Révision

1. a) *Définissez le concept de gestion de données.*

   La gestion de données désigne l'ensemble des tâches et fonctions opérationnelles, organisationnelles et techniques dans les domaines de l'architecture, de l'administration et de la technologie de bases de données, visant à assurer le stockage et l'usage de l'ensemble des données de l'entreprise.
   Section 1.4

   b) *Présentez trois profils professionnels en gestion de données.*

   Les architectes de données sont responsables de la maintenance de l'architecture de données d'entreprise et de la conception logique des bases de données. Les administrateurs de données assurent la gestion des définitions de tables et d'attributs adoptées par l'entreprise et stockées dans un système de dictionnaire de données. Les experts en bases de données sont chargés de la conception physique et de l'installation des bases de données, de la définition des procédures de sauvegarde et de restauration des bases de données après panne.
   Sections 1.4, 2.6, 3.1, 3.7, 4.5

2. a) *Que signifie un système de gestion de bases de données relationnelles (SGBDR) ?*

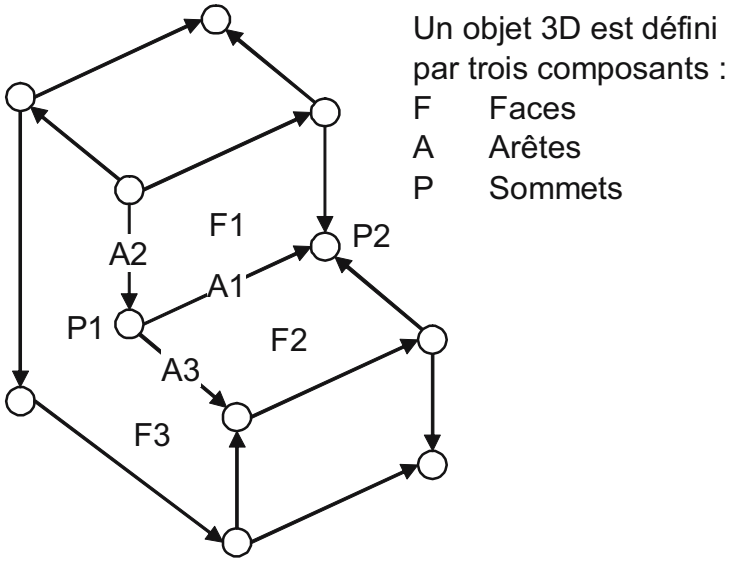
   Un SGBDR est constitué de deux modules «relationnels» : le module de stockage et le module de gestion. Le module de stockage permet d'enregistrer les données et leurs liaisons dans des tables. Outre les tables de données appartenant aux utilisateurs, il existe des tables systèmes prédéfinies où sont stockées les définitions de données. Dans le module de

gestion, le composant le plus important est un langage relationnel de définition et de manipulation de données (par exemple, le langage normalisé SQL). Ce langage inclut également les fonctions utilitaires qui garantissent l'intégrité, la protection et la sécurité des données.
Sections 1.2, 1.3

b) *Que signifie l'indépendance des données ? Quel est le module d'un SGBDR qui assure cette propriété ?*

Dans un système de bases de données, nous parlons d'indépendance des données lorsque les fonctions du système permettent de séparer les données des programmes d'application. Grâce à cette propriété, nous pouvons effectuer des modifications dans les bases de données sans devoir adapter nos programmes. C'est le module de gestion d'un SGBDR qui permet de réaliser l'indépendance des données.
Section 1.3

3. a) *Créez une liste d'informations factuelles pertinentes pour décrire des objets tridimensionnels limités par des faces planes (polyèdre).*



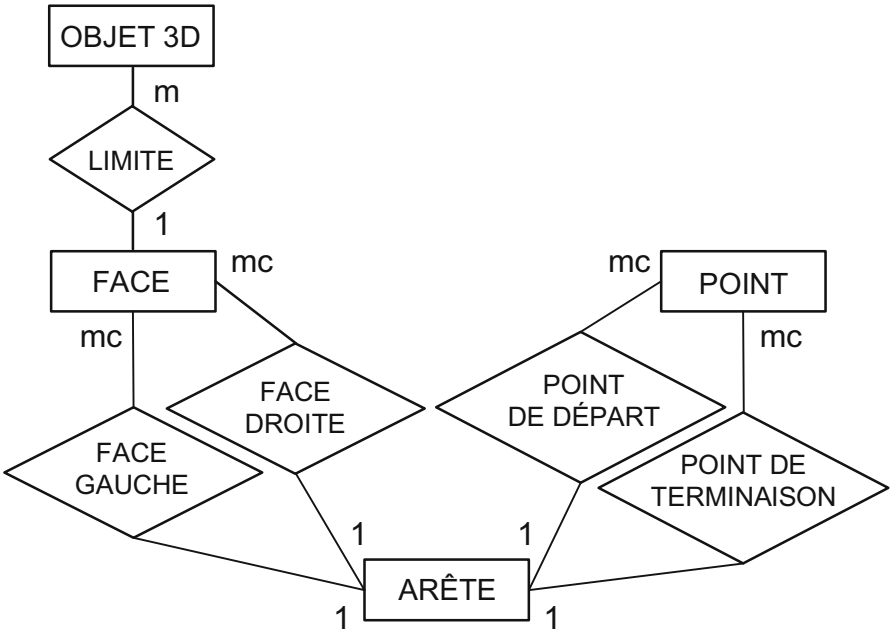
Analyse de données des objets 3D à faces planes :

- un objet 3D est formé de plusieurs faces. Chaque face appartient à un seul objet.
- une face est définie par plusieurs arêtes orientées. Chaque arête a *exactement deux* faces adjacentes (par exemple, l'arête A1 possède deux faces adjacentes, F1 à gauche et F2 à droite).
- une arête possède exactement deux sommets ou points extrêmes : un point de départ et un point de terminaison (par exemple, pour l'arête A1, le point de départ est P1, et le point de terminaison P2). Plusieurs arêtes peuvent partir d'un sommet ou y aboutir.

Section 2.1

b) *Développez un modèle entité-association pour les objets 3D à faces planes (Hypothèse : objets 3D sans ouverture).*

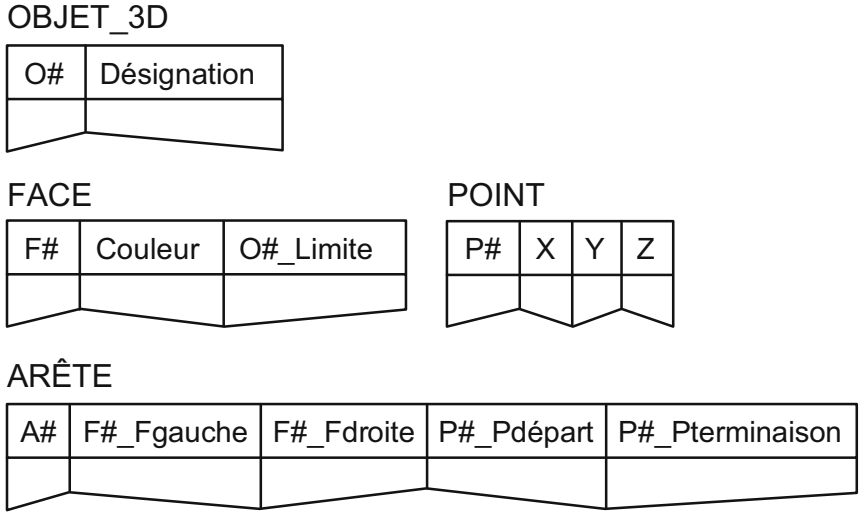
Les relations entre les faces, les arêtes et les sommets s'expriment par deux ensembles de liens duals. L'ensemble d'entités POINT contient les coordonnées cartésiennes.
Section 2.2



4. a) *Transformez le modèle entité-association des objets 3D à faces planes (voir exercice 3b) en un schéma de base de données relationnelle. Quelles sont les règles de transformation appliquées à ce cas précis ?*

Chacun des quatre ensembles d'entités et des cinq ensembles de liens donne lieu à une table (en vertu des règles 1 et 2), soit au total 9 tables. Nous optimisons ce schéma de base de données en appliquant la règle 4 aux liaisons de type simple-complexe. Le nouveau schéma qui en résulte contient 4 tables : l'ensemble de liens LIMITE s'exprime au travers de la table FACE en y définissant une clé étrangère O#_Limite. De même, les ensembles de liens duals FACE DROITE et FACE GAUCHE sont intégrés dans la table ARÊTE en y déclarant les clés étrangères F#_Fdroite et F#_Fgauche. Enfin, les ensembles de liens duals POINT DE DÉPART et POINT DE TERMINAISON sont également intégrés dans la table ARÊTE en y ajoutant les clés étrangères P#_Pdépart et P#_Pterminaison.
Section 2.2

b) *Représentez le schéma de base de données relationnelle optimal pour les objets 3D à faces planes.*



Section 2.2

5. a) *Que signifie un langage relationnel complet ?*

Un langage de requête relationnel est complet au sens de l'algèbre relationnelle, s'il supporte au moins les trois opérateurs ensemblistes de base (l'union, la différence et le produit cartésien) et les deux opérateurs de relation (la projection et la sélection).
Sections 3.2, 3.3

Remarque : Les opérateurs d'intersection, de jointure (*join*, en anglais) et de division sont facultatifs, car nous pouvons les exprimer en fonction des opérateurs susmentionnés.

b) *Quels sont les éléments additionnels que nous devons inclure dans un langage relationnel complet pour qu'il réponde aux exigences de la pratique ? (Citez au moins trois éléments)*

Il faut ajouter les constructions du langage permettant la définition de données (langage de définition de données), la manipulation de données (langage de manipulation de données), la gestion des droits d'utilisation, l'exécution des fonctions de contrôle (par exemple, la reprise après panne, le redémarrage) et la vérification de l'intégrité référentielle.
Section 3.3

6. a) *Dans la base de données des polyèdres limités par des faces planes (voir le schéma de base de données relationnelle à l'exercice 4b, dressez une liste de toutes les désignations d'objets qui présentent des faces rouges.*

```
SELECT    Désignation
FROM      OBJET_3D, FACE
WHERE     O# = O#_Limite AND Couleur = 'rouge'
```

Sections 3.2, 3.4

b) *Dans l'objet limité par des faces planes, présenté à l'exercice 3a, quelles sont les faces touchant le sommet commun P1 ?*

```
SELECT    F#_Fgauche
FROM      ARÊTE
WHERE     P#_Pdépart = 'P1'
   UNION
SELECT    F#_Fdroite
FROM      ARÊTE
WHERE     P#_Pterminaison = 'P1'
```

Idée de solution : Il faut déterminer d'abord toutes les faces gauches des arêtes qui partent du sommet P1 (à savoir F1 et F2), et les combiner ensuite aux faces droites des arêtes qui aboutissent à P1 (F3).
Sections 3.2, 3.4

7. a) *Que signifie l'intégrité référentielle ?*

Une base de données relationnelle respecte la règle de l'intégrité référentielle si chaque valeur d'une clé étrangère existe comme valeur de la clé primaire correspondante (dans la table référencée).
Section 2.5

b) *Programmez une opération d'insertion INSERT (en langage SQL) dans la table VILLE, qui transgresse la règle d'intégrité référentielle.*

PAYS

Nom	Capitale	Code
Suisse	Berne	CH
Allemagne	Berlin	D
Italie	Rome	I

VILLE

Désignation	Population	État
Berne	35000	CH
Berlin	1860000	D
Florence	40000	I
Bâle	60000	CH
Münich	1290000	D
Zürich	900000	CH
Rome	1140000	I
Fribourg	35000	CH

Réponse :

```
INSERT INTO VILLE
VALUES ('Paris',1400000,'F')
```

Remarque : La valeur «F» (code de la France) n'existe pas (encore) dans la table correspondante PAYS.
Sections 2.5, 3.4

8. a) *On désire obtenir une liste de toutes les villes d'Italie (voir exercice 7. Formulez la requête d'interrogation en SQL.*

```
SELECT    Designation
FROM      PAYS, VILLE
WHERE     Code=Etat AND Nom='Italie'
```

Remarque : La requête produit une table résultat contenant les villes Florence et Rome.
Section 3.4

b) *Exprimez la requête en utilisant les opérateurs de l'algèbre relationnelle (expression algébrique).*

La table résultat s'obtient en évaluant l'expression algébrique suivante :

$\pi_{\text{Désignation}} (\sigma_{\text{Nom='Italie'}} (\text{PAYS} \mid\times\mid_{\text{Code=État}} \text{VILLE}))$

Section 3.2

9. a) *Exprimez en SQL une requête qui produit la table résultat suivante à partir des tables PAYS et VILLE de l'exercice 7b.*

Désignation	Population	État
Berlin	1860000	D
Münich	1290000	D
Rome	1140000	I

Réponse :

```
SELECT    *
FROM      VILLE
WHERE     Population > 1000000
```

ou

```
SELECT    Designation, Population, Etat
FROM      VILLE
WHERE     Population > 1000000
```

Section 3.4

b) *Quelle est l'expression en algèbre relationnelle qui produit la même table résultat qu'en 9a.*

La même table résultat s'obtient en évaluant l'expression algébrique suivante :

$\sigma_{\text{Population} > 1000000}$ (VILLE)

Section 3.2

10. a) *On désire obtenir tous les noms de pays et de villes dans une table unique. Sous quelle condition est-il possible de formuler cette requête ?*

Les tables π_{Nom}(PAYS) et $\pi_{\text{Désignation}}$(VILLE) doivent être compatibles avec l'union, c'est-à-dire qu'elles doivent être de même dimension et définies dans les mêmes domaines.
Section 3.2

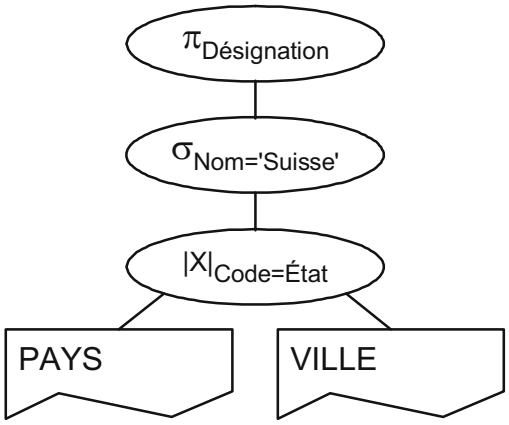
b) *Exprimez la requête en SQL.*

```
SELECT    Nom
FROM      PAYS
   UNION
SELECT    Designation
FROM      VILLE
```

Remarque : Nous admettons ici que les noms des pays et les désignations des villes sont définis dans un même domaine. Sous cette hypothèse, l'union (opérateur UNION en SQL) est possible.
Section 3.4

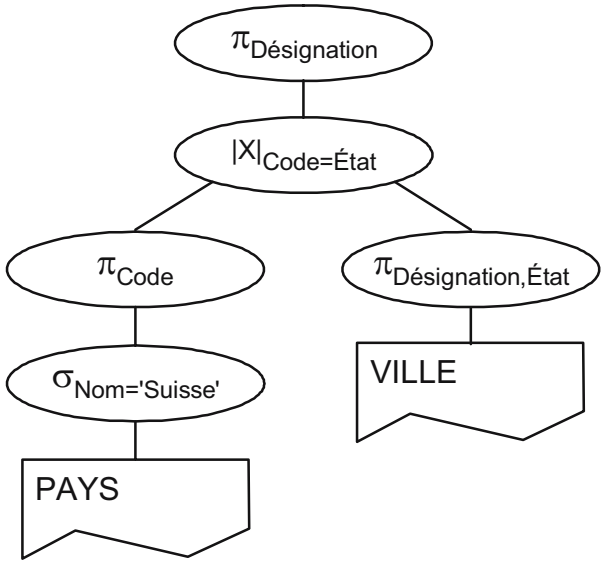
11. a) *Exprimez l'expression algébrique suivante sous la forme d'un arbre d'interrogation :*

$\pi_{\text{Désignation}} (\sigma_{\text{Nom='Suisse'}} (\text{PAYS} \mid\times\mid_{\text{Code=État}} \text{VILLE}))$



Section 4.2

b) *Construisez un arbre d'interrogation optimisé.*



Section 4.2

12. a) *Pourquoi l'adoption d'un modèle à couches multiples est-elle pertinente pour définir l'architecture des systèmes de bases de données relationnelles ?*

L'architecture des systèmes de bases de données repose sur un principe fondamental qui énonce que les modifications

et les extensions doivent pouvoir se faire localement. Comme dans l'implémentation des systèmes d'exploitation ou d'autres composants logiciels, les systèmes de bases de données relationnelles sont aussi bâtis en plusieurs niveaux indépendants qui communiquent entre eux au travers d'interfaces prédéfinies.
Section 4.6

b) *Quelles sont les principales fonctions attribuées à la couche supérieure, c'est-à-dire au niveau de l'interface ensembliste ? (présentez au moins trois fonctions)*

La couche supérieure remplit les fonctions de traduction et d'optimisation des requêtes en exécutant les tâches suivantes : traduction des requêtes et résolution des noms, contrôle et autorisation d'accès, optimisation algébrique, vérification des chemins d'accès, contrôle d'intégrité, et éventuellement, génération de codes.
Section 4.6

13. a) *Que signifie une base de données XML ? À quoi peut-elle servir ?*

Une base de données XML contient des hyperdocuments (documents XML) et repose sur un schéma XML qui permet de décrire et de gérer des données semi-structurées (texte, images, graphiques, etc.). Les bases de données XML sont nécessaires, entre autres, aux applications orientées Web dans les entreprises (commerce électronique ; *e-commerce*, en anglais) ou dans les administrations (gouvernement électronique ; *e-government,* en anglais)
Sections 5.1, 5.2

b) *Comment préserve-t-on les investissements du passé lors d'un changement de système de bases de données ?*

Il existe plusieurs variantes de migration, telles que la conversion assistée par ordinateur des bases de données et des systèmes applicatifs, la transformation des interfaces de langage, la coexistence temporaire avec maintenance

parallèle (synchrone ou asynchrone) des ensembles de données.
Sections 5.3, 5.4, 5.5

14. a) *Quelles sont les faiblesses reconnues aujourd'hui dans l'exploitation des bases de données relationnelles ? Citez au moins trois points critiques.*

En bureautique, dans la conception assistée par ordinateur et la fabrication de circuits ou de composants électroniques, dans les applications orientées Web avec hyperdocuments, ou encore dans les systèmes d'information géographiques, nous constatons des points faibles à trois niveaux suivants :

Description des objets : Les données sont hautement structurées de différentes manières. À côté des données formatées et structurées, il existe aussi des données semi-structurées telles que texte, images et graphiques. Outre les types de données prédéfinis dans les systèmes de bases de données, l'utilisateur doit pouvoir créer de nouveaux types.
Section 6.4

Gestion des transactions : La transaction est l'unité de base dans le contrôle de la cohérence et pour la reprise après panne. Dans le domaine de l'ingénierie, nous observons qu'une transaction dure en général des jours, voire des semaines dans un projet de développement. Or, de telles transactions risquent de ne pas pouvoir être annulées complètement en cas d'erreur ou de panne. En d'autres termes, l'avenir doit nous offrir la possibilité de traiter des transactions de longue durée.
Sections 4.3, 4.5

Archivage : Un système de bases de données doit permettre la gestion des versions. En outre, il doit pouvoir gérer des données temporelles (par exemple, une série chronologique ou une suite de mesures) dans le but de traiter des requêtes d'interrogation sur le passé et le futur.
Section 6.3

b) *Quelles sont les extensions et les améliorations en perspective ?*

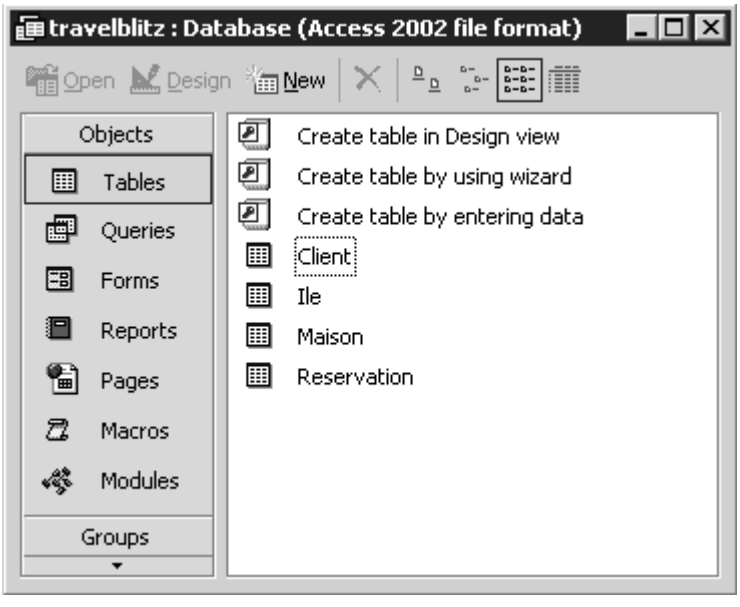
Dans l'avenir, d'une part les bases de données relationnelles bénéficieront des extensions imposées par les besoins de la pratique, et d'autre part de nouvelles générations de bases de données arriveront sur le marché. Les technologies dites post-relationnelles supporteront des systèmes de bases de données réparties, temporelles, relationnelles-objet, multidimensionnelles, floues, déductives.
Sections 6.2 à 6.7

La mise en œuvre d'une base de données avec Access

En suivant ce guide pas à pas, vous réaliserez vous-même un projet de base de données simple dans le domaine du tourisme. Votre première tâche consiste à développer un modèle entité-association. Ensuite, vous le transformerez en une structure de tables constituant votre base de données qui sera implémentée avec Access. Après la saisie des données de l'application, vous formulerez vos requêtes pour extraire les informations désirées de votre base de données.

Access est un logiciel de bases de données qui vous permet de gérer vos tables avec une interface graphique. Tous vos ordres sont traduits en instructions SQL pour l'exécution. Access vous offre aussi la possibilité de programmer directement en SQL, ce qui est très utile pour créer des requêtes complexes.

Les objets d'une base de données Access sont gérés dans différents onglets affichés à l'ouverture de la base considérée, comme le montre la figure suivante.



La «fenêtre de base de données»

Il s'agit d'une base de données nommée *travelblitz* avec ses onglets correspondant aux sept types d'objets. Le premier onglet contient les quatre tables de la base de données.

Une base de données Access comporte au minimum les types d'objets suivants :

- Les *tables* (onglet *Tables*) constituent la pierre angulaire de toute base de données Access. Elles contiennent toutes les informations dans une base de données sous forme de tuples (enregistrements).

- Les *requêtes* (onglet *Queries*) vous permettent d'extraire des informations d'une base de données.

La figure ci-dessus contient d'autres onglets qui renferment des objets nécessaires à l'exploitation efficace d'une base de données :

- Les *formulaires* (onglet *Forms*) vous permettent de saisir, d'afficher et de gérer de manière commode les données à l'aide de masques de saisie.

- Les *états* (onglet *Reports*) vous permettent de présenter les données sous une forme intelligible et agréable à lire (pour établir des factures, par exemple).

- Les *pages* (onglet *Pages*) sont à maints égards analogues aux formulaires. En outre, vous pouvez afficher les pages aussi bien dans Access que dans la fenêtre de votre navigateur Web.

- Les *macros* (onglet *Macros*) et les *modules* (onglet *Modules*) vous permettent d'automatiser des procédures de traitement complexes de votre base de données.

Dans ce guide, les figures et les commandes ont été développées avec la version anglaise d'Access 2002 ; mais il ne vous sera pas difficile de les mettre en correspondance avec d'autres versions du logiciel.

Étude de cas : travelblitz

Votre mission consiste à mettre en œuvre une application de base de données pour l'agence de voyage *travelblitz*, spécialisée dans la location de maisons de vacances situées sur des îles grecques. L'agence de voyage prévoit d'étendre ses offres de location à de nouvelles îles dans le futur. À l'heure actuelle, les données sur les clients et les maisons de vacances sont gérées dans une traditionnelle cartothèque, ce qui restreint les possibilités d'extraction des informations. Ainsi, il faut par exemple beaucoup de temps pour savoir quelle maison est libre dans une période déterminée (durant les trois premières semaines de juillet, par exemple) avec un loyer inférieur à 400 francs suisses la semaine. L'agence de voyage décide de remédier à cette situation par l'implantation d'une base de données pour mieux servir sa clientèle.

Étape 1 : développer un modèle entité-association

Pour simplifier, votre modèle entité-association repose sur les hypothèses suivantes :

1. La saison dure de la semaine 10 à la semaine 40, soit du début avril à la fin septembre. Le loyer reste fixe durant toute la saison ; les maisons sont louées à la semaine.

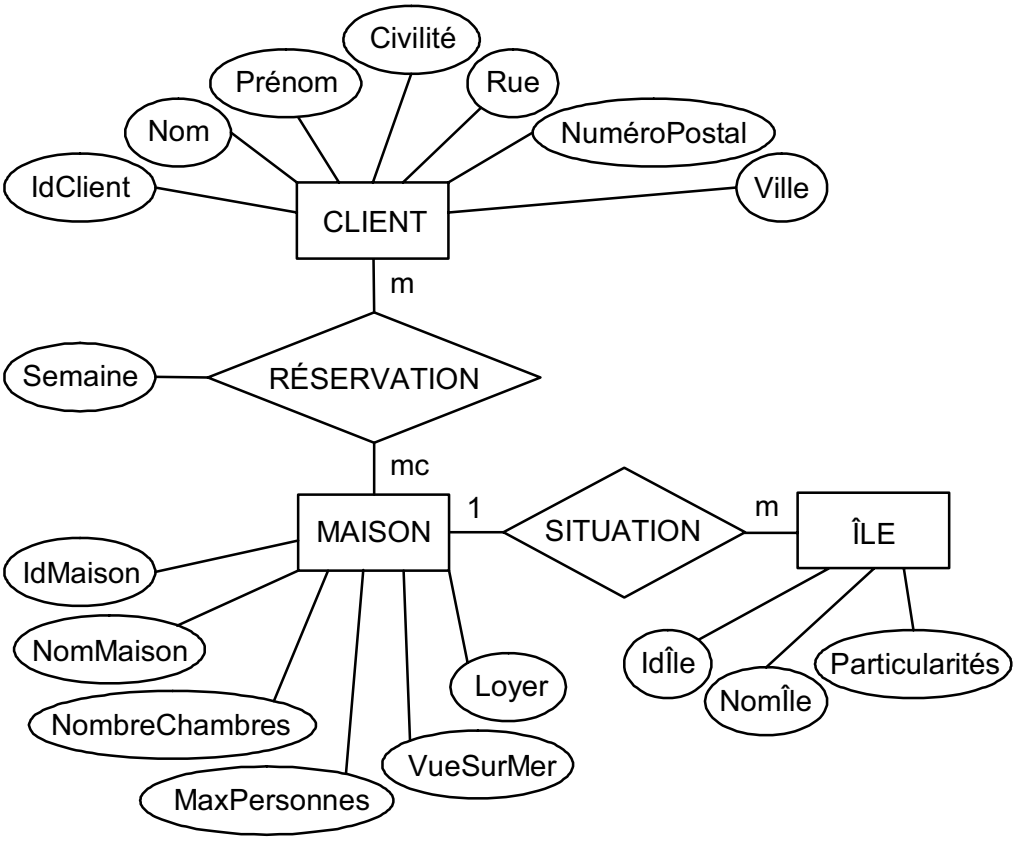
2. A chaque réservation, il faut saisir les informations sur la maison, le client et le numéro de la semaine. Si un client loue une maison sur plusieurs semaines consécutives, la location doit être éclatée en plusieurs réservations (une par semaine).

3. Chaque période de réservation, c’est-à-dire la semaine en question, est indiquée par un numéro compris entre 10 et 40. Une nouvelle base de données est créée chaque année.

4. Pour le moment, la base de données ne contient pas d'informations sur les factures, les délais de paiement, etc.

À des fins d'exercice, les hypothèses énoncées visent à réduire le plus possible la taille du modèle de données. Pour construire le

modèle entité-association, vous devez maintenant réfléchir aux questions suivantes : Quels sont les ensembles d'entités et les ensembles de liens à créer ? Quels sont leurs attributs respectifs ? Quelles sont les clés d'identification à définir pour les ensembles d'entités ?



Pour la gestion des maisons de vacances, le modèle entité-association se construit à partir de trois ensembles d'entités, CLIENT, MAISON et ÎLE. L'ensemble de liens RÉSERVATION traduit l'attribution des maisons aux clients ; la liaison est de type complexe-complexe. Vous constatez que l'attribut Semaine est un attribut de liaison typique, car il détermine la réservation d'une maison de vacances par un client dans le temps. Enfin, vous exprimez l'appartenance hiérarchique des maisons de vacances aux îles par l'ensemble de liens SITUATION.

Étape 2 : concevoir le schéma de base de données

Vous vous posez maintenant les questions suivantes : Comment se présente un schéma de base de données relationnelle pour la

gestion des maisons de vacances ? Quelles sont les règles de transformation à appliquer pour convertir le modèle entité-association précédent en tables ?

En vous référant à la section 2.3.2, vous créez pas à pas la structure des tables à partir de votre modèle entité-association :

1. En vertu de la règle de conversion 1, vous devez créer une table distincte pour chaque ensemble d'entités. Pour simplifier, donnez à la table un nom identique à celui de l'ensemble d'entités correspondant. Vous obtenez ainsi les tables suivantes :

- CLIENT (*IdClient*, Nom, Prénom, Civilité, Rue, NuméroPostal, Ville)
- MAISON (*IdMaison*, NomMaison, NombreChambres, MaxPersonnes, VueSurMer, Loyer)
- ÎLE (*IdÎle*, NomÎle, Particularités)

Par convention, écrivez les clés d'identification en italique pour les mettre en évidence.

2. En vertu de la règle de conversion 3, vous devez absolument définir une table distincte pour l'ensemble de liens de type complexe-complexe RÉSERVATION. Elle servira à stocker tous les liens générés par la location des maisons. Outre les clés étrangères qui identifient les maisons (IdMaison) et les clients (IdClient), la table renferme aussi l'attribut de liaison Semaine pour indiquer la période de location, plus précisément, le numéro de la semaine louée.

- RÉSERVATION (*IdMaison*, IdClient, *Semaine*)

Mettez en évidence l'attribut IdMaison et le numéro de la semaine qui composent la clé d'identification de la table en question. Elle permet d'éviter de manière sûre la double réservation ou surréservation d'une chambre donnée.

3. L'ensemble de liens SITUATION qui relie les maisons de vacances aux îles est de type simple-complexe. Vous avez deux

possibilités de le traduire dans le schéma de base de données : soit comme une table distincte (règle de conversion 2), soit par l'ajout d'une clé étrangère dans la table des maisons de vacances (règle de conversion 4). Choisissez la deuxième variante en agrandissant la table MAISON comme suit :

- MAISON (*IdMaison*, NomMaison, NombreChambres, MaxPersonnes, VueSurMer, Loyer, IdÎle_Situation)

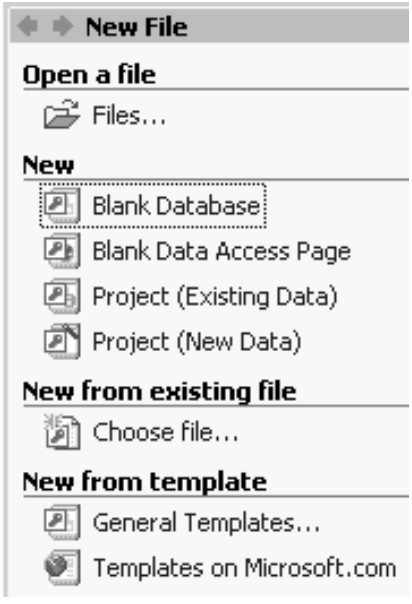
4. En conclusion, vous obtenez les quatre tables suivantes dont vous devez encore vérifier les propriétés par rapport à la troisième forme normale :

- CLIENT (*IdClient*, Nom, Prénom, Civilité, Rue, NuméroPostal, Ville)
- MAISON (*IdMaison*, NomMaison, NombreChambres, MaxPersonnes, VueSurMer, Loyer, IdÎle_Situation)
- ÎLE (*IdÎle*, NomÎle, Particularités)
- RÉSERVATION (*IdMaison*, IdClient, *Semaine*)

À l'exception de la table CLIENT, toutes les autres sont en troisième forme normale. En effet, l'attribut Ville dépend de l'attribut IdClient par transitivité, via NuméroPostal. C'est pourquoi, vous auriez dû définir une table supplémentaire LOCALITÉ contenant deux attributs, NuméroPostal et Ville. Cependant, pour des raisons pratiques, n'appliquez pas cette décomposition dans le présent exercice.

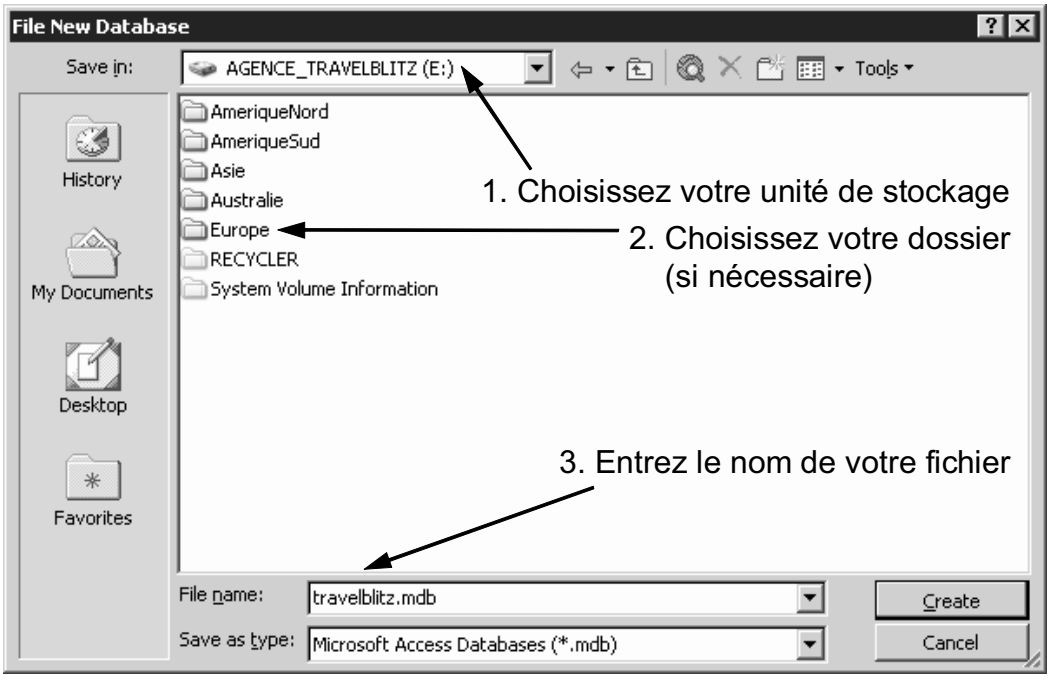
Étape 3 : implanter le schéma de base de données en Access

Démarrez Access et indiquez au système que vous désirez partir avec une nouvelle base de données vide (*Blank Database*) :



Création d'une nouvelle base de données

Access a besoin de savoir où stocker le fichier qui contiendra votre base de données. À cet effet, vous lui précisez l'unité de stockage, le dossier et le nom du fichier dans la fenêtre *File New Database* :



Une fenêtre de base de données s'ouvre aussiôt. Vous l'avez déjà vue au début de ce guide. Naturellement, tous les onglets sont vides au départ.

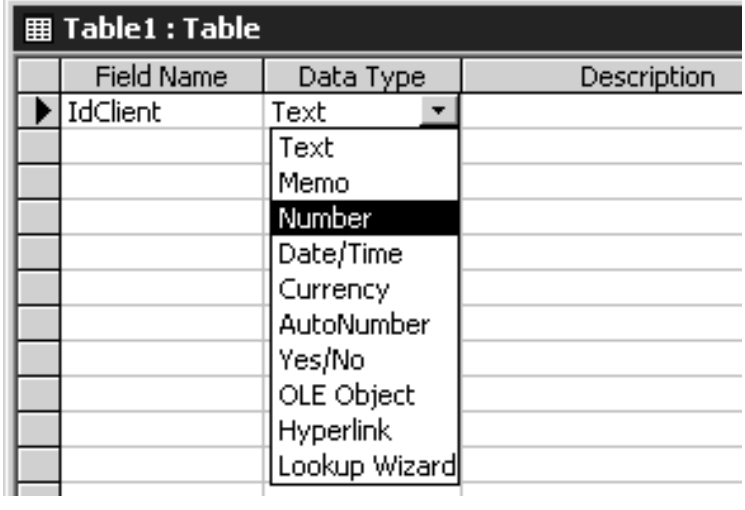
Vous êtes maintenant prêt à définir le schéma de base de données. En premier lieu, vous créez la table des clients en suivant la procédure suivante :

1. Dans l'onglet *Tables*, cliquez sur le bouton *New*. Choisissez ensuite le *mode Création* (*Design View*). Access affiche alors une fenêtre de création de table. Vous entrez ligne par ligne les attributs de la table à créer en spécifiant les propriétés suivantes :

- Nom,
- Type de donnée,
- Description concise de l'attribut.

Le nom et la description de l'attribut sont saisis manuellement. Le type de donnée est choisi dans une liste déroulante : dans la colonne *Data Type*, cliquez sur la flèche pour ouvrir une liste de choix, puis sélectionnez un type de donnée approprié à l'attribut :

Définition d'une table dans Access (choisir le type de donnée)



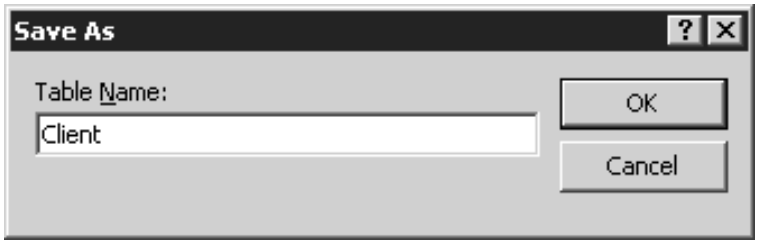
Les types de données les plus importants dans Access sont : Texte, Numérique, Monétaire, Valeur de vérité (Oui/Non) et Date/Heure.

2. Pour définir une clé primaire, marquez l'attribut clé en cliquant sur le sélecteur d'enregistrement placé en regard de l'attribut en question, puis choisissez *Edit/Primary Key*. Si la clé primaire se compose de plusieurs attributs, marquez d'abord un attribut

clé, puis, tout en maintenant la touche CTRL enfoncée, marquez les autres attributs membres de la clé.

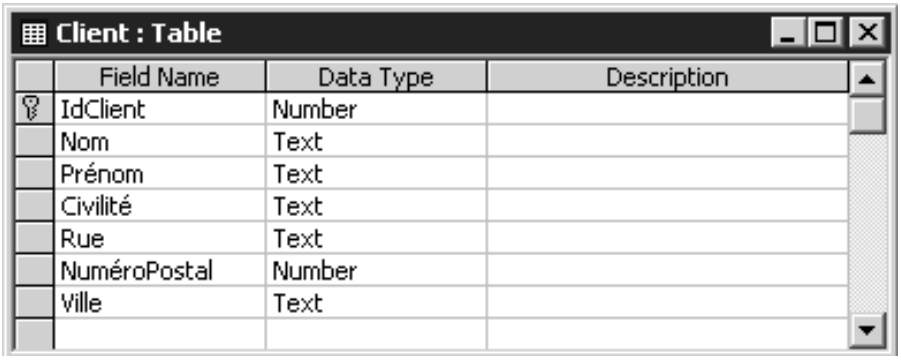
3. Quand vous fermez la fenêtre de création de table, Access affiche une boîte de dialogue dans laquelle vous introduisez le nom de la nouvelle table :

Introduire le nom d'une table



La fenêtre de création de table suivante contient la définition complète de la table CLIENT :

Définition de la table des clients



Il vous reste maintenant à créer les tables MAISON, ÎLE et RÉSERVATION par la même procédure.

Par la suite, vous définirez les contraintes d'intégrité structurelle qui ont été étudiées dans les sections 2.5 et 3.8. En Access, il existe trois types de règles d'intégrité :

- les contraintes de domaine (traitées à l'étape 4),
- les règles de validation de tuples (traitées à l'étape 5) et
- l'intégrité référentielle (traitée à l'étape 6)

Ces règles d'intégrité peuvent être définies indépendamment les unes des autres. Vous allez introduire maintenant des contraintes de

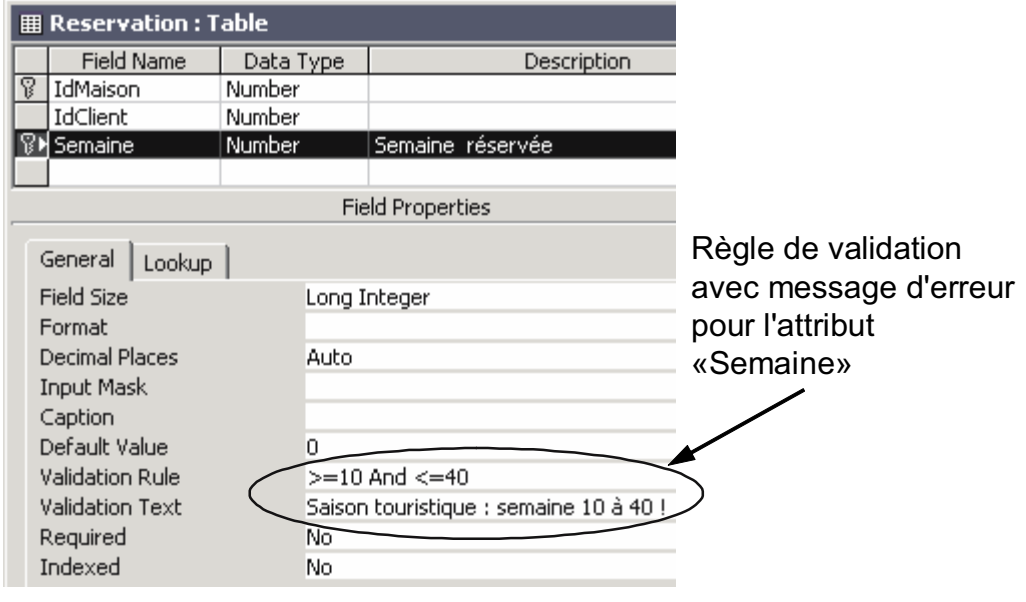
domaine et de tuple, ainsi que la contrainte d'intégrité référentielle pour les attributs clés étrangères.

Étape 4 : spécifier les contraintes de domaine

Comment restreindre l'ensemble des valeurs possibles d'un attribut ? Par exemple, pour l'attribut Semaine dans la table RÉSERVATION, il faut absolument introduire des nombres compris entre 10 et 40, car l'agence de voyage ne loue pas ses maisons de vacances durant les autres semaines. Comment activer cette contrainte d'intégrité en Access ?

Pour spécifier une contrainte de domaine imposée à l'attribut Semaine, ouvrez la table RÉSERVATION en mode Création, puis cliquez sur le sélecteur de ligne  en regard de l'attribut concerné pour le marquer. Des informations détaillées sur l'attribut marqué s'affichent dans la partie inférieure de la fenêtre. Introduisez une règle de validation pour cet attribut dans le champ *Validation Rule*. Plus tard, lorsque l'utilisateur introduit un numéro de semaine qui enfreint cette règle d'intégrité, le logiciel de base de données rejettera l'opération en affichant un message d'erreur. Le champ *Validation Text* vous permet de personnaliser ce message par un texte intelligible à l'utilisateur. Si vous laissez ce champ vide, le message d'erreur sera un texte standard, pas très parlant, tel que «Valeur interdite».

Contrainte de domaine d'un attribut



Définissez la contrainte d'intégrité de l'attribut Semaine et formulez un message d'erreur le plus parlant possible.

Vous admettez les mots suivants comme titre de civilité : "Monsieur", "Madame", "Herr", "Frau". Comment formulez-vous la contrainte d'intégrité correspondante en Access ?

Parmi les contraintes de domaine imposées à un attribut, vous pouvez encore spécifier deux autres propriétés : *Field Size* et *Required* :

- Avec la propriété *Field Size* vous affinez la définition du type de donnée d'un attribut. Ainsi, vous pouvez fixer la longueur maximale des valeurs d'un attribut de type texte, ou déclarer le type de nombre associé à un attribut numérique (Integer pour des nombres entiers, Single ou Double pour des nombres en virgule flottante).

- La propriété *Required* vous permet de rendre obligatoire la saisie d'une valeur non nulle de l'attribut concerné. Le contenu d'un attribut caractérisé par *Required*=Yes ne peut pas être vide.

Remarque : Vous pouvez réaliser simultanément les étapes 3 et 4. En d'autres termes, au fur et à mesure que vous définissez les attributs, vous déclarez leurs propriétés dans la partie inférieure de la fenêtre de création de table.

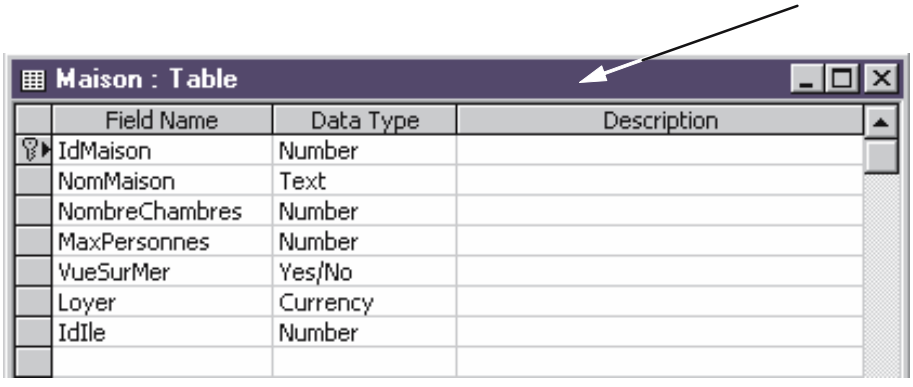
Étape 5 : définir les règles de validation de tuples

Cette catégorie de contraintes d'intégrité implique plusieurs attributs dont les valeurs sont liées entre elles dans un même tuple. Une règle de ce type énonce par exemple que le loyer de chaque maison doit se baser sur un tarif minimum de 100 francs suisses par chambre. Cette contrainte établit un lien entre deux attributs, NombreChambres et Loyer, selon l'expression suivante :

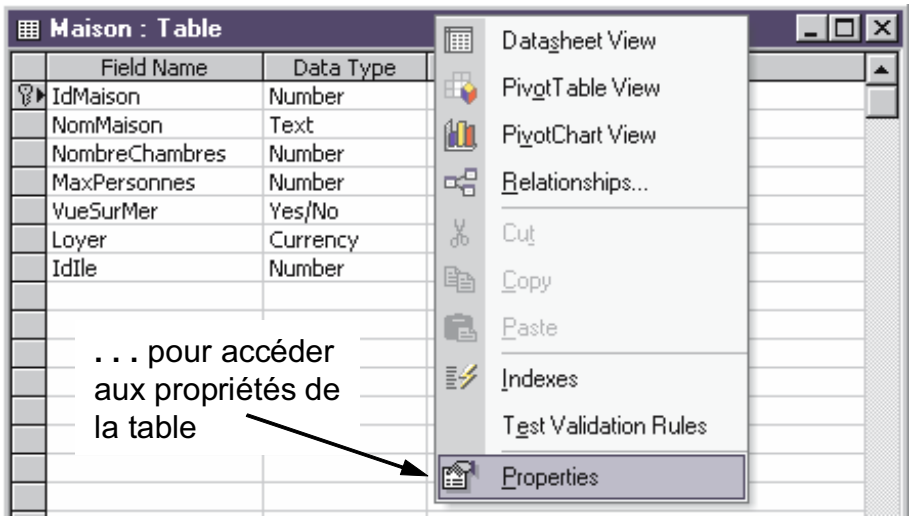
Loyer ≥ NombreChambres * 100

Pour spécifier une règle de validation de tuples, ouvrez la table en mode Création, puis cliquez sur sa barre de titre bleue avec le bouton droit :

Ouverture d'un menu contextuel ...

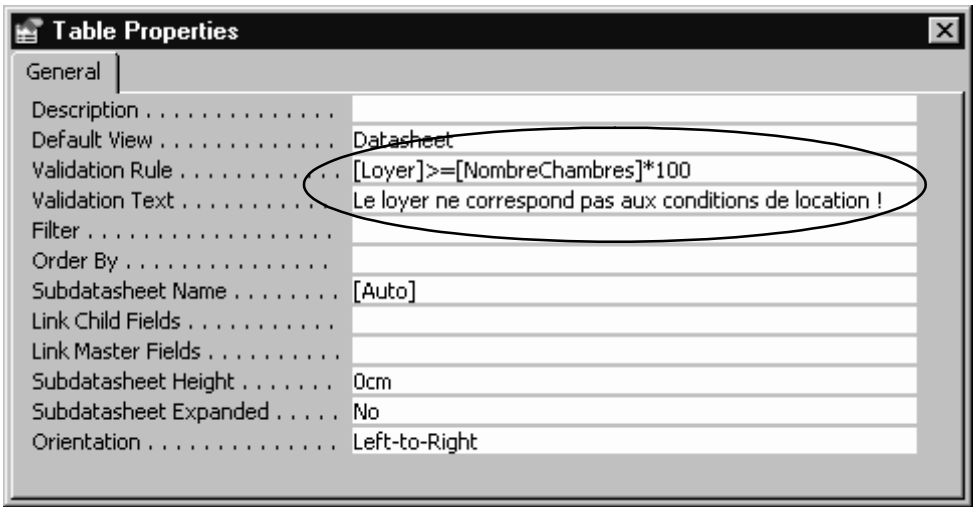


Un menu contextuel s'affiche. Sélectionner *Properties* :



Une fenêtre apparaît avec les propriétés de la table considérée. Dans le champ *Validation Rule*, vous entrez une règle que chaque tuple de la table doit respecter. Le champ *Validation Text* vous permet de personnaliser le message d'erreur qui s'affichera lorsqu'un utilisateur viole cette règle lors de la saisie d'un tuple dans la base de données.

Entrez maintenant la règle d'intégrité définie précédemment pour la table MAISON :



Règle de validation de tuples dans la table MAISON (avec message d'erreur personnalisé)

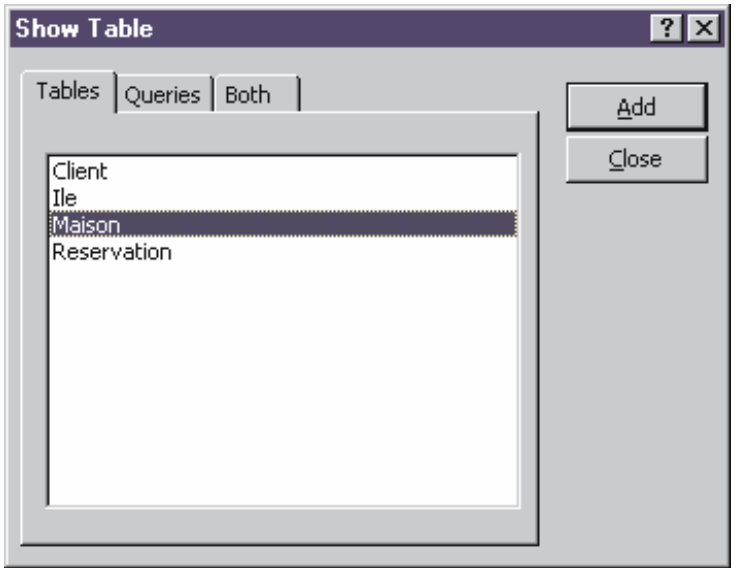
Étape 6 : définir les contraintes d'intégrité référentielle

Vous souvenez-vous de la règle d'intégrité référentielle expliquée dans les sections 2.5 et 3.8 ? Elle empêche qu'un attribut déclaré comme clé étrangère ne reçoive une valeur à laquelle ne correspond aucun tuple dans la table référencée. Dans la base de données de *travelblitz*, vous devez prévenir de telles erreurs de saisie pour chaque attribut clé étrangère en spécifiant des contraintes d'intégrité référentielle appropriées.

Du point de vue conceptuel, chaque lien défini par une clé étrangère sera représenté graphiquement ci-après par un trait qui la connecte à la clé primaire correspondante dans la table référencée. Définissez tout d'abord la contrainte d'intégrité référentielle pour la clé étrangère IdÎle dans la table des maisons de vacances. Pour ce faire, suivez les opérations 1 à 3 suivantes :

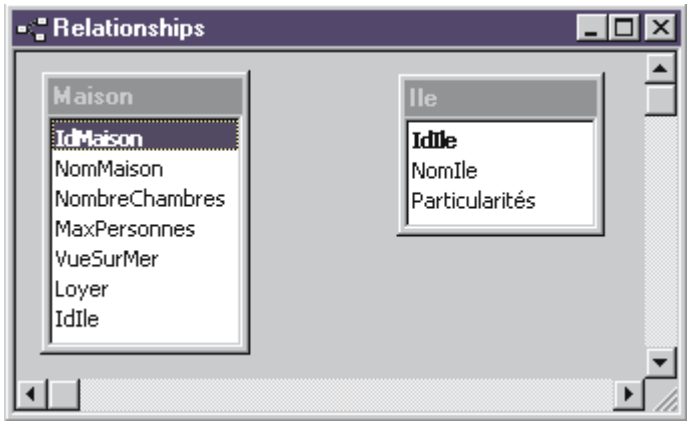
1. Choisissez la commande *Tools/Relationships*. La fenêtre *Show Table* vous propose une liste des tables pouvant participer à la définition d'une règle d'intégrité référentielle :

La fenêtre Show Table pour désigner les tables dans une contrainte d'intégrité référentielle



2. Choisissez les tables MAISON et ÎLE, cliquez le bouton *Add*, puis le bouton *Close* pour terminer. Les deux tables apparaissent schématiquement dans une fenêtre intitulée *Relationships* (les clés primaires s'affichent en gras) :

Placement des tables avant de définir la contrainte d'intégrité référentielle ...

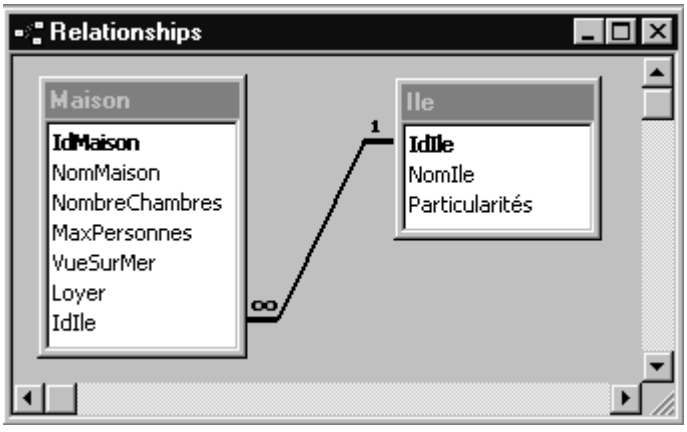


Agrandissez le cadre contenant la table des maisons pour visualiser tous ses attributs comme dans la figure ci-dessus. Glissez la table des îles vers la droite pour augmenter l'espace séparant les deux boîtes (pour déplacer une boîte, cliquez sur la barre de titre et glissez la boîte vers l'endroit désiré tout en pressant le bouton gauche de la souris).

3. Placez à présent le pointeur de la souris sur l'attribut *Maison.IdÎle*, cliquez avec le bouton gauche et, en le maintenant enfoncé, glissez le pointeur vers l'attribut *Île.IdÎle*. À ce moment-là, dans la boîte de dialogue qui apparaît, cochez la case en

regard de l'option *Enforce Referential Integrity*, puis cliquez sur le bouton *Create*. Les deux attributs mentionnés sont alors connectés par un trait continu :

... ensuite

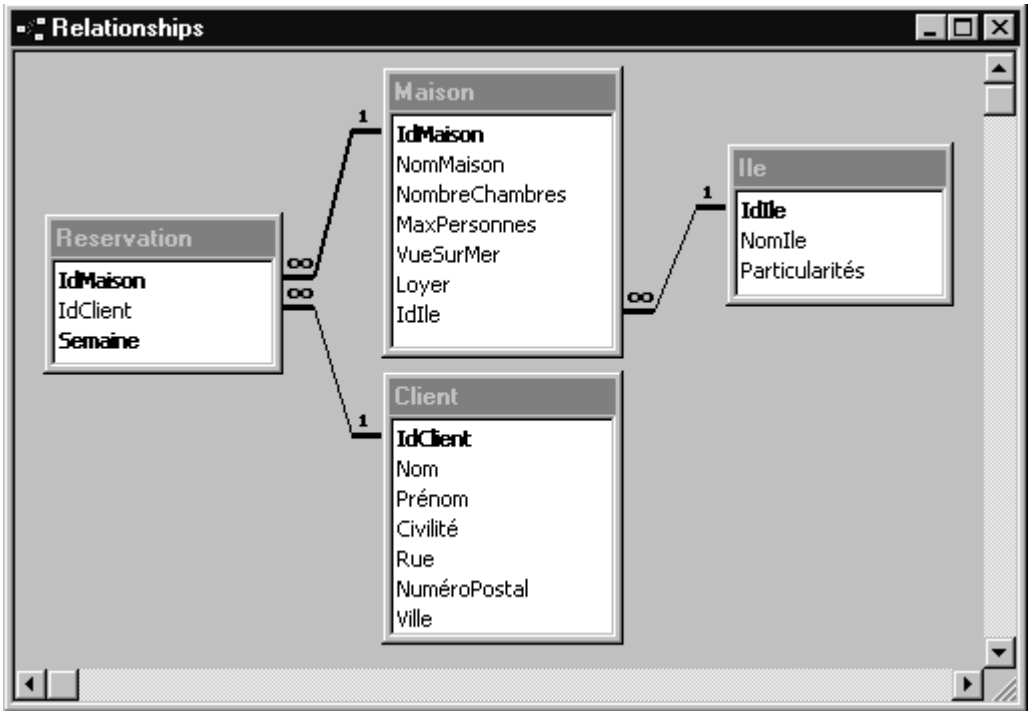


(Au cas où le trait de liaison ne s'affiche pas comme dans la figure ci-dessus, vous avez fait probablement une erreur. Vérifiez si la règle d'intégrité référentielle a été correctement définie. À cette fin, cliquez sur le trait de liaison avec le bouton droit et choisissez *Edit Relationship*. Si cela ne vous permet pas de découvrir l'erreur, vérifiez si les deux attributs ont le même type de donnée, et faites la correction si nécessaire. Si l'erreur persiste, supprimez le trait de liaison et recommencez au début.)

4. Pour terminer la définition des liens par clés étrangères, choisissez la commande *File/Close* ou cliquer sur le bouton Fermer dans le coin supérieur droit de la fenêtre *Relationships*. Sauvegardez les résultats de votre travail.

Définissez maintenant les contraintes d'intégrité référentielle associées aux autres attributs déclarés comme clés étrangères. Appliquez de nouveau la commande *Tools/Relationships.* (Pour réafficher la liste des tables, cliquez avec le bouton droit sur une zone vide dans la fenêtre *Relationships*, sélectionnez *Show Table* dans le menu contextuel, procédez ensuite comme auparavant.)

Si vos manipulations se déroulent correctement, la fenêtre *Relationships* se présente finalement comme suit :



Vous venez de terminer la définition du schéma de base de données. La saisie des données peut maintenant débuter.

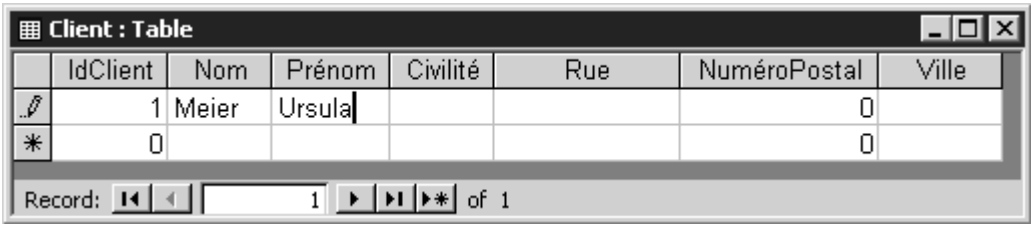
Étape 7 : introduire les données

Ouvrez en premier la table CLIENT. Pour ce faire, vous devez d'abord sélectionner l'onglet *Tables* dans la fenêtre Base de données. Il existe deux possibilités d'ouvrir une table : double-cliquer sur la table considérée, ou sélectionner la table et cliquer sur le bouton *Open*.

Une table, vide pour l'instant, s'affiche avec une seule ligne destinée à recevoir les données. Dans Access, le chiffre zéro s'affiche par défaut à l'emplacement des attributs numériques et signale ainsi à l'utilisateur qu'il doit y taper des nombres.

Introduisez maintenant quelques valeurs de donnée dans la table CLIENT :

Saisie des données dans Access



Voici quelques instructions pratiques pour la saisie des données :

1. Pour passer au champ de donnée suivant, tapez la touche Tabulation ou Entrée. Pour retourner au champ précédent, tapez simultanément la touche Majuscule (⇧, appelée aussi touche *Shift*) et la touche Tabulation.

2. Les touches fléchées (flèche haut, flèche bas, flèche gauche, flèche droite) vous permettent d'atteindre n'importe quel endroit d'une table.

3. Pour annuler une entrée, vous avez besoin de la touche *Esc* (Escape) : tapez cette touche une fois pour rétablir l'ancien contenu de la cellule courante, deux fois pour rétablir le contenu d'un tuple entier.

4. Pour supprimer un enregistrement, marquez la ligne correspondante avec le sélecteur d'enregistrement, puis enfoncez la touche *Del* (Delete).

5. Pour introduire des valeurs logiques (par exemple, pour l'attribut VueSurMer), utilisez la souris ou la barre d'espacement.

6. Pour fermer une table, choisissez *File/Close* ou cliquer sur ☒.

Introduisez les données suivantes dans les quatre tables indiquées :

CLIENT

IdClient	Nom	Prénom	Civilité	Rue	NuméroPostal	Ville
1	Meier	Ursula	Frau	Lindenstrasse 13	4051	Basel
2	Aeby	Paul	Herr	Rosenweg 26	3001	Bern
3	Gendre	Hélène	Madame	Grand-Rue 11	1700	Fribourg
4	Zumsteg	Irene	Frau	Brückenstrasse 23	3005	Bern
5	Wyss	Beat	Herr	Wallisellenstrasse 243	8050	Zürich

ÎLE

IdIle	NomIle	Particularités
1	Rhodes	Planche à voile
2	Samos	Golf, Randonnées
3	Crète	Sites historiques, Rochers

MAISON

IdMaison	NomMaison	NombreChambres	MaxPersonnes	VueSurMer	Loyer	IdIle
1	Paphos	3	5	☑	SFr. 500.00	1
2	Arethoussa	2	4	☐	SFr. 400.00	2
3	Malia	4	6	☑	SFr. 750.00	1
4	Atrium	3	4	☑	SFr. 450.00	3

RÉSERVATION

IdMaison	IdClient	Semaine
1	2	28
1	2	29
1	4	31
1	4	32
1	4	33
2	5	17
2	1	31
2	1	32
4	3	25
4	3	26
4	3	27

Si les données sont saisies exactement comme c'est indiqué ci-dessus, aucune contrainte d'intégrité référentielle ne sera violée. En outre, grâce à la clé primaire formée de deux attributs dans la table RÉSERVATION, Access n'admettra aucun doublon causé par la double réservation d'une même chambre à la même période.

Vérifiez si la base de données garantit aussi les règles d'intégrité définies en effectuant les tests suivants :

- Introduire un numéro de semaine invalide dans la table RÉSERVATION.
- Changer "Madame" en "Mademoiselle" dans la formule de civilité d'une cliente.
- Réduire sensiblement le loyer d'une maison (voir la règle de validation des loyers).
- Associer à une maison le numéro d'une île inexistante.
- Supprimer la maison Arethoussa dans la base de données (Pourquoi cette opération est-elle rejetée ?).

La base de données est maintenant mise en exploitation pour gérer les affaires de votre agence de voyage. Par exemple, vous devez

enregistrer un client, Monsieur Ernst Bircher, domicilié au 10 Seestrasse, 6004 Luzern, qui réserve la maison de vacances Malia à la fin août pendant trois semaines (semaines 33 à 35). Traitez cette transaction avec votre base de données.

L'agence de voyage vient d'acquérir Pegasos, une nouvelle maison de vacances sur l'île de Crète. Elle dispose de 5 chambres pouvant héberger 8 personnes au maximum. Son emplacement n'offre pas la vue sur mer. Enregistrez Pegasos dans la base de données.

Étape 8 : interroger la base de données avec QBE

Vous vous intéressez à une multitude de questions : quelles sont les maisons de vacances situées sur l'île de Crète ? Quelle est la maison réservée par la cliente Ursula Meier ? Quelles sont les maisons libres durant les semaines 31 à 33 ? Naturellement, vous pouvez y répondre en consultant les tables ci-dessus. Dans la réalité, lorsque la base de données atteint une certaine taille, l'usage des langages de requête s'impose.

Dans Access, vous pouvez formuler une requête soit en mode QBE avec son interface graphique (voir la section 3.4.3), soit en programmant directement des instructions SQL (voir la section 3.4.1 et la prochaine étape 9).

Vous désirez établir une liste de toutes les maisons, triées dans l'ordre croissant des loyers :

Une requête de sélection simple

NomMaison	Loyer
Arethoussa	SFr. 400.00
Atrium	SFr. 450.00
Paphos	SFr. 500.00
Pegasos	SFr. 650.00
Malia	SFr. 750.00

En QBE, votre requête est formulée puis exécutée en quatre étapes :

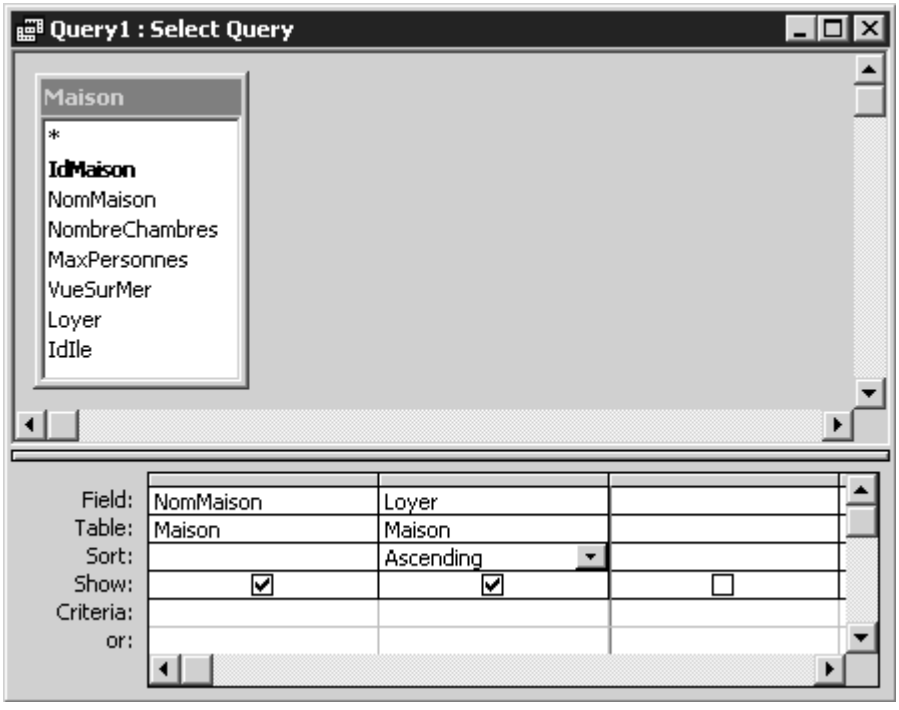
1. Dans la fenêtre Base de données, passez à l'onglet *Queries* et cliquez sur le bouton *New*. Choisissez le mode Création (*Design View*).

2. Dans la fenêtre *Show Table*, sélectionnez les tables que vous voulez interroger et cliquez sur le bouton *Add*, puis sur le bouton *Close*. Une fenêtre de création de requête s'affiche afin que vous puissiez y formuler votre requête en mode QBE.

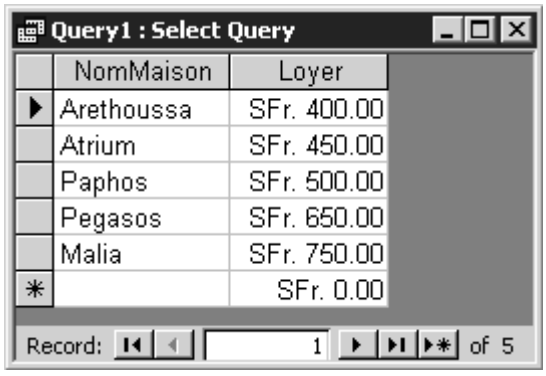
3. Vous activez une ou plusieurs options suivantes pour chaque attribut dans la fenêtre de création de requête :

- Afficher l'attribut (*Show*),
- Trier les enregistrements (*Sort*) dans l'ordre croissant ou décroissant des valeurs de cet attribut,
- Sélectionner les enregistrements par filtrage d'après un critère de recherche (*Criteria*).

. . . formulée en QBE (mode Création)



4. Pour exécuter votre requête, passez en mode Feuille de données par la commande *View/Datasheet View*. Access produit alors le résultat suivant :



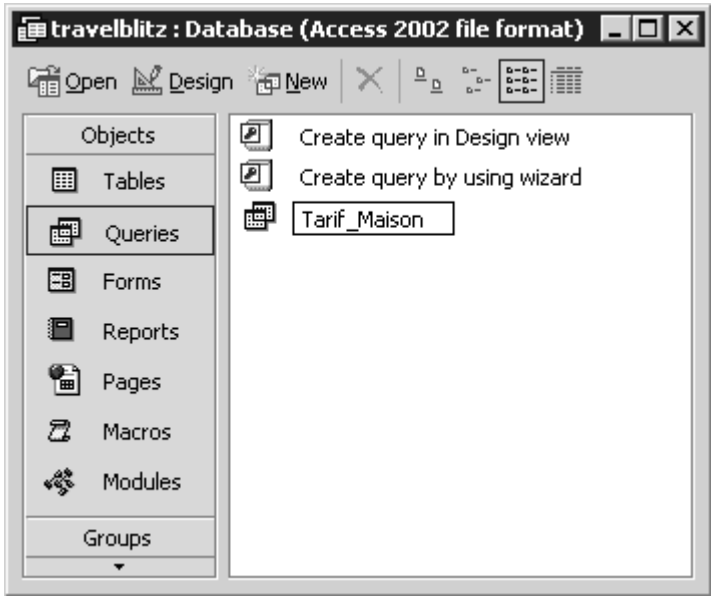
Résultat de la requête en mode Feuille de données

(Remarque : L'enregistrement vide à la fin de la fenêtre résultat signifie qu'en principe, vous pouvez profiter de cette requête pour introduire un nouvel enregistrement dans la table de base. Cependant, cette pratique est déconseillée pour entrer de nouvelles données.)

Si vous désirez modifier une requête, retournez au mode Création par la commande *View/Design View*, faites les modifications, puis vérifiez le résultat en mode Feuille de données.

5. Si vous envisagez de réutiliser la requête dans le futur, vous devez la sauvegarder (voir la section suivante).

La sauvegarde se fait soit par la commande *File/Save*, soit en fermant la requête. Dans le deuxième cas, Access vous demande automatiquement si vous désirez sauvegarder votre requête. Une fois sauvegardée, la requête figure dans l'onglet *Queries* :



L'onglet Queries contient des requêtes considérées comme tables virtuelles

À tout moment, vous pouvez exécuter une requête en la sélectionnant puis en cliquant sur le bouton *Open*. Le bouton *Design* vous permet d'ouvrir une requête en mode Création.

Remarquez que le résultat de l'exécution d'une requête se présente comme une table. Toutefois, les données affichées ne sont pas conservées sous forme de table. En revanche, elles sont générées à chaque exécution. Par conséquent, lorsque les données d'une table sont mises à jour, toutes les requêtes qui interrogent la table modifiée produisent automatiquement des résultats différents.

Étape 9 : exécuter des requêtes en SQL

Avant d'exécuter une requête, Access la transforme en commande SQL que vous pouvez aussi examiner. Pour ce faire, ouvrez la requête en mode Création ou Feuille de données, puis choisissez la commande *View/SQL View*.

Exécutez maintenant la requête Tarif_Maison que vous venez de créer, puis passez en mode SQL :

La requête Tarif_Maison exprimée en SQL

Access préfixe chaque nom d'attribut par le nom de la table correspondante. Dans la plupart des cas, vous n'avez pas besoin d'apporter cette précision et la simple commande SQL suivante suffit :

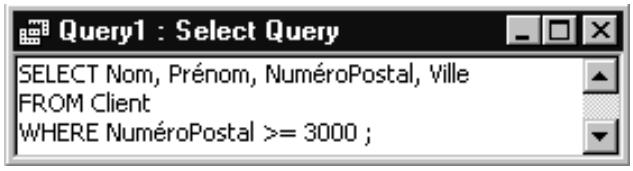
```
SELECT      NomMaison, Loyer
FROM        Maison
            ORDER BY Loyer
```

Vous désirez obtenir une liste de tous les clients de la Suisse orientale (NuméroPostal ≥ 3000) avec noms, prénoms, numéros postaux et villes.

Pour créer cette requête, choisissez l'onglet *Queries*, puis la commande *View/SQL View* pour travailler en mode SQL. Editez

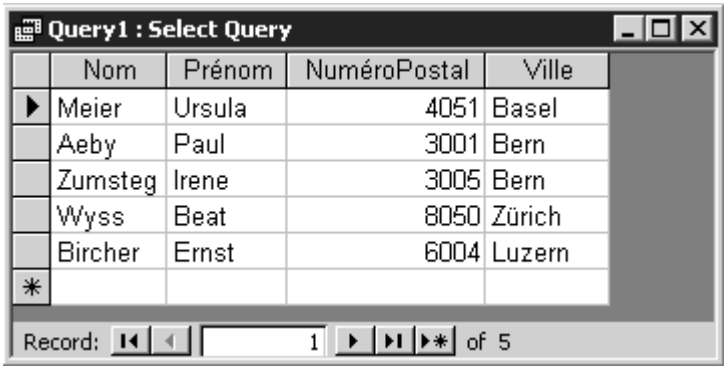
maintenant la commande SELECT appropriée dans la fenêtre de création de requête SQL :

Exemple d'une requête SQL ...



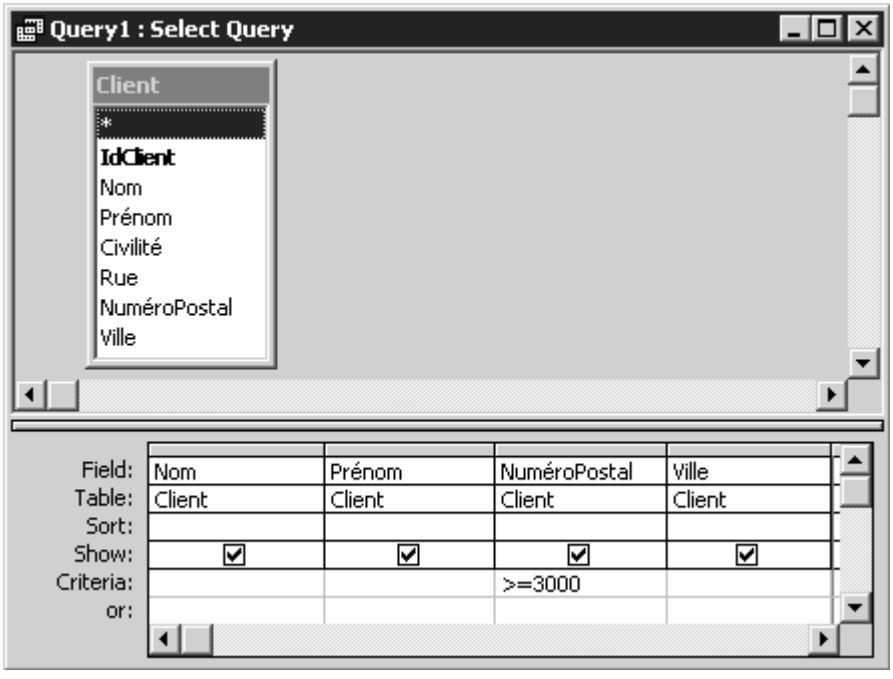
Par la commande *View/Datasheet View*, Access exécute votre requête et produit le résultat suivant :

... qui produit les résultats désirés



Pour voir la formulation équivalente de votre requête en QBE, passez en mode Création par la commande *View/Design View* :

... la même requête formulée en QBE



En comparant la commande SQL à la formulation de la même requête en QBE, vous constatez qu'il faut, dans les deux cas, introduire

les mêmes éléments constitutifs de la requête en question. Par conséquent, pour créer des requêtes simples, choisissez entre SQL et QBE le langage avec lequel vous vous sentez à l'aise. En revanche, vous devez recourir à SQL pour définir des requêtes complexes (par exemple, des commandes SQL emboîtées, voir l'étape 10).

Étape 10 : créer des requêtes complexes (avec jointures ou instructions SQL imbriquées)

Vous avez souvent besoin d'extraire des données provenant de plusieurs tables simultanément. Par exemple, vous désirez avoir une vue d'ensemble de toutes les maisons de vacances avec les noms des îles où elles se trouvent :

Requête nécessitant des données extraites de deux tables

Maison_sur_Ile : Select Query

NomMaison	NomIle	NombreChambres	MaxPersonnes	VueSurMer	Loyer
Arethoussa	Samos	2	4	☐	SFr. 400.00
Atrium	Crète	3	4	☑	SFr. 450.00
Malia	Rhodes	4	6	☑	SFr. 750.00
Paphos	Rhodes	3	5	☑	SFr. 500.00
Pegasos	Crète	5	8	☐	SFr. 650.00

Record: 1 of 5

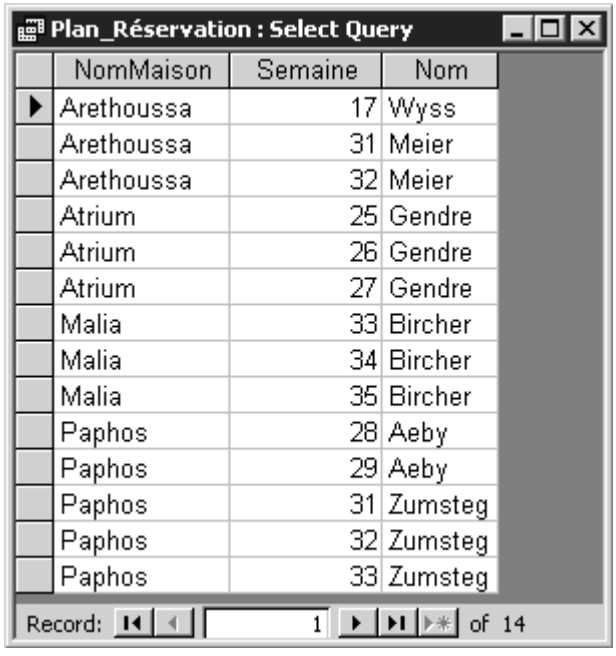
Le résultat ci-dessus est issu de deux tables, MAISON et ÎLE, liées par l'attribut *IdÎle* :

```
SELECT      NomMaison, NomIle, NombreChambres,
            MaxPersonnes, VueSurMer, Loyer
FROM        Maison, Ile
WHERE       Maison.IdIle = Ile.IdIle
ORDER BY    NomMaison
```

Créez cette requête et sauvegardez-la sous le nom Maison_sur_Ile.

Comment réalisez-vous la jointure de plus de deux tables ? Par exemple, vous désirez établir une liste de toutes les réservations, contenant les noms des clients et des maisons réservées :

Requête impliquant une double jointure



Plan_Réservation : Select Query

NomMaison	Semaine	Nom
Arethoussa	17	Wyss
Arethoussa	31	Meier
Arethoussa	32	Meier
Atrium	25	Gendre
Atrium	26	Gendre
Atrium	27	Gendre
Malia	33	Bircher
Malia	34	Bircher
Malia	35	Bircher
Paphos	28	Aeby
Paphos	29	Aeby
Paphos	31	Zumsteg
Paphos	32	Zumsteg
Paphos	33	Zumsteg

Record: 1 of 14

Votre requête nécessite trois tables, MAISON, RÉSERVATION et CLIENT :

```
SELECT      NomMaison, Semaine, Nom
FROM        Maison, Reservation, Client
WHERE       Maison.IdMaison = Reservation.IdMaison
            AND Client.IdClient = Reservation.IdClient
ORDER BY    NomMaison, Semaine
```

Créez cette requête et sauvegardez-la sous le nom Plan_Réservation.

Dans une requête imbriquée, une commande SQL renferme d'autres commandes. Vous désirez obtenir par exemple une liste des maisons de vacances sur l'île de Crète. Tout d'abord, il faut extraire le numéro d'identification de Crète par une commande SQL interne ; ce numéro est ensuite utilisé dans une commande SQL externe :

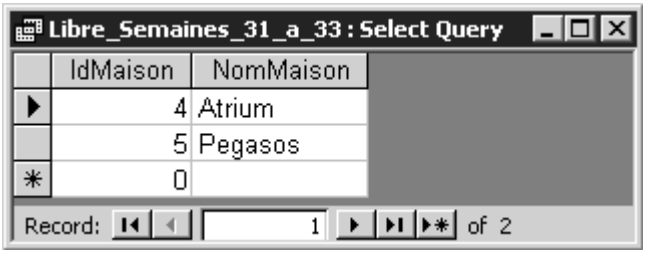
```
SELECT  NomMaison                                          <- instruction SQL externe
FROM    Maison
WHERE   IdIle = ( SELECT IdIle FROM Ile WHERE NomIle='Crète' )
                                                           <- instruction SQL interne
```

Créez et testez cette requête en y insérant à chaque fois un nom d'île différent.

SQL permet de formuler de puissantes requêtes imbriquées, notamment lorsqu'une commande SQL interne traite non pas une seule valeur de comparaison, mais plusieurs valeurs successives provenant des tuples issus d'une requête externe. Pour connaître par exemple les maisons de vacances libres durant les semaines 31 à 33, vous formulerez la requête SQL imbriquée suivante :

```
SELECT    IdMaison, NomMaison
FROM      Maison
WHERE     NOT EXISTS
            ( SELECT    *
              FROM      Reservation
              WHERE     Reservation.IdMaison =
                        Maison.IdMaison
                        AND Semaine >= 31
                        AND Semaine <= 33 )
```

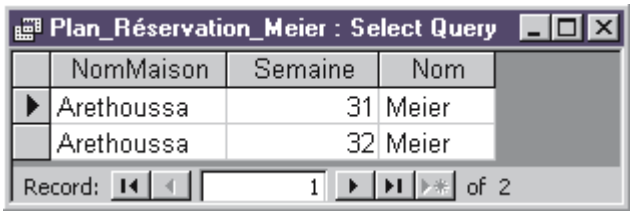
Access vous fournit le résultat suivant que vous pouvez vérifier par rapport à celui de la précédente requête Plan_Réservation.



Nous vous présentons encore deux techniques pour exploiter d'autres potentialités du langage SQL. La première consiste à interroger des requêtes considérées elles-mêmes comme des tables :

```
SELECT      *
FROM        Plan_Réservation
WHERE       Nom = 'Meier'
```

Vous obtenez comme résultat tous les enregistrements du client Meier extraits de la requête Plan_Réservation :

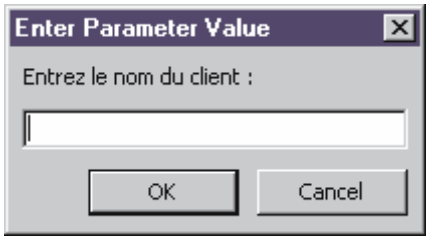


La seconde technique consiste à créer en Access des requêtes avec valeurs de comparaison variables. Il s'agit de valeurs que vous devez seulement introduire au lancement d'une requête. Par cette technique, vous pouvez généraliser la requête ci-dessus pour qu'elle puisse s'appliquer à n'importe quel client.

Pour ce faire, remplaçez la valeur de comparaison Meier par un message entre crochets, invitant l'utilisateur à entrer le nom d'un client :

```
SELECT      *
FROM        Plan_Réservation
WHERE       Nom = [Entrez le nom du client  :]
```

Au lancement de cette requête, le texte entre crochets apparaît dans une boîte de dialogue dans laquelle vous introduirez la valeur de comparaison manquante :



Access insère cette valeur dans votre requête d'interrogation avant d'exécuter la commande SQL.

Grâce à cette technique, vous êtes en mesure de modifier les deux commandes imbriquées ci-dessus pour qu'elles soient applicables à n'importe quelle île ou à n'importe quelle période de la saison touristique.

Glossaire

Administrateur de données

L'administrateur de données est chargé de la gestion des définitions de données et de fonctions utilisées dans l'entreprise à l'aide d'un système de dictionnaire de données.

Agrégation

L'agrégation est le regroupement des ensembles d'entités comme un tout. Une structure d'agrégation peut être hiérarchique ou en réseau.

Algèbre relationnelle

L'algèbre relationnelle constitue le cadre formel des langages de base de données relationnels. Elle définit les opérateurs suivants : l'union, la différence, le produit cartésien, la projection et la sélection.

Anomalie

Les anomalies sont des écarts par rapport à la réalité, qui surviennent dans une base de données lors des opérations d'insertion, de mise à jour et de suppression .

Arbre

Un arbre est une structure de données dans laquelle chaque nœud, sauf le nœud racine, possède un seul nœud prédécesseur, et chaque feuille est connectée à la racine par un chemin unique.

Association

L'association d'un ensemble d'entités à un autre définit la liaison des deux ensembles dans cette direction. Le type attribué à chaque association exprime le degré de cette liaison orientée.

Base de données floue

Une base de données floue repose sur la logique floue qui lui permet de prendre en charge des faits incomplets, vagues ou imprécis.

Calcul relationnel

Le calcul relationnel repose sur la logique des prédicats. Il utilise les combinaisons logiques de prédicats et les quantificateurs («quel que soit …» ou «il existe …»).

Clé

Une clé est une combinaison minimale d'attributs qui identifie de manière unique chaque tuple dans une table.

Contrainte d'intégrité

Les contraintes d'intégrité contiennent la spécification formelle des clés, des attributs et des domaines. Elles visent à garantir la cohérence des données.

Dictionnaire de données

Un système de dictionnaire de données sert à décrire et documenter les éléments de données, les structures de la base de données, les transactions, etc., ainsi que leurs liaisons.

Entité

Les entités sont des objets du monde réel ou dans notre pensée. Elles se caractérisent par leurs attributs et se regroupent en ensembles d'entités.

Entrepôt de données

Un entrepôt de données est un système de bases de données multidimensionnel permettant l’analyse de données représentées dans un cube multidimensionnel.

Forme normale

Les formes normales sont des règles qui permettent de détecter les dépendances à l'intérieur des tables en vue d'éliminer les informations redondantes et les anomalies qui en résultent.

Généralisation

La généralisation est un processus visant à regrouper les attributs communs à plusieurs ensembles d'entités dans un ensemble d'entités supérieur ; dans une hiérarchie de généralisation, les sous-ensembles d'entités sont appelés spécialisations.

Gestion de curseurs

La gestion de curseurs permet de traiter, à l'aide d'un pointeur, un ensemble de tuples enregistrement par enregistrement.

Hachage

Le hachage est l'organisation fragmentée du stockage de données, basée sur une fonction de hachage qui calcule, à partir d'une clé, l'adresse correspondante d'un enregistrement de données.

Indépendance des données

Un système de base de données garantit l'indépendance des données grâce à ses fonctions qui permettent de séparer les données des programmes d'application.

Index

Un index est une structure de données physique qui contient les adresses internes des enregistrements de données, associées aux attributs désignés dans une table.

Jointure

La jointure est une opération de base de données qui génère une table résultat en rapprochant deux tables par leur attribut commun.

Langage de manipulation de données

Un langage relationnel de manipulation de données permet d'effectuer des opérations ensemblistes sur une base de données pour insérer, modifier et supprimer les entrées dans les tables.

Langage de requête

Un langage de requête relationnel permet le traitement ensembliste des tables, basé sur des critères de sélection ; dans

l'approche ensembliste, le résultat d'une requête consiste toujours en un ensemble de tuples stockés dans une table.

Migration

La migration des programmes de bases de données est le passage, assisté par ordinateur, d'un système de bases de données à un autre par la conversion des données et des applications du système traditionnel vers le système cible.

Modèle de données

Un modèle de données décrit de manière formelle et structurée les données et leurs liaisons dans un système d'information.

Modèle entité-association

Le modèle entité-association est un modèle de données qui définit des classes de données (ensembles d'entités) et leurs liaisons. Les ensembles d'entités sont représentés graphiquement par des rectangles, et les ensembles de liens par des losanges.

Modèle relationnel

Le modèle relationnel est un modèle de données dans lequel les données et leurs liaisons sont toutes deux représentées sous forme de tables.

Optimisation

L'optimisation d'une requête de base de données consiste à transformer l'expression algébrique correspondante (optimisation algébrique) et à exploiter de manière efficace les structures d'accès et de stockage en vue de réduire le coût de traitement.

Orientation objet

Dans l’approche orientée objet, les données sont encapsulées avec leurs méthodes. En outre, le concept de l’héritage s’applique aux propriétés des classes de données.

Protection des données

La protection des données vise à prévenir l'accès non autorisé aux données et leur usage illicite.

Protocole de validation à deux phases

Dans une base de données répartie, le protocole de validation à deux phases assure que toutes les transactions locales se terminent avec succès et que la mise à jour des données s'effectue correctement. Sinon, il défait les transactions, annulant ainsi leur effet sur la base de données.

Protocole de verrouillage à deux phases

Le protocole de verrouillage à deux phases empêche une transaction d'acquérir un autre verrou après le premier déverrouillage d'un objet de la base de données.

Redondance

La redondance désigne le stockage multiple d'un même fait dans une base de données.

Reprise

La reprise consiste à restaurer l'état cohérent d'une base de données après panne.

Schéma de base de données

Un schéma de base de données relationnelle est la spécification formelle d'une base de données et des tables. Il définit tous les attributs clés et non clés ainsi que les contraintes d'intégrité.

Sécurité des données

La sécurité des données englobe des moyens techniques et des outils logiciels destinés à protéger un système d'information contre la corruption, la destruction et la perte de données.

Sélection

La sélection est une opération de base de données qui extrait des tuples dans une table d'après un critère spécifié par l'utilisateur.

SQL

SQL (Structured Query Language) est le plus important langage relationnel de requête et de manipulation de données. Il est normalisé par l'ISO (International Organization for Standardization).

Synchronisation

Dans un environnement multi-utilisateur, la synchronisation est la coordination des accès simultanés à une base de données. En mode synchronisation pessimiste, le système élimine le plus tôt possible les conflits entre les transactions exécutées en parallèle. En mode synchronisation optimiste, les transactions conflictuelles sont annulées le plus tard possible.

Système de gestion de base de données

Un système de gestion de base de données se compose d'un module de stockage et d'un module de gestion. Le module de stockage permet d'enregistrer les données et leurs liaisons. Le module de gestion offre des fonctionnalités et des langages nécessaires à la maintenance et la gestion des données.

Système hérité

Dans le contexte des bases de données, nous parlons de système hérité (système légué, système patrimoine) lorsque ses structures de données présentent des faiblesses accumulées dans le passé. Cela concerne aussi des ensembles de données qui ne peuvent pas être automatiquement transformés lors d'un changement de système de bases de données.

Table

Une table (relation) est un ensemble de tuples (enregistrements de données) formés d'un nombre déterminé d'attributs. Un attribut ou une combinaison d'attributs identifie de manière unique les tuples dans une table.

Transaction

Une transaction est une suite d'opérations caractérisée par les propriétés d'atomicité, de cohérence, d'isolation et de durabilité. La gestion des transactions permet à plusieurs utilisateurs simultanés de travailler sans conflit dans un système de bases de données.

Utilisateur final

L'utilisateur final d'une application est rattaché à un département fonctionnel de l'entreprise et possède des connaissances de base en informatique.

Valeur nulle

La valeur nulle représente une valeur de donnée qui n'est pas encore connue à un moment donné dans le système de bases de données.

Verrou mortel

Un verrou mortel est un interblocage des transactions dans un environnement multi-utilisateur, c'est-à-dire une situation où les transactions se bloquent mutuellement. Un système de gestion de bases de données doit être capable de détecter et de résoudre les verrous mortels.

XML

Le langage de balisage XML (eXtensible Markup Language) permet de décrire ou de représenter de manière hiérarchique des données et des documents-textes semi-structurés.

Lexique anglais-français

access path	chemin d'accès
after image	image après
aggregation	agrégation
association	association, liaison
atomicity	atomicité, unité indivisible
attribute	attribut, caractéristique, propriété
before image	image avant
B-tree	arbre B
B*-tree	arbre B*
built-in function	fonction prédéfinie, fonction intégrée
candidate key	clé candidate
cascaded deletion	suppression en cascade
checkpoint	point de reprise, point de sauvegarde
commit	valider les changements effectués dans la base de données
concurrency control	synchronisation
conditional	conditionnel
consistency	cohérence
corporate-wide data architecture, enterprise data architecture	architecture de données d'entreprise
create	créer
cursor	curseur

database management system	système de gestion de bases de données
database schema	schéma de base de données
data definition language	langage de définition de données
data dictionary system	système de dictionnaire de données
data manipulation language	langage de manipulation de données
data mining	fouille de données, forage de données
data model	modèle de données
data protection	protection des données
data security	sécurité des données
data warehouse	entrepôt de données, destiné aux applications décisionnelles
date	date
deadlock	verrou mortel, interblocage
declare	déclarer
deductive database	base de données déductive
delete	supprimer
descriptive language	langage descriptif
design	conception, concevoir
difference	différence
distributed database	base de données répartie
domain	domaine
durability	durabilité
enduser	utilisateur final
entity	entité
entity-relationship model	modèle entité-association
entity set	ensemble d'entités
equi-join	équijointure
exclusive lock	verrou exclusif

fetch	chercher
foreign key	clé étrangère
functional dependency	dépendance fonctionnelle
fuzzy database	base de données floue
generalization	généralisation
grant	accorder
grid file	fichier grilles
hash function	fonction de hachage
hashing	hachage
identification key	clé d'identification
index	index
information system	système d'information
insert	insérer
integrity	intégrité, absence d'incohérence
intersection	intersection
is-a structure	structure de généralisation (structure «est un»)
isolation	isolation
join	jointure, joindre
key	clé
key hashing	transformation de clé
knowledge base	base de connaissances
lock	verrou, verrouiller
locking protocol	protocole de verrouillage
log	journal, journaliser
log file	fichier journal
loop	boucle

manipulation language	langage de manipulation de données
multi-dimensional data structure	structure de données multi-dimensionnelle
multi-valued dependency	dépendance multivaluée
nested join	jointure imbriquée
normal form	forme normale
null value	valeur nulle
object-relational database	base de données relationnelle-objet
optimistic concurrency control	mode de synchronisation optimiste
optimization	optimisation
page	page
part-of structure	nomenclature (structure «membre de»)
performance	performance
pessimistic concurrency control	mode de synchronisation pessimiste
point query	requête singulière
precedence graph	graphe de précédence
primary key	clé primaire
procedural language	langage procédural
projection	projection
query language	langage de requête, langage d'interrogation
query tree	arbre d'interrogation
range query	requête d'intervalle
record	enregistrement
recovery	reprise après panne

redundancy	redondance
referential integrity	intégrité référentielle
relation	relation, table
relational algebra	algèbre relationnelle
relational calculus	calcul relationnel
relationship	lien, liaison
repeating group	groupe répétitif
restart	redémarrer
restricted deletion	suppression restreinte
retrieve	extraire
revoke	révoquer
role	rôle
rollback	annuler, défaire
save	sauvegarder
selection	sélection
serializability	sérialisabilité
snapshot	cliché instantané
sort-merge join	jointure par tri-fusion
synchronization	synchronisation
temporal database	base de données temporelle
time	temps
transaction	transaction
transitive dependency	dépendance transitive
transitive closure	fermeture transitive
tree	arbre
two-phase commit protocol	protocole de validation à deux phases
two-phase locking protocol	protocole de verrouillage à deux phases
union	union
unlock	déverrouiller

update	changer, actualiser, mettre à jour
view	vue

Bibliographie

Akoka J., Comyn-Wattiau I. (2001) Conception des bases de données relationnelles en pratique : Concepts, méthodes et cas corrigés. Vuibert

Astrahan M. M. et al. (1976) System R : A Relational Approach to Data Base Management. ACM Transactions on Database Systems, Vol.1, No. 2, pp. 97-137

Balzert H. (Hrsg.) (1993) CASE-Systeme und Werkzeuge. Bibliographisches Institut

Balzert H. (1999) Lehrbuch der Objektmodellierung - Analyse und Entwurf. Spektrum Akademischer Verlag

Bancilhon F. et al. (1992) Building an Object-Oriented Database System : The Story of O2. Morgan Kaufmann

Banos D., Mouyssinat M. (1991) De MERISE aux bases de données - SGBD hiérarchique, réseau, relationel. Eyrolles

Bayer R. (1972) Symmetric Binary B-Trees : Data Structures and Maintenance Algorithms. Acta Informatica, Vol. 1, No. 4, pp. 290-306

Bernstein P.A. et al. (1987) Concurrency Control and Recovery in Database Systems. Addison-Wesley

Berson A., Smith S. (1997) Data Warehousing, Data Mining and OLAP. McGraw-Hill

Bertino E., Martino L. (1993) Object-Oriented Database Systems. Addison-Wesley

Biethahn et al. (2000) Ganzheitliches Informationsmanagement (Band II : Daten- und Entwicklungsmanagement). Oldenbourg

Booch G. (1993) Object-Oriented Analysis and Design with Applications. Benjamin Cummings

Bordogna G., Pasi G. (Eds.) (2000) Recent Issues on Fuzzy Databases. Physica-Verlag

Bosc P., Liétard L., Pivert O., Rocacher D. (2004) Gradualité et imprécision dans les bases de données - Ensembles flous, requêtes flexibles et interrogation de données mal connues. Éditions Ellipses

Bosc P., Kacprzyk J. (Eds.) (1995) Fuzzyness in Database Management Systems. Physica-Verlag

Brodie M.L., Stonebraker M. (1995) Migrating Legacy Systems - Gateways, Interfaces & The Incremental Approach. Morgan Kaufmann

Brouard F., Soutou C. (2005) SQL. Collection Synthex : Synthèse de cours et exercices corrigés. Pearson Education France

Castano S. et al. (1995) Database Security. Addison-Wesley

Cattell R.G.G. (1997) Bases de données orientées objets. International Thomson Publishing

Celko J. (1999) Joe Celko's Data & Databases - Concepts in Practice. Morgan Kaufmann

Celko J. (2000) SQL AVANCÉ - Programmation et techniques avancées. Vuibert

Ceri S., Pelagatti G. (1985) Distributed Databases - Principles and Systems. McGraw-Hill

Chen G. (1992) Design of Fuzzy Relational Databases Based on Fuzzy Functional Dependency. Ph.D. Dissertation Nr. 84, Leuven Belgium

Chen G. (1998) Fuzzy Logic in Data Modeling - Semantics, Constraints and Database Design. Kluwer Academic Publishers

Chen P.P.-S. (1976) The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1, pp. 9-36

Clifford J., Warren D.S. (1983) Formal Semantics for Time in Databases. ACM Transactions on Database Systems, Vol. 8, No. 2, pp. 214-254

Clocksin W.F., Mellish C.S. (1994) Programming in Prolog. Springer

Coad P., Yourdon E. (1991) Object-Oriented Design. Yourdon Press

Codd E.F. (1970) A Relational Model for Large Shared Data Banks. Communications of the ACM, Vol. 13, No. 6, pp. 377-387

Connolly T., Begg C. (2005) Database Systems - A Practical Approach to Design, Implementation and Management. Addison-Wesley

Cremers A.B. et al. (1994) Deduktive Datenbanken - Eine Einführung aus der Sicht der logischen Programmierung. Vieweg

Dadam P. (1996) Verteilte Datenbanken und Client/Server-Systeme. Springer

Date C.J. (1986) Relational Database - Selected Writings. Addison-Wesley

Date C.J. (2004) Introduction aux bases de données. Vuibert

Darwen H., Date C.J. (1997) A Guide to the SQL Standard. Addison-Wesley

Delobel C. et al. (1991) Bases de données : des systèmes relationnels aux systèmes à objets. InterEditions

Dietrich S.W., Urban S.D. (2005) An Advanced Course in Database Systems - Beyond Relational Databases. Pearson Prentice Hall

Dippold R., Meier A., Ringgenberg A., Schnider W., Schwinn K. (2001) Unternehmensweites Datenmanagement - Von der Datenbankadministration bis zum modernen Informationsmanagement. Vieweg

Dittrich K.R. (Ed.) (1988) Advances in Object-Oriented Database Systems. Lecture Notes in Computer Science, Vol. 334, Springer

Dürr M., Radermacher K. (1990) Einsatz von Datenbanksystemen - Ein Leitfaden für die Praxis. Springer

Dutka A.F., Hanson H.H. (1989) Fundamentals of Data Normalization. Addison-Wesley

Eilers H. et al. (1986) SQL in der Praxis. Addison-Wesley

Elmasri R., Navathe S.B. (2004) Conception et architecture des bases de données. Pearson Education France

Etzion O., Jajodia S., Sripada S. (Eds.) (1998) Temporal Databases - Research and Practice. Lecture Notes in Computer Science, Springer

Ferstl O.K., Sinz E.J. (1991) Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). Wirtschaftsinformatik, Jhrg. 33, Nr. 6, S. 477-491

Findler N.V. (Ed.) (1979) Associative Networks - Representation and Use of Knowledge by Computers. Academic Press

Gadia S.K. (1988) A Homogeneous Relational Model and Query Languages for Temporal Databases. ACM Transactions on Database Systems, Vol. 13, No. 4, pp. 418-448

Gallaire H. et al. (1984) Logic and Databases : A Deductive Approach. ACM Computing Surveys, Vol. 16, No. 2, pp. 153-185

Garcia-Molina H., Ullman J.D., Widom J. (2000) Database System Implementation. Prentice-Hall

Gardarin G. (2000) Bases de données - Objet & relationnel. Eyrolles

Gardarin G. (2003) Bases de données. Eyrolles

Gardarin G., Valduriez P. (1989) Relational Databases and Knowledge Bases. Addison-Wesley

Gemünden G., Schmitt M. (1991) Datenmanagement in deutschen Grossunternehmen - Theoretischer Ansatz und empirische Untersuchung. Information Management, Jhrg. 6, Nr. 4, S. 22-34

Geppert A. (2002) Objektrelationale und objektorientierte Datenbankkonzepte und -systeme. dpunkt

Gillenson M.L. (1990) Physical Design Equivalences in Database Conversion. Communications of the ACM, Vol. 33, Nr. 8, pp. 120-131

Gluchowski P., Gabriel R., Chamoni P. (1997) Management Support Systeme - Computergestützte Informationssysteme für Führungskräfte und Entscheidungsträger. Springer

Goglin J.-F. (1998) La construction du datawarehouse : du datamart au dataweb. Hermes

Gouarné J.-M. (1998) Le projet décisionnel : enjeux, modèles et architectures du Data Warehouse. Eyrolles

Gray J., Reuter A. (1993) Transaction Processing - Concepts and Techniques. Morgan Kaufmann

Härder T. (1978) Implementierung von Datenbanksystemem. Hanser

Härder T., Rahm E. (1999) Datenbanksysteme - Konzepte und Techniken der Implementierung. Springer

Härder T., Reuter A. (1983) Principles of Transaction-Oriented Database Recovery. ACM Computing Surveys, Vol. 15, Nr. 4, pp. 287-317

Heinrich L.J. (1999) Informationsmanagement - Planung, Überwachung und Steuerung der Informationsinfrastruktur. Oldenbourg

Hernandez M.J., Viescas J.L. (2001) Introduction aux requêtes SQL. Eyrolles

Heuer A. (1997) Objektorientierte Datenbanken. Addison-Wesley

Heuer A., Saake G. (2000) Datenbanken - Konzepte und Sprachen. MITP

Hitz M., Kappel G. (2002) UML@Work - Von der Analyse zur Realisierung. dpunkt

Hoffer J.A., Prescott M.B., McFadden F.R. (2004) Modern Database Management. Pearson Prentice Hall

Hughes J.G. (1991) Object-Oriented Databases. Prentice-Hall

Hüsemann F.C. (2002) Datenbankmigration - Methodik und Softwareunterstützung. VDI Verlag

Inmon W.H. (2002) Building the Data Warehouse. Wiley

Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P. (2000) Fundamentals of Data Warehouses. Springer

Jensen C.S., Clifford J., Gadia S.K., Seger A., Snodgrass R.T. (1992) A Glossary of Temporal Database Concepts. In : SIGMOD RECORD, Vol. 21, Nr. 3. ACM Press

Jones A., Stephens R.K., Plew R.R., Garrett R.F., Kriegel A. (2005) SQL Functions - Programmer's Reference. Wiley

Kacprzyk J., Zadrozny S. (1995) FQUERY for Access - Fuzzy Querying for a Windows-Based DBMS. In : Bosc and Kacprzyk (1995), pp. 415-433

Kazakos W., Schmidt A., Tomczyk P. (2002) Datenbanken und XML - Konzepte, Anwendungen, Systeme. Springer

Kemper A., Eickler A. (2001) Datenbanksysteme - Eine Einführung. Oldenbourg

Kerre E.E., Chen G. (1995) An Overview of Fuzzy Data Models. In : Bosc and Kacprzyk (1995), pp. 23-41

Kimball R., Reeves L., Ross M., Thornthwaite W. (2000) Concevoir et déployer un data warehouse - Guide de conduite de projet. Eyrolles

Kim W. (1990) Introduction to Object-Oriented Databases. The MIT Press

Klettke M., Meyer H. (2003) XML & Datenbanken - Konzepte, Sprachen und Systeme. dpunkt

Kuhlmann G., Müllmerstadt F. (2000) SQL - Der Schlüssel zu relationalen Datenbanken. Rowohlt

Kurbel K., Strunz H. (Hrsg.) (1990) Handbuch der Wirtschafts-informatik. C.E. Poeschel

Lang S.M., Lockemann P.C. (1995) Datenbankeinsatz. Springer

Lans R.F. van der (1988) Das SQL-Lehrbuch. Addison-Wesley

Lausen G., Vossen G. (1996) Objekt-orientierte Datenbanken : Modelle und Sprachen. Oldenbourg

Lentzner R. (2004) SQL3 avec Oracle, MySQL, Microsoft SQL Server et Access. Dunod

Lockemann P.C., Schmidt J.W. (Hrsg.) (1993) Datenbank-Handbuch. Springer

Loeser H. (2001) Web-Datenbanken - Einsatz objekt-relationaler Datenbanken für Web-Informationssysteme. Springer

Lorie R.A. et al. (1985) Supporting Complex Objects in a Relational System for Engineering Databases. In : Kim W. et al. (Ed.) Query Processing in Database Systems. Springer, pp. 145-155

Ma Z. (2005) Fuzzy Database Modeling with XML. Advances in Database Systems, Vol. 29. Springer Science & Business Media

Maier D. (1983) The Theory of Relational Databases. Computer Science Press

Maier D., Warren D.S. (1988) Computing with Logic - Logic Programming with Prolog. Benjamin/ Cummings

Martin J. (1986) Einführung in die Datenbanktechnik. Hanser

Martin J. (1990) Information Engineering - Planning and Analysis. Prentice-Hall

Martin J., Odell J.J. (1992) Object-Oriented Analysis and Design. Prentice-Hall

Martiny L., Klotz M. (1989) Strategisches Informationsmanagement. Oldenbourg

Maurer W.D., Lewis T.G. (1975) Hash Table Methods. ACM Computing Surveys, Vol. 7, Nr. 1, pp. 5-19

Meier A. (1987) Erweiterung relationaler Datenbanksysteme für technische Anwendungen. Informatik-Fachberichte, Bd. 135, Springer

Meier A. (1994) Ziele und Aufgaben im Datenmanagement aus der Sicht des Praktikers. Wirtschaftsinformatik, Jhrg. 36, Nr. 5, S. 455-464

Meier A. (1997) Datenbankmigration - Wege aus dem Datenchaos. Praxis der Wirtschaftsinformatik, Jhrg. 34, Nr. 194, S. 24-36

Meier A. (Hrsg.) (2000) WWW & Datenbanken. Praxis der Wirtschaftsinformatik, Jhrg. 37, Heft 214, dpunkt

Meier A., Dippold R. (1992) Migration und Koexistenz heterogener Datenbanken - Praktische Lösungen zum Wechsel in die relationale Datenbanktechnologie. Informatik-Spektrum, Jhrg. 15, Nr. 3, S. 157-166

Meier A., Graf H., Schwinn K. (1991) Ein erster Schritt zu einem globalen Datenmodell. Information Management, Jhrg. 6, Nr. 2, S. 42-48

Meier A., Haltinner R., Widmer-Itin B. (1993) Schutz der Investitionen beim Wechsel eines Datenbanksystems. Wirtschaftsinformatik, Jhrg. 35, Nr. 4, S. 331-338

Meier A., Johner W. (1991) Ziele und Pflichten der Datenadministration. Theorie und Praxis der Wirtschaftsinformatik, Jhrg. 28, Nr. 161, S. 117-131

Meier A., Savary C., Schindler G., Veryha Y. (2001) Database Schema with Fuzzy Classification and Classification Query Language. Proc. of the International Congress on Computational Intelligence - Methods and Applications, CIMA 2001, University of Wales, Bangor U.K.

Meier A., Wüst T. (1995) Objektorientierte Datenbanksysteme - Ein Produktvergleich. Theorie und Praxis der Wirtschaftsinformatik, Jhrg. 32, Nr. 183, S. 24-40

Meier A., Wüst T. (2003) Objektorientierte und objektrelationale Datenbanken - Ein Kompass für die Praxis. dpunkt

Miranda S. (2002) Bases de données - Architecture, modèles relationnels et objets, SQL3. Dunod

Morin A., Bosc P., Hebrail G., Lebart L. (2002) Bases de données et statistique. Dunod

Mucksch H., Behme W. (Hrsg.) (1996) Das Data-Warehouse-Konzept - Architektur, Datenmodelle, Anwendungen. Gabler

Nanci D., Espinasse B. (2001) Ingénierie des systèmes d'information : Merise - Deuxième génération. Vuibert

Nievergelt J., Hinterberger H., Sevcik K.C. (1984) The Grid File : An Adaptable, Symmetric Multikey File Structure. ACM Transactions on Database Systems, Vol. 9, No. 1, pp. 38-71

Olle T.W. et al. (1988) Information Systems Methodologies - A Framework for Understanding. Addison-Wesley

Ortner E. et al. (1990) Entwicklung und Verwaltung standardisierter Datenelemente. Informatik-Spektrum, Jhrg. 13, Nr. 1, S 17-30

Österle H. et al. (1991) Unternehmensführung und Informationssystem - Der Ansatz des St. Galler Informationssystem-Managements. Teubner

Özsu M.T., Valduriez P. (1991) Principles of Distributed Database Systems. Prentice-Hall

Panny W., Taudes (2000) Einführung in den Sprachkern SQL-99. Springer

Paredaens J. et al. (1989) The Structure of the Relational Database Model. Springer

Petry F.E. (1996) Fuzzy Databases - Principles and Applications. Kluwer Academic Publishers

Pistor P. (1993) Objektorientierung in SQL3 : Stand und Entwicklungstendenzen. Informatik-Spektrum, Jhrg. 16, Nr. 2, S. 89-94

Pons O., Vila M.A., Kacprzyk J. (Eds.) (2000) Knowledge Management in Fuzzy Databases. Physica-Verlag

Pucheral P. (2004) Bases de données avancées - Modèles, systèmes et usages. Numéro spécial de la revue RSTI - Technique et science informatiques. Hermès/Lavoisier

Rahm E. (1994) Mehrrechner-Datenbanksysteme. Addison-Wesley

Rahm E., Vossen G. (Hrsg.) (2003) Web & Datenbanken - Konzepte, Architekturen, Anwendungen. dpunkt

Ramakrishnan R., Gehrke J. (2003) Database Management Systems. McGraw-Hill

Reingruber M.C., Gregory W.W. (1994) The Data Modeling Handbook - A Best-Practice Approach to Building Quality Data Models. Wiley

Reuter A. (1981) Fehlerbehandlung in Datenbanksystemen. Hanser

Riordan R.M. (2005) Designing Effective Database Systems. Addison-Wesley

Rothnie J.B. et al. (1980) Introduction to a System for Distributed Databases. ACM Transactions on Database Systems, Vol. 5, No. 1, pp. 1-17

Rumbaugh J. et al. (1991) Object Oriented Modelling and Design. Prentice-Hall

Saake G. et al. (1997) Objektdatenbanken. International Thomson Publishing

Sauer H. (1992) Relationale Datenbanken - Theorie und Praxis inklusive SQL-2. Addison-Wesley

Scheer A.-W. (1991) Architektur integrierter Informationssysteme - Grundlagen der Unternehmensmodellierung. Springer

Schek H.-J., Scholl M.H. (1986) The Relational Model with Relation-Valued Attributes. Information Systems, Vol. 11, No. 2, pp. 137-147

Schindler G. (1998) Fuzzy Datenanalyse durch kontextbasierte Datenbankanfragen. Deutscher Universitäts-Verlag

Schlageter G., Stucky W. (1983) Datenbanksysteme : Konzepte und Modelle. Teubner

Schöning H. (2003) XML und Datenbanken - Konzepte und Systeme. Hanser

Shenoi S., Melton A., Fan L.T. (1992) Functional Dependencies and Normal Forms in the Fuzzy Relational Database Model. Information Sciences, Vol. 60, pp. 1-28

Silberschatz A., Korth H.F., Sudarshan S. (2005) Database System Concepts. McGraw-Hill

Silverston L. (2001a) The Data Model Resource Book, Revised Edition, Volume 1 - A Library of Universal Data Models for All Enterprises. Wiley

Silverston L. (2001b) The Data Model Resource Book, Revised Edition, Volume 2 - A Library of Universal Data Models by Industry Types. Wiley

Simsion G.C., Witt G.C. (2005) Data Modeling Essentials. Morgan Kaufmann

Smith J.M., Smith D.C.P. (1977) Database Abstractions : Aggregation and Generalization. ACM Transactions on Database Systems, Vol. 2, No. 2, pp. 105-133

Snodgrass R.T. (1987) The Temporal Query Language TQuel. ACM Transactions on Database Systems. Vol. 12, No. 2, pp. 247-298

Snodgrass R.T. et al. (1994) A TSQL2 Tutorial. SIGMOD-Record, Vol. 23, No. 3, pp. 27-33

Stein W. (1994) Objektorientierte Analysemethoden - Vergleich, Bewertung, Auswahl. Bibliographisches Institut

Stephens R.K., Plew R.R. (2001) Conception de bases de données. CampusPress

Stonebraker M. (Ed.) (1986) The Ingres Papers. Addison-Wesley

Stonebraker M. (1996) Object-Relational DBMS's - The Next Great Wave. Morgan Kaufmann

Takahashi Y. (1995) A Fuzzy Query Language for Relational Databases. In : Bosc and Kacprzyk (1995), pp. 365-384

Taylor A.G. (2003) SQL For Dummies. Wiley

Tsichritzis D.C., Lochovsky F.H. (1982) Data Models. Prentice-Hall

Türker C. (2003) SQL:1999 & SQL:2003 - Objektrelationales SQL, SQLJ & SQL/XML. dpunkt

Ullman J. (1982) Principles of Database Systems. Computer Science Press

Ullman J. (1988) Principles of Database and Knowledge-Base Systems. Computer Science Press

Vetter M. (1998) Aufbau betrieblicher Informationssysteme mittels pseudo-objektorientierter, konzeptioneller Datenmodellierung. Teubner

Vossen G. (2000) Datenmodelle, Datenbanksprachen und Datenbank-managementsysteme. Oldenbourg

Vossen G., Witt K.-U. (1988) Das SQL/DS-Handbuch. Addison-Wesley

Wedekind H. (1991) Datenbanksysteme - Eine konstruktive Einführung in die Datenverarbeitung in Wirtschaft und Verwaltung. Spektrum Akademischer Verlag

Weikum G. (1988) Transaktionen in Datenbanksystemen - Fehlertolerante Steuerung paralleler Abläufe. Addison-Wesley

Weikum G., Vossen G. (2002) Transactional Information Systems - Theory, Algorithms and the Practice of Concurrency Control and Recovery. Morgan Kaufmann

Wiederhold G. (1983) Database Design. McGraw-Hill

Williams K. et al. (2001) XML et les bases de données. Wrox Press et les Éditions Eyrolles

Williams R. et al. (1982) R* : An Overview of the Architecture. In : Scheuermann P. (Ed.) Improving Database Usability and Responsiveness. Academic Press, pp. 1-27

Witten I.H., Frank E. (1999) Data Mining. Morgan Kaufmann

Zehnder C.A. (2002) Informationssysteme und Datenbanken. vdf Hochschulverlag

Zloof M.M. (1977) Query-by-Example : A Data Base Language. IBM Systems Journal. Vol. 16, No. 4, pp. 324-343

Index