ESIR1 / BDD jeudi 14 avril 2015
Contrôle continu 30mn

Répondre de façon **concise mais précise**

- Concis veut dire court, c'est-à-dire plus petit que la boîte de réponse à la question.
- Précis veut dire avec suffisamment de détail, c'est-à-dire pas trop plus petit que la boîte de réponse à la question.
- Conclusion : essayez de vous mettre à votre avantage. Montrez que vous avez compris la question, et que vous connaissez la réponse. Un contrôle, continu ou pas, c'est fait pour vérifier vos acquis, pas votre capacité à éluder les questions.

aux questions suivantes en utilisant les cadres. Aucun document n'est autorisé, hormis un aide-mémoire manuscrit de format A4 recto seul constitué par chaque étudiant. Ce sujet comporte 2 pages. Ne pas oublier d'indiquer vos noms et prénoms.

Certaines questions font référence au schéma suivant :

```
Client( num-client, nom-client )
Produit( num-produit, nom-produit, prix-produit )
Commande( num-commande, num-client )
Item( num-commande, num-produit, qté-produit )
Facture( num-facture, num-commande )
```

Donner une expression en algèbre relationnelle de l'ensemble des triplets (nom-produit, qté-produit, prix-produit) des factures d'un client de nom donné. N'hésitez pas à découper l'expression en sous-expressions.

[Vu en cours/TD ; c'est pour ceux qui affirment qu'il n'y a pas de question sur les TD. Ce n'est évidemment pas rigoureusement un exercice travaillé en TD, mais c'est une question qui relève de l'expertise acquise en TD]

Il suffisait d'une jointure entre les tables qui relient nom-client, nom-produit, qté-produit et prix-produit.

Ne pas confondre num-client et nom-client. Et ne pas répondre en SQL ; la compétence en SQL ne compense pas la compétence en algèbre relationnelle. C'est parce qu'ils raisonnent en AR que les interpréteurs SQL arrivent à répondre efficacement aux requêtes. Attention aux Oméga et Théta qui n'ont rien à voir avec la notation de l'algèbre relationnelle. Pi comme P(rojection) et Sigma comme S(élection) c'est facile, non ?

Donner une expression en SQL qui calcule les totaux des factures d'un client donné. N'hésitez pas à découper l'expression en sous-expressions.

[Vu en cours/TD/TP]

Il suffisait d'une jointure entre les tables qui relient nom-client, nom-produit, qté-produit et prix-produit, mais aussi Facture pour réaliser la spécification « factures d'un client donné ». Il fallait aussi utiliser proprement des fonctions de calcul et d'agrégation. Mais aussi ne pas oublier de faire un groupage par facture pour ne pas calculer le total de toutes les factures, mais bien les « totaux des factures d'un client donné » comme demandé. N'oubliez-pas qu'avec les fonctions d'agrégation on n'obtient qu'une ligne, sauf avec le groupage où on obtient une ligne par groupe. Un groupage par client ne sert à rien puisque la requête ne concerne qu'un client. Pensez aussi que des totaux sans clé de lecture ne riment à rien ; il faut donc agréger et projeter sur la colonne de groupage. Par contre, projeter sur d'autres colonnes que celle de groupage n'a aucun sens.

Pour l'anecdote, j'ai vu une proposition de sommer les numéros de factures ! Et pourquoi pas faire la moyenne de vos numéros de téléphone ?

Quel problème résout JDBC ?

[Vu en cours/TP]

Je vois bien que certains ont du mal à distinguer le pourquoi du comment, mais c'est pourtant ce qui vous donne l'autorité d'un ingénieur sur un technicien. C'est toujours la même chose ; il faut répondre exactement à la question. On ne répond pas à un pourquoi par un comment, ni à un quel est la définition de par ça sert à.

Attention : JDBC n'a rien à voir avec BCNF. Cette erreur, surtout répétée, m'intrigue beaucoup ; un regard un peu trop appuyé sur la copie d'un voisin ? Évidemment, il y a le BC, mais quand même n'impliquez pas NBC, ABC et EBCDIC dans nos affaires. JDBC n'a rien à voir non plus avec interface graphique ; une application peut bien sûr avoir et une interface vers une base de données, et une interface graphique vers les utilisateurs, mais JDBC ne s'occupe que de la première. Autre bizarrerie : relier JDBC et transaction. JDBC doit bien sûr s'en préoccuper, mais on ne l'a pas traité, et de toute façon, il ne fait rien pour vous, c'est à vous de décider quoi faire. Dernière bizarrerie : affirmer que JDBC résout le problème du shell ???

Quelle solution apporte JDBC ?

[Vu en cours/TP]

Une interface Java vers des méthodes programmées par chaque fournisseur de BD, ces méthodes réalisant la connexion, l'envoi de requête et l'analyse des réponses.

En tout cas, pas de service d'interface homme-machine, encore moins graphique.

Je dois signaler le cas intéressant de deux poètes qui ont été inspirés par la même muse. Ils ont répondu par une très belle phrase, presque la même, l'un évoquant une passerelle l'autre un pont ! Malheureusement, la même muse leur a inspiré des réponses identiques, mais idiotes, à d'autres questions. C'est vraiment dommage pour leur réputation. Moralité, bien choisir ses voisins.

Rappeler ce qu'est une dépendance fonctionnelle.

[Vu en cours/TD/TP]

Juste dire ce que c'est, pas à quoi ça sert. Je me moque dans une certaine mesure que vous répondiez à une question que je n'ai pas posé, mais ça ne remplace pas la réponse à la question que j'ai posée. Par contre dire que c'est un lien entre deux ensembles d'attributs répond à la question mais ne suffit pas, il faut aussi dire en quoi consiste le lien (précision !).

Ne pas confondre avec clé. Les dépendances fonctionnelles permettent de tenir les raisonnements qui aboutissent à déterminer les clés. Les dépendances fonctionnelles sont en entrée de la conception, et les clés en sortie.

Ne pas non plus se tromper sur la lecture de A -> B. Strictement parlé, ça veut juste dire que si (a, b1) et (a, b2) sont dans cette relation, alors b1 = b2. Et c'est en ce sens seulement qu'on dit que A détermine B. En particulier, connaissant 'a' tout seul, il n'y a pas forcément moyen de connaître 'b' si il n'y a pas déjà de (a, b) dans la table. Donc A détermine B au sens où à chaque valeur de A ne correspond qu'une valeur de B, mais pas au sens au connaissant une valeur de A une règle de calcul permettrait de connaître la valeur correspondante de B (cas des règles de calcul des tableurs). Par exemple, R I -> U est vrai au second sens (et au premier évidemment), mais num-facture -> num-client n'est vrai qu'au premier sens car il n'y a pas de règle de calcul qui donne un numéro de client à partir d'un numéro de facture. Pour conclure, A -> B indique une relation qui peut être arbitraire entre A et B, mais qui néanmoins est une fonction. Et la meilleure représentation d'une fonction arbitraire est souvent une table, d'où un boulevard pour les BD (annuaires, catalogues, etc).

Rappeler les axiomes de Armstrong.

[Vu en cours/TD]

Avec leur nom c'est mieux, mais uniquement les noms ne suffit pas.

À quoi servent les axiomes de Armstrong ?

[Vu en cours/TD]

D'abord et avant tout à raisonner formellement sur les dépendances fonctionnelles. Qui dit axiome dit logique, et c'est parce que c'est complètement formalisé que c'est automatisable. On peut dire après à quoi sert de raisonner sur les dépendances fonctionnelles : les trouver toutes, même celles qui sont implicites, en rechercher une base élémentaire, trouver des clés, piloter la normalisation.

On souhaite apporter les modifications suivantes au schéma. Pour chaque modification dire brièvement de quelle façon la réaliser et pourquoi de cette façon. Ne pas chercher à produire le code SQL qui réalise la modification.

[Vu en cours/TD/TP]

- **Chaque client n'a qu'une adresse de facturation.**

  Donc client -> adresseF. On a déjà une clé num-client pour représenter cette dépendance ; pas la peine d'inventer autre chose.

- **Chaque client a plusieurs adresses de livraison qui lui sont propres.**

  Donc adresseL -> client. Ici ce n'est pas très raisonnable de faire directement de l'adresse une clé. Il vaut mieux créer un attribut num-adresse qui servira de clé, l'adresse proprement dite n'étant que la description de la clé. Surtout ne pas créer des attributs adresse1, adresse2, etc, pour toutes les adresses de livraison d'un client. Rappel : les tables sont flexibles en hauteur (nb lignes) mais rigides en largeur (nb colonnes).

- **Chaque commande a une adresse de livraison et une adresse de facturation.**

  L'adresse de facturation c'est facile, ça ne peut être que celle du client ; ne rien faire. Pour l'adresse de livraison, ça doit être une des adresses de livraison du client, utiliser la clé de l'adresse visée. Mettre directement l'adresse oblige à trop de redondance ; ex. quand le code postal change, il faudrait aller le vérifier dans toutes les commandes en cours.

- **Chaque commande peut être livrée en plusieurs lots qu'on veut pouvoir suivre individuellement. Un item ne peut pas être divisé.**

  J'attendais une solution du genre intercaler un table Lot entre Commande et Item de telle sorte qu'une commande soit constituée de lots, et chaque lot d'items. Le suivi aurait pu alors consister en une table Suivi(num-lot, info-suivi-lot). Certains ont proposé une autre solution, tout à fait recevable, de noter un numéro de lot dans la table Item. Dans ce cas, il faut quand même s'assurer que la table Suivi permette de vérifier qu'il s'agit bien d'un lot de la commande ; on doit avoir lot -> commande dans tous les cas. C'est pourquoi je préfère ma solution, même si je n'ai pas pénalisé l'autre.