INFO-H-303: Synthèse de questions THÉORIQUES issues des anciens examens et du livre de ref. fundamentals of database systems, third edition, Elmasri,Navathe

Arabella Brayer

Table des matières

1 janvier 2008

1.1 question 1

Une manière de supprimer la généralisation dans un modèle entité-association (EA) est de conserver la super-entité, d'y ajouter les attributs et les relations des sous-entités, d'y ajouter un attribut 'type' et ajouter toutes les contraintes nécessaires. Donner deux autres manières de supprimer la généralisation dans un modèle EA. Spécifier et justifier pour chaque type de généralisation si chacun des deux cas peut ou non s'appliquer.

(tentative de) réponse : Les deux autres façons sont :

- Conserver la sous-identité
- Modéliser avec des relations ordinaires

(slides chapitre 5 du cours)

Conserver la sous-identité : Propage à chaque sous-entité les attributs (et identificateurs) et les relations de la super-entité.
Il y a redondance des attributs et relations. Les applications deviennent plus complexes, et il y a des contraintes inter-relations.

Cette méthode n'est applicable que dans le cas de l'héritage total et exclusif.

Modéliser avec des relations ordinaires : On créé de nouvelles relations. Elles sont mandataires de la sous-entité et optionnelles pour la super-entité. Les sous-identités deviennent des entités faibles, la super-entité devient une simple entité. Les attributs et anciennes relations sont conservées.

On ajoute de la redondance (les nouvelles relations ont la même signification). Les contraintes doivent exprimer le type de généralisation (t, e), certaines opérations deviennent plus complexes.

1.2 question 2

Donner et expliquer les trois manières vues au cours de gérer la suppression en conservant l'intégrité référentielle dans le modèle relationnel. Illustrer à l'aide d'exemples bien choisis chacun des cas.

1.3 question 3

Définir l'opération de division de l'algèbre relationnelle. Donner un équivalent en calcul relationnel tuple.

1.4 question 4

Donner un exemple bien choisi d'utilisation de la jointure externe (outer join).

1.5 question 5

Définir troisième forme normale et forme normale de Boyce-Codd. Donner un exemple de relation qui est :

- en deuxième mais pas en troisième forme normale,

- en troisième mais pas en forme normale de Boyce-Codd.

Donner les anomalies de mise à jour pour chacun des deux exemples.

2 Janvier 2009

2.1 question 1

A l'aide d'un exemple, expliquer comment remplacer une association ternaire par un modèle équivalent sans relation ternaire. Donner la traduction relationnelle des 2 modélisations.

2.2 question 2

Expliquer la notion de contrainte d'intégrité du modèle relationnel. Citer et expliquer brièvement les différents types de contraintes dans le modèle relationnel : celles présentes dans le modèle et celles annexées à celui-ci.

2.3 question 3

Définir l'opération de produit cartésien en algèbre relationnelle. Expliquer le lien entre cette opération et la jointure, la jointure naturelle et la jointure externe.

Réponse : Slides chapitre 6, Relational Algebra
Le produit cartésien associe chaque tuple d'une relation avec tous les tuples d'une autre.
La jointure associe deux relations sur base d'une condition :
La jointure naturelle lie deux relations sur la condition d'égalité de deux attributs (equijoin), et ne retourne qu'un des deux champs :
$T *_{A=B} S$ ne va retourner que A.
$R * S$ va tester l'égalité des attributs qui ont le même nom.

La jointure externe, quant à elle, sélectionner les éléments d'une table indépendamment du fait qu'ils soient contenus ou non dans la deuxième table.

2.4 question 4

Expliquer la séquence intuitive d'exécution (sémantique opérationnelle) d'une requête de type : SELECT (...) FROM (...) WHERE (...) GROUP BY (...) HAVING (...)

2.5 question 5

Définir la notion de dépendance multi-valuée. Définir également la quatrième forme normale. Expliquer le lien entre les dépendances multi-valuées et la première forme normale.

3 Juin 2013

3.1 Modèle relationnel

Une manière de supprimer la généralisation dans un modèle entité-association (EA) est de conserver la super-entité, d'y ajouter les attributs et les relations des sous-entités, d'y ajouter un attribut "type", et ajouter toutes les contraintes nécessaires. Donner deux autres manières

de supprimer la généralisation dans un modèle EA. Spécifier et justifier pour chaque type de généralisation si chacun des deux cas peut ou non s'appliquer.

redo : Les deux autres façons sont :
- Conserver la sous-identité
- Modéliser avec des relations ordinaires

(slides chapitre 5 du cours)

Conserver la sous-identité : Propage à chaque sous-entité les attributs (et identificateurs) et les relations de la super-entité.
Il y a redondance des attributs et relations. Les applications deviennent plus complexes, et il y a des contraintes inter-relations.

Cette méthode n'est applicable que dans le cas de l'héritage total et exclusif.

Modéliser avec des relations ordinaires : On créé de nouvelles relations. Elles sont mandataires de la sous-entité et optionnelles pour la super-entité. Les sous-identités deviennent des entités faibles, la super-entité devient une simple entité. Les attributs et anciennes relations sont conservées.

On ajoute de la redondance (les nouvelles relations ont la même signification). Les contraintes doivent exprimer le type de généralisation (t, e), certaines opérations deviennent plus complexes.

3.2 Calcul tuple

(a) Donner la forme générale d'une requête en calcul relationnel tuple.
(b) Le quantificateur universel ($\forall$) est souvent associé à l'implication ($\Longrightarrow$). Expliquer.

3.3 SQL

Donner la syntaxe et le contexte dans lequel la clause HAVING du langage SQL peut être utilisée. Illustrez par un exemple.

3.4 Normalisation

Définissez la notion de *dépendance fonctionnelle*.
Définissez les trois premières formes normales et donnez trois exemples illustrant ces différentes formes.

4 Juin 2015

4.1 Modèle entité association

Donnez 3 manières de supprimer la généralisation dans un modèle entité-association.
Pour chacune de ces trois manières, spécifiez si elle est adaptée ou non lorsque la généralisation est du type *partiel*.
Quelle contrainte supplémentaire doit-on ajouter lorsque la généralisation est de type *exclusif* ?

réponse incomplète Les trois façons sont :

- Conserver la super-entité
- Conserver la sous-identité
- Modéliser avec des relations ordinaires

(slides chapitre 5 du cours)

Conserver la super-entité : On ajoute les attributs et les relations des sous-entités, on ajoute un attribut 'type' et on ajoute toutes les contraintes nécessaires.

Conserver la sous-identité : Propage à chaque sous-entité les attributs (et identificateurs) et les relations de la super-entité.
Il y a redondance des attributs et relations. Les applications deviennent plus complexes, et il y a des contraintes inter-relations.

Cette méthode n'est applicable que dans le cas de l'héritage total et exclusif.

Modéliser avec des relations ordinaires : On créé de nouvelles relations. Elles sont mandataires de la sous-entité et optionnelles pour la super-entité. Les sous-identités deviennent des entités faibles, la super-entité devient une simple entité. Les attributs et anciennes relations sont conservées.

On ajoute de la redondance (les nouvelles relations ont la même signification). Les contraintes doivent exprimer le type de généralisation (t, e), certaines opérations deviennent plus complexes.

4.2 Algèbre relationnelle

Expliquer comment se passer de l'opération de division en algèbre relationnelle.

slides chapitre 5, fin la division peut être remplacée par la différence et le produit cartésien :
$R \div S = T$ est équivalent à
$T1 \leftarrow \pi_A(R)$
$T2 \leftarrow \pi\_A((T1xS) - R)$
$T \leftarrow T1 - T2$

4.3 SQL

Soit les relations Employee(SSN, Name, Salary) et WorksOn(ESSN, PNO, Hours), où WorksOn.ESSN référence Employee.SSN. Parmi les trois requêtes suivantes, lesquelles expriment la question : *"Donner la moyenne des salaires des employés qui travaillent plus de 10 heures sur un projet ? "* Justifiez à l'aide d'un exemple les requêtes incorrectes.

1.
```
SELECT AVG(Salary)
FROM Employee, WorksOn
WHERE SSN = ESSN AND Hours > 10

SELECT AVG(DISTINCT Salary)
FROM Employee, WorksOn
WHERE SSN = ESSN AND Hours > 10
```

3. SELECT AVG(DISTINCT Salary)
FROM Employee, WorksOn
WHERE SSN IN
(SELECT ESSN
FROM WorksOn
WHERE Hours > 10)

4.4 Normalisation

Définissez la notion de *Dépendance fonctionnelle*.
Définissez les trois premières formes normales et donnez trois exemples illustrant ces différentes formes.

5 Questions chapitre 1

5.1 Définir les termes

- Donnée (data)
- Base de données (database)
- SGBD (DBMS)
- Système de bases de données (database system)
- Database catalog
- Program-data independance
- User View
- Administrateur de base de données (DBA)
- Utilisateur final (end user)
- Canned Transaction
- Persistent object
- meta-data
- transaction processing application

Rep (prop) :
Database : une collection de données connexes ; Une DB ne représente qu'une partie du monde réel. On peut parler de "miniworld" ou "Universe of Discourse" (UoD).
Data : des faits connus qui peuvent être enregistrés et ont une signification implicite.
SGBD : Collection de programmes qui permettent de créer et de maintenir une base de données. Définir une base de données signifie d'en spécifier les types de données, les structures, et les contraintes pour les données qui seront stockées. Construire une base de données est le processus où l'on enregistre les données dans un espace de stockage contrôlé par le SGBD. Manipuler une db implique certaines fonctions comme "interroger" (querying) la db pour retrouver certaines données spécifiques, en mettre à jour (update) pour refléter les changements du miniworld, la génération de rapports à partir des données.
Db System : On appelle "database system" l'ensemble database et DBMS.
Database catalog : Le catalogue d'une base de données est l'ensemble des définitions des objets "bases de données", comme les tables, les vues (tables virtuelles), les indexes, utilisateurs et groupes, contraintes, etc. Les informations stockées dans le catalogue sont les "meta-data".
Program-data independance Dans le traitement de fichiers traditionnel, la structure des data files est intégrée dans le programme d'accès. Donc un changement dans la structure nécessite de changer tous les programmes qui accèdent à ces fichiers. Au contraire, avec un SGBD, les programmes d'accès ne nécessitent pas de changement, la plupart du temps. C'est ce qu'on appelle "Program-data independance".
Cela permet de changer le schéma à un niveau, sans avoir à changer au niveau supérieur.

indépendance logique des données : Un schéma externe est isolé des changements qui ne concernent pas le community schema.

indépendance physique des données : Les schémas sont isolés des changements de données
Illustration

- un lien entre deux enregistrements peut être conceptuel ou logique (pointeur)
- l'ajout d'un index à une table ne modifie pas la programmation. La seule modification est l'efficacité.
- Ajout de nouveaux champs : seuls les programmes désirant y accéder devront être modifiés

User View Typiquement, une DB possède plusieurs utilisateurs, et chacun d'eux peut avoir besoin de "vues/perspectives" différentes. Une vue peut être un sous-ensemble de la db, ou contenir des données virtuelles, dérivées de la db stockée, mais pas directement stockées elles-mêmes.
Administrateur de base de données : Dans un environnement de DB, la ressource primaire est la base de données elles-même, et la seconde ressource est le SGBD, (et les softwares relatifs). L'administration de ces ressources est la responsabilité de l'administrateur BD : DBA. Il donne

accès à la db, pour coordonner et gérer l'usage, acquérir des programmes et ressources au besoin.
End User : Les personnes dont le job nécessite un accès à la db, pour interroger, mettre à jour, générer des rapports. C'est en général pour eux que la db existe. Il y a plusieurs types de end-users :

- Casual end user (décontracté) : Il accède peu souvent, mais il a besoin de certaines infos à chaque fois.
- Naive : Il met à jour souvent la db, au travers de "canned transactions" (requêtes sans danger, qui ont été vérifiées et ne risquent pas de causer de pb)
- Sophistiqué : ingénieurs, scientifiques, etc. qui implémentent leurs solutions et ont des besoins plus complexes
- Stand-alone (pas trop compris)

canned transaction : transaction en "conserve", standard, qui a été vérifiée, qui peut mettre à jour des données sans que l'on craigne pour l'intégrité de la base, toujours le même type de transaction.
Persistent object : Comme en POO, c'est un objet qui persiste même après fermeture du programme. Opposé à transient, qui est "transitoire", n'existe plus à la fin de l'exécution.
meta-data : ce qui est contenu dans le catalogue ;
transaction processing application : *En informatique, et particulièrement dans les bases de données, une transaction telle qu'une réservation, un achat ou un paiement est mise en œuvre via une suite d'opérations qui font passer la base de données d'un état A — antérieur à la transaction — à un état B postérieur et des mécanismes permettent d'obtenir que cette suite soit à la fois atomique, cohérente, isolée et durable.* (wikipedia)

5.2 Citer trois types différents d'actions qu'impliquent les bases de données ? Discuter chacune d'elles

- Interroger (querying)
- Modifier, la structure ou les données
- Analyser (report)

... to be continued

5.3 Discuter les principales caractéristiques de l'approche "Bases de données" et des différences par rapport à une gestion de fichier traditionnelle

Dans la gestion de fichiers traditionnelle, chaque utilisateur définit ses propres fichiers et leur structure. Ainsi, cela peut être très difficile de centraliser les données. Par exemple, si chaque professeur stocke lui-même les notes des élèves, et que l'on souhaite comparer les notes de deux classes, on va avoir du mal à récupérer l'ensemble des notes de deux professeurs différents, car les fichiers seront structurés de façon différente. Dans l'approche DB au contraire, la structure est définie une seule fois, et on évite la redondance. L'accès y est géré par utilisateur.

Description Par rapport à un système de fichier, la DB contient une description de la structure et des contraintes. (catalogue). Dans un système de fichier, la définition des données fait partie de l'application elle-même.

Indépendance L'indépendance des données est également une différence : Dans un système de fichier, un changement de structure devra être répercuté sur chaque application. En SGBD, la majeure partie du temps, cela ne provoquera aucun changement dans les applications, puisque le catalogue est stocké avec la DB.

Multiples utilisateurs et vues Une DB peut fournir plusieurs types d'accès aux données, voire à des données "virtuelles".

Accès et modifications multiples Une grande différence par rapport au système de fichier traditionnel est que la DB permet un accès multiple et simultané. Un SGBD doit avoir un système de contrôle de concurrence (concurrency control) afin d'assurer l'intégrité des données. Par exemple, plusieurs réservations en même temps : on doit s'assurer que deux utilisateurs n'achètent pas la même place au même moment, qu'une seule place est vendue. (online transaction processing).

5.4 Quelles sont les responsabilités du DBA et du Database Designer ?

Le DBA donne accès à la db, pour coordonner et gérer l'usage, acquérir des programmes et ressources au besoin. Les DB Designers sont responsables du choix des structures pour stocker les données. Il doit comprendre les besoins afin de fournir la structure adaptée. La plupart du temps, le designer est dans l'équipe du DBA. Il devra communiquer avec tous les types d'utilisateurs/groupes et fournir des vues adaptées.

5.5 Quels sont les différents types d'utilisateurs finaux de bases de données (end users) ?

- Casual end user (décontracté) : Il accède peu souvent, mais il a besoin de certaines infos à chaque fois.
- Naive : Il met à jour souvent la db, au travers de "canned transactions" (requêtes sans danger, qui ont été vérifiées et ne risquent pas de causer de pb)
- Sophistiqué : ingénieurs, scientifiques, etc. qui implémentent leurs solutions et ont des besoins plus complexes
- Stand-alone (pas trop compris)

5.6 Discuter des différentes "fonctionnalités" qui devraient être fournies par un SGBD

6 Questions chapitre 2

6.1 Définir les termes

- Data Model
- Database Schema
- Database state
- Internal schema
- Conceptual schema
- External schema
- Indépendance des données
- DDL
- DML
- SDL
- VDL
- query language
- Database utility

Rep (prop) :
Data Model : Collection de concepts qui peuvent être utilisés pour décrire la structure d'une base de données. Fournit des mécanismes (langages) pour définir :

- la structure
- les opérations pour retrouver/modifier les informations
- des contraintes d'intégrité

Les data models peuvent avoir des niveaux d'abstraction différents : haut niveau ou conceptuel ressemblent plus à ce que l'on propose aux utilisateurs habituels, bas niveau ou data model physique sont plutôt pour les spécialistes, plus complexes. Entre les deux, il y a "representational" (ou implementation data model), qui cache un certain nombre de détails mais qui peut être implémenté directement dans un système informatique.
Dans le cours, on parle de "relational data model". Il fournit :

- Des mécanismes pour définir les domaines et les structures de relations
- plusieurs langages de malipulation (DML)
- mécanismes pour spécifier les contraintes (clé étrangères, ...) et un langage pour spécifier les contraintes arbitraires

Database Schema : Un schéma décrit l'organisation/structure d'une db. On distingue les données de leur structure. Cette structure est spécifiée par le design de la db.
Database state : L'état - aussi appelé snapshot, est la situation des données à un instant t.
Internal schema : Dans une architecture "three-schema architecture" (3 schémas, donc), le niveau "interne" décrit le stockage interne de la db : accès physique aux données, chemins d'accès. (physique)
Conceptual schema Décrit la structure pour toute la communauté d'utilisateurs : cache les détails d'accès physiques aux données, et décrit les entités, les types de données, les relations, les opérations permises et les contraintes.
External schema (logique) : (vue) inclus des schémas externes ou des vues d'utilisateurs. Chaque schéma externe décrit une part de la db auquel un groupe/utilisateur est intéressé et cache le reste de la db.
Indépendance des données L'architecture 3-schémas (décrite au dessus) peut être utilisée pour expliquer l'indépendance des données. On peut citer deux différentes sortes d'indépendance :

- L'indépendance logique des données : La capacité de changer le schéma conceptuel sans devoir changer le schéma externe.
- L'indépendance physique des données : la capacité à changer le schéma interne sans devoir changer le modèle conceptuel (externe).

DDL : Data Definition Language, utilisé par le DBA pour définir le schéma. Dans les SGBD avec une séparation entre le niveau conceptuel et les données, DLL est utilisé pour le schéma, et

spécifier les contraintes.
DML : La manipulation de la base de données directement se fait à l'aide du Data Manipulation Language (retrouver, modifier, insérer, supprimer une information) avec un langage de requête ou de programmation. user-friendly interfaces (graphiques, menus, formulaires, langage naturel, ...)
Il y a plusieurs niveaux. Pour le haut niveau :

- Précise plus CE qui doit être cherché que COMMENT
- utilise avec propre langage ou intègre un langage hôte (COBOL, C, PASCAL)
- aussi appelé "déclaratif"

Plus bas niveau :

- Retrouve chaque enregistrement séparément et les traite individuellement (? ?)
- appelé aussi "procédural" contenu...

SDL : Storage Definition Language, utilisé pour le schéma interne, lorsque la séparation conceptuel/data est moins nette.
VDL : View Definition Language. ? ?
query language : Un langage de requête est utilisé pour accéder aux données stockées dans la db.
Database utility : Beaucoup de SGBD proposent des utilitaires avec ce type de fonctions :

- Loading : charge des fichiers existants dans la db. On précise le format de départ du fichier et le format désiré. L'utilitaire va automatiquement convertir.
- Backup : Créé une copie de la db, par "dumping". Utilisé dans des cas extrêmes. La plupart du temps, on procède à des backup incrementiels : gain de place.
- réorganisation de fichiers : Afin d'améliorer la performance, on peut souhaiter réorganiser.
- Performance monitoring : Propose des statistiques.

6.2 Discuss the main categories of data models

6.3 Quelle est la différence entre un schéma de base de données t un état de la base de données ?

Un schéma représente la structure de la base, il ne concerne pas les données en elle-mêmes mais donne des indications sur les contraintes, le type de données etc. L'état représente l'état des données à un moment m, on peut également parler de snapshot.Il s'agit là des données en elles-mêmes.

6.4 Describe the three-schema architecture. Why do we need mappings between schema levels ? How do different shema definition languages support this architecture ?

6.5 What is the difference between logical data independence and physical data independence ?

Data independence refers to the possibility to change the schema at one level without having to change it at the next higher level (nor having to change programs that access it at that higher level). The two types are :

- **Logical data independence :** An external schema (and programs that access it) is insulated from changes that do not concern it in in the community schema (and in the physical schema)
- **Physical data independence :** the community and external schemas are insulated from changes in the physical schema

Logical data independence therefore refers to the fact that data will only be modified if it is logical to do so, if changes do not concern it, it is left untouched. On the other hand physical data independence therefore refers to the fact that logical schemas of the data are unaffected by

what the data actually is. If a person changes their name it will only affect the physical data of his name, it will not change the fact that he HAS a name.

6.6 What is the difference between procedural and non-procedural DMLs ?

6.7 Discuss the different types of users-friendly interfaces and the types of users who typically use each

6.8 With what other computer system software does a DBMS interact ?

6.9 Discuter des types d'utilitaires et outils de db et leurs fonctions

Beaucoup de SGBD proposent des utilitaires avec ce type de fonctions :

- Loading : charge des fichiers existants dans la db. On précise le format de départ du fichier et le format désiré. L'utilitaire va automatiquement convertir.
- Backup : Créé une copie de la db, par "dumping". Utilisé dans des cas extrêmes. La plupart du temps, on procède à des backup incrementiels : gain de place.
- réorganisation de fichiers : Afin d'améliorer la performance, on peut souhaiter réorganiser.
- Performance monitoring : Propose des statistiques.

7 Questions chapitre 3 (livre)

7.1 Lister les différents cas où l'usage d'une valeur null peut être appropriée

- L'attribut est optionnel (exemple : on possède ou pas un téléphone, ce n'est pas obligatoire)
- L'attribut null permet d'enlever une généralisation (attribut "type" + autres optionnels)
- ?

7.2 Définir les termes suivants :

- entité
- attribut
- valeur d'attribut
- instance de relation
- attribut composé
- attribut multivalué
- attribut dérivé
- attribut complexe
- value set (domain)
- domaine
- relation
- clé
- clé primaire

Rep (prop) :

Entité : Un objet important, un concept dans le monde réel (ma voiture rouge...) qui peut être conceptuel (travail), ou physique (voiture, ..)
Une entité est souvent le terme utilisé pour décrire une instance, mais aussi pour *entity type, entity set, entity class.* En pratique, dans ce cours, "entité" est similaire à ce qu'on entend par "objet".

attribut : Une entité ou une relation a des valeurs pour chaque attribut. Celles-ci appartiennent à un set de valeurs, ou à un domaine.

valeur d'attribut : appartient à un set de valeurs (0 ou 1, par exemple), ou à un domaine

instance de relation :

attribut composé : Un attribut peut être composé de plusieurs champs, exemple : une adresse (city, nro, etc.) Conceptuellement, il s'agit du même attribut. On peut le spécifier dans la table en conservant le même préfixe (addr_nro, addr_city etc.)

attribut multivalué : un attribut qui peut avoir plusieurs instances. Exemple : un élève qui suit 0..n cours, qui possède 0..n numéros de téléphone...

attribut dérivé : se déduit d'autres attributs. ex. nombre d'élèves dans une classe, âge d'une personne, statut majeur ou mineur...
attribut complexe :
value set (domain) : détermine l'ensemble des valeurs possibles. Exemple : un numéro de téléphone fixe belge est composé de 9 chiffres

domaine : Il donne de précieuses indications sur la structure de l'information. En SQL, en général, cela se limite au type de donnée traditionnel dans les langages de programmation : integer, real, etc.

Relation :

clé : Lorsqu'un ensemble de clés possèdent la propriété d'identification d'un tuple, une clé est l'unité la plus petite de cet ensemble. En général, une relation a plusieurs clés (clés candidates). La définition des clés appartient au schéma. **clé primaire :** Les SGBD demandent souvent qu'une relation possède une clé primaire. Donc si la relation a plusieurs clés, l'une d'elle est privilégiée.

Ne pas confondre "clé" et "index". Si un attribut est une clé, il n'y a pas forcément d'index. La clé est une notion conceptuelle, l'index est un concept physique utilisé dans le but de l'optimisation.

7.3 Expliquer la différence entre un attribut et un set de valeurs

Un attribut est conceptuellement ce que représente une valeur dans une colonne, un set de valeurs est ce que peut prendre cet attribut comme valeurs. Exemple : une note peut aller de 0 à 20 (set), et cela correspond à la note (attribut)

7.4 Qu'est-ce qu'un type de relation ? Expliquer la différence entre une instance de relation, un type de relation

Un type de relation est conceptuel. Une instance est une entrée de cette relation dans une table.

7.5 What is a participation role ? When is it necessary to use role names in the description oh a relationship types ?

7.6 Décrire deux alternatives pour spécifier les contraintes structurelles d'un type de relation. Quels sont les avantages et désavantages des deux ?

Les contraintes peuvent être classifiées :
- clés
- dépendances
- intégrité référentielle
- contraintes "ad hoc" (spécifiques du domaine)

Une contrainte est tout ce qu'on veut exprimer au niveau de la structure de la db qui n'est pas possible d'exprimer à l'aide des mécanismes de description du data-model.

On peut spécifier une contrainte structurelle par une contrainte référentielle : une clé étrangère. Dans ce cas, le lien est symétrique. C'est une contrainte très présente dans les db. Cela signifie que la clé référencée doit exister.

Des contraintes complexes peuvent exister : les salaires des employés doivent être inférieurs au salaire du supérieur. Dans ce cas, cela peut être spécifié dans l'application elle-même. Le problème est que le développeur doit connaître les contraintes, et que si elles changent, il faut modifier l'application.

Notons qu'il n'y a pas de différence fondamentale de nature entre la structure de données et des contraintes. Cela dépend du sens que l'on souhaite donner. Le même morceau d'information peut être considéré comme un fait ou une contrainte (les joueurs de telle équipe de foot habitent dans telle ville).

? ?

7.7 Under what conditions can an attribute of a binary relationship type be migrated to become an attribute of one of the participating entity types ?

7.8 Quelle est la signification d'une relation récursive ? Donner des exemples

C'est une relation qui porte sur le même type d'entité une fois ou plus. Exemple : la supervision des employés.
Un employé peut superviser un employé, qui peut superviser un employé... (attention à la gestion des cycles)

7.9 Quand utilise-t-on le concept de faible entité ? Définir les termes

:

- owner entity type
- weak entity type
- identifying relationship type
- partial key

7.10 Est-ce qu'une faible entité peut être de degré supérieur à 2 ? Donner des exemples

prop : Oui. On peut imaginer une table dont les entrées représentent l'ajout d'un certain label (dont le texte est stocké sur une autre table) concernant un certain établissement (dont l'id est répertorié dans une autre table) apposé par un certain membre (dont l'id est stocké ailleurs). Dans ce cas, trois clés étrangères sont utilisées, et l'entité reste faible.

8 Questions chapitre 4

8.1 What is a subclass ? When is a subclass needed in data modeling ?

8.2 Define the following terms :

- superclass of a subclass
- superclass/subclass relationship
- IS-A-relationship
- specialization
- generalization
- category
- specific (local) attribute
- specific relationships

8.3 Discuss the mechanism of attribute/relationship inheritance. Why is it useful ?

8.4 Discuss user-defined and predicate-defined subclasses, and identify the differences between the two

8.5 Discuss user-defined and attribute-defined specializations, and identify the difference between the two

8.6 Discuss the two main types of constraints on specializations and generalizations

8.7 What is the difference between a specialization hierarchy and a specialization lattice ?

8.8 What is the difference between specialization and generalization ? Why do we not display this difference in schema diagrams ?

8.9 How does a category differ from a regular shared subclass ? What is a category used for ? Illustrate your answer with examples

8.10 For each of the following UML terms, discuss the corresponding term in the EER model, if any :

- object
- class
- association
- aggregation
- generalization
- multiplicity
- attributes
- discriminator
- link
- link attribute
- reflexive association
- qualified association

8.11 Discuss the main differences between the notation for EER schema diagram and UML class diagrams by comparing how common concepts are represented in each

8.12 Discuss the two notations for specifying constraints on n-ary relationships, and what each can be used for

8.13 List the various data abstraction concepts and the corresponding modeling concepts in the EER model

8.14 What aggregation feature is missing from the EER model? How can the EER model be further enhanced to support it?

8.15 What are the main similarities and differenes between conceptual database modeling techniques and knowledge representation techniques

9 Questions sur le modèle relationnel (chap. 7 du livre)

9.1 Define the following terms :

- domain
- attribute
- n-tuple
- relation schema
- relation-state
- degree of a relation
- relational database schema
- relational database state

9.2 Why are tuples in a relation not ordered ?

9.3 Why are duplicate tuples not allowed in a relation ?

9.4 What is the difference between a key and a superkey ?

9.5 Why do we designate one of the candidate key of a relation to be a primary key ?

9.6 Discuss the characteristics of relations that make the different from ordinary tables and files

9.7 Discuss the various reasons that lead to the occurence of null values in relations

9.8 Discuss the entity integrity and referential integrity constraints. Why is each considered important ?

9.9 Define foreign key. What is this concept used for N How does it play a role in the join operation ?

9.10 List the operations of a relational algebra and the purpose on each

9.11 What is union compatibility ? Why do the UNION, INTERSECTION and DIFFERENCE operations require that relations on which they are applied be union compatible ?

9.12 Discuss some types of queries for which renaming of attributes is necessary in order to specify the query unambiguously

9.13 Discuss the various types of JOIN operations. Why is theta join required ?

9.14 What is yhe FUNCTION operation ? What is used for ?

9.15 How are the OUTER JOIN operations different from the (inner) JOIN operations ? How is the OUTER UNION operation different from UNION ?