Analyse des données

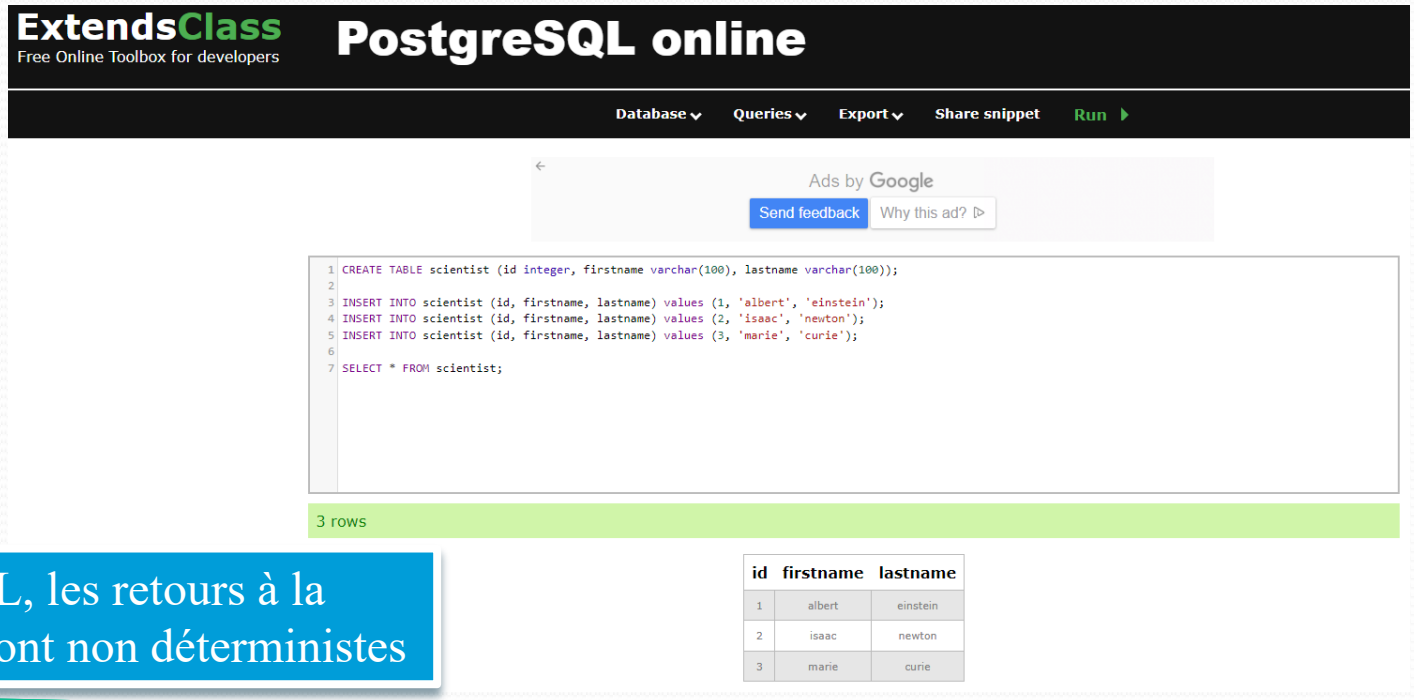
CM2 : Skyline

Mickaël Martin Nevot

V1.2.0

Rappel : Pour tester

- Interpréteur en ligne : https://extendsclass.com/postgresql-online.html
- Documentation : https://www.postgresql.org/docs/current



En SQL, les retours à la ligne sont non déterministes

Rappel : Existence

- EXISTS : vrai si ensemble non nul

```
-- Quels sont les étudiants n'ayant réalisé aucun stage en entreprise ?
SELECT ide
FROM Etudiant
WHERE NOT EXISTS (
        SELECT * FROM Convention
        -- ou SELECT ide FROM Convention
        WHERE Convention.ide = Etudiant.ide);
```

Rappel : Table commune

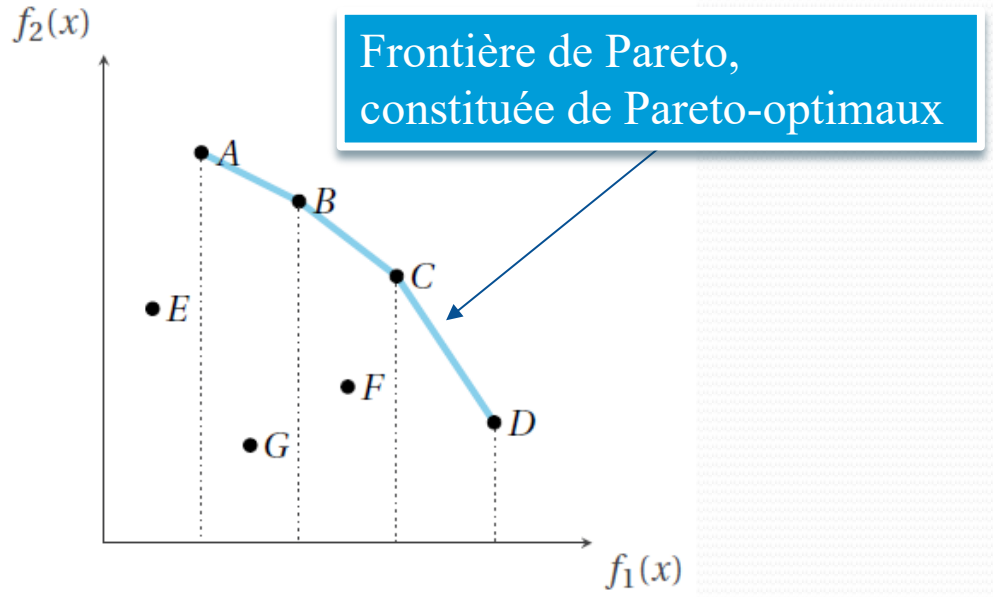
- **Relation précalculée** avant la requête principale

```
WITH
T1 (nom, nb_ade) AS (
     SELECT nom, COUNT(DISTINCT adresse) AS nb_ade
     FROM Etudiant
     GROUP BY nom
),
T2 (nom, nb_ads) AS (
     SELECT P.nom, COUNT(DISTINCT adresse) AS nb_ads
     FROM Personnel P INNER JOIN Societe S
       ON P.ids = S.ids
     GROUP BY P.nom
)
SELECT T1.nom, nb_ade, nb_ads
FROM T1, T2
WHERE T1.nom = T2.nom;
```

Plusieurs tables communes peuvent être imbriquées

Analyse multicritère

- Opérateur Skyline
  (dominance entre tuples, dite de Pareto)
- Permet d'avoir les meilleurs résultats
  (même quand il n'en existe pas d'optimaux)
- Permet une analyse
  efficace de préférences
- Préférences Skyline :
  - MIN
  - MAX



Opérateur Skyline

```
-- Requête avec l'opérateur Skyline.
SELECT * FROM Personnel
SKYLINE OF salaire MIN, anciennete MAX;
```

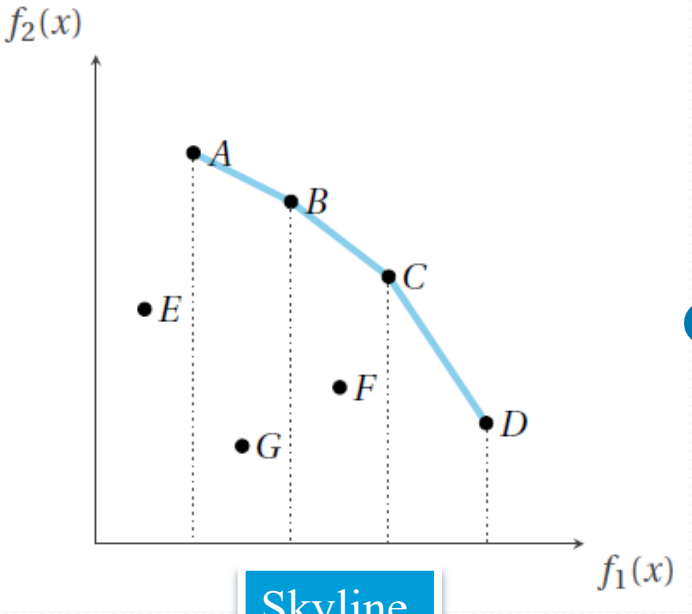
Préférence Skyline : MIN

Préférence Skyline : MAX

```
-- Requête équivalente si l'opérateur Skyline n'est pas implémenté pas dans le SGBD.
SELECT * FROM Personnel AS P1
    WHERE NOT EXISTS (
        SELECT * FROM Personnel AS P2
        WHERE (P2.salaire <= P1.salaire
          AND P2.anciennete >= P1.anciennete)
          AND (P2.salaire < P1.salaire
           OR P2.anciennete > P1.anciennete));
```

idp	nom	prenom	ids	salaire	anciennete
4	Dupond	Emile	8	2420	20
19	Petitjean	Helene	21	1750	12

Skycube



Skyline

Cube de données

Skycube

Table des faits récapitulative

RowId	Type	Propriété	...	Parangon	Rareté	Gemmes	Succès [a]
1	*Féticheur*	*Vitesse*	...	600	5	30	3
2	*Féticheur*	*Vitesse*	...	900	5	85	7
3	*Sorcière*	*Chance*	...	600	10	50	7
4	*Sorcière*	*Vitesse*	...	100	10	85	5
5	*Croisé*	*Chance*	...	900	10	50	7

a. * Par milliers, arrondi au nombre inférieur. Par exemple 5 équivaux à environ 5000.

Préférences Skyline :
- Parangon : MAX
- Rareté : MAX
- Gemmes : MAX
- Succès : MAX

Jeu vidéo Diablo III : Reaper of Souls

Treillis de Skycube

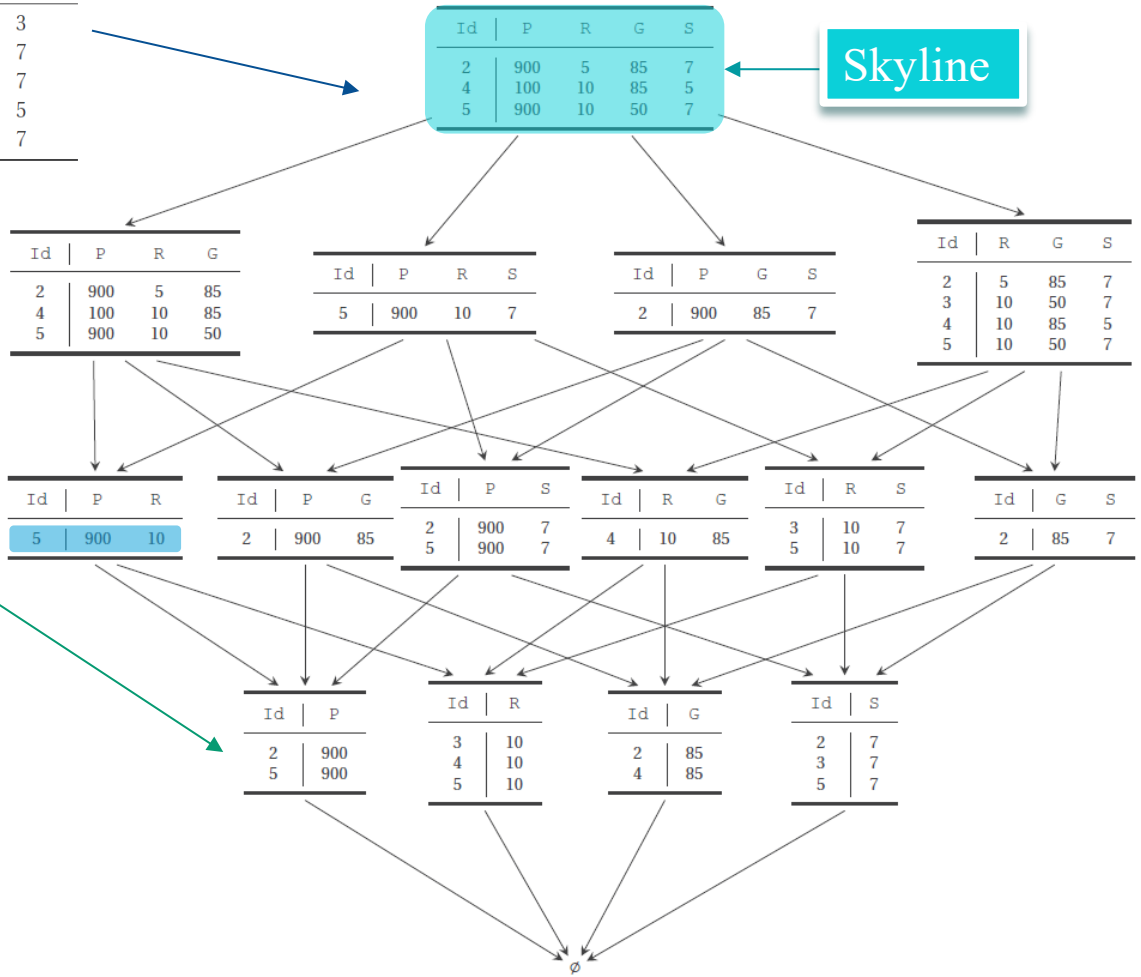
RowId	Type	Propriété	...	Parangon	Rareté	Gemmes	Succès [a]
1	*Féticheur*	*Vitesse*	...	600	5	30	3
2	*Féticheur*	*Vitesse*	...	900	5	85	7
3	*Sorcière*	*Chance*	...	600	10	50	7
4	*Sorcière*	*Vitesse*	...	100	10	85	5
5	*Croisé*	*Chance*	...	900	10	50	7

a. Par milliers, arrondi au nombre inférieur. Par exemple 5 équivaux à environ 5000.

Préférences Skyline : MAX

Skyline

Skycuboïdes

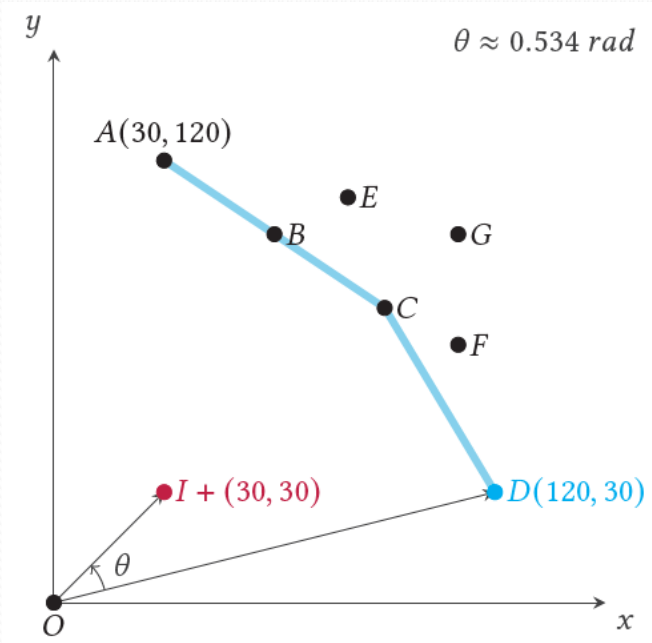


Classement de Skyline : CoSky

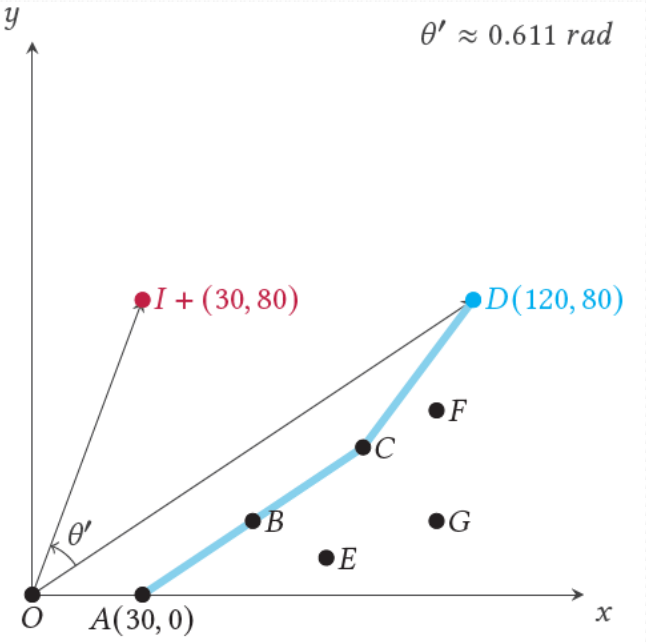
- Méthode de type TOPSIS
- Étapes de CoSky :

Pour cosinus (de Salton) Skyline

I. Préparation : unification des préférences Skyline



Préférences : (MIN, MIN)



Préférences : (MIN, MAX)

Classement de Skyline : CoSky

- Étapes de CoSky : ← Faisable avec une requête SQL

1. Normalisation des attributs par la somme (échelle de 0 à 1)
2. Pondération avec indice de Gini (poids des attributs)
3. Détermination du point idéal
4. Calcul des scores avec le cosinus de Salton (avec angle entre le point et le point idéal)
5. Classement des résultats (ordonnancement)

id	parangon	rarete	gemmes	succes	score
2	900	5	85	7	0.978
5	900	10	50	7	0.976
4	100	10	85	5	0.914

CoSky

En SQL

```
WITH S AS (SELECT * FROM Diablo
           SKYLINE OF parangon MAX, rarete MAX, gemmes MAX, succès MAX
), SN AS (SELECT id,
                 parangon / tParangon AS nParangon,
                 rarete / tRarete AS nRarete,
                 gemmes / tGemmes AS nGemmes,
                 succes / tSucces AS nSucces
          FROM S, (SELECT SUM(parangon) AS tParangon,
                          SUM(rarete) AS tRarete,
                          SUM(gemmes) AS tGemmes,
                          SUM(succes) AS tSucces FROM S) AS ST
), SGini AS (SELECT 1 - SUM(nParangon * nParangon) AS giniParangon,
                    1 - SUM(nRarete * nRarete) AS giniRarete,
                    1 - SUM(nGemmes * nGemmes) AS giniGemmes,
                    1 - SUM(nSucces * nSucces) AS giniSucces
             FROM SN
), SW AS (SELECT giniParangon / COALESCE(NULLIF(giniParangon + giniRarete + giniGemmes + giniSucces, 0), 1) AS wParangon,
                 giniRarete / COALESCE(NULLIF(giniParangon + giniRarete + giniGemmes + giniSucces, 0), 1) AS wRarete,
                 giniGemmes / COALESCE(NULLIF(giniParangon + giniRarete + giniGemmes, 0), 1) AS wGemmes,
                 giniSucces / COALESCE(NULLIF(giniParangon + giniRarete + giniSucces, 0), 1) AS wSucces
          FROM SGini
), SP AS (SELECT id,
                 wParangon * nParangon AS pParangon,
                 wRarete * nRarete AS pRarete,
                 wGemmes * nGemmes AS pGemmes,
                 wSucces * nSucces AS pSucces
          FROM SN, SW
), Ideal AS (SELECT MAX(pParangon) AS iParangon,
                    MAX(pRarete) AS iRarete,
                    MAX(pGemmes) AS iGemmes,
                    MAX(pSucces) AS iSucces
             FROM SP
), SScore AS (SELECT id,
                     (iParangon * pParangon + iRarete * pRarete + iGemmes * pGemmes + iSucces * pSucces) /
                     COALESCE(NULLIF(SQRT(pParangon * pParangon + pRarete * pRarete + pGemmes * pGemmes + pSucces * pSucces) *
                     SQRT(iParangon * iParangon + iRarete * iRarete + iGemmes * iGemmes + iSucces * iSucces), 0), 1) AS score
              FROM Ideal, SP
)
SELECT P.id AS id, parangon, rarete, gemmes, succes, CAST(score AS DECIMAL(4,3)) AS score
FROM S P INNER JOIN SScore rs ON P.id = rs.id ORDER BY score DESC;
```

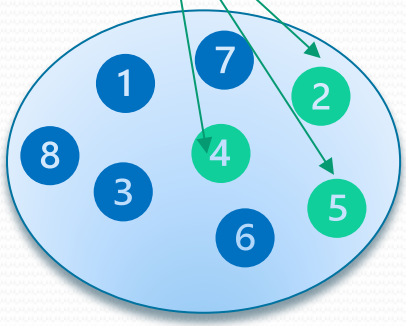
Si cardinalité du Skyline égale à 1

DeepSky

- Algorithme couplant les principes de :
  - CoSky (pour l'ordonnancement)
  - Skyline multiniveaux

Méthode permettant de trouver les *top-k* points non ordonnés entre eux

Skyline ordonné : 2, 5 et 4



Si $k = 2$: 2 et 5 sont retournés

Si $k = 3$: 2, 5 et 4 sont retournés

Si $k = 4$: 2, 5 et 4 sont ajoutés à la réponse, et en supprimant ces points, un nouveau Skyline ordonné est calculé, et ainsi de suite, jusqu'à ce qu'au plus 1 autre point ait été trouvé

Aller plus loin

- Cube de données quotient
- Cube de données partition
- Cube de données hiérarchique
- Calcul de cube de données
- Fermeture du cube de données
- Calcul de Skycube
- PageRank
- Skycube émergent
- Méthode TAGED

Liens

- Document classique :
  - Mickaël Martin Nevot. *Extraction multidimensionnelle et multicritère de données émergentes pour l'équilibrage de jeux vidéo*.
  - Jim Gray. *Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total*.
  - Kevin Beyer. *Bottom-Up Computation of Sparse and Iceberg CUBEs*.
  - Stephan Börzsönyi. *The Skyline Operator*.

Liens

- Document classique :
  - Yidong Yuan. *Efficient Computation of the Skyline Cube*.
  - Young-Jou Lai. *TOPSIS for MODM*.
  - Timotheus Preisinger. *Looking for the Best, but not too Many of Them: Multi-Level and Top-k Skylines*.
  - Laurent Carmignac. *Programmation et administration des bases de données*.

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com