

Ús de Bases de Dades

PAC 2: Disseny físic de bases de dades

Proposta de solució

Presentació

En aquesta Prova d'Avaluació Continuada s'exerciten els aspectes que convé tenir en compte en el disseny físic d'una base de dades. L'objectiu d'aquesta prova és comprovar el grau de comprensió del mòdul 2. Aquesta prova consta de 4 exercicis. Cal destacar que és necessari haver assimilat el contingut del mòdul 2 per a la correcta realització d'aquesta PAC.

Competències

En aquesta PAC es desenvolupen les següents competències del Grau en Multimèdia:

- Capacitat d'analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per abordar-lo i resoldre'l.
- Capacitat per aplicar les tècniques específiques de tractament, emmagatzematge i administració de dades.

Objectius

Els objectius concrets d'aquesta Prova d'Avaluació Continuada són:

- Aprendre a fer el disseny físic de la base de dades a partir del disseny lògic, adaptat a les característiques d'un SGBD concret.
- Definir els índexs necessaris i convenients en cada taula, per tal que les aplicacions tinguin un bon rendiment quan accedeixen a la base de dades.
- Comprendre la influència dels paràmetres del sistema en el funcionament i rendiment del gestor.
- Entendre la informació obtinguda del pla de les consultes i predir els accessos del gestor.
- Saber interpretar la informació subministrada pels monitors de rendiment.

Exercici 1 [25%]

Indiqueu si són certes o falses les afirmacions següents, justificant breument la vostra resposta:

1. Una bona mesura per tal de millorar el rendiment dels processos en línia és ubicar els fitxers d'índexs en una unitat de disc diferent que els fitxers de dades.

Proposta de solució

CERT. Si ubiquem els fitxers dels índex en una unitat de disc diferents dels fitxers de dades s'eviten interferències d'accessos de lectura.

2. En una empresa amb un alt volum de compra-vendes emmagatzemen, en un camp de la taula `CLIENTS`, l'import total de facturació per a cadascun d'ells. Per tal d'obtenir una alta eficiència en l'obtenció dels clients ordenats pel volum de facturació la millor solució és definir un índex per aquest camp.

Proposta de solució

FALS. Atès que el volum de compravenda de l'empresa és elevat, és de suposar que el contingut de la columna corresponent a l'import total de facturació de cada client s'actualitzarà freqüentment. Cal evitar la creació d'índexs sobre columnes que s'actualitzen amb molta freqüència ja que, cada operació d'actualització d'aquesta columna a la taula, implicarà també que l'SGBD haurà d'actualitzar el contingut de la clau en el fitxer índex, amb la qual cosa s'incrementarà molt el cost en cada actualització.

3. Quan creem un `TABLESPACE` normalment podem definir els diferents fitxers on s'emmagatzemaran les dades enregistrades a les taules que s'associïn a aquest espai virtual.

Proposta de solució

CERT. La creació d'un espai per taules varia en funció de cada SGBD. Tant Oracle com DB2 disposen de la sentència `TABLESPACE`, mentre que SQL Server utilitza la sentència `FILEGROUP`. Depenent de cada SGBD, aquesta sentència té diferents paràmetres, però en general permet definir quins seran els fitxers físics on s'emmagatzemaran les dades de les taules associades al `TABLESPACE`. Per exemple, amb Oracle els fitxers físics s'especifiquen amb el paràmetre `filespec`, on es pot definir la mida i la ubicació i nom del fitxer.

4. En una base de dades d'un supermercat s'ha dissenyat una taula pels diferents productes que estan a la venda i una altra per les diferents seccions de la botiga (làctics, conserves, embotits etc...). Tenint en compte que hi ha una molt baixa activitat pel que fa a la inserció de les seves dades, pot interessar emmagatzemar amb dues taules en un mateix espai per a taules simple.

Proposta de solució

CERT. Cal tenir en compte que en un supermercat poden haver uns pocs milers de productes i unes poques seccions. Valorem doncs que les taules són relativament petites i com diu l'enunciat amb molt poques insercions. També cal considerar que l'espai per a taules simple està compost per pàgines, i cada pàgina pot contenir files de diferents taules. És a dir, permet emmagatzemar files barrejades de diferents taules. Aquest fet pot afavorir el procés de lectura si s'emmagatzemen conjuntament les files de les dues taules mantenint un cert ordre, és a dir, de manera que s'emmagatzemi cadascuna de les seccions seguides dels productes assignats a aquesta.

Exercici 2 [20%]

Cerca informació sobre índexs GIN. Quina és la seva funcionalitat? En quins SGBD els trobem implementats? Posa un exemple d'ús . (límit de la resposta 2 pàgines)

Proposta de solució

Els índexs GIN (*Generalized Inverted Index*) són índexs de tipus invertit, que poden gestionar valors que contenen més d'un element clau, on l'element clau es concep o pot ser vist com multiavaluat. Això els fa especialment interessants per indexar valors compostos en cerques que requereixin identificar files que contenen determinats valors components.

En aquest tipus d'índex s'utilitza el terme *ítem* per a referir-se al valor compost a indexar (document) i la paraula *clau* per a referir-se als valors a buscar (les paraules). Els índexs GIN emmagatzemen un conjunt de parells (clau, RID₀, RID₁,... RID_n) que indiquen que el valor de la clau es troben en les files amb adreces RID₀, RID₁,... RID_n.

Els índexs GIN es troben implementats en els SGBD de *PostgreSQL* i a *Informix Database Server (IDS)* de *IBM*. *Oracle* no fa servir índexs GIN i disposa del seu propi tipus d'índex *Oracle Text*.

Com en els índexs GiST i SP-GiST poden donar suport a la creació d'estratègies d'indexació per part de l'usuari. Bàsicament, aquest tipus d'índex és l'element central dels algorismes d'indexació dels motors de cerca, similar a la cerca mitjançant paraules clau de *Google*. Per exemple els elements a indexar poden ser documents i les consultes tenen com a objectiu recuperar els documents que contenen un conjunt específic de paraules.

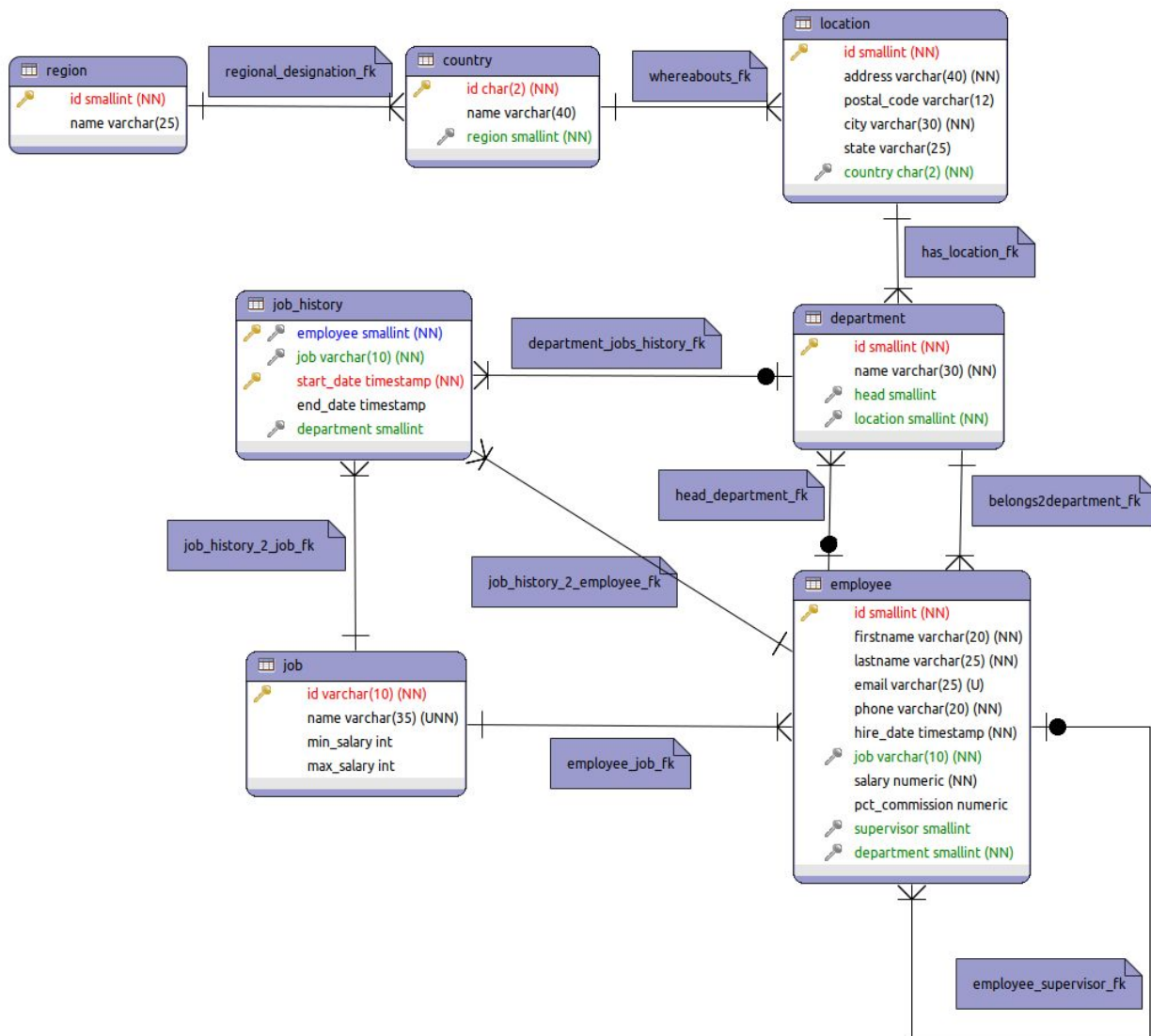
Més informació a:

- <https://www.postgresql.org/docs/10/gin-intro.html>
- <https://www.postgresql.org/docs/10/textsearch-indexes.html>

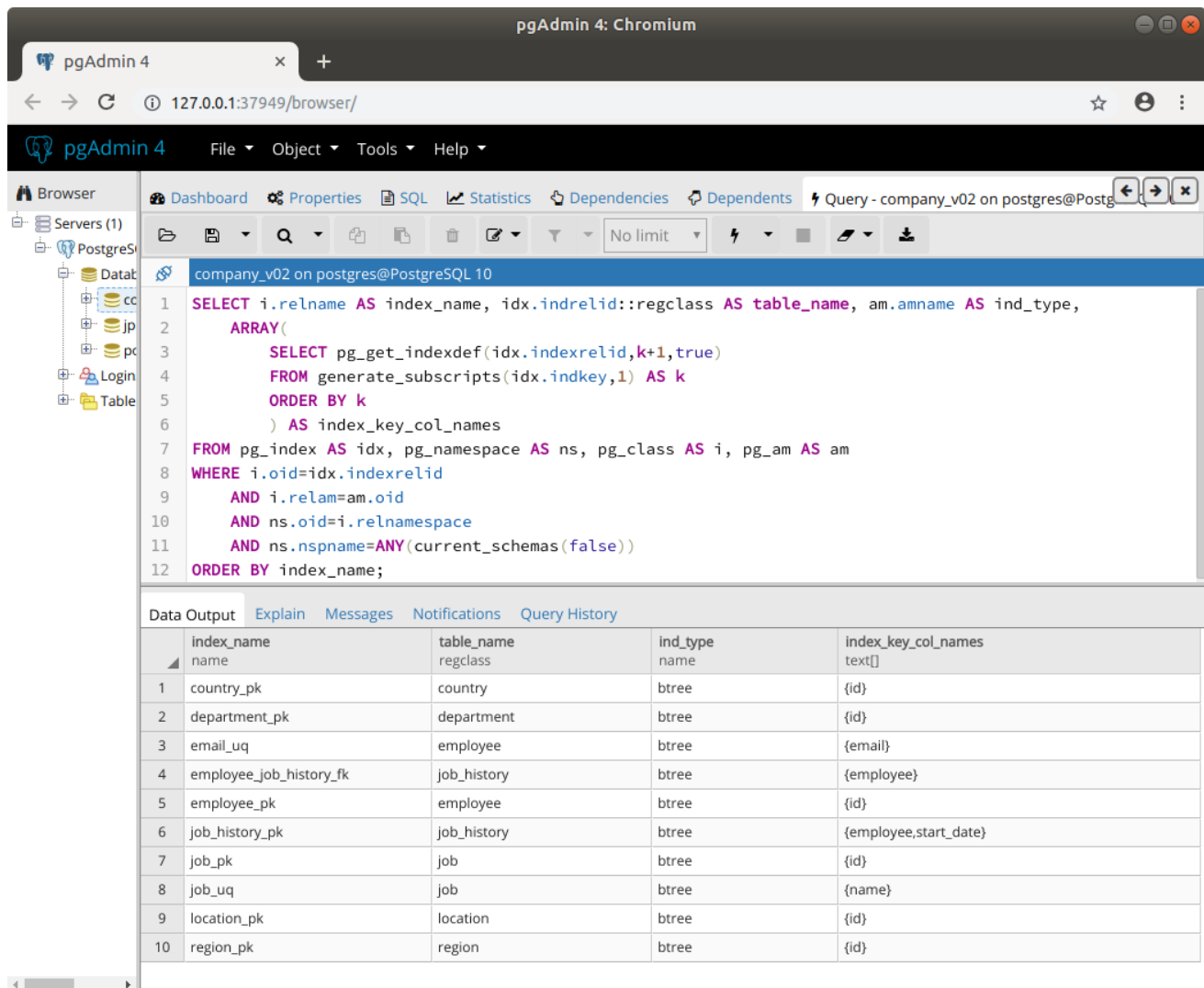
Exercici 3 [25%]

Suposem que estem treballant amb una nova base de dades anomenada **company_v02**. Aquesta, ens permet gestionar els treballs que estan realitzant, i l'històric dels treballs realitzats anteriorment, pels empleats d'una empresa. A l'històric no s'emmagatzema el valor del salari. El valor del salari emmagatzemat correspon a l'import que cobra actualment. Aquests empleats s'organitzen jeràrquicament i són assignats en departaments. Cadascun d'aquests departaments està ubicat en una localització i un dels seus empleats n'és el seu cap. Degut a que la informació correspon a una multinacional hi ha localitzacions en diferents països i aquests es troben agrupats per regions internacionals.

En el següent diagrama podem veure una descripció de les diferents taules implicades:



Un cop realitzada la implementació de les diferents taules i de realitzar una càrrega massiva de dades, l'administrador de la base de dades ha executat la següent consulta al catàleg i el resultat obtingut ha estat el següent:



The screenshot shows the pgAdmin 4 interface in a Chromium browser window. The query editor displays a SQL query to list all indexes in the database. The query is as follows:

```

1 SELECT i.relname AS index_name, idx.indrelid::regclass AS table_name, am.amname AS ind_type,
2     ARRAY(
3         SELECT pg_get_indexdef(idx.indexrelid,k+1,true)
4         FROM generate_subscripts(idx.indkey,1) AS k
5         ORDER BY k
6     ) AS index_key_col_names
7 FROM pg_index AS idx, pg_namespace AS ns, pg_class AS i, pg_am AS am
8 WHERE i.oid=idx.indexrelid
9     AND i.relam=am.oid
10    AND ns.oid=i.relnamespace
11    AND ns.nspname=ANY(current_schemas(false))
12 ORDER BY index_name;

```

The results are displayed in a table with the following columns: index_name, table_name, ind_type, and index_key_col_names. The results are as follows:

	index_name	table_name	ind_type	index_key_col_names
1	country_pk	country	btree	{id}
2	department_pk	department	btree	{id}
3	email_uq	employee	btree	{email}
4	employee_job_history_fk	job_history	btree	{employee}
5	employee_pk	employee	btree	{id}
6	job_history_pk	job_history	btree	{employee,start_date}
7	job_pk	job	btree	{id}
8	job_uq	job	btree	{name}
9	location_pk	location	btree	{id}
10	region_pk	region	btree	{id}

Podries explicar quin és l'origen i la funcionalitat dels índexs esmentats?

Proposta de solució

Tal i com podem veure en la pàgina 25 dels apunts d'aquest mòdul, s'esmenta que:

- En la majoria dels SGBD, la definició d'una columna de la taula amb l'atribut únic porta implícita la definició de l'índex.
- En la majoria dels SGBD, la definició d'una clau primària porta implícita la definició d'un índex únic i no nul.
- Les claus foranes formen, en la majoria dels casos, un altre índex.

Així doncs, podem veure en la consulta al catàleg els següents índexs de la base de dades `company_v02` en un SGBD PostgreSQL a partir de la definició de les taules del diagrama adjunt:

country_pk: Es tracta de la *primary key* del camp `id` de la taula `country`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

department_pk: Es tracta de la *primary key* del camp `id` de la taula `department`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

email_uq: El camp `email` de la taula `employee` s'estableix com a `UNIQUE`, i es crea un índex sobre aquests camp per a garantir que els valors no es repeteixen.

employee_job_history_fk: En garantir la integritat referencial entre les taules `employee` i `job_history` (un registre de l'històric de treballs realitzats sempre correspon a un empleat) on el camp `employee` de la taula `job_history` és una clau forana relacionada amb el camp `id` de la taula `employee`. S'ha considerat recomanable disposar d'un índex sobre la clau forana per a optimitzar els accessos d'inserció o esborrat de files.

employee_pk: Es tracta de la *primary key* del camp `id` de la taula `employee`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

job_history_pk: Es tracta de la *primary key* formada pels camps `employee` i `job` de la taula `job_history`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

job_pk: Es tracta de la *primary key* formada pel camp `id` de la taula `job`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

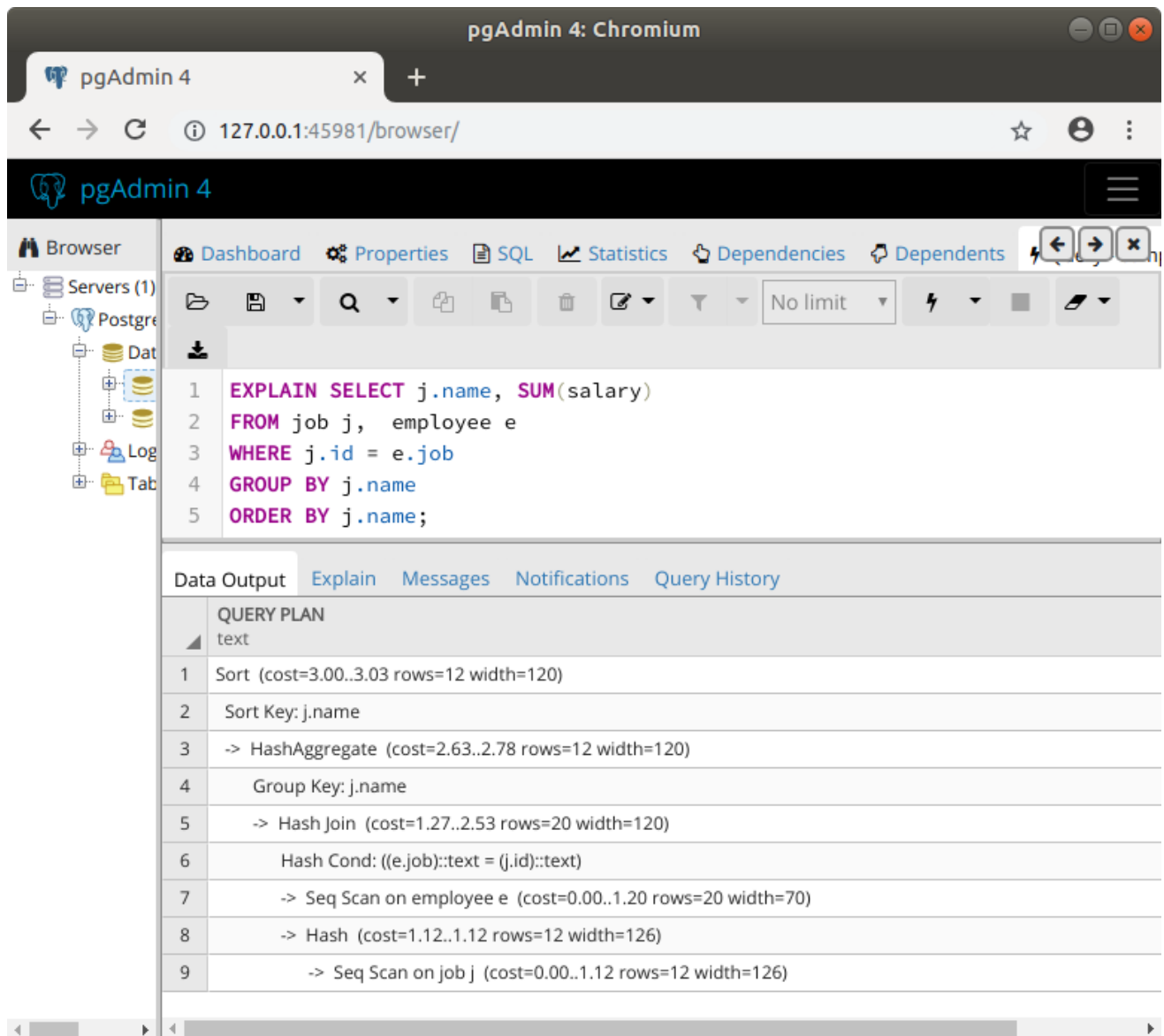
job_uq: El camp `name` de la taula `job` s'estableix com a `UNIQUE`, i es crea un índex sobre aquests camps per a garantir que la parella de valors no es repeteixi.

location_pk: Es tracta de la *primary key* formada pel camp `id` de la taula `location`. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

region_pk: Es tracta de la *primary key* del camp *id* de la taula *region*. En crear una *primary key*, automàticament es crea un índex, que garantirà la unicitat i evitarà duplicitats.

Exercici 4 [30%]

En un SGBD PostgreSQL s'ha realitzat el pla d'accés de la següent consulta sobre la base de dades `company_v02`:



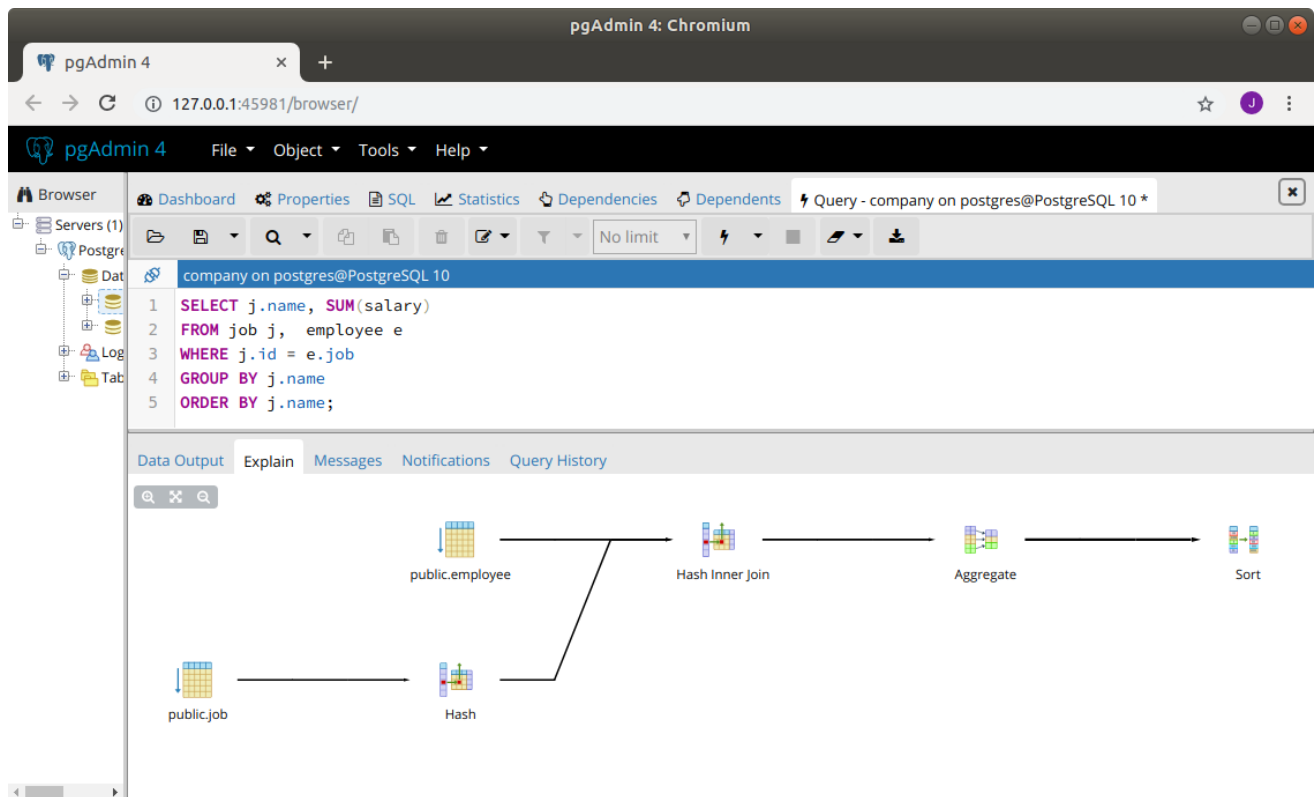
The screenshot shows the pgAdmin 4 interface in a Chromium browser window. The SQL query being executed is:

```
1 EXPLAIN SELECT j.name, SUM(salary)
2 FROM job j, employee e
3 WHERE j.id = e.job
4 GROUP BY j.name
5 ORDER BY j.name;
```

The 'Data Output' tab is selected, displaying the 'QUERY PLAN' for the query. The plan consists of the following steps:

Step	Operation	Cost	Rows	Width
1	Sort	3.00..3.03	12	120
2	Sort Key: j.name			
3	-> HashAggregate	2.63..2.78	12	120
4	Group Key: j.name			
5	-> Hash Join	1.27..2.53	20	120
6	Hash Cond: ((e.job)::text = (j.id)::text)			
7	-> Seq Scan on employee e	0.00..1.20	20	70
8	-> Hash	1.12..1.12	12	126
9	-> Seq Scan on job j	0.00..1.12	12	126

i que es pot representar gràficament de la següent forma:



Es demana que realitzeu una interpretació d'aquest pla d'accés de la consulta realitzada de la manera més acurada possible. (límit de la resposta 1 pàgina).

Proposta de solució

La comanda `EXPLAIN`, dissectiona una consulta en els seus components, i analitza el temps d'execució, nombre de files que retorna i ús de memòria per a cada component.

El resultat està imbricat, i per llegir-lo cal llegir de més intern a més extern, i de baix a dalt.

Els nombres indicats entre parèntesi signifiquen el següent:

cost: indica el temps estimat que triga a executar-se. Mostra dos nombres, el primer, indica el cost estimat per iniciar l'operació i el segon el cost estimat final que triga en executar-se la operació. Aquests valors són acumulats, és a dir, es referencien a partir de l'inici d'execució de l'anàlisi del pla de consulta.

rows: indica el nombre estimat de files que retorna.

width: Ample mitjà de registres produïts en aquest node (en bytes)

Els costos es mesuren en unitats arbitràries, determinades pels paràmetres de costos del planificador. Tradicionalment es mesuren en unitats de càrregues de pàgina. Cal tenir en compte que el cost d'un node de nivell superior inclou el cost de tots els seus nodes secundaris, per tant, el cost total estimat de la consulta realitzada serà de 3,03.

En el resultat de la sentència `EXPLAIN` de la imatge, es veu el següent:

- a.- **Línies 8 i 9**: Fa un escaneig de la taula `JOB` per clau primària. Comença als 0,0, triga 1. Retorna 12 files, amb un ample mitjà estimat 126 bytes.
- b.- **Línia 7**: Lectura seqüencial de la taula `EMPLOYEE` i, comença als 0,0, acaba al 1,20 i torna 20 files amb un ample mitjà estimat 70 bytes. Fixem-nos que aquesta operació es realitza paral·lelament a les esmentades anteriorment.
- c.- **Línies 6 i 5**: L'operador `Hash Join` s'utilitza en fer una operació de combinació entre els registres de les dues taules enllaçades, en aquest cas `EMPLOYEE` i el *hash* de la taula `JOB`, calculat anteriorment, que compleixen la condició `e.job=j.id`. Comença en 1,27 i acaba en 2,53, retorna 20 files amb un ample mitjà estimat 120 bytes. Fixem-nos que aquesta operació es realitzarà posteriorment a les anteriors, d'aquí prové el valor de l'estimació temporal inicial.
- d.- **Línies 4 i 3**: `HashAggregate`, ja que la consulta necessita calcular `SUM(salary)` i s'ha indicat que el resultat estigui agrupat per l'atribut `JOB.name`. Comença als 2,63 i acaba als 2,78 i retorna 12 files amb un ample mitjà estimat 120 bytes.
- e.- **Línies 1 i 2**: S'executa la funció `Sort`. Ordena per `JOB.name`. Comença als 3,00 i acaba als 3,03 i retorna 12 files amb un ample mitjà estimat de 120 bytes.

Recursos

Els següents recursos són d'utilitat per la realització de la PAC:

Bàsics

- Mòdul didàctic 2. Disseny físic de bases de dades

Criteris de valoració

La ponderació dels exercicis és la següent:

- Exercici 1: 25%
- Exercici 2: 20%
- Exercici 3: 25%
- Exercici 4: 30%

Aquesta PAC s'ha de fer de manera estrictament individual. Qualsevol indicati de còpia serà penalitzat amb un suspens (D) per a totes les parts implicades i la possible avaluació negativa de l'assignatura en la seva totalitat.

Format i data de lliurament

1. El format del fitxer ha de ser PDF.
2. El nom del fitxer ha de tenir el format següent: "nomUsuariUOC_PAC2.pdf", per exemple "jperezbr_PAC2.pdf".
3. El nom de l'alumne ha d'aparèixer a la portada i en cada pàgina del document.

Data límit de lliurament : 26 de març de 2019 a les 24:00 hores.

La data de lliurament d'aquesta PAC ha de ser estrictament respectada, i no s'acceptarà cap lliurament després de la data establerta. Si es considera per alguna raó justificada que no es va a poder complir amb aquesta data, l'estudiant s'haurà de posar en contacte amb el seu consultor de l'assignatura amb suficient anterioritat per poder buscar conjuntament una solució al respecte. Si s'acorda el lliurament amb posterioritat, la nota màxima d'aquesta PAC serà un aprovat (C+).