Apunts2BigDataDataModelsQuery.pdf

Aridevi

Desenvolupament d'Aplicacions de Dades Massives

3º Grado en Ingeniería de Datos

Escuela de Ingeniería
Universidad Autónoma de Barcelona

BIG DATA

DESENVOLUPAMENT D'APLICACIONS DE DADES MASSIVES

1. DATA MODELS AND QUERY LANGUAGES

Un data model és una part molt important del desenvolupament d'un programari. Son representacions visuals dels elements de dades d'una empresa i les connexions entre ells. Ens ajuda a definir i estructurar les dades. Els models, donen suport al desenvolupament d'un sistema. Permeten decidir de manera col·laborativa com s'emmagatzemaran i s'aprofitaran les dades. Se centra en com es dissenya i desenvolupa el programari i també ens ajuden quan trobem un problema a solucionar.

Existeixen diferents models da dades:

1. El model més conegut és el d'**SQL**, que es basa en el **model relacional**. Les dades s'organitzen en relacions (anomenades taules en SQL), on cada relació és una col·lecció no ordenada de tuples (files en SQL).

   El model relacional té nombrosos beneficis, però també té limitacions:
   - ✓ És un model molt simple, no requereix consultes complexes.
   - ✓ Té precisió i integritat de les dades.
   - ✓ Permet manejar grans volums de dades. El problema d'això serà que el manteniment serà difícil a causa de l'augment de les dades. És costós de configurar i mantenir.
   - ✓ Gestiona un gran volum d'operacions d'escriptura de dades per segon.
   - ✓ Realitza operacions de cerca específiques.
   - ✓ Models de dades dinàmics. Amb esquema fàcil de canviar.

2. Un altre model molt conegut és el de **NoSQL** (documental). És un codi obert, BD no relacional distribuïda.
3. **Graphs**. Representat com un graf connectat de nodes i relacions amb propietats i etiquetes.
4. Resource Description Framework **RDF**. Model d'intercanvi de dades a la Web. Facilita la fusió de dades.

Representació JSON:

JSON és adequat per a un document com ara un perfil. Una avantatge de JSON és la manca d'un esquema. No hi ha necessitat d'unions múltiples. Les relacions d'un a molts des del perfil de l'usuari fins als ítems de l'usuari, impliquen una estructura d'arbre a les dades i la representació JSON fa que aquesta estructura d'arbre sigui clara (**explícita**).

La diferència més gran entre una base de dades i JSON és que la BD realitza consultes múltiples entre usuaris i altres taules mentre que JSON ho pot realitzar tot en una sola consulta degut a la bona localitat de les dades.

Quan parlem de que té una bona localitat de les dades, ho diem perquè normalment un fitxer JSON s'emmagatzema en una única cadena. A més a més, al tenir totes les dades en un mateix document, el temps és òptim, ja que, ja hi és a memòria. En canvi, quan les dades es divideixen en diverses taules, es requereixen múltiples cerques d'índex per recuperar les dades, això comporta més cerques al disc.

Si utilitzéssim JSON com a base de dades hauríem de preguntar-nos com gestionar les relacions amb l'entitat (**entity relationships**). Per tal de gestionar les relacions, fem ús d'identificadors que ens serveixen per no confondre ambigüitat dels camps, per facilitar: les actualitzacions dels valors, la cerca i les traduccions.

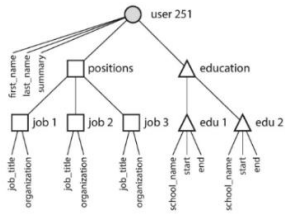
En l'exemple del perfil de LinkedIn, emmagatzemem regions d'una persona com a identificadors en comptes de text senzill és eliminar la duplicació, que és la idea clau darrere de la normalització a les bases de dades.

Ariadna De Vicente Viladesau

Exemple de perfil de Linkedin:

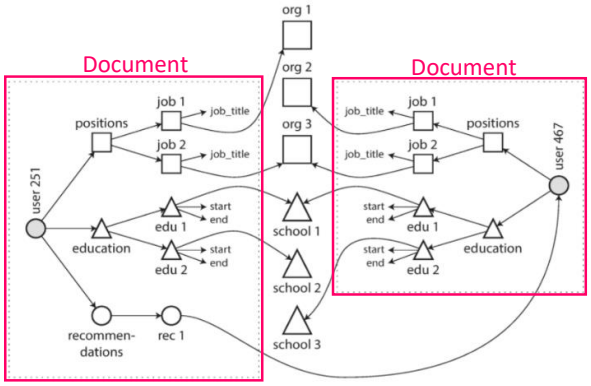
Una relació **one-to-many** d'un usuari a molts ítems, es pot representar de moltes maneres.

- SQL model. Posar la informació en taules separades amb una clau a la taula de l'usuari.



En aquest exemple, emmagatzemem regions d'una persona com a identificadors. Al perfil de LinkedIn, es poden afegir noves funcions com les organitzacions i les escoles com a entitats, però no ho podem emmagatzemar fàcilment al document, ja que, aquestes requeriran de relacions **many-to-many**.

La figura següent il·lustra com aquestes noves funcions requereixen relacions de molts a molts. Les dades de cada rectangle es poden agrupar en un sol document, però les referències a organitzacions, escoles i altres usuaris s'han de representar com a referències i requereixen unions.



Quin és el millor data model?

Per tal de saber quin seria el millor model de dades, farem una comparativa entre les **BD de Documents** i les **BD Relacionals**. Les BD de Documents, no son massa útils per consultes d'unió entre entitats. En les BD Relacionals, l'operació join és una bona eina per a fer les consultes.

Per a saber quin seria el millor model de dades, ens hauríem de preguntar quines son les operacions més comunes que haurem de realitzar. I depenent d'això, ja sabrem quin model utilitzar.

En la BD de Documents, cada document correspon a un objectes i cada objecte conté diferents atributs. En canvi, una BD Relacional, en una taula hi haurà diversos objectes i atributs emmagatzemats com a files i columnes.

1. **BD de Documents:** Les dades són majoritàriament documents.
   - Relació de one-to-many
   - Cal carregar l'arbre només una vegada
2. **DB Relacionals:** Diferents entitats i necessitat d'analitzar les relacions
   - Consultes de many-to-many
   - L'estructura de la informació no canviarà massa en el futur
   - No hi ha una gran quantitat d'informació que canvia en el temps

Gestió d'un canvi de data model:

Ara descobrirem si és possible canviar de model de dades i en cas de ser possible, quin cost tindria fer-ho. Per tal de fer-ho, hauríem de canviar el format de les dades i això, es tractaria de forma diferent segons el tipus de model.

Quan parlem de BD Documents, es tracta d'un esquema **dinàmic**. Per tal de canviar el format de les dades, escriuríem nous documents amb nous camps (per exemple, nom i cognom). D'aquesta manera, l'aplicació substituirà els documents antics.

En canvi, una BD Relacional, es tracta d'un esquema **estàtic**. Cada canvi obliga a modificar tota la BD.

Ariadna De Vicente Viladesau

Schema-on-read / schema-on-write:

Un **schema-on-read**, és aquella creació en al que no apliquem cap esquema durant la recollida de dades.

Primer de tot carrega les dades i després immediatament, ja realitzarem les consultes o *querys* de les dades. Podem veure com realment mai hem creat un esquema. D'aquesta manera, podem simplement modificar l'script sense necessitat de tornar a carregar totes les dades.

```
1. Load the data    hdfs dfs -copyFromLocal /temp/custfile*.txt/user/hadoop/customer

2. Query the data   hadoop jar Hadoop-streaming.jar
                        -mapper customer-mapper.py
                        -reducer customer-reducer.py
                        -input /user/hadoop/customer/*.txt
                        -output /user/hadoop/output/query1
```

L'estructura de dades és implícita i s'interpreta durant la lectura, és similar als tipus dinàmics de Python i té màxima flexibilitat.

Exemples: JSON google maps, missatge de twitter. #Pregunta examen

Un **schema-on-write**, es defineix com la creació d'un esquema (crear una taula) per a dades abans d'escriure a la BD.

Un cop existeix l'esquema o taula, podem afegir les dades. I un cop fet això, podem realitzar consultes o *querys* de les dades. En SQL, no pots afegir dades fins que no estigui l'esquema de taula declarat. Si l'esquema s'ha de redefinir, la taula és eliminada i es torna a carregar.

```
1. Create schema    CREATE TABLE Costumers (Key int, Name varchar (40), ...)

2. Add data         BULK INSERT Costumers
                    FROM 'c:\temp\custfile.txt'
                    WITH FIELDTERMINATOR = ' ","  '

3. Query data       SELECT Key, Name FROM Costumers
```

La BD força totes les dades emmagatzemades a seguir el model definit. Té un esquema explícit de dades, és similar als tipus estàtics C++ i té una estructura sòlida.

#Pregunta examen
Exemples: Compte corrent bancària, llistat d'alumnes matriculats a l'assignatura, beques del ministeri d'Educació.

Query languages:

És un llenguatge de programació informàtic que sol·licita i recupera dades de BD mitjançant l'enviament de consultes. El LQ s'utilitza per crear, accedir i modificar dades. El client recuperarà totes les dades dels registres/taula del client.

Els LQ **imperatius**, s'utilitzen per descriure com volem que es faci alguna cosa en concret. Això s'aconsegueix amb un control explícit de manera detallada i pas a pas; la seqüència i la redacció de cada línia de codi tenen un paper crític. Alguns llenguatges de programació imperatiu general coneguts inclouen Python, C i Java.

- ✓ Especifica un ordre particular de realitzar les operacions, no és fàcil de fer en paral·lel. #Pregunta examen
- ✓ Defineix un ordre.

Els LQ **declaratius**, es defineixen com qualsevol llenguatge de consulta de bases de dades que no sigui imprescindible. Permeten als usuaris expressar quines dades han de recuperar. Donen instruccions àmplies i de forma general sobre la tasca a completar en comptes d'indicar com completar-la. Alguns llenguatges de programació declaratius generals coneguts inclouen SQL, Ruby, R i Haskell.

- ✓ Oculta els detalls d'implementació. #Pregunta examen
- ✓ No es defineix cap ordre en les operacions a realitzar. #Pregunta examen
- ✓ Fa més senzilla l'execució en paral·lel de les operacions.

Imperatiu (Python)

```
var sharks = [];
for (var i = 0; i < animals.length; i++) {
    if (animals[i].family === "Sharks") {
        sharks.push(animals[i]);
    }
}
return sharks;
```

Declaratiu (SQL)

```
SELECT * FROM animals WHERE family = 'Sharks';
```

Ariadna De Vicente Viladesau

Consultes de MapReduce:

És un model de dades específic per un gran volum de processament de dades. És un model de programació de baix nivell popularitzat per Google per a l'execució distribuïda en clústers d'ordinador que es basa en dues primitives: mapejar i reduir.

S'utilitza com a mecanisme per realitzar consultes de només lectura a molts documents. MapReduce no és ni un llenguatge de consulta declaratiu ni una API de consulta totalment imperativa, sinó un punt intermedi: la consulta s'expressa amb fragments de codi, que són cridats repetidament pel marc de processament. Es basa en les funcions de **mapa** (també conegudes com a col·leccionar) i **reduir** (també conegudes com a plegar o injectar) que existeixen en molts llenguatges de programació.

Models de dades semblants a grafs:

El model de document és adequat si existeixen moltes relacions one-to-many no hi ha cap relació entre registres. El model relacional pot gestionar casos simples de relacions de molts a molts, però a mesura que les connexions dins de les dades es tornen més complexes, és millor començar a modelar les dades com a **graf**.

Un graf consta de dos tipus d'objectes: **vèrtexs** (també coneguts com a nodes o entitats) i **arestes** (també coneguts com a relacions). Molts tipus de dades es poden modelar com a graf.

Propietats de les taules dels grafs:

Els grafs no es limiten a dades *homogènies*: els vèrtexs poden **representar diferents tipus d'objectes** en un graf. Els nodes poden ser persones (gràfiques socials), pàgines web (gràfiques web), encreuament de carrers (gràfiques de mapa),...

- ✓ Cada **vèrtex** consta de: identificador únic, arestes sortints, arestes entrants i propietats.
- ✓ Cada **aresta** consta de: identificador únic, vèrtex on comença l'aresta, vèrtex on acaba l'aresta, etiqueta del tipus de relació i propietats.
- ✓ Cada node es pot **connectar** amb qualsevol altre amb un nou enllaç.
- ✓ El graf es pot **recórrer** mitjançant enllaços.
- ✓ S'utilitzen enllaços per **etiquetar** diferents tipus de relacions en un mateix graf.

Alguns algoritmes de grafs son: Google PageRank, Vehicle navigation Systems, Facebook on els nodes son persones, llocs, comentaris i les arestes son relacions, qui està comentant,...

Models de dades per grafs:

Formes d'estructurar i consultar dades en grafs:

- Model de **propietats** de graf: Implementat per Neo4j, Titan i InfiniteGraf.
- Model de **Triple-Store**: Implementat per Datomic i AllegroGraph.
- Llenguatges de consulta de grafs **declaratius**: Cypher, SPARQL i Datalog.
- Llenguatges de consulta de grafs **imperatius**: Gremlin.
- Marcs de **processament** de grafs: Pregel.

El llenguatge de consulta Cypher:

Cypher és un llenguatge de consulta declaratiu per a grafs de propietats, creat per a la BD Neo4j. El podem utilitzar per inserir informació en una BD i per consultar-ne informació.

Llenguatge de consulta SPARQL – Web com a BD:

Quan parlem de **web semàntica** ens referim a que una web sigui vista com una BD àmpliament distribuïda. Sabem que els llocs web ja publiquen informació com a text i imatges perquè els humans la llegeixin, així que per què no publiquen també informació com a dades llegibles per màquina perquè els llegeixin els ordinadors?

El Resource Description Framework (RDF) estava pensat com un mecanisme perquè diferents llocs web publiquin dades en un format coherent, permetent que les dades de diferents llocs web es combinessin automàticament en una web de dades, una mena de "base de dades de tot" a tota Internet.

Llegir aquest article: https://medium.com/the-definitive-notes/chapter-2-data-models-and-query-languages-15cdd29578f

Ariadna De Vicente Viladesau