Q#1

1. Billet & numero\_representation
2. Billet & numero\_siege
3. Personne & nom-prenom
4. CREATE INDEX billet\_numero\_representation\_idx ON Billet(numero\_representation)

b) CREATE INDEX billet\_numero\_siege\_idx ON Billet(numero\_siege)

b) CREATE INDEX personne\_nom\_prenom\_idx ON personne(nom, prenom)

1. Ont pourrait vouloir le numero de representation au lieu de linker l’entièreté de la table pour faire des traitements externes

c) Un peu dans la même logique que celui-ci haut, ont pourrait vouloir récuperer le siege pour éviter joindre la table Siege en elle-même pour faire des traitements subséquent

c) Un classique !

Q#2

CREATE INDEX representation\_code\_idx ON representation(code)

CREATE INDEX conference\_code\_idx ON conference(code)

CREATE INDEX siege\_code\_idx ON siege(code)

En réalité je n’arrive pas vraiment à trouver de bonnes raisons, j’ai simplement observer que la valeur (field) se retrouve à plusieurs endroits dans la base de données.

Q#3

La cardinalité n’est pas assez élever pour justifier a l’optimisateur l’utilisation de l’index

1. Si ont rajoute plus de salle, avec un certain ‘threshold’ sa vas se mettre à jour

Q# 4

Selon moi si tu index dans historique le numero\_billet et numero\_client afin de ‘sauver’ un join :   
  
CREATE INDEX historique\_numero\_client\_numero\_billet\_idx ON historique(numero\_client, numero\_billet)

Q#5

CREATE SEQUENCE billet\_numero\_representation\_seq

OWNED BY billet.numero\_representation;

SELECT setval('billet\_numero\_representation\_seq', (SELECT MAX(numero\_representation) FROM billet));  
  
Comprend pas pourquoi on peut pas faire un sous-select pour starter la séquence : ////

Q#6

CREATE SEQUENCE representation\_code\_seq START WITH 0 MINVALUE 0 MAXVALUE 20 CYCLE;

ALTER TABLE representation

ALTER COLUMN code SET DEFAULT CONCAT('R', NEXTVAL('representation\_code\_seq'));

Q#7

Simplement parce qu’avec le temps la base de données va changer, et donc, les besoins également. Autrement dit, à moins que ton buisness marche pas, tu vas devoir modifier, delete et créé de nouveau index pour faire fonctionner ton buisness, example :   
  
tu vends des savons au début, mais 4 ans + tard tu vends plein de produit différent, bah un index pour les produits devrait être là :D

Q#8

Je dirais juste que rendu à un certain moment l’optimisation à ces limites, c’est comme un dév web qui veut rendre son site super rapide, (I mean le web c’est lent de base) pis sa ramène aussi à la logique que si ton utilisateur doit attendre 2min pour une opération, que tu optimise a 1 :30, malgré le gain relativement énorme de 25% bah sa prend quand même 1 :30, tandis que si ta mettons une opé de 3sec, que tu descend sa a 0.75sec, j’peut te dire que tu vas gagner ton augmentation ahahah, mais c’est relativement ce que je crois.

Q#9

Je dirais que c’est d’avantage parce que contrairement au index sur les PK (qui sont extrêmement utile de manière générale) les séquences sont beaucoup plus à la discretion des gens qui travaille sur la BD, les index se créé généralement tous de la même manière également pis sa sa fait que c’est beaucoup plus ‘machine-like’ a générer tandis que les séquences sont plus complexe et ont un certain niveau d’abstraction dans la manière de les créé et les raisons pour lesquelles elle vont être utilisé.

Q#10

Alrighttttt donc la requête original est :

EXPLAIN ANALYZE SELECT c.nom, p.prenom || ' ' || p.nom, r.date, s.nom, s.adresse, COUNT(b.numero)

FROM salle s

JOIN representation r ON r.numero\_salle = s.numero

JOIN conference c ON c.numero = r.numero\_conference

JOIN conference\_conferencier cc ON cc.numero\_conference = c.numero

JOIN personne p ON p.numero = cc.numero\_conferencier

JOIN billet b ON b.numero\_representation = r.numero

JOIN historique h ON h.numero\_billet = b.numero AND h.annule = false

WHERE p.prenom = 'Daren' AND p.nom = 'Gouth'

GROUP BY c.nom, p.numero, r.date, s.nom, s.adresse

ORDER BY r.date DESC  
  
Sa me donne un analyze :   
  
cost=37.61..37.68

rows=2

width=518

actual time=2.684..2.739

rows=23

loops=1

Execution Time: 3.095 ms

Bon première affaire que je voit c’est :  
  
SELECT c.nom, p.prenom || ' ' || p.nom, r.date, s.nom, s.adresse, COUNT(b.numero)

FROM personne p

JOIN conference\_conferencier cc ON p.numero = cc.numero\_conferencier

JOIN conference c ON c.numero = cc.numero\_conference

JOIN representation r ON r.numero\_conference = c.numero

JOIN salle s ON r.numero\_salle = s.numero

JOIN billet b ON b.numero\_representation = r.numero

JOIN historique h ON h.numero\_billet = b.numero AND h.annule = false

WHERE p.prenom = 'Daren' AND p.nom = 'Gouth'

GROUP BY c.nom, p.prenom, p.nom, r.date, s.nom, s.adresse

ORDER BY r.date DESC  
  
Fake c’est juste de passer par la table personne en premier, pis de réorganiser les JOINS, en fesant sa comme sa j’ai remarquer que sa vas réduire le nombre de filtre à faire

cost: 41.93..42.18

rows: 7

width: 527

actual time: 1.973..2.043

rows (réel): 23

loops: 1

Execution Time: 2.480 ms  
  
I guess que sa aura vallu la peine, sachant que le temps est passer de 3.095ms à 2.480ms

Ensuite l’autre chose que je pense qui pourrait être optimiser :   
  
SELECT c.nom, r.date, COUNT(b.numero) AS tickets\_sold

FROM personne p

JOIN conference\_conferencier cc ON p.numero = cc.numero\_conferencier

JOIN conference c ON c.numero = cc.numero\_conference

JOIN representation r ON r.numero\_conference = c.numero

JOIN billet b ON b.numero\_representation = r.numero

JOIN historique h ON h.numero\_billet = b.numero AND h.annule = false

WHERE p.prenom = 'Daren' AND p.nom = 'Gouth'

GROUP BY c.nom, r.date

ORDER BY r.date DESC;  
  
  
Plus grosse update ever, me suis rendu compte que le nom ont l’avait en double, ensuite ont peut aussi enlever l’addresse parce que ont veut juste le nombre de billet vendu, donc ont supprime un JOIN, en roulant sa j’obtiens :   
  
cost: 40.64..40.78

rows: 7

width: 46

actual time: 1.620..1.655

rows (réel): 23

loops: 1

Execution Time: 1.852 ms

Donc encore une fois, ont a réduit le temps d’execution et la ‘bonne nouvelle’ c’est que le cost à déscendu par rapport à la derniere query mais tjrs supérieur à l’original, par contre le temps à 1.852ms, presque la moitié du temps original :DDDDDD et la width aussi assez insane

Pour le reste je voit pas vraiment, j’ai tenter avec des index, pour les confériencier notamment, mais sa na pas fait de grosse différence, ce qui fait relativement du sens les confériencer c’Est relativement petit comme dataset et en plus leur cardinalité est relativement faible et je pense que l’optimisateur passe outre vue que c’est juste des +1 sur le ID qui à déjà la seq