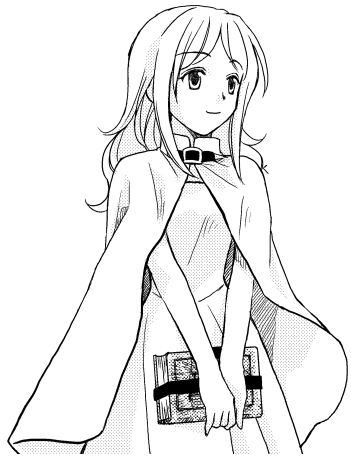
LES GUIDES MANGAS

BASES DE DONNÉES



Auteur	MANA TAKAHASHI	
Dessins	SHOKO AZUMA	
Studio	TREND-PRO	
Traduction	BENJAMIN MONMEGE	enseignant-chercheur
Révision	JEAN-YVES FÉVRIER	agrégé

Ohmsha

no starch press

H&K

Avant-propos

Les bases de données sont des composants essentiels de presque tous les systèmes informatiques d'entreprise, et pourtant leur nature profonde est difficile à cerner.

Ce livre reprend tout à zéro. Il vous plaira si vous avez besoin de comprendre comment fonctionne une base de données, si vous avez envie d'en déployer une pour votre usage personnel, si vous vous demandez comment fonctionnent les sites web, ou encore si vous avez la responsabilité de gérer des données dans votre entreprise.

Les points clefs sont présentés dans les pages de manga. Chaque chapitre se termine par des approfondissements, des compléments et des exercices. Vous pourrez ainsi acquérir une vue d'ensemble des technologies et vous assurer de votre compréhension.

- Le chapitre 1 montre l'intérêt des bases de données et les difficultés auxquelles s'attendre si on s'en prive.
- Le chapitre 2 aborde la terminologie élémentaire ; différents modèles de bases de données sont présentés.
- Le chapitre 3 explique comment concevoir une base de données, et plus spécifiquement une base relationnelle – le modèle le plus utilisé.
- Le chapitre 4 traite du langage SQL, qui sert à gérer les bases de données relationnelles.
- Le chapitre 5 explique le fonctionnement des transactions au sein d'une base de données, les autorisations accordées aux utilisateurs et la protection des données même en cas de panne.
- Le chapitre 6 décrit des applications des bases de données, notamment sur le web.

Ce livre est le fruit des efforts conjoints de nombreuses personnes : Shoko Azuma pour les dessins, TREND-PRO pour la production et Ohmsha pour l'organisation, l'édition et le marketing. Je leur exprime toute ma gratitude.

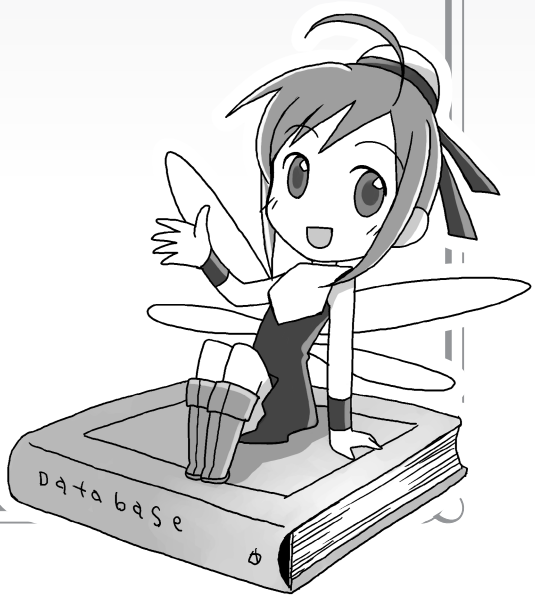
J'espère que ce livre sera utile à tous ses lecteurs !

Mana Takahashi

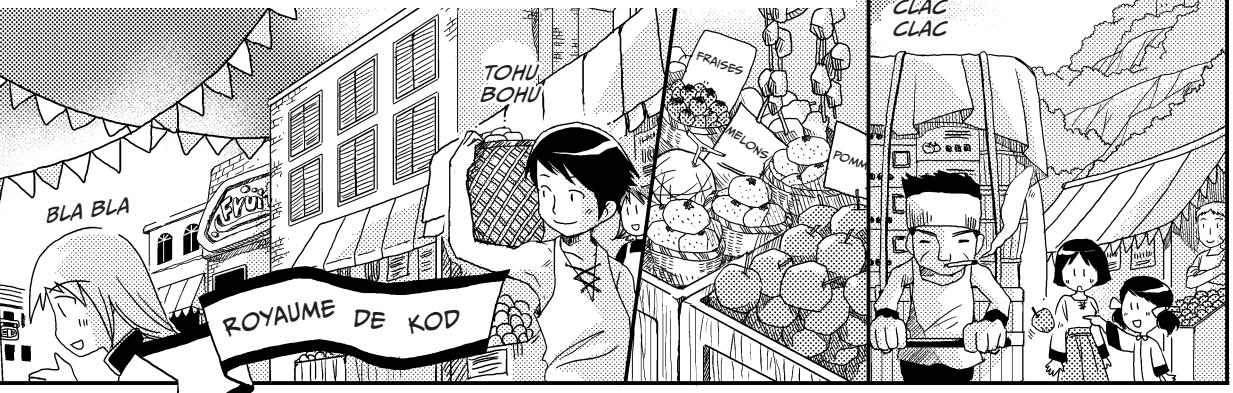
Table des matières

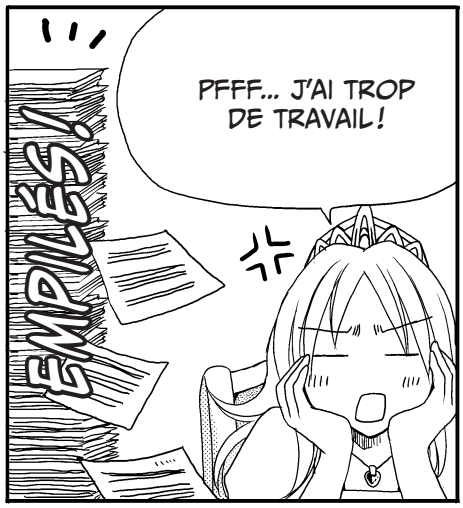
1

QU'EST-CE QU'UNE BASE DE DONNÉES ?



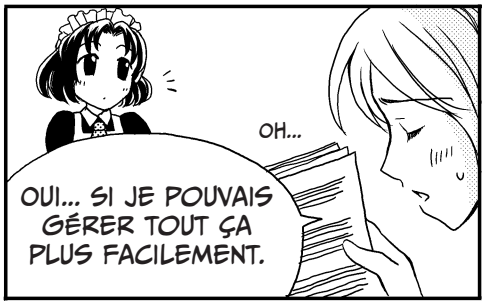
POURQUOI UTILISER UNE BASE DE DONNÉES?

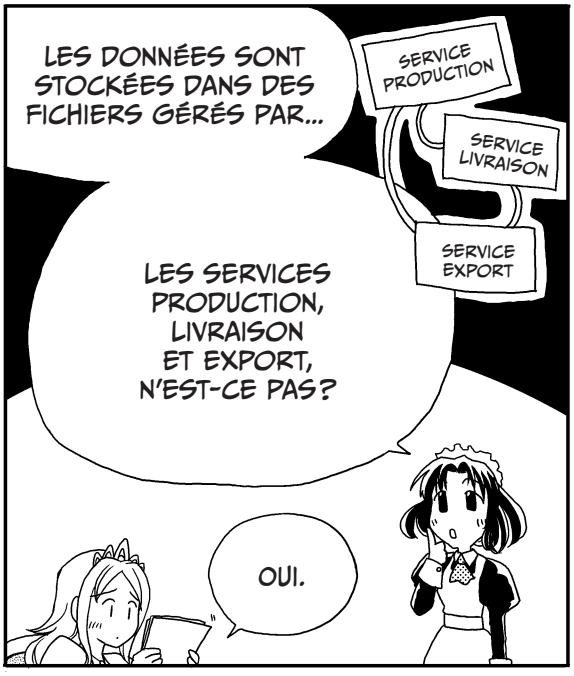



EMPILÉS !
PFFF... J'AI TROP DE TRAVAIL !


LE ROYAUME DE KOD, C'EST «LE PAYS DU FRUIT» !
TERRE BÉNIE
PLÉTHORE DE FRUITS JAMAIS NE NUIT !


VOUS DEVRIEZ VOUS RÉJOUIR DE TOUTES CES COMMANDES.
POUAH !
ET VOILÀ.


OH...
OUI... SI JE POUVAIS GÉRER TOUT ÇA PLUS FACILEMENT.

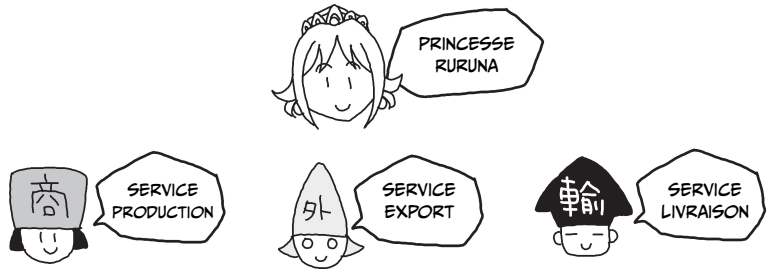

LES DONNÉES SONT STOCKÉES DANS DES FICHIERS GÉRÉS PAR...
SERVICE PRODUCTION
SERVICE LIVRAISON
SERVICE EXPORT
LES SERVICES PRODUCTION, LIVRAISON ET EXPORT, N'EST-CE PAS ?
OUI.


JE SUIS SÛRE QUE CE SYSTÈME EST TRÈS EFFICACE.
MAINTENANT, AU TRAVAIL !!
LA LI DA LA LA LI DUM !
BLAM !

QUOI DE NEUF DANS LE ROYAUME ?

Le royaume de Kod est prospère grâce à l'exportation de fruits. Son activité est répartie en trois services :

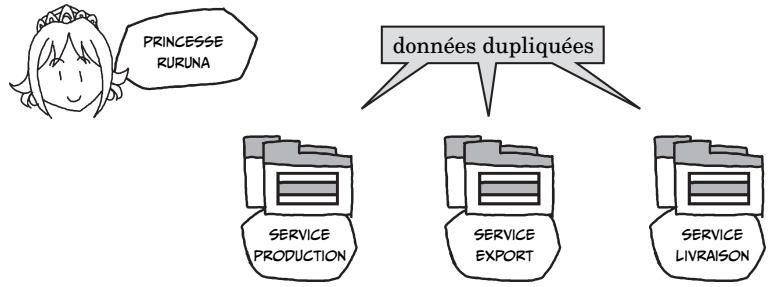
- le service *production* s'occupe des arbres, des récoltes et des stocks ;
- le service *export* gère les relations commerciales avec les clients étrangers ;
- le service *livraison* achemine les fruits jusqu'aux clients.



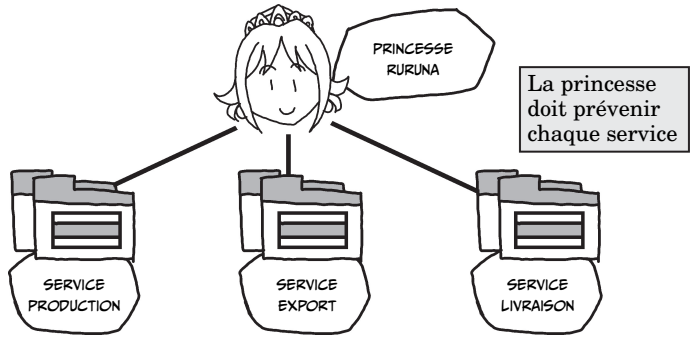
Rien de compliqué à première vue : cette organisation ne devrait guère poser de problème. Pourtant, la princesse Ruruna se plaint d'un manque d'efficacité, de redondances et d'erreurs. Que se passe-t-il ?

ORGANISATION DES DONNÉES

Chaque service stocke les informations dont il a besoin dans un fichier qui lui est propre. Mais ces informations sont également utiles aux autres services. Par exemple, quand le service export réalise une vente, il doit vérifier auprès du service production que les stocks sont suffisants et transmettre au service livraison le nom du client, les quantités et les délais. Chaque service doit ainsi rentrer ses informations dans son fichier, recopier des informations des autres services, et imprimer des reçus qui conservent la trace des échanges. Les données sont dupliquées au lieu d'être partagées.



Mais ce n'est pas tout : la mise à jour des informations communes à tous les services est pénible. Par exemple, quand la princesse souhaite changer le prix des pommes, elle doit prévenir chaque service individuellement. Quelle perte de temps !



LES DONNÉES PEUVENT DEVENIR INCOHÉRENTES

Prévenir chaque service n'est déjà pas amusant, mais en plus plein de problèmes peuvent surgir à cette occasion. D'abord, les services ne modifieront pas tous leurs fichiers en même temps, ce qui posera des problèmes pour les transactions enregistrées à cette période-là. Ensuite, un service peut oublier de mettre à jour son fichier tandis qu'un autre peut rentrer une valeur incorrecte. De la sorte, les services n'auront plus les mêmes informations et potentiellement, aucun n'aura la bonne. Quel enfer !

SERVICE PRODUCTION

Nom	Prix
Melons	800 G
Fraises	150 G
Pommes	120 G
Citrons	200 G

SERVICE EXPORT

Nom	Prix
Melons	800 G
Fraises	150 G
Pommes	100 G
Citrons	200 G

SERVICE LIVRAISON

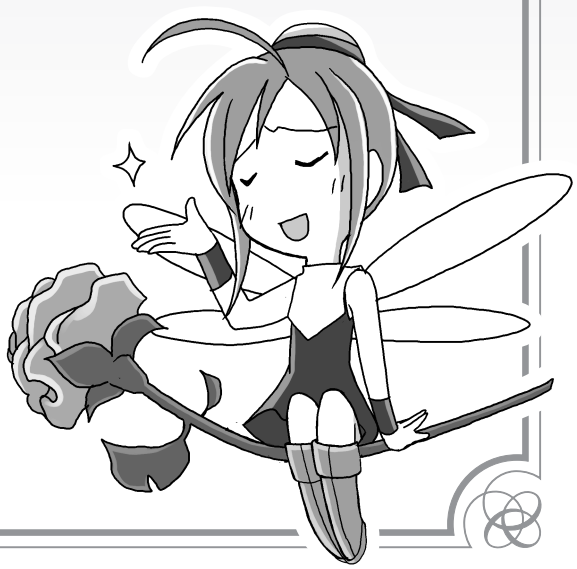
Nom	Prix
Melons	800 G
Fraises	150 G
Pommes	300 G
Citrons	200 G

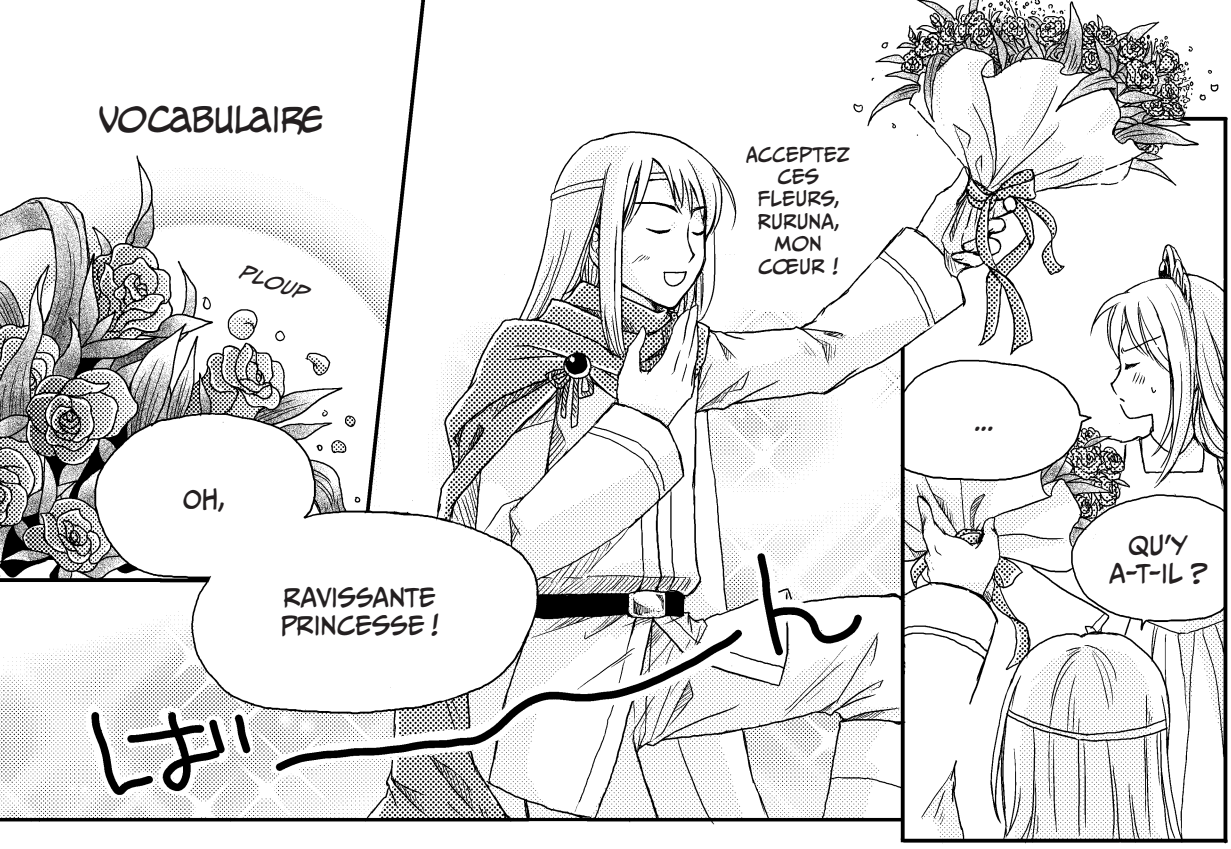
IL EST DIFFICILE D'AJOUTER UN SERVICE

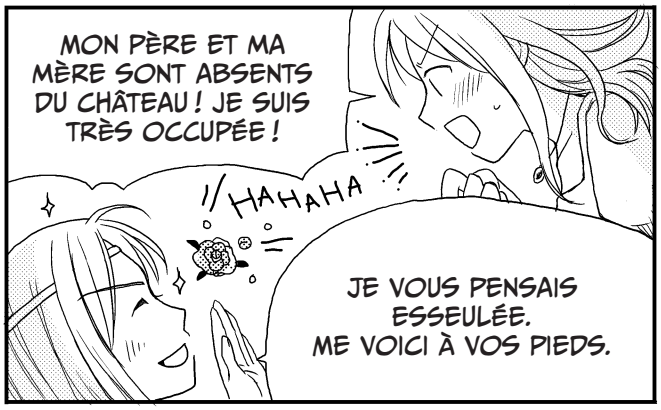
Un autre défaut du système actuel est que sa complexité s'accroît avec le nombre de services. Supposons par exemple que le roi lance une activité touristique à Kod. Quand un guide animera une visite des vergers et parlera de la production de fruits du royaume, il voudra disposer des chiffres les plus récents. Hélas, le service tourisme n'aura pas

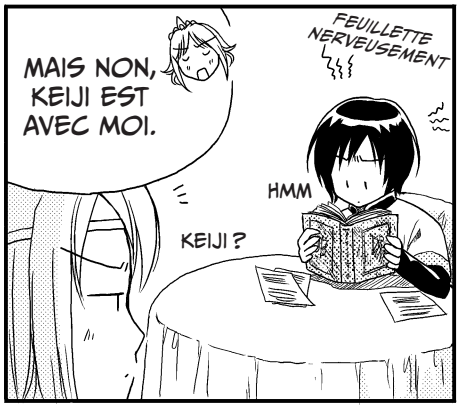
2

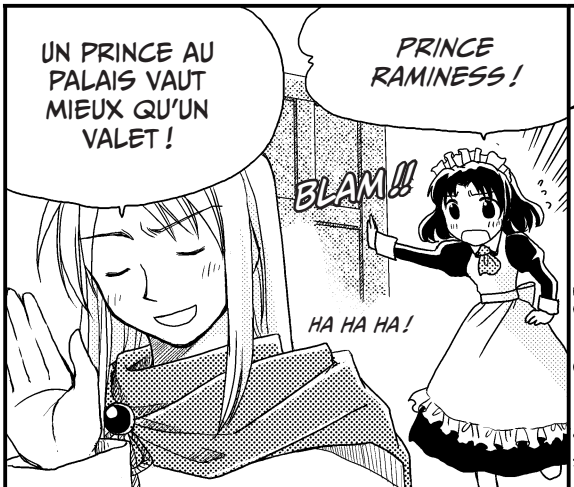
QU'EST-CE QU'UNE BASE DE DONNÉES RELATIONNELLE ?

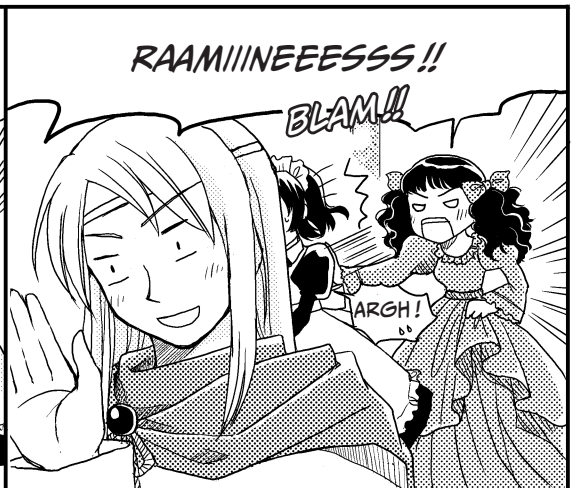



VOCABULAIRE
PLOUP
OH,
RAVISSANTE PRINCESSE !
ACCEPTEZ CES FLEURS, RURUNA, MON CŒUR !
...
QU'Y A-T-IL ?


MON PÈRE ET MA MÈRE SONT ABSENTS DU CHÂTEAU ! JE SUIS TRÈS OCCUPÉE !
HAHAHA
JE VOUS PENSAIS ESSEULÉE. ME VOICI À VOS PIEDS.


MAIS NON, KEIJI EST AVEC MOI.
FEUILLETTE NERVEUSEMENT
HMM
KEIJI ?


UN PRINCE AU PALAIS VAUT MIEUX QU'UN VALET !
PRINCE RAMINESS !
BLAM !!
HA HA HA !

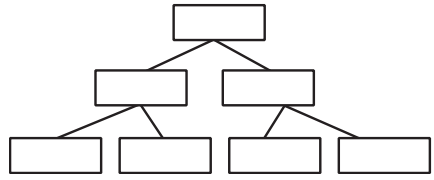

RAAMIIINEEESSS !!
BLAM !!!
ARGH !


PIÉGÉ !
C'EST QUI, ÇA ?
VOUS AVIEZ PROMIS DE ME CONTER FLEURETTE.
OUI... HMM... EH BIEN...
C'EST POUR VOUS.
...
ET MES FLEURS ALORS ?
TIRE, TIRE...
JE VOUS REVIENDRAI, PRINCESSE RURUNA...
OUF !
IL EST SI ENNUYEUX.
PLOUM !
MAIS LE PRINCE RAMINESS EST L'HÉRITIER DU PAYS VOISIN.
VOUS NE DEVRIEZ PAS LE TRAITER SI LÉGÈREMENT.
JE SAIS BIEN...
BONJOUR !

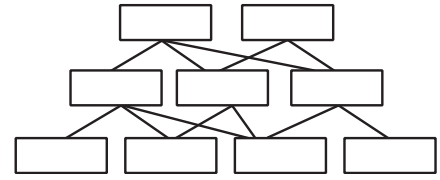
Les modèles de données

Une base de données, oui, mais de quel type ? Un *modèle de données* décrit à la fois la manière dont les données sont organisées et les opérations que l'on peut faire sur elles. Comme je l'ai dit à Ruruna et Keiji, il y a trois grandes manières d'organiser les données.

La première est *hiérarchique* (pensez à la chaîne de commandement dans une armée) : chaque donnée est reliée à un seul parent. La deuxième est *en réseau* (pensez à une guérilla) : chaque donnée peut être reliée à plusieurs parents. Ces deux organisations complexes rendent peu efficaces les recherches sur les données et limitent la flexibilité des requêtes.



organisation hiérarchique



organisation en réseau

La troisième organisation est dite *relationnelle* : les données sont rangées dans des tables qui explicitent leurs relations. C'est visuel et facile à comprendre. Parlons de ce modèle plus en détail.

Opérations sur les données

Dans une base de données relationnelle, on extrait et on traite les données en utilisant des opérations mathématiques définies rigoureusement. Il y a huit opérations principales, que l'on peut répartir en opérations ensemblistes et opérations relationnelles.

Opérations ensemblistes

Les opérations entre ensembles sont l'union, l'intersection, la différence et le produit cartésien. Elles utilisent un ensemble de lignes pour produire un nouvel ensemble de lignes. Les trois premières ne font que sélectionner les lignes de l'entrée qui figureront dans la sortie.

Pour illustrer ces commandes, utilisons deux tables fictives, `table 1` et `table 2`.

table 1

nom	prix
melons	800 G
fraises	150 G
pommes	120 G
citrons	200 G

table 2

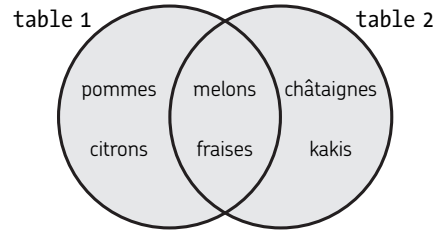
nom	prix
melons	800 G
fraises	150 G
châtaignes	100 G
kakis	350 G

UNION

L'union de table 1 et de table 2 est l'union de leurs lignes : toute ligne qui apparaît dans l'une ou l'autre table figure dans le résultat.

table 1 ∪ table 2

nom	prix
melons	800 G
fraises	150 G
pommes	120 G
citrons	200 G
châtaignes	100 G
kakis	350 G

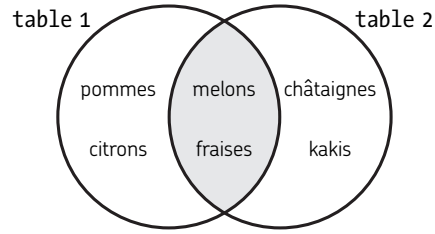


INTERSECTION

L'intersection de table 1 et de table 2 est l'intersection de leurs lignes : seules les lignes qui apparaissent dans les deux tables figurent dans le résultat.

table 1 ∩ table 2

nom	prix
melons	800 G
fraises	150 G



DIFFÉRENCE

La différence de table 1 et de table 2 (comprendre « table 1 moins table 2 ») est l'ensemble des lignes qui n'appartiennent qu'à table 1.

table 1 ⧵ table 2

nom	prix
pommes	120 G
citrons	200 G

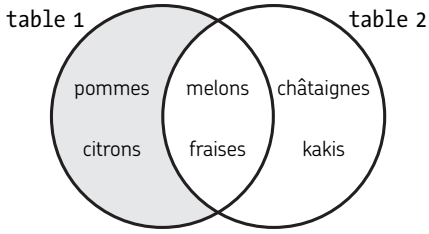
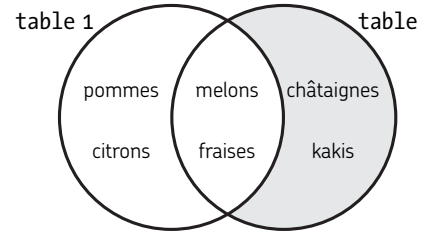


table 2 ⧵ table 1

nom	prix
châtaignes	100 G
kakis	350 G



3

Concevons une base de données !

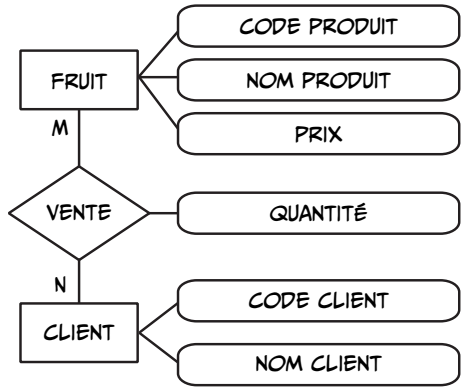

LE MODÈLE E-a
FROU
FROU
KEIJI ?
OÙ ES-TU ?
PSST
OUI...
OUI...
PSST
QU'EST-CE QU'IL A, KEIJI ?
!!
PSST
IL MARMONNE TOUT SEUL...
...BASE DE DONNÉES, JE CROIS...
CHUT !
BONJOUR LES FILLES !
OH, KEIJI...
PRINCESSE RURUNA !
B... B... BONJOUR !
...
D'ACCORD, J'AI COMPRIS.
JE VOIS.
KEIJI !!


REGARD FIXE...
OH, BONJOUR PRINCESSE RURUNA.
SALUT !
BONJOUR TICO.
KEIJI, IL FAUT QUE TU FASSES PLUS ATTENTION !
À QUOI ?
BONJOUR RURUNA !!
NOUS SOMMES LES SEULS À VOIR TICO.
ON DIRAIT UN FOU !
ÇA FAIT BIZARRE QUAND TU LUI PARLES DEVANT D'AUTRES GENS.
OH !
DÉSOLÉ... VOUS AVEZ RAISON.
C'EST VRAI.
DE QUOI AVEZ-VOUS PARLÉ ?
JE NE SAIS PAS PAR OÙ COMMENCER POUR CRÉER UNE BASE DE DONNÉES.
TICO ME DONNAIT DES CONSEILS.
BIEN !
VOUS AVEZ DÛ TRAVAILLER TOUTE LA MATINÉE.
ET ?

QU'EST-CE QUE LE MODÈLE E-A ?

Avant de créer une base, il faut définir ce qu'elle va contenir. Pour cela, il est judicieux de modéliser le système qui produit les données, c'est-à-dire d'en élaborer une représentation simplifiée. C'est ce que Ruruna et Keiji ont fait avec les exportations du royaume de Kod.

Le type de modélisation le mieux adapté aux bases de données s'appelle E-A, pour entité-association ; il est similaire à la modélisation Merise. On appelle *entités* les choses du monde réel sur lesquelles on possède des données qu'il faut gérer, par exemple des fruits ou des clients. Chaque entité a des *propriétés*. Une *association* est une relation entre entités, par exemple la relation de vente qui existe entre l'entité fruit et l'entité client. Dans le modèle E-A, on tient compte de la *cardinalité*, c'est-à-dire du nombre d'associations entre deux entités.

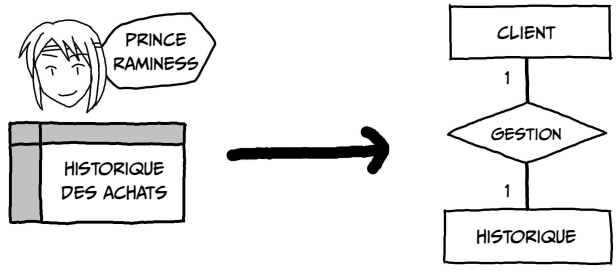


EXEMPLES

Essayez de modéliser les systèmes ci-dessous après avoir lu leur description – donnez-vous quelques instants de réflexion avant de regarder la solution. Gardez à l'esprit qu'il existe souvent plusieurs modélisations correctes : la solution n'est pas toujours unique.

CAS 1

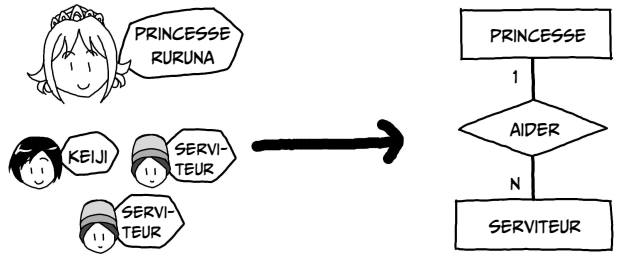
Chaque client du royaume de Kod tient à jour un historique de ses achats.



L'association est de type *un-à-un*.

CAS 2

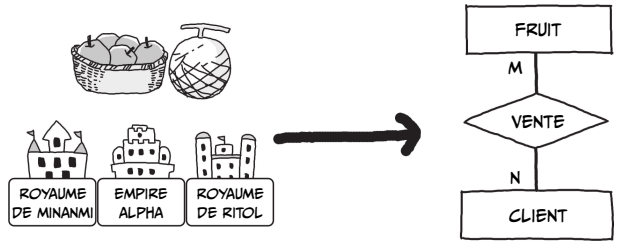
Une princesse est entourée de plusieurs personnes, qui ne servent qu'elle.



L'association est de type *un-à-plusieurs*.

CAS 3

Les clients du royaume de Kod importent plusieurs types de fruits ; chaque type de fruit est exporté vers plusieurs clients.

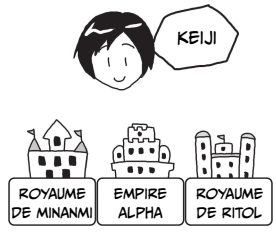


L'association est de type *plusieurs-à-plusieurs*.

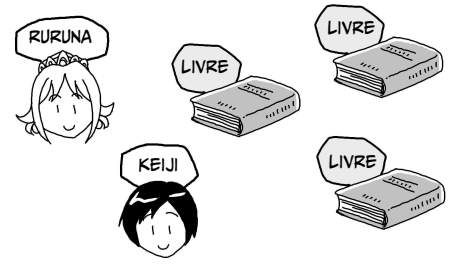
QUESTIONS

Testez votre compréhension du modèle E-A ! Les réponses figurent page 80.

Q 1 Chaque client du royaume de Kod a un seul interlocuteur. Cet employé gère plusieurs clients.

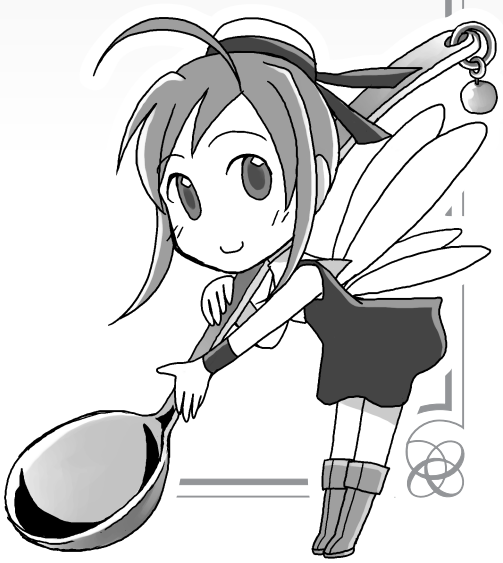


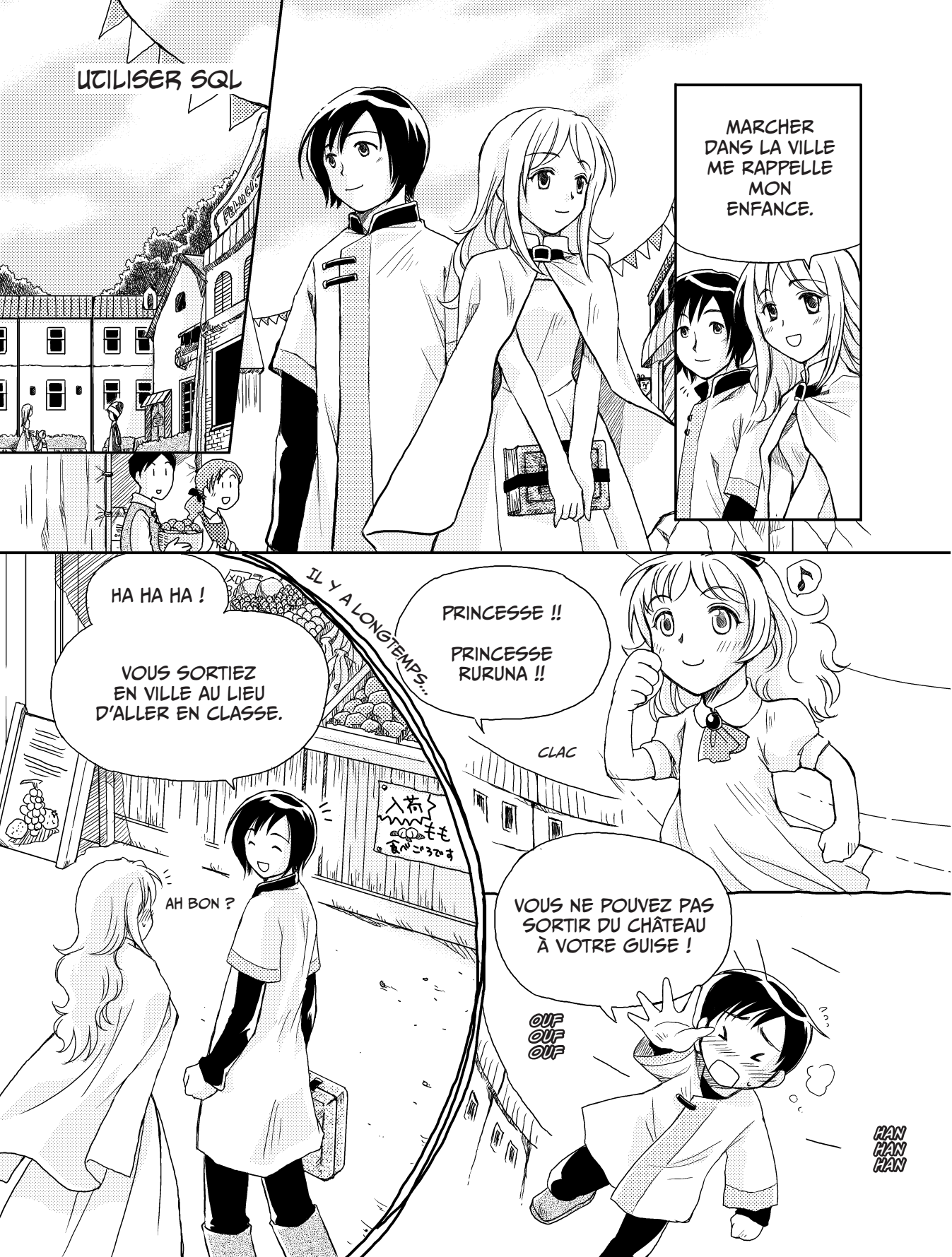
Q 2 Chaque adhérent d'une bibliothèque peut emprunter plusieurs livres. Chaque livre peut être emprunté par plusieurs adhérents.



4

APPRENONS SQL !




UTILISER SQL
MARCHER DANS LA VILLE ME RAPPELLE MON ENFANCE.
IL Y A LONGTEMPS...
PRINCESSE !!
PRINCESSE RURUNA !!
CLAC
HA HA HA !
VOUS SORTIEZ EN VILLE AU LIEU D'ALLER EN CLASSE.
AH BON ?
VOUS NE POUVEZ PAS SORTIR DU CHÂTEAU À VOTRE GUISE !
OUF OUF OUF
HAN HAN HAN


TU SAIS QUOI, KEIJI ? TU FERAIS MIEUX DE RESTER AU CHÂTEAU SI C'EST POUR ME CRIER DESSUS TOUT LE TEMPS.
SOUFFLE
OUF !
HAN
M... MAIS...
UN VALET DOIT SUIVRE SA PRINCESSE !
HH
SENS DU DEVOIR !
TU ME SUIS PAR DEVOIR ?
PAS PARCE QUE TU ES MON AMI ?
EH BIEN...
VOYONS VOIR...
QUEL IDIOT !
OH NON, C'EST PAS CE QUE JE VOULAIS DIRE !
FROU
PRINCESSE...
OUCH...
JE VOUS AI RETROUVÉE, PRINCESSE RURUNA.
CAPITAINE DE LA GARDE I... IGOR... !
ど-B-O-U-M-!!

Survol de SQL

Dans ce chapitre, Ruruna et Keiji ont commencé à apprendre le langage SQL, qui sert à dialoguer avec une base de données. Ses commandes se répartissent en trois groupes, qui sont comme des sous-langages.

– Le premier groupe (*Data Definition Language*) sert à manipuler les **tables** d'une base de données : création, modification ou destruction.
– Le deuxième groupe (*Data Manipulation Language*) permet d'ajouter, mettre à jour ou effacer des **données** dans une table.
– Le troisième groupe (*Data Control Language*) gère les **utilisateurs** et leurs permissions. Il permet aussi de s'assurer qu'il ne naîtra pas de conflit entre les données.

Quel que soit le groupe, on dialogue avec la base en entrant des *requêtes* (ou *instructions*), qui ressemblent à des phrases. Chaque requête est constituée de *clauses* faites de *commandes* (ou *mots clefs*) suivies d'arguments dont certains peuvent être remplacés par des *jokers*.

Les commandes sont des mots anglais, ce qui donne à SQL l'apparence d'une langue naturelle. Il n'y a pas besoin d'être informaticien ou expert en bases de données pour écrire les requêtes usuelles. N'oubliez pas le point-virgule à la fin de la requête !

Traduction des commandes	
AVG	moyenne
COUNT	compter
CREATE	créer
DELETE	effacer
DISTINCT	distinct
FROM	depuis
GROUP BY	grouper par
HAVING	ayant
IN	parmi
INSERT	insérer
INTO	vers
NULL	vide
ORDER BY	trier par
PRIMARY KEY	clef primaire
SELECT	sélectionner
SUM	somme
UPDATE	mettre à jour
VALUES	valeurs
WHERE	où

Chercher des informations avec SELECT

S'il ne fallait connaître qu'une seule commande SQL, ce serait `SELECT`. Elle permet de rechercher des données dans une table (ou plusieurs) en précisant des conditions. Par exemple, la requête ci-dessous recherche dans la table `produits` toutes les lignes dans lesquelles le `prix` est égal à 200 (G). On peut l'écrire sur une ou plusieurs lignes.

```
SELECT *
FROM produits
WHERE prix = 200;
```

En français : `SÉLECTIONNER` *toutes les lignes*
`DE LA TABLE produits`
`DANS LESQUELLES prix = 200 ;`

Le joker *, placé à l'endroit où l'on aurait attendu le nom d'une colonne (ou plusieurs), signifie « toutes les colonnes ».

Remarquons que l'ordinateur accepte aussi bien `SELECT` que `select` ou `Select` ou toute autre variante. On n'emploie les majuscules dans les noms de commandes que pour aider à les distinguer des arguments.

ÉCRIRE DES CONDITIONS

Dans l'exemple précédent, on demandait à la base de n'indiquer que les produits dont le prix valait 200 (G) ; cette précision est ce que l'on appelle une *condition*. Il faut l'entendre comme « afficher cette ligne à condition que... »

En SQL, les conditions sont introduites par le mot clef `WHERE`. Voyons maintenant les principales manières de les exprimer.

OPÉRATEURS DE COMPARAISON

Lorsque la comparaison est numérique, on emploie les symboles mathématiques usuels. On utilise `<>` pour signifier « différent de ».

Opérateur	Exemple
`=`	`prix = 200`
`<`	`prix < 200`
`<=`	`prix <= 200`
`>`	`prix > 200`
`>=`	`prix >= 200`
`<>`	`prix <> 200`

Pour tester si une case a été laissée vide, on utilise le mot clef `IS NULL`, comme dans

```
prix IS NULL
```

OPÉRATEURS LOGIQUES

Les opérateurs logiques permettent de combiner des conditions à l'aide des mots clefs `AND` (et), `OR` (ou) et `NOT` (négation).

Opérateur	Exemple
`AND`	`prix = 100 AND quantité > 1000`
`OR`	`prix = 100 OR quantité > 1000`
`NOT`	`NOT prix = 100`

équivalent à `prix <> 100`

MOTIFS

Lorsque la condition porte sur une chaîne de caractères, il arrive que l'on ne soit pas sûr de l'écriture qui a été utilisée lors de l'insertion des données dans la table. Vend-on des `pommes`, des `pomme`, des `Pommes` ou des `Pomme` ? A-t-on convenu d'écrire `pasteque` au lieu de `pastèque` pour éviter les problèmes de codage des caractères ? Et pour éviter d'introduire des espaces dans les noms, ce qui oblige à les entourer ensuite des guillemets, peut-être a-t-on écrit `pommes-Gala` au lieu de `pommes Gala` ?

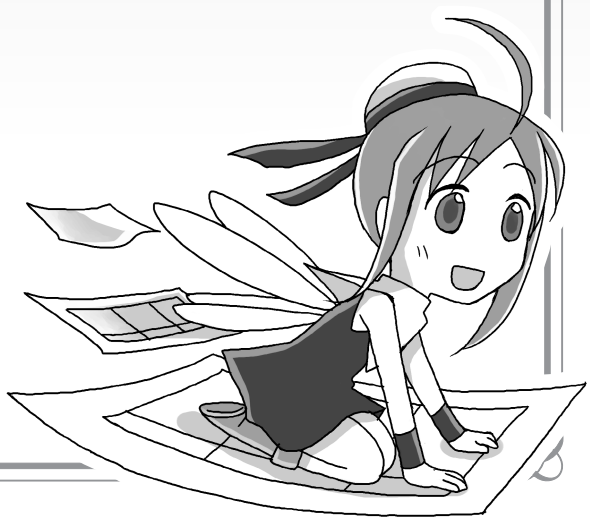
D'autres fois, on est sûr de l'écriture mais on recherche exprès une information partielle. Si par exemple on a enregistré des adresses électroniques, en se concentrant sur ce qui suit `@` (comme `@chateau.kod`) on peut savoir combien de personnes utilisent tel ou tel domaine.

Joker	Description	Exemple	Correspondances
`_`	remplace 1 caractère	`_omme`	pomme, Pomme, homme, tomme...
`%`	remplace 0 ou plusieurs caractères	`%ons`	melons, citrons, cornichons...
		`n%`	noix, nectarines, navets...
		`n%s`	nectarines, navets...

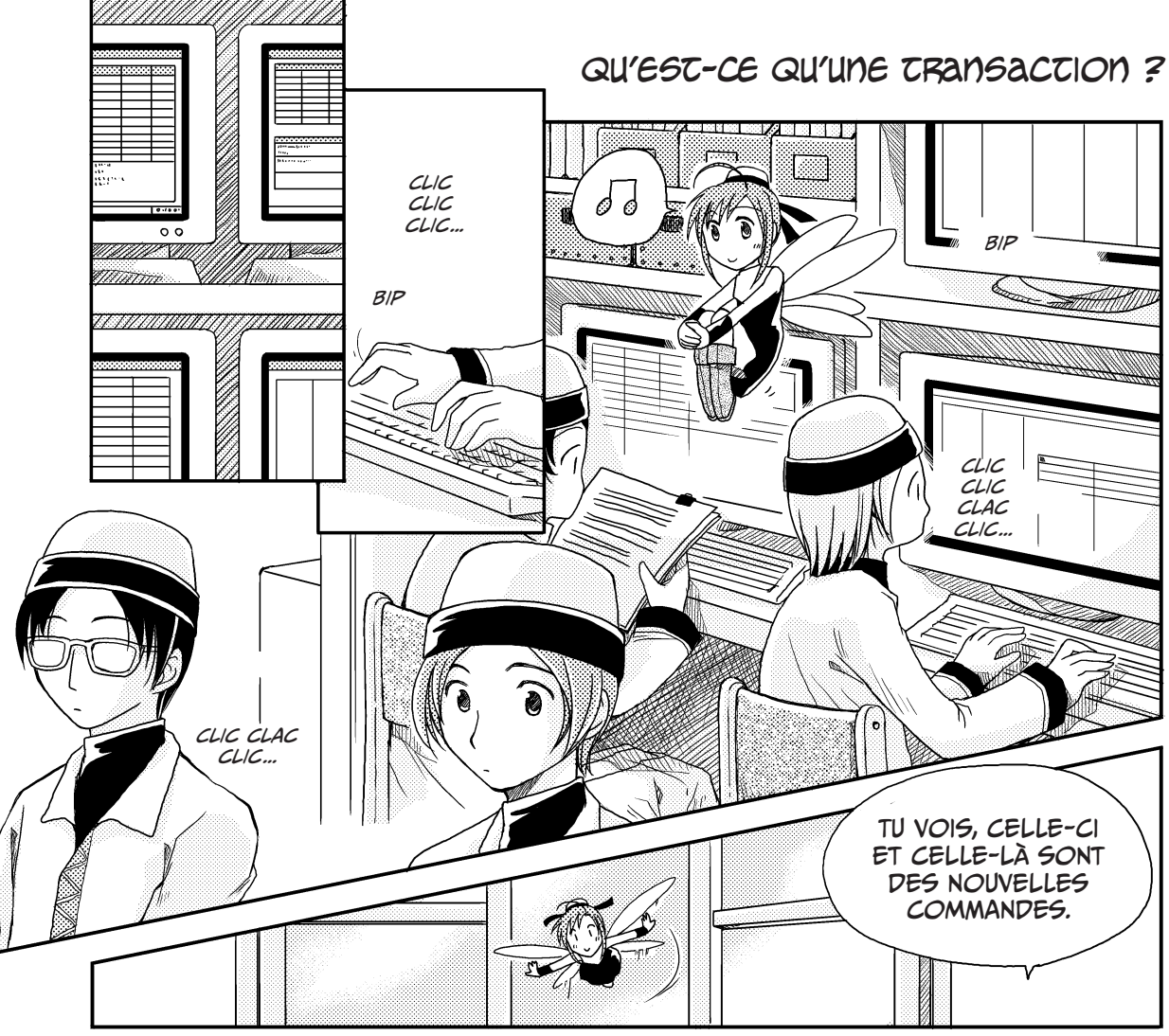
```
SELECT nom_produit FROM produits WHERE nom_produit LIKE '%ons' ;
```

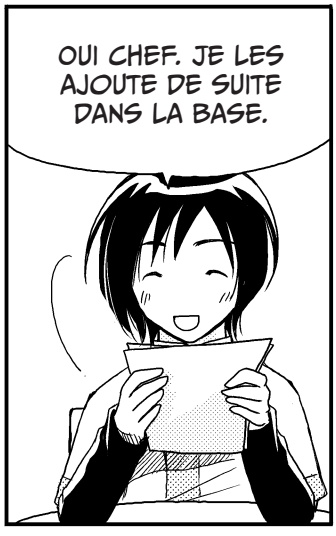
5

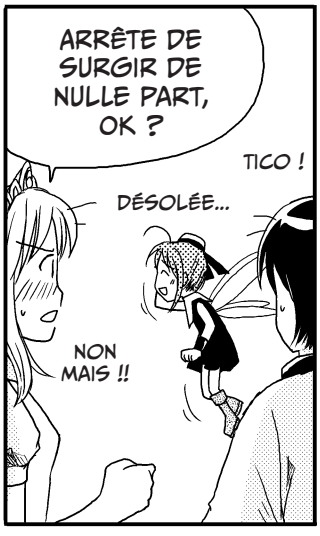
APPRENONS À GÉRER UNE BASE DE DONNÉES !



QU'EST-CE QU'UNE TRANSACTION ?



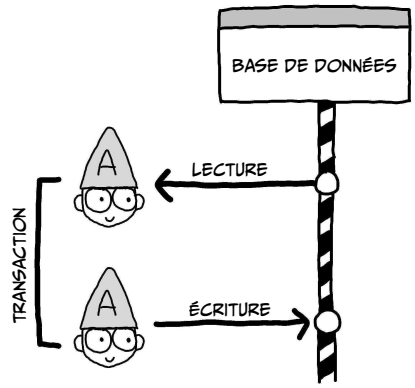





BON... JE VEUX AUSSI TE REMERCIER...
MAIS IL NOUS RESTE BEAUCOUP À APPRENDRE.
PAR EXEMPLE, JE ME DEMANDE COMMENT LA BASE ARRIVE À FONCTIONNER ALORS QUE TANT D'UTILISATEURS Y ACCÈDENT EN MÊME TEMPS.
À CE PROPOS, LA QUESTION DE LA SÉCURITÉ M'INQUIÈTE UN PEU.
APPAREMMENT, VOUS AVEZ QUELQUES DOUTES.
ON PEUT DIRE ÇA.
POUR FAIRE LE POINT,
HUM !
J'AI FAIT QUELQUES RECHERCHES.
AH OUI ?
MON EXPOSÉ S'INTITULE :
COMMENT LA BASE GÈRE-T-ELLE DE NOMBREUX ACCÈS SIMULTANÉS ?
FLASH !
OH !
J'AI MÊME FAIT DES DESSINS !
WOUAAH, SUPER !
PETIT THÉÂTRE DE LA BASE
UN SPECTACLE !

PROPRIÉTÉS DES TRANSACTIONS

Une base de données permet de rechercher, insérer, mettre à jour et supprimer des données. On appelle *transaction* un ensemble d'opérations réussies exécutées par un seul utilisateur.



Il est indispensable que les transactions s'exécutent sans introduire d'incohérences dans les données, y compris si la base est partagée par un grand nombre d'utilisateurs ou si une panne intervient en plein milieu d'une transaction. Pour offrir cette garantie, il est nécessaire et suffisant que la base respecte les quatre propriétés ci-dessous. On peut les retenir en remarquant que leurs premières lettres forment le mot *ACID* (« acide » en anglais).

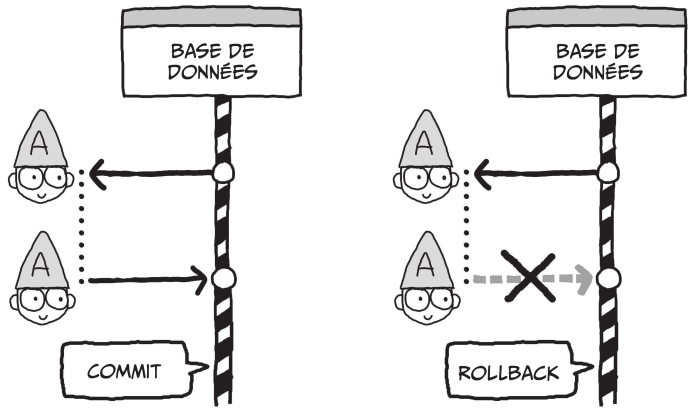
PROPRIÉTÉS REQUISES D'UNE TRANSACTION

Propriété	Description
Atomicité	Une transaction doit se terminer par un `commit` ou par un `rollback`.
Cohérence	L'exécution d'une transaction n'entraîne jamais de perte de cohérence dans les données de la base.
Isolement	Que des transactions soient exécutées en parallèle ou les unes après les autres, les résultats doivent être les mêmes.
Durabilité	Une panne ne doit pas modifier le résultat d'une transaction terminée.

Examinons ces propriétés en détail.

ATOMICITÉ

En grec ancien, *atomos* signifie « indivisible ». Les transactions d'une base sont dites atomiques parce qu'elles ne sont pas découpables en morceaux qui doivent être réussis tandis que d'autres pourraient échouer. C'est tout ou rien. Si toutes les actions de la transaction sont réussies, celle-ci se termine par un `commit` qui la valide définitivement. Sinon, toutes les actions sont annulées et la transaction s'achève par un `rollback`.



Usuellement, la base effectue toute seule le `commit` ou le `rollback`, mais on peut aussi prendre la main et dire explicitement à la base ce que l'on souhaite qu'elle fasse. On tape alors l'une des commandes suivantes.

```
COMMIT ;
```

```
ROLLBACK ;
```

QUESTIONS

(réponses page 156)

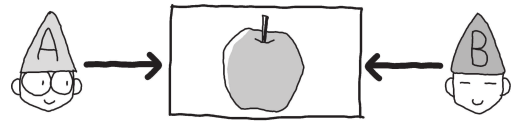
Avez-vous bien compris l'atomicité ?

Q 1 Quelle instruction SQL est utilisée pour finaliser une transaction ?

Q 2 Quelle instruction SQL est utilisée pour annuler une transaction ?

COHÉRENCE

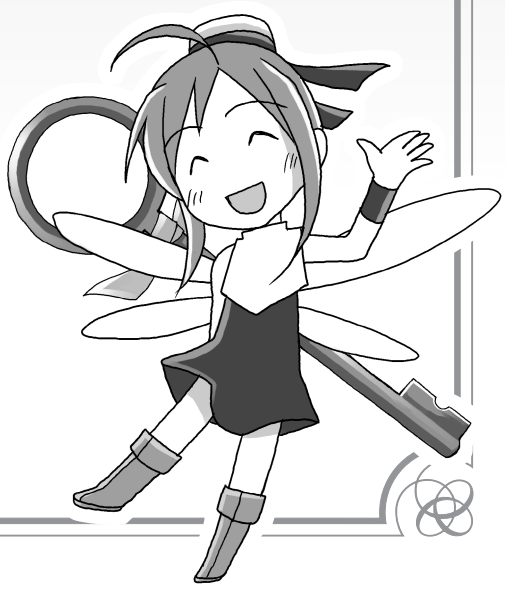
Une transaction ne doit pas introduire d'erreur. Si les données de la base étaient cohérentes avant la transaction, elles doivent aussi l'être après. Les problèmes risquent de surgir surtout lorsque plusieurs utilisateurs manipulent les mêmes données au même moment. On dit qu'ils accèdent aux mêmes *ressources*.



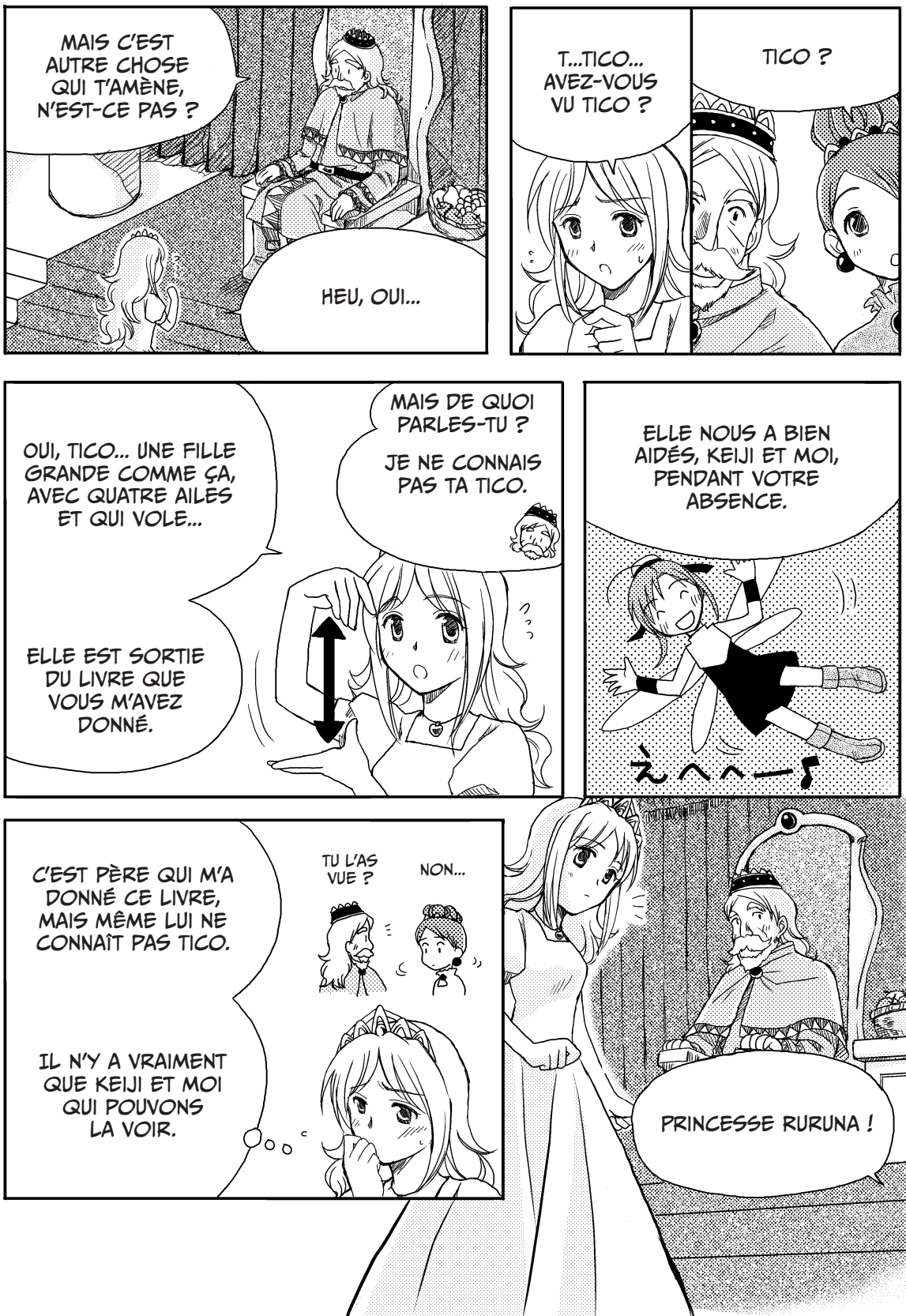
Ainsi, Keiji a donné l'exemple d'Alex et de Béa qui essayaient tous deux d'ajouter 10 pommes au stock initial de 30 pommes. Après les transactions, la base commençait par indiquer 40 pommes au lieu de 50 ; ce type d'erreur s'appelle une *mise à jour perdue*. Une base doit permettre à plusieurs transactions d'utiliser les mêmes ressources en même temps sans créer de mise à jour perdue.

6

LES BASES DE DONNÉES SONT PARTOUT !

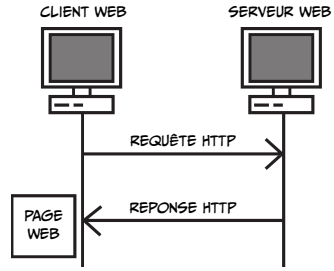


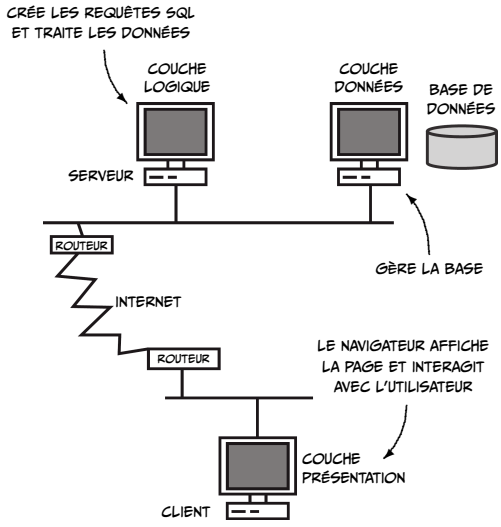

はむっ*
* SCRONCH SCRONCH
はっはっは!!
DÉLICIEUX ! C'EST BIEN UN FRUIT DU ROYAUME DE KOD !
PÈRE !
OUI ? QU'Y A-T-IL ? TOI AUSSI TU VEUX UNE BANANE, RURUNA ?
MAIS NON !
PÈRE, VOUS NE FAITES QUE MANGER DES FRUITS DEPUIS VOTRE RETOUR.
PARDON ! MAIS CE SONT VRAIMENT LES MEILLEURS.
RURUNA A BIEN TENU LES RÊNES PENDANT MON ABSENCE.
LE ROYAUME EST PLUS PROSPÈRE QUE JAMAIS !
はっはっは
CES BASES DE DONNÉES SONT VRAIMENT PRATIQUES !
ホホホホ


MAIS C'EST AUTRE CHOSE QUI T'AMÈNE, N'EST-CE PAS ?
HEU, OUI...
T...TICO... AVEZ-VOUS VU TICO ?
TICO ?
OUI, TICO... UNE FILLE GRANDE COMME ÇA, AVEC QUATRE AILES ET QUI VOLE...
ELLE EST SORTIE DU LIVRE QUE VOUS M'AVEZ DONNÉ.
MAIS DE QUOI PARLES-TU ?
JE NE CONNAIS PAS TA TICO.
ELLE NOUS A BIEN AIDÉS, KEIJI ET MOI, PENDANT VOTRE ABSENCE.
えへへー♪
C'EST PÈRE QUI M'A DONNÉ CE LIVRE, MAIS MÊME LUI NE CONNAÎT PAS TICO.
IL N'Y A VRAIMENT QUE KEIJI ET MOI QUI POUVONS LA VOIR.
TU L'AS VUE ?
NON...
PRINCESSE RURUNA !

Les bases de données sur le web

Les bases de données sont devenues indispensables tant aux entreprises qu'aux particuliers, qui les utilisent sans le savoir à travers des pages web. Schématiquement, un clic sur un lien fait émettre au navigateur (le *client web*) une requête HTTP (*HyperText Transfer Protocol*) en direction d'un *serveur web*, qui renvoie une réponse HTTP contenant la page web demandée.





Lorsque le serveur web est relié à une base de données située sur un autre ordinateur, le schéma devient celui ci-contre. Le travail de création de la page web a été décomposé en trois niveaux[1] : la couche données, la couche logique et la couche présentation.

La **couche données** reçoit des requêtes SQL, interroge la base (qui est située dans le serveur de données) et renvoie les données demandées.

La **couche logique** crée les requêtes SQL et, à l'aide d'un langage de programmation, traite les données reçues. Si le serveur est trop chargé, ses tâches peuvent être réparties sur un serveur applicatif (qui traite les données) et un serveur web (qui crée la page web).

La **couche présentation**, qui apparaît dans le navigateur de l'utilisateur, affiche la page web en combinant un balisage HTML, des instructions de présentation en CSS, éventuellement du code JavaScript, et des images ou vidéos spécifiées par leur URL (*Uniform Resource Locator*). Elle attend également les actions de l'utilisateur.

Cette configuration en trois couches est conceptuellement simple car elle découple les fonctions, mais aussi flexible en pratique puisque l'on peut travailler sur un aspect sans modifier les autres.

Utiliser des procédures stockées

Si l'ordinateur qui génère les requêtes SQL est distinct de celui qui héberge la base de données, le réseau qui les relie peut devenir encombré, donc lent, voire saturé. Pour y remédier, on peut stocker des programmes dans la base elle-même, ce qui facilite en

[1] Ce système client/serveur est souvent dit en « trois tiers » ; ceci résulte hélas d'une traduction erronée du faux-ami anglais *tiers*, qui signifie « niveaux » ou « couches ».

outre le développement d'applications en fournissant aux programmeurs des interfaces de haut niveau faisant abstraction des tâches basiques et répétitives. Ces programmes stockés peuvent être de trois types.

type	caractéristiques
procédure stockée	effectue un traitement mais ne renvoie rien
fonction stockée	renvoie des valeurs lors de son exécution
déclencheur	lancé automatiquement avant ou après des requêtes

QUESTIONS

(réponses page 188)

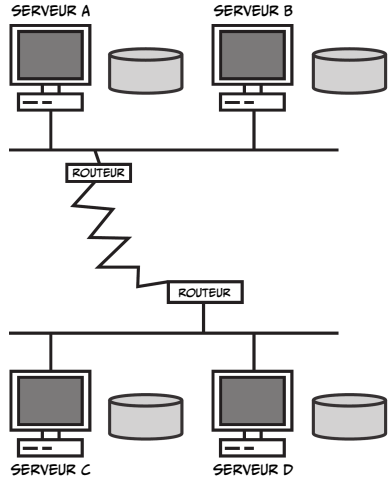
Q 1 Dans un système client/serveur « trois tiers », sur quelle couche la base de données fonctionne-t-elle ?

Q 2 Et sur quelle couche reçoit-on les actions de l'utilisateur ?

QU'EST-CE QU'UNE BASE DE DONNÉES DISTRIBUÉE ?

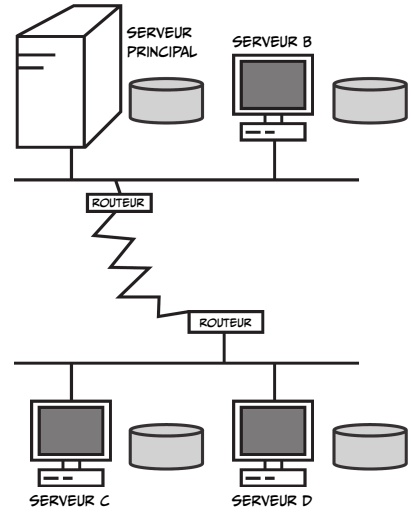
Une base de données peut être répartie sur plusieurs serveurs situés sur le même réseau local ou géographiquement éloignés. Dans tous les cas, la base offre une interface unique aux utilisateurs, comme si elle n'utilisait qu'un seul serveur. Selon leur rôle, on dit que les serveurs sont distribués horizontalement ou verticalement.

DISTRIBUTION HORIZONTALE



– aucun serveur n'est privilégié
– les serveurs s'échangent des données
– protection élevée contre les pannes
– peut servir rapidement des clients situés dans des pays éloignés

DISTRIBUTION VERTICALE



– un seul serveur répond aux requêtes
– les autres lui fournissent des données
– permet d'isoler les données entre les services d'une même entreprise

aide-mémoire SQL

recherche élémentaire

```
SELECT prix FROM fruits ;
SELECT prix, nom FROM fruits ;
```

recherche conditionnelle

```
SELECT nom FROM fruits WHERE prix >= 100 ;
```

recherche par motif

```
SELECT code FROM fruits WHERE nom LIKE '%anan%' ;
```

recherche triée

```
SELECT nom FROM fruits WHERE prix > 100 ORDER BY prix ;
```

recherche et regroupement

```
SELECT région, AVG(prix) FROM fruits GROUP BY région HAVING AVG(prix) > 100 ;
```

jointure

```
SELECT fruits.code, tonnes FROM fruits, stocks WHERE fruits.code = stocks.code ;
```

créer une table

```
CREATE TABLE fruits (
    code INT NOT NULL,
    nom VARCHAR(255),
    prix INT,
    PRIMARY KEY(code)
) ;
```

créer une vue

```
CREATE VIEW mille_lots AS
    SELECT * from relevés
    WHERE quantité >= 1000 ;
```

effacer (irrémédiablement) **une table**

```
DROP TABLE fruits ;
```

effacer une vue

```
DROP VIEW mille_lots ;
```

ajouter une ligne

```
INSERT INTO fruits (code, nom, prix) VALUES (420, 'ananas', 750) ;
```

mettre à jour une ligne

```
UPDATE fruits SET prix = 570 WHERE code = 420 ;
```

effacer une ligne

```
DELETE FROM clients WHERE nom = 'Fayite' ;
```

Index