INFO-H-303: Synthèse de questions théoriques (issues du livre de ref.)

Arabella Brayer

Table des matières

1 Questions chapitre 1

1.1 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".
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.
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)

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

1.3 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 contôle 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 trasaction processing).

1.4 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.

1.5 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)

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

2 Questions chapitre 2

2.1 Définir les termes

- Data Model
- Database Schema
- Database state
- Internal schema
- Conceptual schema
- External schema
- Indépendance des données
- DDL
- DML
- SDL
- VDL
- query language
- Host language
- Data sublanguages
- Database utility
- Catalog
- Client-Server architecture

2.2 Discuss the main categories of data models

2.3 What is the difference between a database schema and a database state ?

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

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

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

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

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

2.9 Discuss some types of database utilities and tools and their fonctions

3 Questions chapitre 3

3.1 Discuss the role of a high-level data model in the database design process

3.2 List the various cases where use of null value would be appropriate

3.3 Define the following terms :

- entity
- attribute
- attribute value
- relashionship instance
- composite attribue
- multivalued attribute
- derived attribute
- complex attribute
- key attribute
- value set (domain)

3.4 What is an entity type ? Entity set ? Explain the differences among an entity an entity type, an and an entity set

3.5 Explain the difference between an attribue and a value set

3.6 What is a relationship type ? Explain the difference among a relationship instance, a relationship type, and a relationship set

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

3.8 Describe the two alternatives for specifying structural constraints on the relationship types. What are the advantages and disadvantages of each ?

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

3.10 When we think of relationships as attribute, what are the value sets of these attributes ? What class of data models is based on this concept ?

3.11 What is the meant by a recursive relationship type ? Give some examples of recursive relationships types

3.12 When is the concept of a weak entity used in data modeling ? Define the terms

:

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

3.13 **Can an indentifying relationship of a weak entity type be of a degree greater than two ? Give examples to illustrate your anser**

3.14 **Discuss the conventions for displaying an ER schema as an ER diagram**

3.15 **Discuss the naming conventions used for ER schema diagrams**

4 Questions chapitre 4

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

4.2 Define the following terms :

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

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

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

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

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

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

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

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

4.10 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

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

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

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

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

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