Problématique des larges graphes

Camelia Constantin – LIP6

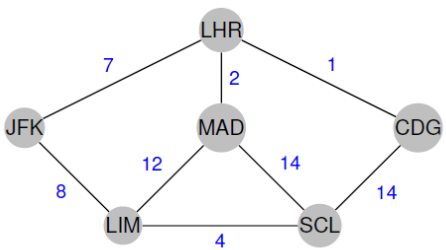
Motivation

Les graphes sont largement utilisés pour représenter les données dans de nombreuses applications, comme:

- Le transport
- Information géographique
- Les documents semi-structurés
- Les citations bibliographiques
- Les processus biologiques
- La représentation de la connaissance (Web sémantique)
- Les systèmes de workflows
- La provenance de données
- Les réseaux sociaux
- Les recommandations sur les sites de vente
- Les pages Web
- …

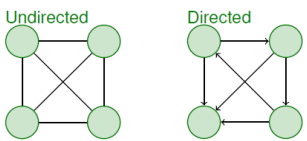
Exemple de graphe

Graphe des aéroports et durée de vol entre.



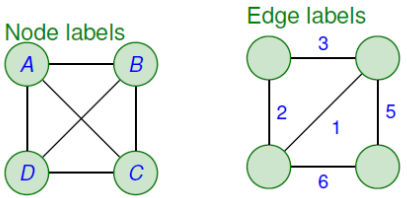
Types de graphe

Les graphes peuvent être dirigés (réseau social, workflow, références bibliographiques, etc) ou non (réseau routier, réseau des connaissances, etc)



Types d'étiquettes

Les nœuds et les arêtes peuvent être étiquettés (coût, classe, nature du lien, etc)



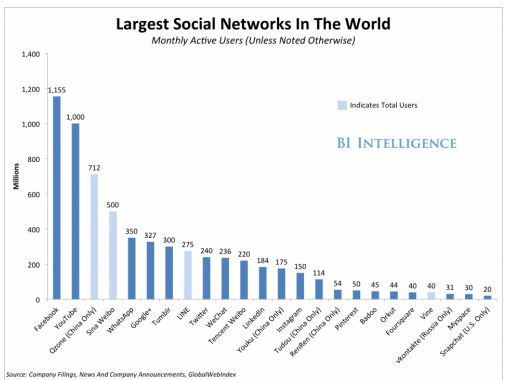
Exemples de requêtes

- Dans un réseau de type transport, alimentation, communication
  - Atteignabilité: puis-je aller de *a* à *b* ?
  - Court-chemin: trouver le plus court chemin (ou le moins cher, etc) entre *a* et *b*
- Dans un réseau de représentation de connaissance:
  - Une classe (élément) *A* est elle un sous-classe d'une classe *B*?
  - Existe-t'il un lien entre l'entité *A* et l'entité *B*?
  - Calcul de similarité entre l'entité *A* et l'entité *B* basé sur le graphe sémantique

Exemples de requêtes (suite)

- Pages Web et réseaux sociaux
  - Existence de chemins entre A et B (et nombre, longueur, coût, etc)
  - Recherche du plus court chemin
  - Requête sur le voisinage de A
  - Recommandations basées sur le graphe
  - Détection de cycles de longueur 2, 3, quelconque
  - Provenance d'une information
  - Détection de fraude pour score de recommandation
  - Calcul du diamètre du graphe
  - Calcul de la couverture d'un noeud
  - Etc etc etc

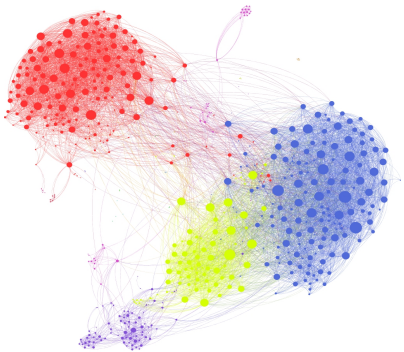
Réseaux sociaux: des graphes gigantesques



Quelques exemples (2014)

Facebook

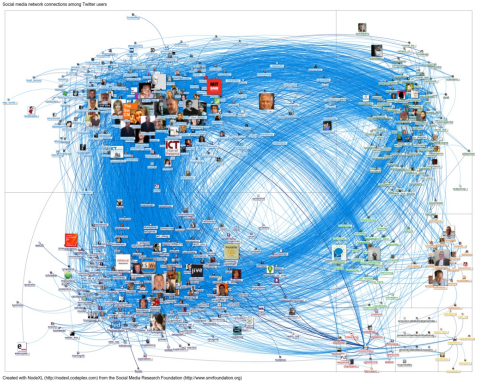
- 1.3 milliards d'utilisateurs actifs
- 400 milliards d'arcs dans le graphe



Quelques exemples (2014)

Twitter

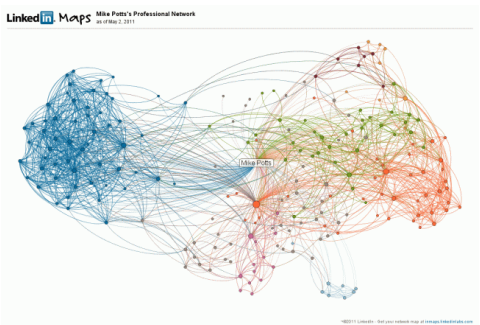
- 650 millions d'utilisateurs
- 130 milliards d'arcs dans le graphe



Quelques exemples (2014)

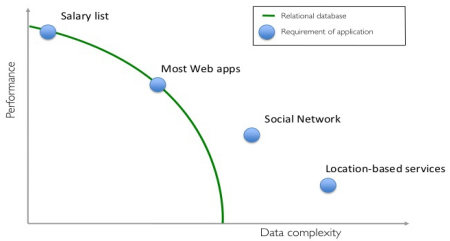
LinkedIn

- 330 millions d'utilisateurs
- Environ 63 milliards d'arcs



RDBMS Performance Curve

Side note: RDBMS performance



Représentation d'un graphe

Matrice d'adjacence

noeud	N1	N2	N3
N1	0	1	1
N2	0	0	0
N3	0	1	0

Ajout/modification/suppression d'un arc?

Ex: *Update matrice set N2=1 where nœud='N3'*

Ajout d'un nouvel utilisateur?

Alter table matrice ADD N4 boolean !!!!!

Représentation d'un graphe avec matrice

Quelle taille pour la matrice?

- Exemple LinkedIn (330 millions d'utilisateurs, 63 milliards d'arcs)
  - Avec un fichier (stockage bien, interrogation coûteuse surtout si pas en mémoire)
  - Avec une BD
- Exemple Facebook (1.3 milliards d'utilisateurs, 400 milliards d'arcs)
  - Avec un fichier
  - Avec une BD

Représentation d'un graphe avec matrice

Quelle taille pour la matrice?

- Exemple LinkedIn (330 millions d'utilisateurs, 63 milliards d'arcs)
  - Avec un fichier
    - Matrice de booléens de taille 330.10^6x330.10^6=108,9.10^15 bits
    - Soit… environ 12 Po !! -> sur disque
  - Avec une BD
    - Un tuple= 1 id de nœud (double), 330.10^6 bits
    - 330.10^6 entrées
    - Donc du même ordre!
    - Mais par contre mécanisme d'accès par index permettant d'accéder efficacement au disque pour interrogation, maj, etc

Représentation d'un graphe avec matrice

Quelle taille pour la matrice?

- Et Facebook (1.3 milliards d'utilisateurs, 400 milliards d'arcs)?
  - Avec un fichier / BD
    - Matrice de booléens de taille 1,3.10^9x1,3.10^9 = 1,69.10^18 bits
    - Soit… environ 1,5 Exaoctets !!

Représentation d'un graphe dans une BDR

Meilleure solution: Liste d'adjacence: table *edge et node*

compte	nom	email	…
N1	Jean	…	
N2	Lucie	…	
N3	Marc	…	

follower	followee
N1	N2
N1	N3
N2	N1
N2	N3
…	…

Ajout/modification/suppression d'un arc?

Ex: *Insert into edge values ('N3','N2')*

Ajout d'un nouvel utilisateur?

Insert into node values ('N4','Anne',…) → pas de LDD, juste du LMD

Représentation d'un graphe avec listes

Quelle taille pour les listes (avec BD)?

- Exemple LinkedIn (330 millions d'utilisateurs, 63 milliards d'arcs)
- Exemple Facebook (1.3 milliards d'utilisateurs, 400 milliards d'arcs)

Représentation d'un graphe avec listes

Quelle taille pour les listes (avec BD)?

- Exemple LinkedIn (330 millions d'utilisateurs, 63 milliards d'arcs)
  - Table des comptes: 330 millions de tuples de taille suivant info. stockée
  - Table d'adjacence: 63 milliards de tuples de taille 2xdouble, soit 1008 milliards d'octets, soit environ 940 Go
- Exemple Facebook (1.3 milliards d'utilisateurs, 400 milliards d'arcs)
  - Table d'adjacence d'environ 6 To

Exemple de requête

- Trouver les voisins directs :
  - Le nom des utilisateurs qui suivent 'Marc'?

Trouver voisins directs

Nom des utilisateurs qui suivent 'Marc'?

Select B.nom
From node A, node B, edge
where A.nom='Marc'
and A.compte=followee
and follower=B.compte

Trouver voisins directs

Nom des utilisateurs qui suivent 'Marc' ?

Et si le graphe est non dirigé?

… encore plus compliqué! (trouvez par vous-même)

Atteignabilité?

Liste des nœuds atteignables depuis le compte de 'Marc' ?

- Implique d'explorer le graphe depuis un nœud donné
- Avec une profondeur d'exploration fixée (à réaliser en TME)

La récursivité en SQL

4 possibilités:

- SQL2 : PL/SQL avec boucles et condition d'arrêt suivant la requête (pas de suivant, profondeur voulue, etc.)
- Requêtes hiérarchiques (clause CONNECT BY) dans Oracle7
- SQL3:Requêtes récursives (clause WITH) dans Oracle 11gR2
- DATALOG : modèle théorique base sur les clauses de Horn, des implantations mais pas de produits véritables

On va utiliser les 3 premières

Exemple : PL/SQL

Liste des ancêtres
EMP(EMPNO,NAME,VILLE,MGR)
Tous les supérieurs parisiens de Pierre?

```
DECLARE cur_emp EMP%ROWTYPE;
        cur_mgr EMP.EMPNO%TYPE;
BEGIN
SELECT MGR INTO cur_mgr FROM EMP
    WHERE NAME='Pierre' ;
WHILE (cur_mgr IS NOT NULL) LOOP
    SELECT * INTO cur_emp FROM EMP
    WHERE EMPNO=cur_mgr;
    IF(cur_emp.VILLE='Paris') THEN
        DBMS_OUTPUT.PUT_LINE(cur_emp.NAME);
    END IF;
    cur_mgr:=cur_emp.MGR;
END LOOP;
END;
/
```

Exemple: PL/SQL

Tous les subordonnés parisiens de Pierre?

Plus compliqué!

→ Utilisation de curseurs

Exemple: PL/SQL

Les subordonnés des subordonnés de Pierre?
EMP(EMPNO,NAME,VILLE,MGR)

```
DECLARE
CURSOR c1(cur_emp EMP.EMPNO%TYPE) is SELECT * FROM EMP WHERE MGR=cur_emp;
CURSOR c2 IS SELECT e1.EMPNO FROM EMP e1, EMP e2 WHERE e1.MGR=e2.EMPNO and E2.Name= 'Pierre' ;
sub_pierre c2%ROWTYPE;
sub_sub c1%ROWTYPE ;
BEGIN
   OPEN c2;
   FETCH c2 INTO sub_pierre;
   WHILE(c2%FOUND) LOOP
      OPEN c1(sub_pierre.EMPNO)
      FETCH c1 INTO sub_sub ;
      WHILE(c1%FOUND) LOOP
          DBMS_OUTPUT.PUT_LINE(sub_sub.NAME);
          FETCH c1 INTO sub_sub ;
      END LOOP ;
      FETCH c2 INTO sub_pierre;
   END LOOP ;
END ;
/
```

Requêtes hiérarchiques

```
SELECT select_list
FROM table_expression
[ WHERE ... ]
[ START WITH start_expression ]
CONNECT BY [NOCYCLE] { PRIOR parent_expr =
    child_expr | child_expr = PRIOR parent_expr }
[ ORDER SIBLINGS BY column1 [ ASC | DESC ]
    [, column2 [ ASC | DESC ] ] ...
[ GROUP BY ... ]
[ HAVING ... ]
...
```

Requêtes hiérarchiques

- START WITH indique le noeud de départ
- CONNECT BY PRIOR : règle de connexion entre les nœuds, spécifie la relation entre les tuples parent/enfant dans la hiérarchie.
- WHERE supprime les tuples de la hiérarchie qui ne satisfont pas la condition (on n'arrête pas la récursion)
- LEVEL : attribut permettant de retourner la profondeur du nœud par rapport à la racine
- NOCYCLE : ne retourne pas un message d'erreur si un cycle est rencontré
- SYS_CONNECT_BY_PATH : permet de construire le chemin depuis la racine
- CONNECT_BY_ROOT : utiliser le nœud racine dans une condition
- ……..

Exemple de requêtes hiérarchiques

EMP(EMPNO,NAME,VILLE,MGR)
Tous les subordonnés parisiens de Pierre?

```
Select NAME,LEVEL
from EMP E
where E.VILLE='Paris'
start with E.NAME='Pierre'
connect by E.MGR = prior E.EMPNO;
```

Exemple de requêtes hiérarchiques

Liste des ancêtres
EMP(EMPNO,NAME,VILLE,MGR)
Tous les supérieurs parisiens de Pierre?

Exemple de requêtes hiérarchiques

Liste des ancêtres
EMP(EMPNO,NAME,VILLE,MGR)
Tous les supérieurs parisiens de Pierre?

```
Select NAME,LEVEL
from EMP E
where E.VILLE='Paris'
start with E.NAME='Pierre'
connect by prior E.MGR = E.EMPNO;
```

Récursivité (SQL3)

WITH [RECURSIVE] <surnom_requête>
[(<liste_colonne>)]
AS (<requête_select>)

Puis:
<requête_utilisant_surnom_requête>

NB: le mot clé RECURSIVE n'est pas utile en général sauf certains systèmes où obligatoire (ex. postgres)

Exemple récursion SQL3

EMP(EMPNO,NAME,VILLE,MGR)
Tous les subordonnés parisiens de Pierre?

```
WITH subordonne(monSubNo,monsubNom)
AS
(SELECT DISTINCT EMPNO,NAME
FROM EMP
WHERE NAME='Pierre'
UNION ALL
SELECT sub.EMPNO, sub.NAME
FROM EMP sub, subordonne sup where sup.ENO=sub.MGR)

Select * FROM subordonne where ville='Paris';
```

Exemple récursion SQL3

Liste des ancêtres
EMP(EMPNO,NAME,VILLE,MGR)
Tous les supérieurs parisiens de Pierre?

Exemple récursion SQL3

Liste des ancêtres
EMP(EMPNO,NAME,VILLE,MGR)
Tous les supérieurs parisiens de Pierre?

```
WITH supérieur(monSupNo,monSupNom)
AS
(SELECT DISTINCT EMPNO,NAME
FROM EMP
WHERE NAME='Pierre'
UNION ALL
SELECT sup.EMPNO, sup.NAME
FROM EMP sup, supérieur sub on sup.ENO=sub.MGR)

Select * FROM supérieur where ville='Paris';
```

Limites des SGBD relationnels

Résultats sur MySQL avec 1,000 utilisateurs atteignables

Depth	Execution Time (sec)	Records Returned
2	0.028	~900
3	0.213	~999
4	10.273	~999
5	92,613.150	~999

Limites des SGBD relationnels

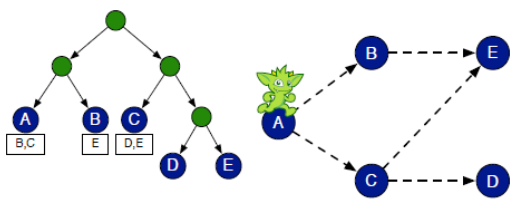
Résultats sur avec 1,000,000 utilisateurs atteignables

Depth	Execution Time (sec)	Records Returned
2	0.016	~2,500
3	30.267	~125,000
4	1,543.505	~600,00
5	Did not finish after an hour	N/A

Limites des SGBD

- SGBD Relationnels offrent
  - un système de jointure entre les tables permettant de construire des requêtes complexes impliquant plusieurs entités
  - un système d'intégrité référentielle permettant de s'assurer que les liens entre les entités sont valides
- Contexte fortement distribué: Ces mécanismes ont un coût considérable
  - avec la plupart des SGBD relationnels, les données d'une BD liées entre elles sont placées sur le même nœud du serveur
  - si le nombre de liens important, il est de plus en plus difficile de placer les données sur des nœuds différents.

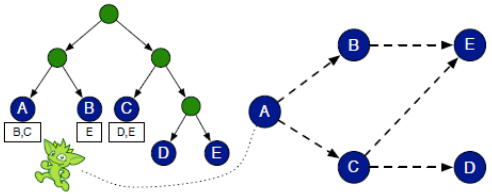
Limites des SGBD



Supposons qu'on recherche les nœuds atteignables depuis A

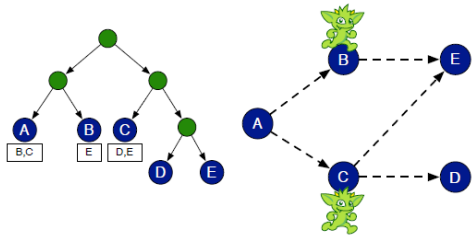
(images issues de le présentation de M. Rodriguez)

Limites des SGBD



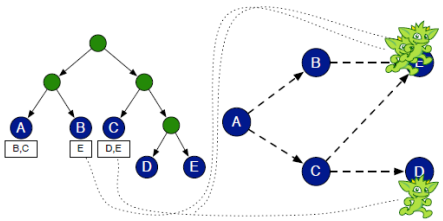
- On doit d'abord interroger l'index pour savoir quels nœuds sont adjacents à A
- Coût de $\log_k(n)$ pour l'index (sauf si tient en mémoire) plus un accès disque pour lire les ID des voisins.

Limites des SGBD



- On accède aux informations de B et C (si besoin)
- Le graphe implicite est représenté à l'aide de l'index (explicite)

Limites des SGBD



- Et ainsi de suite …

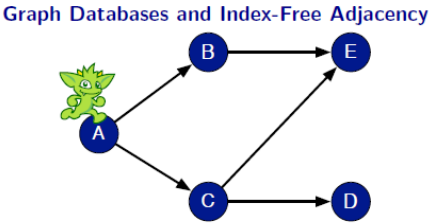
Solutions « Big Data »

- Bases de données graphes
- Approches MapReduce
- Approches Pregel-like

Adjacence sans index

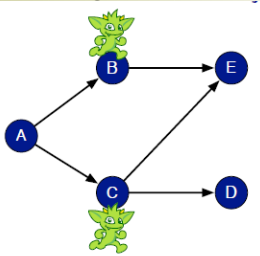
- Une base de données classique peut représenter un graphe uniquement de manière implicite
- Les approches « graphes » comme BD graphes rendent la structure de graphe explicite
- Dans ces approches chaque nœud sert d'index pour ses nœuds adjacents
- Ainsi, même si la taille du graphe augmente, le coût d'une étape de parcours demeure le même.

Approches Big Data



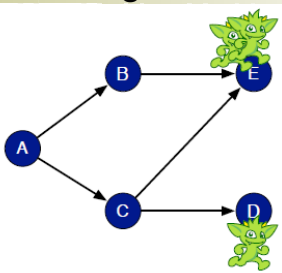
- Dans une BD graphe, un nœud A a des références directes vers ses nœuds adjacents
- Un coût en temps constant pour trouver les voisins de A, indépendamment du nombre de voisins

Approches Big Data



- Parcours du graphe explicite

Approches Big Data



- Parcours du graphe explicite

Perf.: ex. BD graphes (Neo4j) – 1000 utilisateurs atteignables

Depth	Execution Time (sec)	Records Returned
2	0.04	~900
3	0.06	~999
4	0.07	~999
5	0.07	~999

Perf.: ex. BD graphes (Neo4j) – 1000000 utilisateurs atteignables

Depth	Execution Time (sec)	Records Returned
2	0.010	~2,500
3	0.168	~110,000
4	1.359	~600,000
5	2.132	~800,000

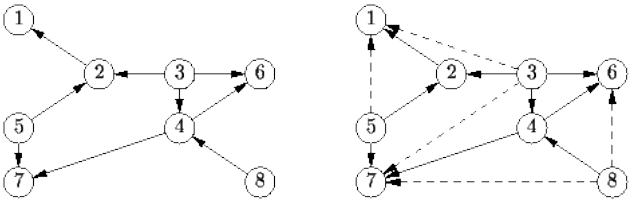
Quelques algorithmes abordés

- Fermeture transitive
- Plus court chemin
- Nombre de voisins communs
- Nombre de triangles dans graphe

Fermeture transitive

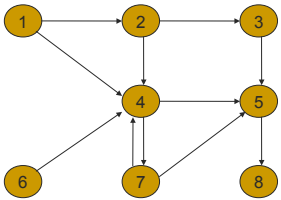
- Créer un arc entre 2 nœuds s'il existe un chemin entre les 2 nœuds
- Algorithme: ajouter les arcs n'existant pas en faisant un BFS depuis chaque nœud du graphe

Fermeture transitive (2)



Fermeture transitive: exercice

Trouver la fermeture du graphe suivant:



Fermeture transitive: utilisation

- Estimation de l'influence de chaque compte dans Twitter
- En gardant les chemins, calcul de la mesure de centralité (si on prend 2 comptes au hasard probabilité qu'un chemin les reliant passe par moi)

Plus court chemin

- Déterminer le plus court chemin entre 2 nœuds
- Éventuellement des poids sur les arcs
- Algorithme: un BFS (avec pruning quand la distance/poids plus grande que la distance/poids chemin déjà trouvé)… ou Dijkstra

Plus court chemin: utilisation

- GPS (avec distance, mais aussi avec coût le plus bas)
- Degré de séparation: trouver le nombre d'utilisateurs entre deux utilisateurs sur Facebook

Voisins communs

- Déterminer le nombre de voisins en commun entre 2 noeuds
- Les voisins peuvent être des comptes (graphes sociaux de type FB, Twitter, LinkedIn)
- … ou alors avoir une autre nature → graphe bi-parti du type Youtube, FourSquare, …

Voisins communs: utilisation

- You Might Also Know de Facebook: si on partage beaucoup d'amis, on doit sans doute se connaître
- Recommandation de lieux dans FourSquare d'après avis des amis: si des amis recommandent un lieu, alors de bonne chance que j'aime aussi

Nombres de triangles

- Observation: dans graphe social beaucoup de cycle de longueur 2 (facile à détecter) mais aussi beaucoup de longueur 3 (triangle)
- Objectif: compter ces triangles
- Algorithme (naïf): pour toute paire de voisins b et c d'un nœud a, vérifier si arc (b,c) existe
- Si orienté, regarder si v est le voisin d'un voisin d'un voisin

Comptage des triangles: exemples

Combien de triangles?