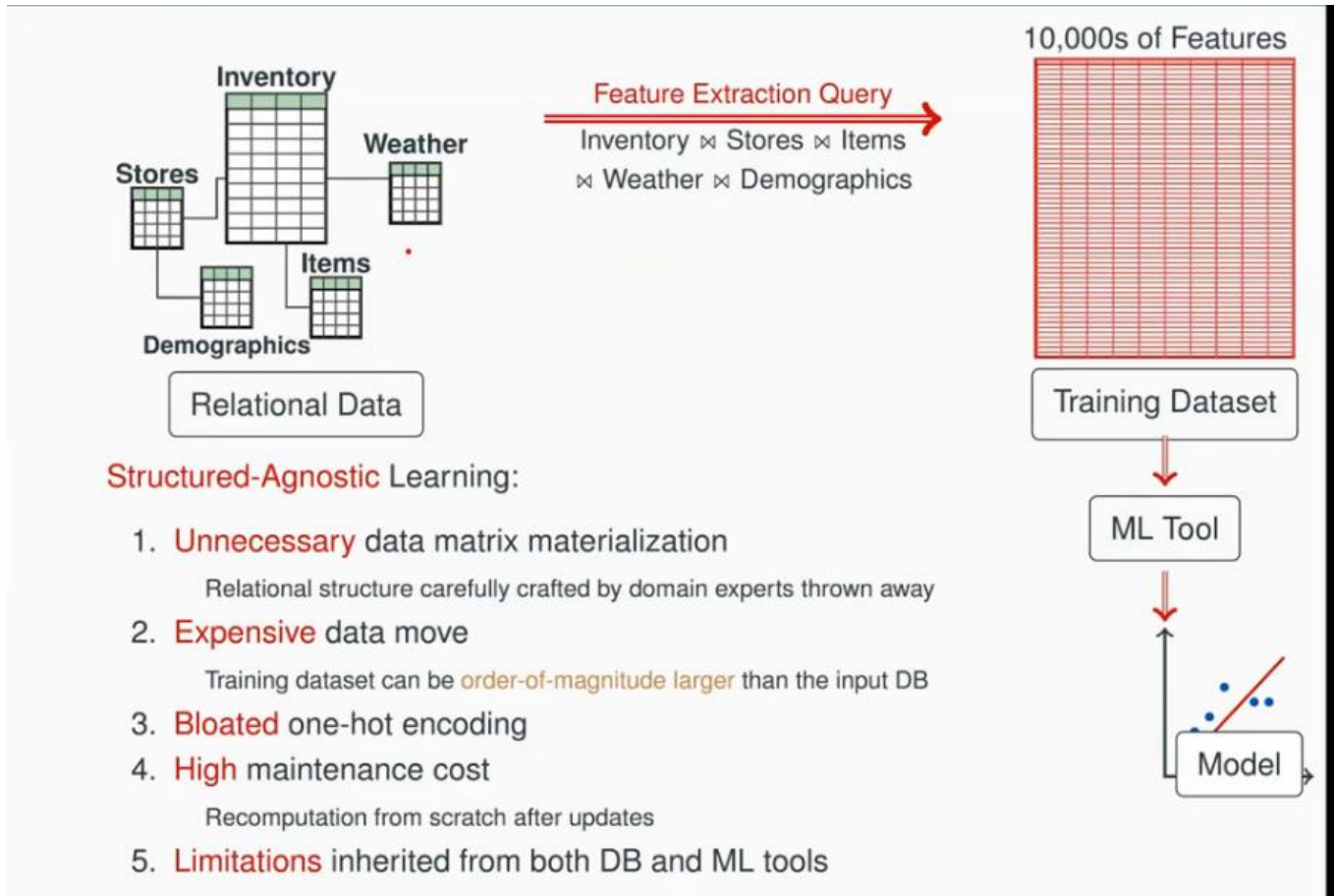
- Startup Indexima

Requêtes analytiques pour du ML plus performant : LMFAO

- A Layered Aggregate Engine for Analytics Workloads
 - SIGMOD 2019 : D. Olteanu à l'Univ Oxford, Relational AI
 - PDF: <https://nuage.lip6.fr/s/2kaRRtPNGKqJZj2>
 - VLDB 2020 : Relational data borg is learning
 - <http://www.vldb.org/pvldb/vol13/p3502-olteanu.pdf>
- Accélérer l'entraînement d'un modèle
 - S'applique pour les arbres de décision, la régression, ...
 - Reformuler la fonction à évaluer par des requêtes d'agrégation
- Solution ad-hoc
 - représenter les données en mémoire
 - optimiser le calcul d'un lot d'agrégats
- Performance
 - bcp + rapide qu'un SGBD en mémoire ou qu'en python (scikit) ou R

LMFAO : Contexte et Défis

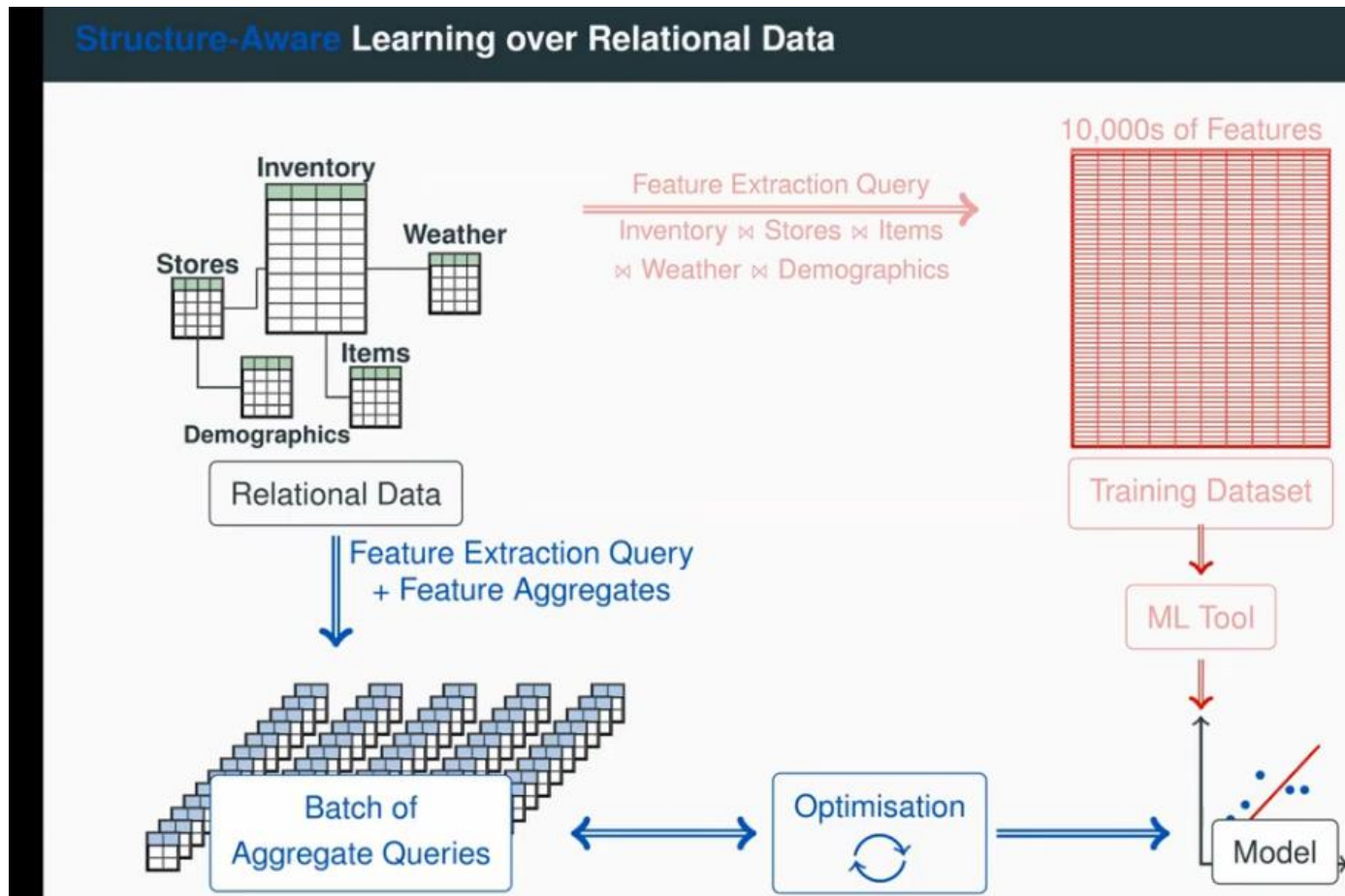
Layered Multiple Functional Aggregate Optimization



Source : <https://www.youtube.com/watch?v=0ic0jMjOpM0>

LMFAO : ML basé sur des agrégats

Un modèle ML est calculé à l'aide d'agrégats



LMFAO : Application à des modèles « classiques » en ML

Workload	Query Batch
Linear Regression	$\text{SUM}(X_i * X_j)$ [WHERE $\sum_k X_k * w_k < c$]
Covariance Matrix	$\text{SUM}(X_i)$ GROUP BY X_j [WHERE ...]
(Non)poly. loss	$\text{SUM}(1)$ GROUP BY X_i, X_j [WHERE ...]
Decision Tree Node	$\text{VARIANCE}(Y)$ WHERE $X_j = c_j$
Mutual Information	$\text{SUM}(1)$ GROUP BY X_i
Rk-means	$\text{SUM}(1)$ GROUP BY X_j $\text{SUM}(1)$ GROUP BY $\text{Center}_1, \dots, \text{Center}_k$



LMFAO : performance

Structure-aware versus Structure-agnostic Learning

Train a linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow		Our approach (SIGMOD'19)	
	Time	Size (CSV)	Time	Size (CSV)
Database	—	2.1 GB	—	2.1 GB
Join	152.06 secs	23 GB	—	—
Export	351.76 secs	23 GB	—	—
Shuffling	5,488.73 secs	23 GB	—	—
Query batch	—	—	6.08 secs	37 KB
Grad Descent	7,249.58 secs	—	0.05 secs	—
Total time	13,242.13 secs		6.13 secs	

2,160× faster while being more accurate (RMSE on 2% test data)

LMFAO : autres modèles considérés

So far:

- Polynomial regression
- Factorisation machines
- Classification/regression trees
- Mutual information
- Chow Liu trees
- k -means clustering
- k -nearest neighbours
- (robust, ordinal) PCA
- SVM

On-going:

- Boosting regression trees
- AdaBoost
- Sum-product networks
- Random forests
- Logistic regression
- Linear algebra:
 - QR decomposition
 - SVD
 - low-rank matrix factorisation

All these cases can benefit from **structure-aware computation**

Perspectives : DIFF

- DIFF: A Relational Interface for LargeScale Data Explanation, VLDB 2018
 - <http://www.vldb.org/pvldb/vol12/p419-abuzaid.pdf>

Motivation de DIFF

- Besoin de comprendre et **expliquer** des données
 - Données dénormalisées = une seule relation
 - Exple UsageApp(date, os, modeleTel, état)
 - Mettre en évidence des caractéristiques communes
 - Caractéristique : attribut = valeur
 - Exple: tel = « S20 », état=« panne »
 - Ensemble de paires (attribut, valeur)
 - [modeleTel = «S20», os = «9.1»] est 3 fois plus fréquent pour l'état « panne» que pour l'état «sans panne»
 - Nombreux cas d'usage réels
 - Détection de signaux faibles dans les données
 - Microsoft: mise à jour win10, Facebook: surveillance appli, Censys: cyber attaques
- Limites des solutions existantes
 - Pas SQL, donc difficile à intégrer dans un workflow d'analyse
 - Centralisée: ne peut analyser qu'une faible portion des données

Opérateur DIFF

- But: exprimer une requête d'*explication*
 - *Déterminer des ensembles fréquents*
 - *qui satisfont une condition portant sur 2 bases*
- Syntaxe déclarative
 - *base1* **DIFF** *base2*
 - **ON** *attributs*
 - **COMPARE BY** *condition*
 - **MAX ORDER** *n*
- Condition
 - *risk_ratio*: taux élevé si l'ensemble est fréquent dans la base 1 et plus fréquent dans la base 1 que dans la base 2
 - $(\text{occ base1} / \text{occ totale}) / (\text{non occ base1} / \text{non occ totale})$
 - *support* : fréquence d'occurrence dans une base
 - *odds_ratio*
 - UDF
- Max order *n* : nombre max d'attributs d'une explication

DIFF: optimisation

- Optimisations logiques
 - Réordonner les opérations :
 - Appliquer les filtres COMPARE BY avant de traiter les jointures
 - Tenir compte des dépendances fonctionnelles
 - Réduit le nombre de combinaisons d'attributs à considérer
- Optimisations physiques
 - Indexation
 - Estimation du coût d'utilisation d'un index
 - Indexer seulement si faible coût

Perspective : Indexima

- Cas de reporting sur des très grands volumes
 - Taille $> 10^9$ tuples
- Besoin de performance
 - évaluer une requête sans accéder aux données mais seulement aux index
- Index dédiés au calcul d'agrégats
 - Index multi attributs
 - plusieurs fonctions précalculées : min,max,sum,count,...
 - Contient le domaine d'un attr pour évaluer count distinct
- Indexation
 - Exploiter l'historique des requêtes
 - Indexer les agrégats fréquents
 - Index commun à plusieurs agrégats
 - Exploiter la sémantique des attributs
- Disponibilité : solution distribuée sur cluster
- Dynamicité : nouvelles données --> mise à jour des index en background