PostgreSQL

Cours basic d’initiation

Sommaire présentation

- - What about postgreSQL ?
- - PostgreSQL history
- - Client-server architecture
- - Feature for postgreSQL
- - Database Object
- - Client tools
- - basic syntaxe
- - Questions
- - Exercices

What about postgreSQL

- SBGDR puissant et robuste
- Supporte une grande partie du standard SQL
- Multi-plateformes
- Supporte de nombreux langages programmation
- Modèle client-serveur

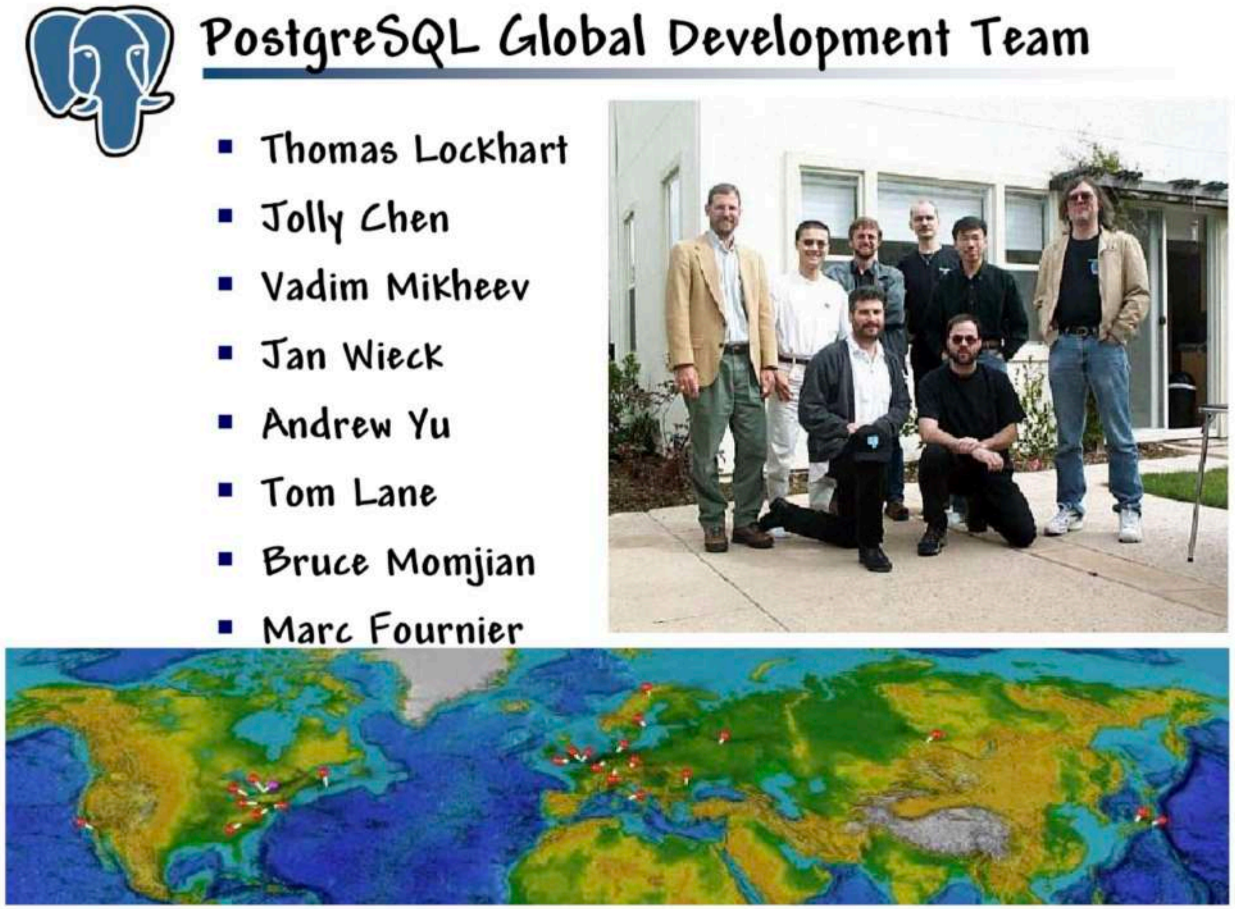
What about postgreSQL (suite)

- Conçu pour prise en charge des données volumineuses
- Déploiement illimitée
- Bonne documentation de référence.
- Excellent support
- Outils graphiques d'administration

Postgres History

- A Brief History of PostgreSQL
- PostgreSQL, originally called Postgres, was created at UCB by a computer science professor named Michael Stonebraker. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres, now owned by Computer Associates.
- **1977-1985** – A project called INGRES was developed.
  - Proof-of-concept for relational databases
  - Established the company Ingres in 1980
  - Bought by Computer Associates in 1994
- **1986-1994** – POSTGRES
  - Development of the concepts in INGRES with a focus on object orientation and the query language - Quel
  - The code base of INGRES was not used as a basis for POSTGRES
  - Commercialized as Illustra (bought by Informix, bought by IBM)
- **1994-1995** – Postgres95
  - Support for SQL was added in 1994
  - Released as Postgres95 in 1995
  - Re-released as PostgreSQL 6.0 in 1996
  - Establishment of the PostgreSQL Global Development Team

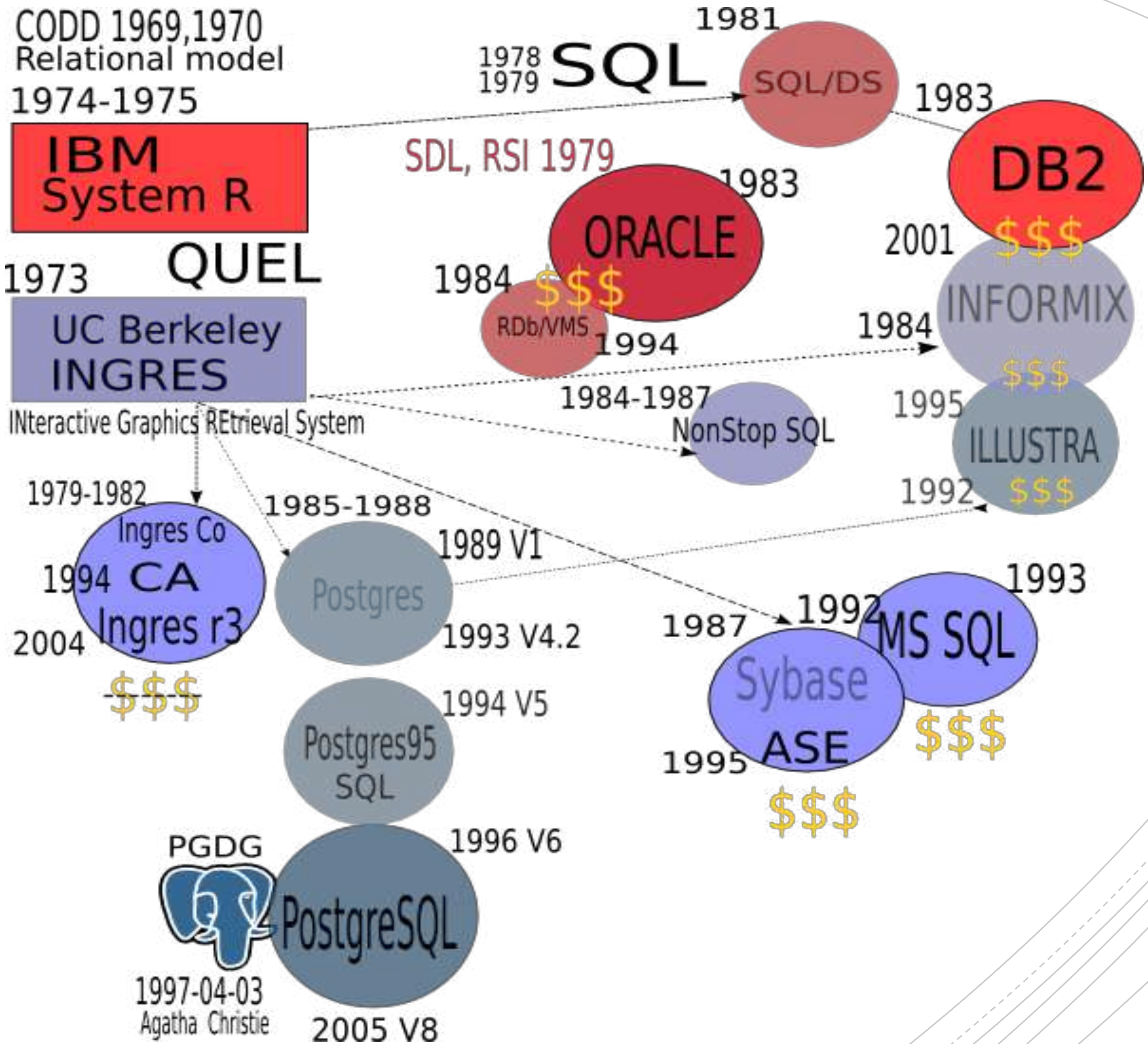
Postgres History



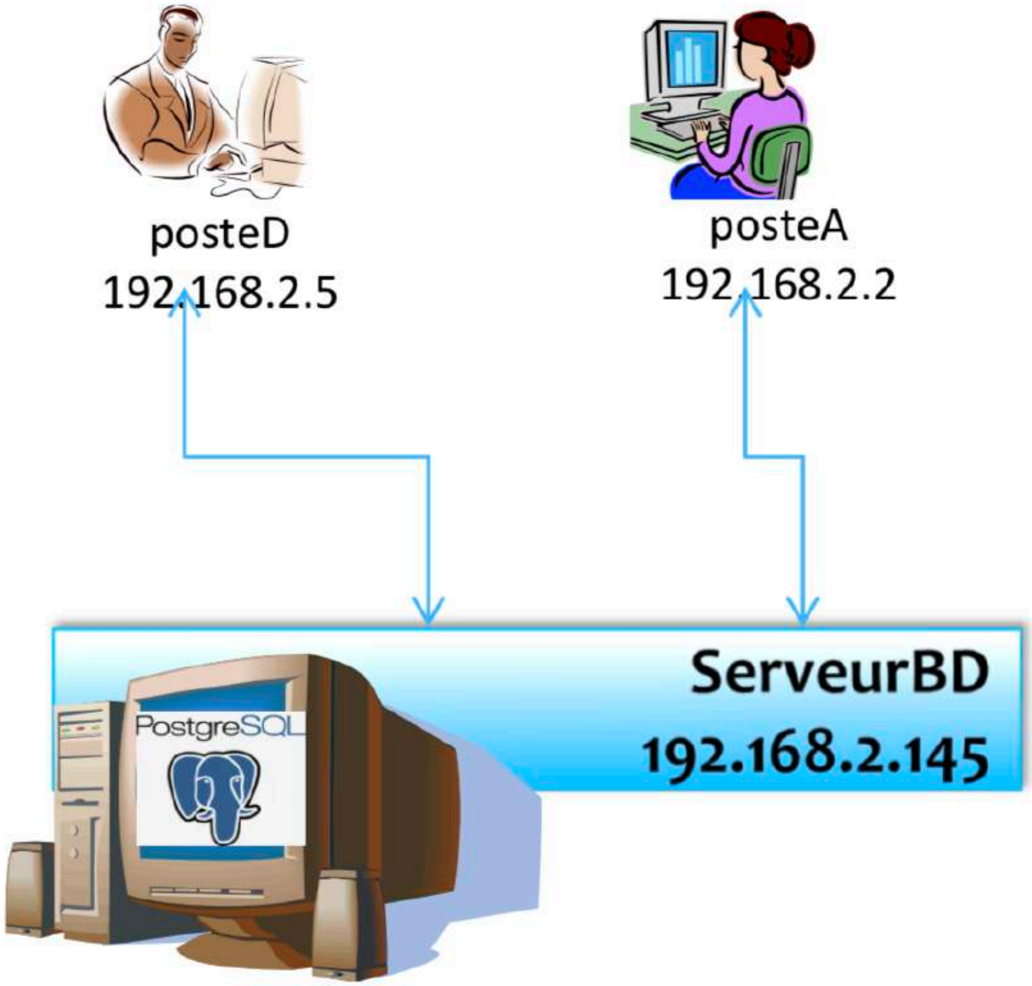
PostgreSQL Global Development Team

- Thomas Lockhart
- Jolly Chen
- Vadim Mikheev
- Jan Wieck
- Andrew Yu
- Tom Lane
- Bruce Momjian
- Marc Fournier

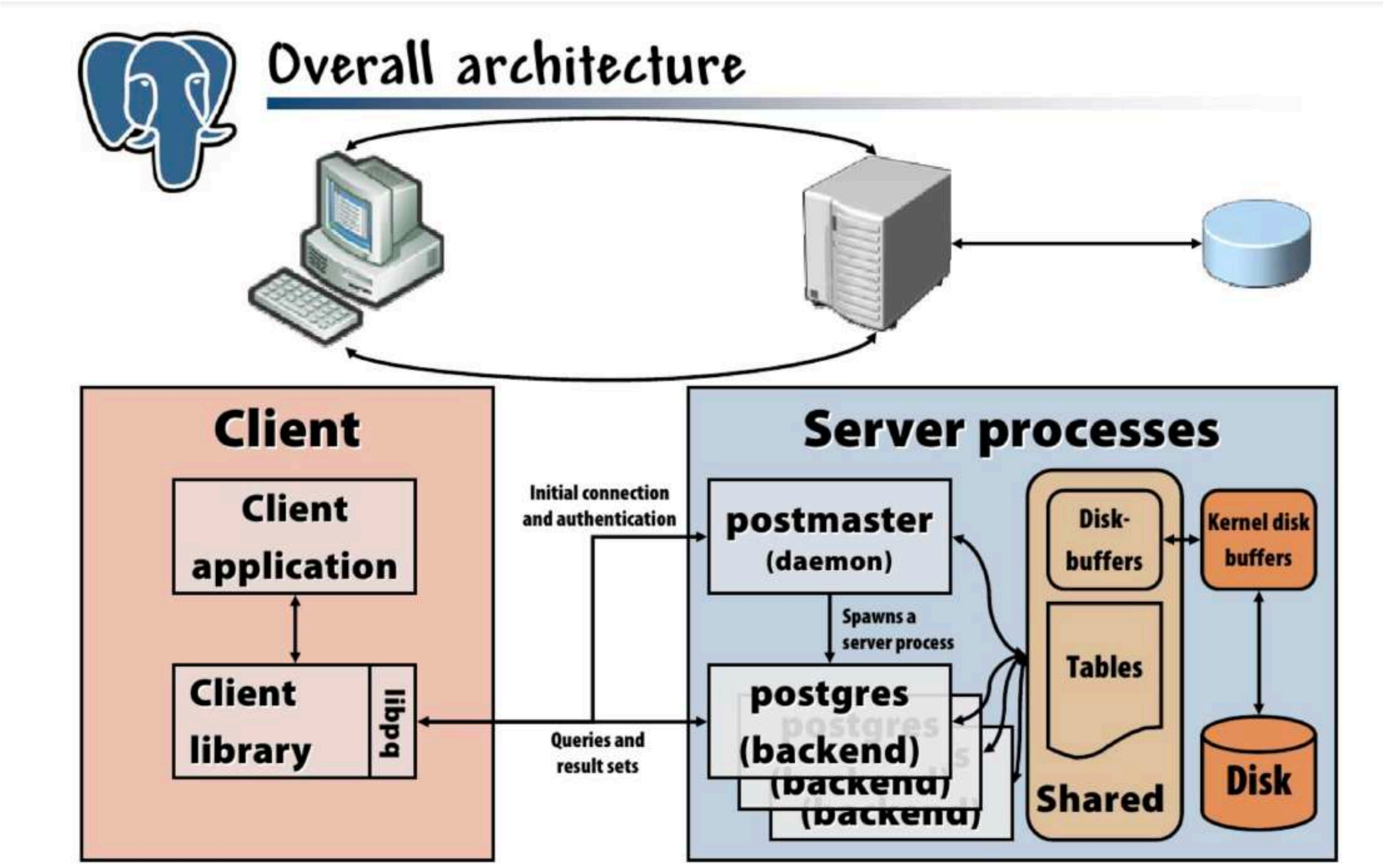
Postgres History



Client-server architecture

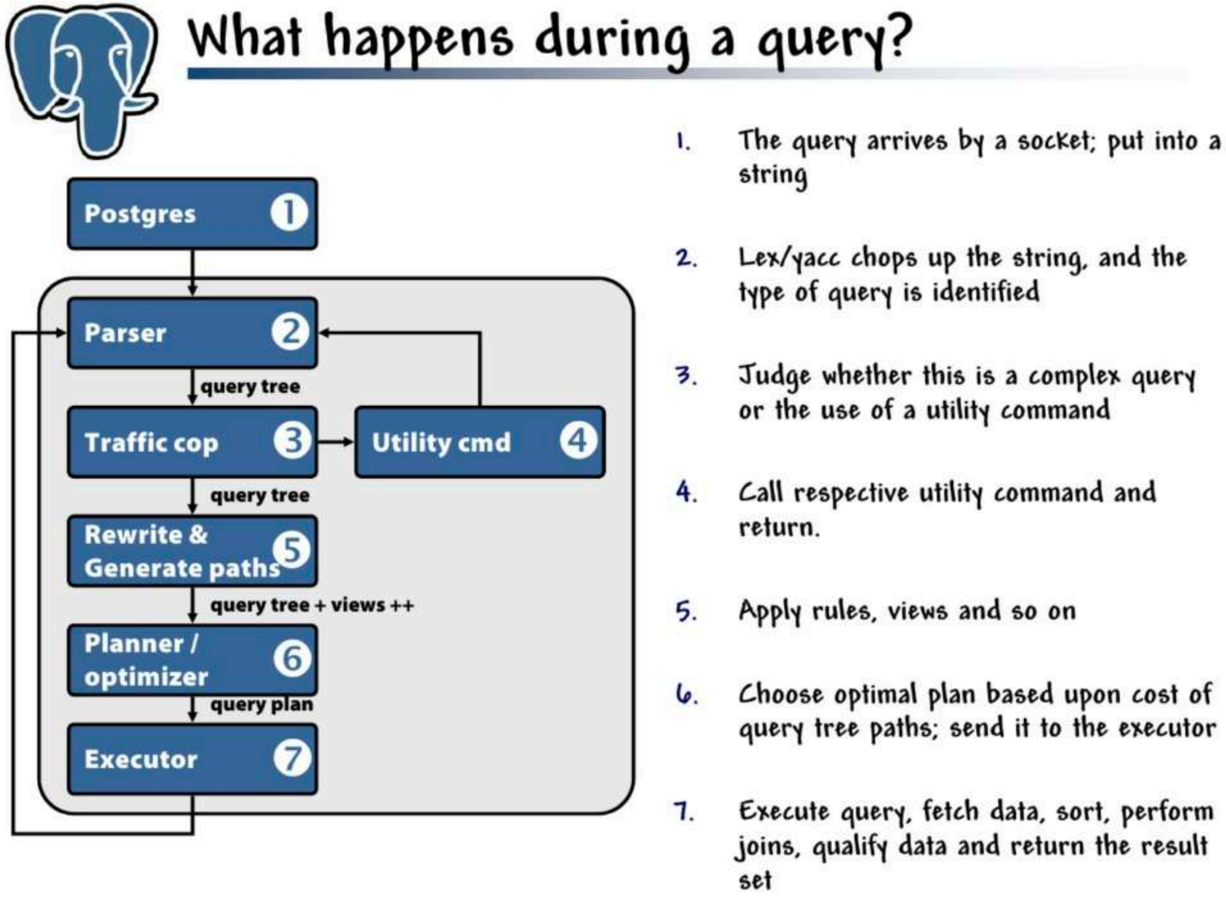


Client-server architecture



Overall architecture

Client-server architecture



What happens during a query?

Postgres 1

Parser 2

query tree

Traffic cop 3

Utility cmd 4

query tree

Rewrite & Generate paths 5

query tree + views ++

Planner / optimizer 6

query plan

Executor 7

1. The query arrives by a socket; put into a string
2. Lex/yacc chops up the string, and the type of query is identified
3. Judge whether this is a complex query or the use of a utility command
4. Call respective utility command and return.
5. Apply rules, views and so on
6. Choose optimal plan based upon cost of query tree paths; send it to the executor
7. Execute query, fetch data, sort, perform joins, qualify data and return the result set

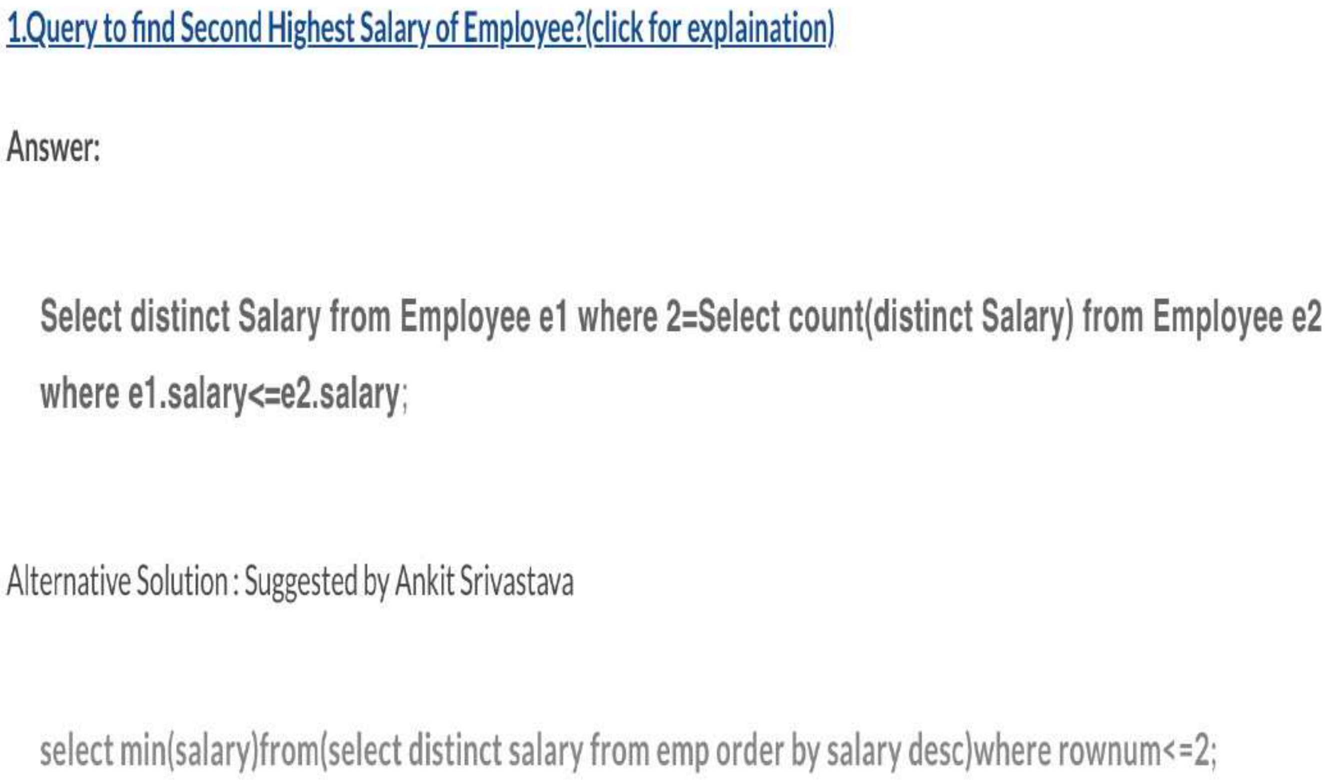
Keys feature for postgreSQL

PostgreSQL supports a large part of the SQL standard and offers many modern features including the following –

- Complex SQL queries
- SQL Sub-selects
- Foreign keys
- Trigger
- Views
- Transactions
- Multiversion concurrency control (MVCC)
- Streaming Replication (as of 9.0)
- Hot Standby (as of 9.0)

Keys feature for postgreSQL

- Complex SQL queries



1.Query to find Second Highest Salary of Employee?(click for explaination)

Answer:

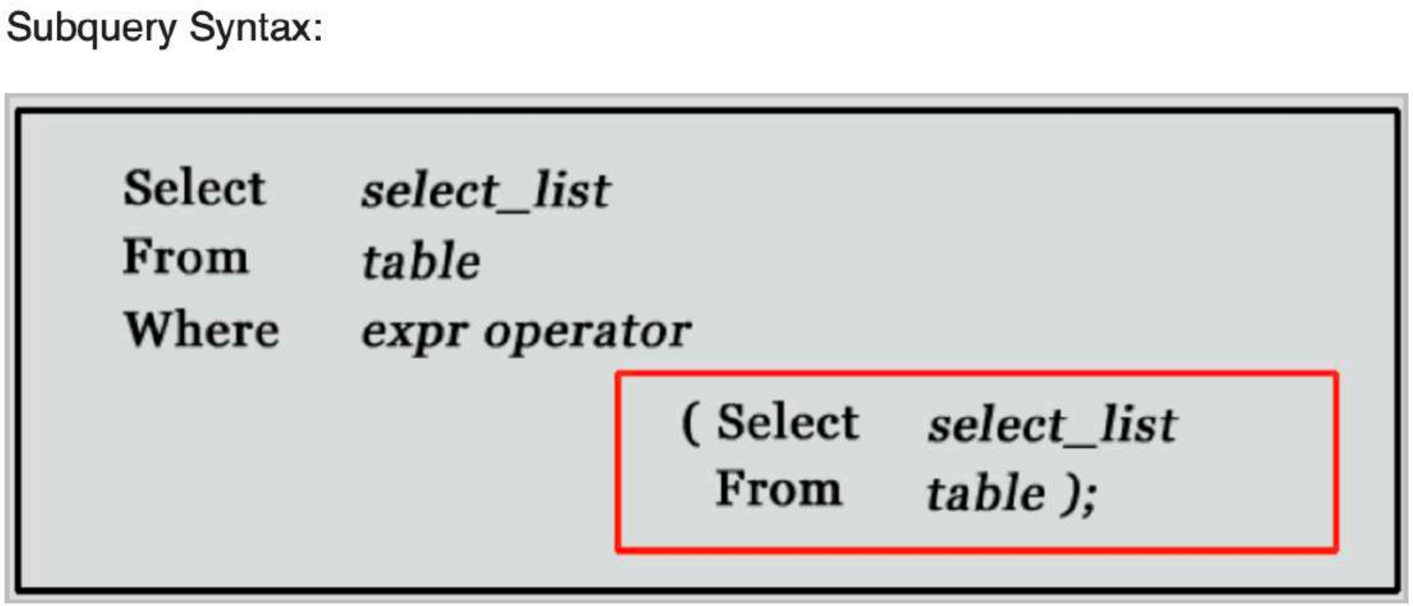
```
Select distinct Salary from Employee e1 where 2=Select count(distinct Salary) from Employee e2
where e1.salary<=e2.salary;
```

Alternative Solution : Suggested by Ankit Srivastava

```
select min(salary)from(select distinct salary from emp order by salary desc)where rownum<=2;
```

Keys feature for postgreSQL

- SQL Sub-selects

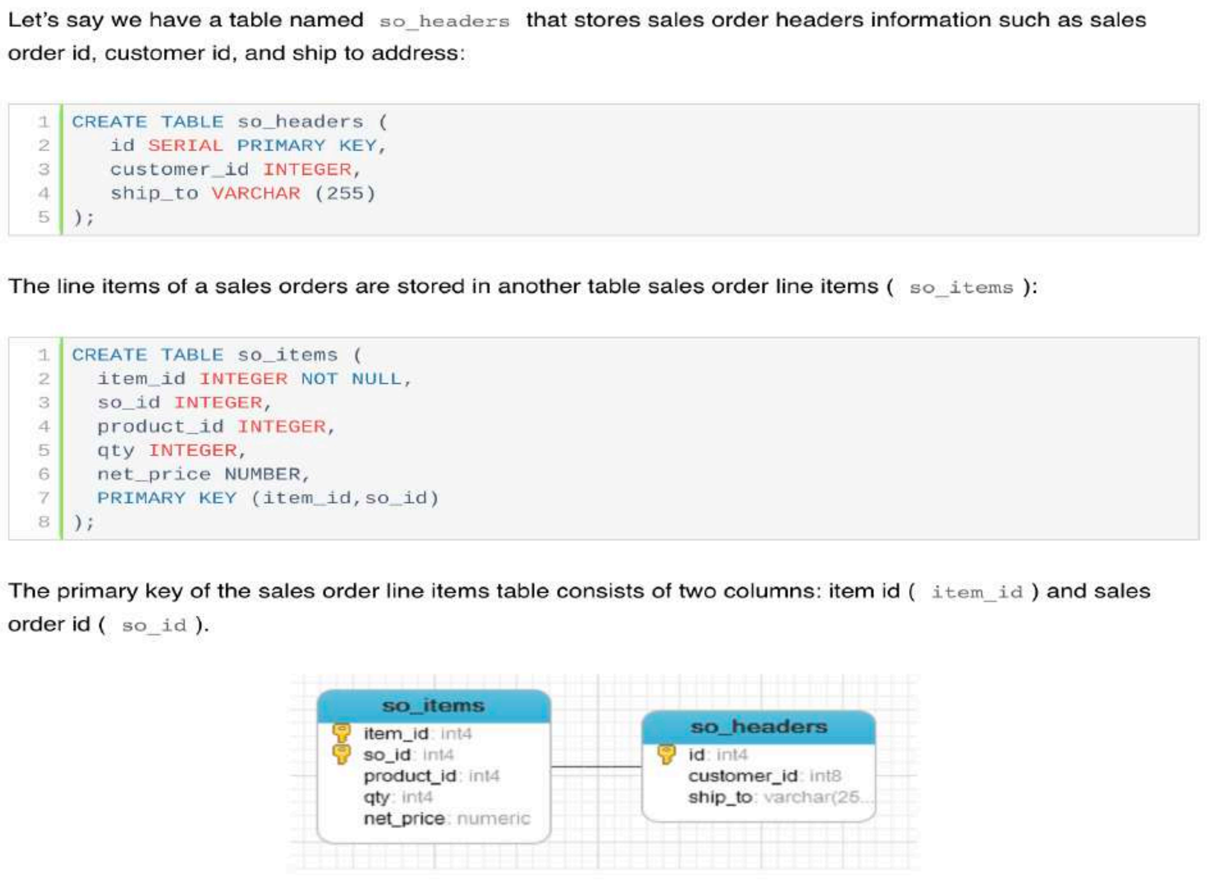


Subquery Syntax:

```
Select     select_list
From       table
Where      expr operator
                          ( Select    select_list
                            From      table );
```

Keys feature for postgreSQL

Foreign keys



Let’s say we have a table named `so_headers` that stores sales order headers information such as sales order id, customer id, and ship to address:

```sql
CREATE TABLE so_headers (
   id SERIAL PRIMARY KEY,
   customer_id INTEGER,
   ship_to VARCHAR (255)
);
```

The line items of a sales orders are stored in another table sales order line items (`so_items`):

```sql
CREATE TABLE so_items (
  item_id INTEGER NOT NULL,
  so_id INTEGER,
  product_id INTEGER,
  qty INTEGER,
  net_price NUMBER,
  PRIMARY KEY (item_id,so_id)
);
```

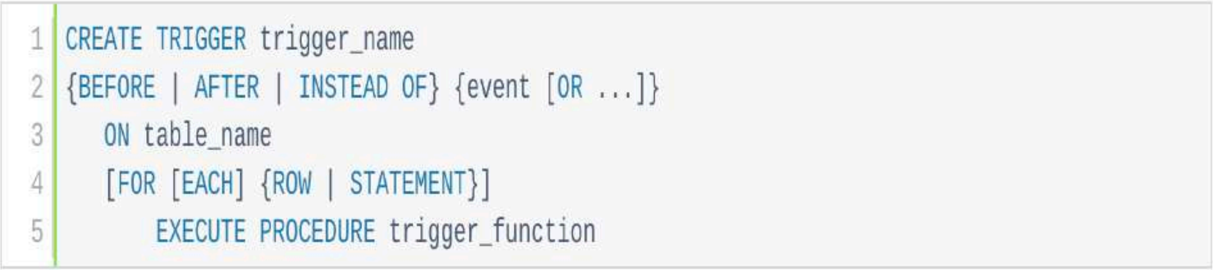
The primary key of the sales order line items table consists of two columns: item id (`item_id`) and sales order id (`so_id`).

so_items
item_id: int4
so_id: int4
product_id: int4
qty: int4
net_price: numeric

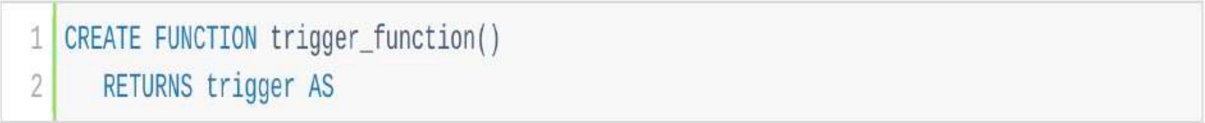
so_headers
id: int4
customer_id: int8
ship_to: varchar(25...

Keys feature for postgreSQL

- **Trigger**



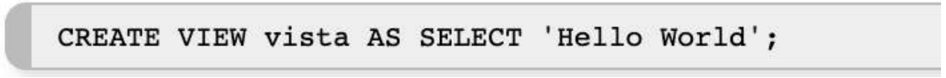
```
CREATE TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF} {event [OR ...]}
   ON table_name
   [FOR [EACH] {ROW | STATEMENT}]
       EXECUTE PROCEDURE trigger_function
```



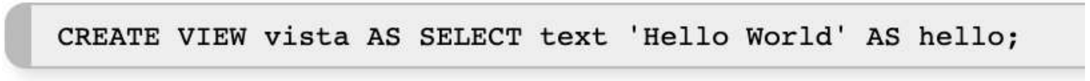
```
CREATE FUNCTION trigger_function()
   RETURNS trigger AS
```

Keys feature for postgreSQL

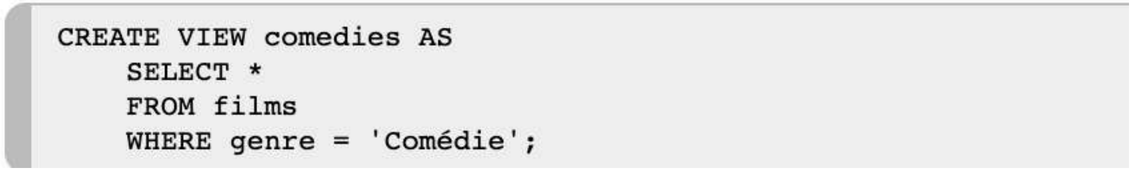
- **Views**



```
CREATE VIEW vista AS SELECT 'Hello World';
```



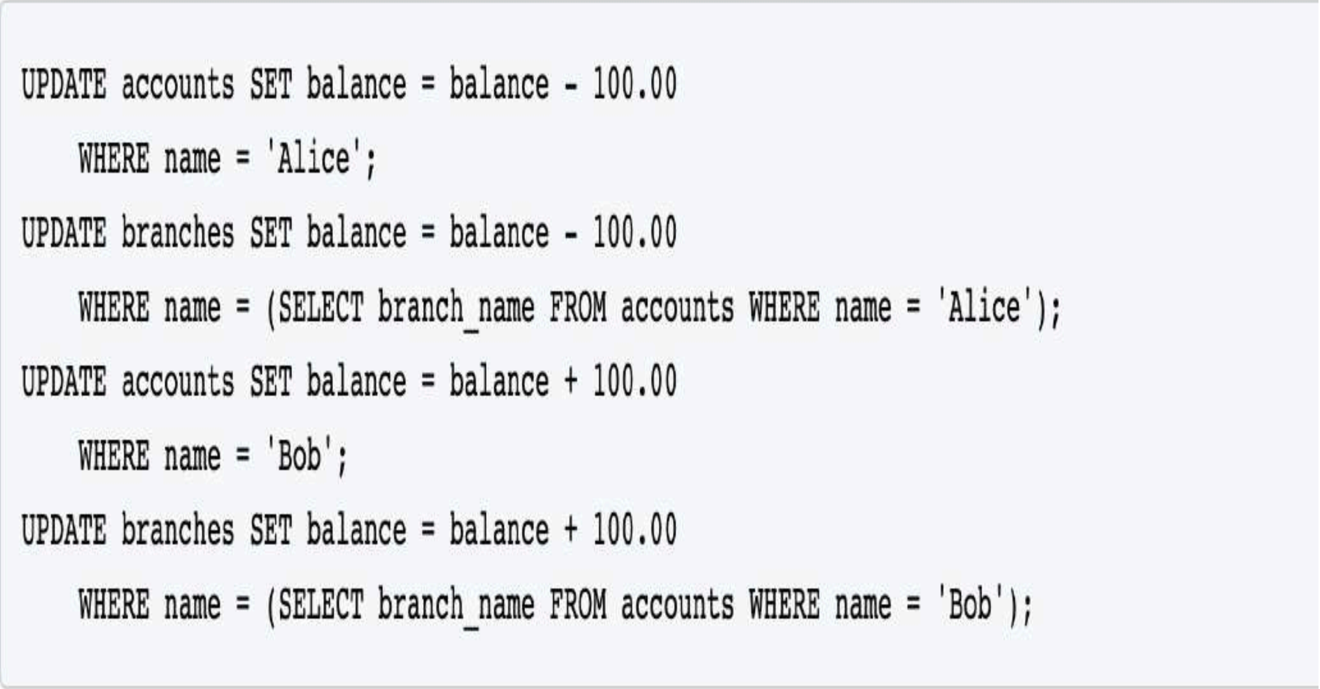
```
CREATE VIEW vista AS SELECT text 'Hello World' AS hello;
```



```
CREATE VIEW comedies AS
    SELECT *
    FROM films
    WHERE genre = 'Comédie';
```

Keys feature for postgreSQL

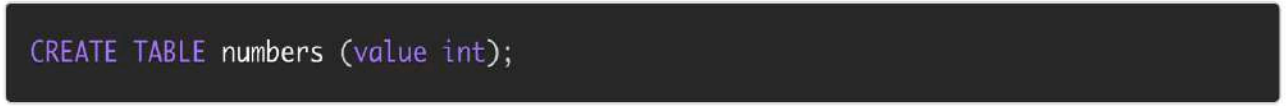
- **Transactions**



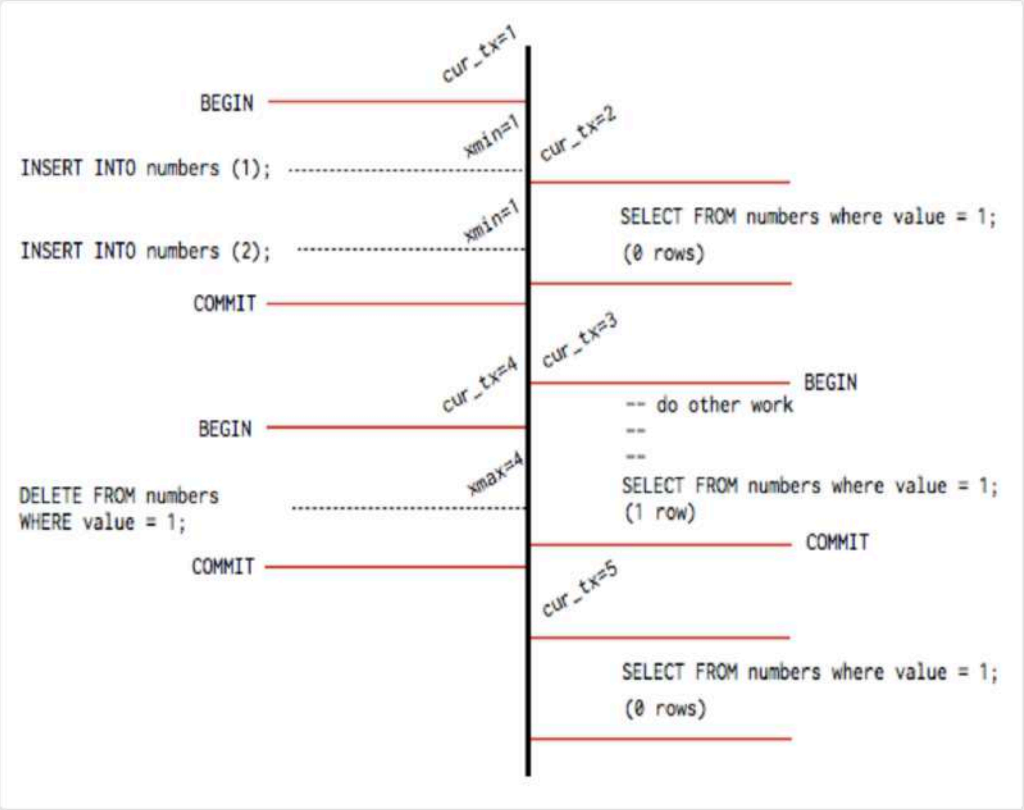
```
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
UPDATE branches SET balance = balance - 100.00
    WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Alice');
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';
UPDATE branches SET balance = balance + 100.00
    WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Bob');
```

Keys feature for postgreSQL

- **Multiversion concurrency control (MVCC)**



```
CREATE TABLE numbers (value int);
```



```
BEGIN
INSERT INTO numbers (1);
INSERT INTO numbers (2);
COMMIT

BEGIN
DELETE FROM numbers
WHERE value = 1;
COMMIT
```

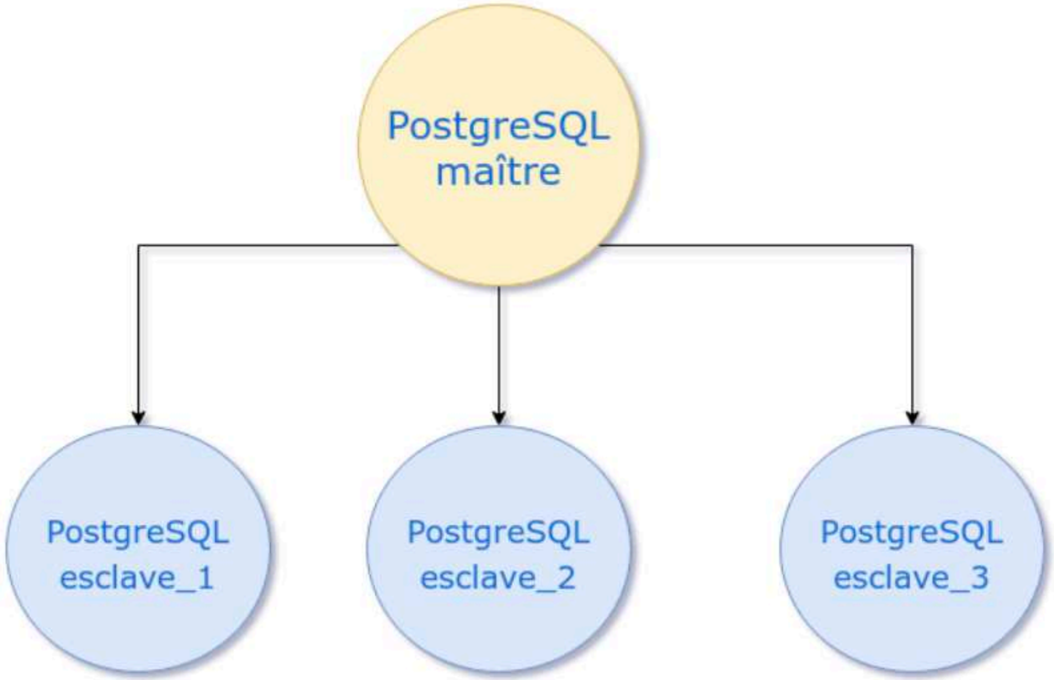
```
SELECT FROM numbers where value = 1;
(0 rows)

BEGIN
-- do other work
--
--
SELECT FROM numbers where value = 1;
(1 row)
COMMIT

SELECT FROM numbers where value = 1;
(0 rows)
```

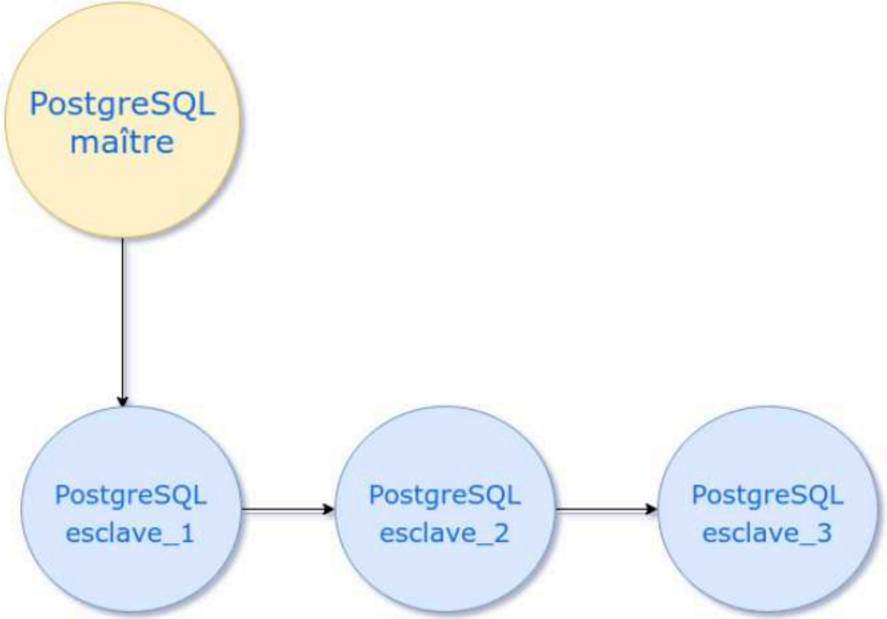
Keys feature for postgreSQL

- Streaming Replication (as of 9.0)
  - Réplication par streaming classique



Keys feature for postgreSQL

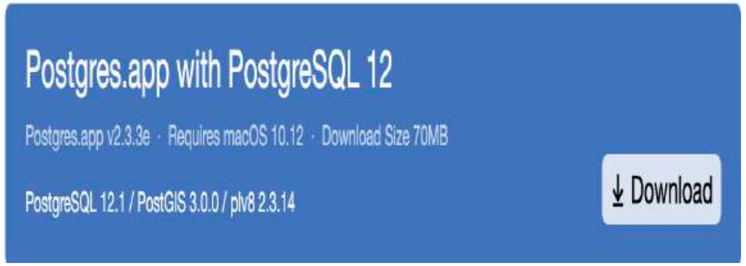
- Streaming Replication (as of 9.0)
  - Réplication par streaming en cascade



Keys feature for postgreSQL

- Hot Standby (as of 9.0)

Instalation tools postgreSQL



Postgres.app with PostgreSQL 12

Postgres.app v2.3.3e · Requires macOS 10.12 · Download Size 70MB

PostgreSQL 12.1 / PostGIS 3.0.0 / plv8 2.3.14

https://postgresapp.com/

pgAdmin 4

https://www.pgadmin.org/download/

Exercice

Enoncé de l'Exercice:

On considère le schéma relationnel suivant qui modélise une application sur la gestion d'une Bibliothèque

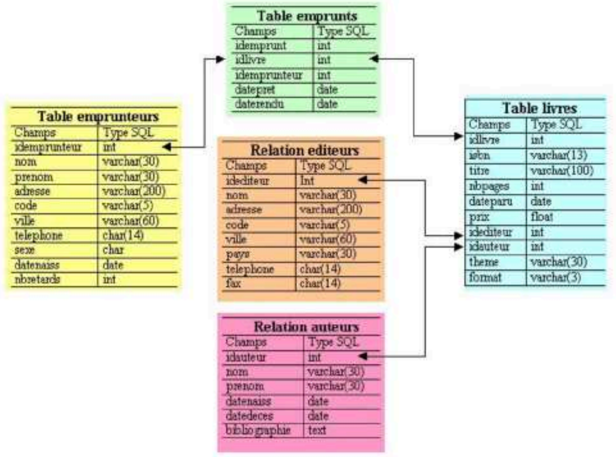


Table emprunts

Champs	Type SQL
idemprunt	int
idlivre	int
idemprunteur	int
datepret	date
daterendu	date

Table emprunteurs

Champs	Type SQL
idemprunteur	int
nom	varchar(30)
prenom	varchar(30)
adresse	varchar(200)
code	varchar(5)
ville	varchar(60)
telephone	char(14)
sexe	char
datenaiss	date
nbretards	int

Relation editeurs

Champs	Type SQL
idediteur	Int
nom	varchar(30)
adresse	varchar(200)
code	varchar(5)
ville	varchar(60)
pays	varchar(30)
telephone	char(14)
fax	char(14)

Table livres

Champs	Type SQL
idlivre	int
isbn	varchar(13)
titre	varchar(100)
nbpages	int
dateparu	date
prix	float
idediteur	int
idauteur	int
theme	varchar(30)
format	varchar(3)

Relation auteurs

Champs	Type SQL
idauteur	int
nom	varchar(30)
prenom	varchar(30)
datenaiss	date
datedeces	date
bibliographie	text

Créer les requêtes suivantes:

1- Ajoutez un champ rendu_theorique à la table livres qui correspondra à la data maximum à laquelle le livre devra être rendu. Ensuite ajoutez une règle qui calculera automatiquement cette date en se servant de la date_emprunt t en y ajoutant 15 jours.

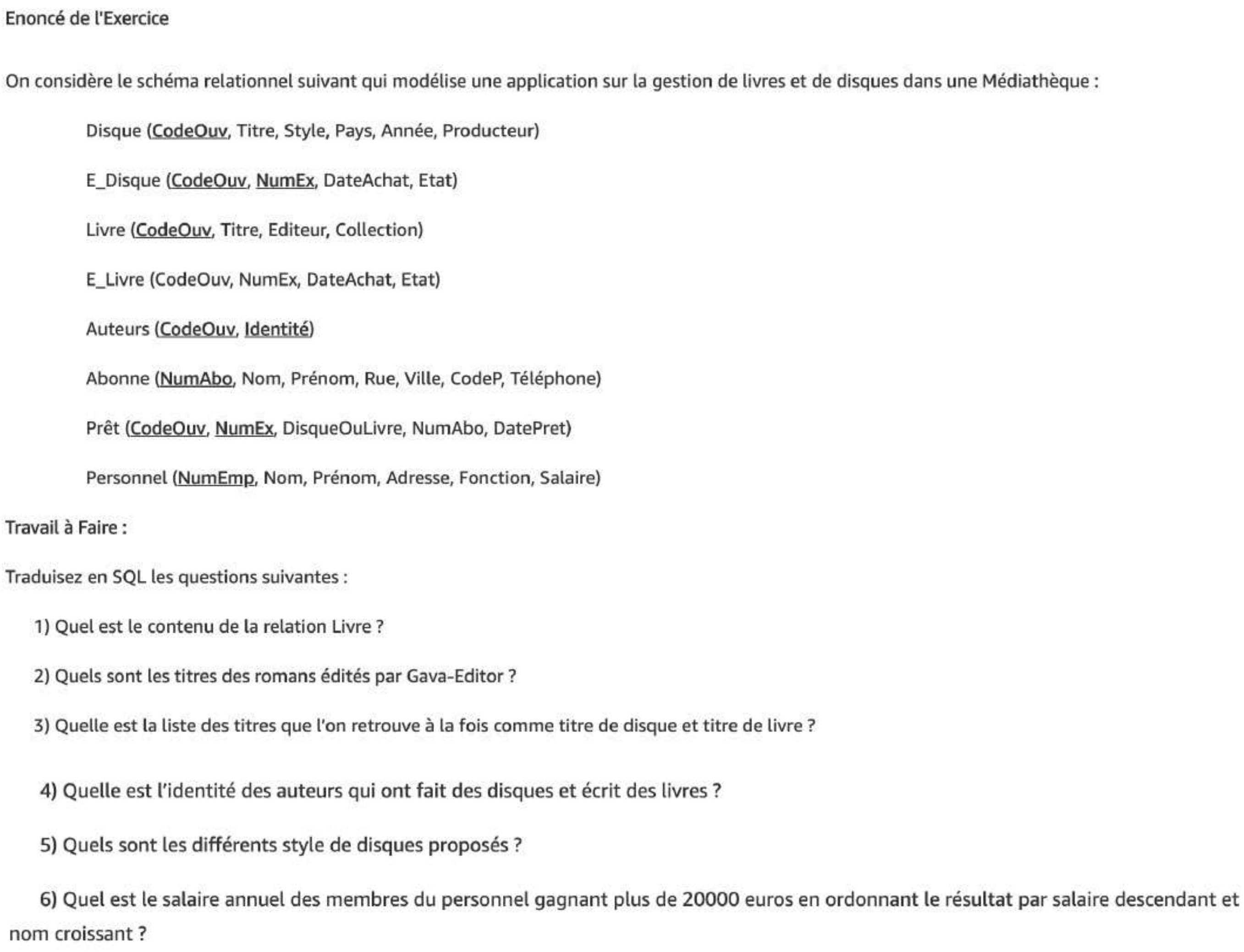
2- Effacez cette règle.

3- Créer un index sur la table emprunteurs

4- Effacez-le

5- Libérez de l'espace occupés par les enregistrements éffacés ou obsolètes.

Exercice 2



Enoncé de l'Exercice

On considère le schéma relationnel suivant qui modélise une application sur la gestion de livres et de disques dans une Médiathèque :

Disque (CodeOuv, Titre, Style, Pays, Année, Producteur)

E_Disque (CodeOuv, NumEx, DateAchat, Etat)

Livre (CodeOuv, Titre, Editeur, Collection)

E_Livre (CodeOuv, NumEx, DateAchat, Etat)

Auteurs (CodeOuv, Identité)

Abonne (NumAbo, Nom, Prénom, Rue, Ville, CodeP, Téléphone)

Prêt (CodeOuv, NumEx, DisqueOuLivre, NumAbo, DatePret)

Personnel (NumEmp, Nom, Prénom, Adresse, Fonction, Salaire)

Travail à Faire :

Traduisez en SQL les questions suivantes :

1) Quel est le contenu de la relation Livre ?

2) Quels sont les titres des romans édités par Gava-Editor ?

3) Quelle est la liste des titres que l'on retrouve à la fois comme titre de disque et titre de livre ?

4) Quelle est l'identité des auteurs qui ont fait des disques et écrit des livres ?

5) Quels sont les différents style de disques proposés ?

6) Quel est le salaire annuel des membres du personnel gagnant plus de 20000 euros en ordonnant le résultat par salaire descendant et nom croissant ?

Question