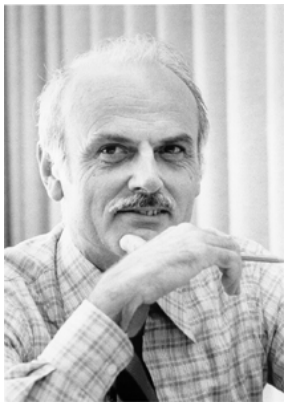
E. F. Codd (Wikipedié)



Born	Edgar Frank Codd August 19, 1923[1][2] Isle of Portland, England
Died	April 18, 2003 (aged 79) Williams Island, Aventura, Florida, USA
Fields	Computer Science
Institutions	University of Oxford University of Michigan IBM
Alma mater	Exeter College, Oxford University of Michigan
Thesis	*Propagation, Computation, and Construction in Two-dimensional cellular spaces* (1965)
Doctoral advisor	John Henry Holland[3]
Known for	OLAP Relational model[4] Codd's cellular automaton Codd's 12 rules Boyce–Codd normal form
Notable awards	Turing Award[5]

Résumé

- Le *Modele relationnel* pour la gestion des Bases De Données (BDD) est un modèle de BDD basé sur la logique du premier ordre proposé et formulé pour la 1ere fois par Edgar F. Codd (1969).

Résumé

- Le *Modele relationnel* pour la gestion des Bases De Données (BDD) est un modèle de BDD basé sur la logique du premier ordre proposé et formulé pour la 1ere fois par Edgar F. Codd (1969).
- Dans une BDD *relationnelle* l'information est organisée dans des tableaux à deux dimensions appelées *relations* ou *tables*.

Résumé

- Le *Modele relationnel* pour la gestion des Bases De Données (BDD) est un modèle de BDD basé sur la logique du premier ordre proposé et formulé pour la 1ere fois par Edgar F. Codd (1969).
- Dans une BDD *relationnelle* l'information est organisée dans des tableaux à deux dimensions appelées *relations* ou *tables*.
- Une BDD est donc un ensemble de tables. Les lignes sont appelées *tuples*, *nuplets* ou encore *enregistrements*.

Résumé

- Le *Modele relationnel* pour la gestion des Bases De Données (BDD) est un modèle de BDD basé sur la logique du premier ordre proposé et formulé pour la 1ere fois par Edgar F. Codd (1969).
- Dans une BDD *relationnelle* l'information est organisée dans des tableaux à deux dimensions appelées *relations* ou *tables*.
- Une BDD est donc un ensemble de tables. Les lignes sont appelées *tuples*, *nuplets* ou encore *enregistrements*.
- Le *modèle relationnel* fournit une méthode déclarative pour spécifier *données* (l'ensemble étudié) et *requêtes* (questions permises sur cet ensemble).

Résumé

- Le *Modele relationnel* pour la gestion des Bases De Données (BDD) est un modèle de BDD basé sur la logique du premier ordre proposé et formulé pour la 1ere fois par Edgar F. Codd (1969).
- Dans une BDD *relationnelle* l'information est organisée dans des tableaux à deux dimensions appelées *relations* ou *tables*.
- Une BDD est donc un ensemble de tables. Les lignes sont appelées *tuples*, *nuplets* ou encore *enregistrements*.
- Le *modèle relationnel* fournit une méthode déclarative pour spécifier *données* (l'ensemble étudié) et *requêtes* (questions permises sur cet ensemble).
- L'utilisateur décrit les informations que contient la BDD et quelles informations il souhaite connaître et laisse le *système de gestion de BDD* (SGBD) gérer la description machine de la base et son stokage ainsi que la manière dont il retrouve l'information.

Quelques considérations générales

- Toutes les données sont représentées comme des *relations n−aires*, des sous-ensembles de produits cartésiens de n ensembles.

Quelques considérations générales

- Toutes les données sont représentées comme des *relations $n-$aires*, des sous-ensembles de produits cartésiens de n ensembles.
- Calculs sur les données : *calcul relationnel* ou *algèbre relationnelle*.

Quelques considérations générales

- Toutes les données sont représentées comme des *relations n−aires*, des sous-ensembles de produits cartésiens de n ensembles.
- Calculs sur les données : *calcul relationnel* ou *algèbre relationnelle*.
- Le concepteur de BDD relationnelle crée un modèle logique *cohérent* (sans contradiction).

Attributs

Définition

On considère donné un ensemble infini $\mathcal{A}$, dont les éléments sont appelés des *attributs*, un ensemble D (ensemble des *domaine*), et une application dom de $\mathcal{A}$ dans D.

Remarque

Si $A \in \mathcal{A}$, l'élément $\mathrm{dom}(A)$ de D est appelé *domaine* de A.
La domaine de A est lui-même un ensemble, par exemple ensemble des entiers, des flottants, des chaînes de caractères...

Attributs

Définition

On considère donné un ensemble infini $\mathcal{A}$, dont les éléments sont appelés des *attributs*, un ensemble D (ensemble des *domaine*), et une application dom de $\mathcal{A}$ dans D.

Exemple

Soit le lycée **Pierre Dupont** contenant des CPGE. Les classes sont des couples **(filière,numéro)** comme **(MPSI,1)** ; **(MPSI,2)** ou **(PCSI,1)**.

- **filière** est un attribut dont le domaine est l'ensemble fini de chaînes de caractères {**MPSI,PCSI,PC,PSI,MP,BCPST,HK**}.
- **numéro** est un attribut dont le domaine est l'ensemble $\mathbb{N}^*$ ou mieux : un intervalle $[\![0, m]\!]$ où m est le nombre maximum de classes de même niveau dans le lycée.

Schéma relationnel

Définition

Soit $\mathcal{A}$ un ensemble d'attributs et dom une application qui associe un domaine à chaque attribut.
Un *Schéma relationnel* est un tuple $S = (A_1, A_2, \ldots A_n) \in \mathcal{A}^n$ où les A_i sont distincts deux à deux (mais peuvent avoir les mêmes domaines).

Schéma relationnel

Définition

Soit $\mathcal{A}$ un ensemble d'attributs et dom une application qui associe un domaine à chaque attribut.
Un *Schéma relationnel* est un tuple $S = (A_1, A_2, \ldots A_n) \in \mathcal{A}^n$ où les A_i sont distincts deux à deux (mais peuvent avoir les mêmes domaines).

Remarque

- Généralement, on écrit le schéma relationnel sous forme de tuples de couples **(attribut, domaine)** comme
  $S = ((A_1, \mathrm{dom}(A_1)), \ldots, (A_n, \mathrm{dom}(A_n)))$
- Le plus souvent, on ajoute au schéma des symboles indiquant les *clés primaires* et *clés étrangères* (voir sections dédiées).

Schéma relationnel

Définition

Soit $\mathcal{A}$ un ensemble d'attributs et dom une application qui associe un domaine à chaque attribut.
Un *Schéma relationnel* est un tuple $S = (A_1, A_2, \ldots A_n) \in \mathcal{A}^n$ où les A_i sont distincts deux à deux (mais peuvent avoir les mêmes domaines).

Exemple

Schéma des classes du lycée :

$$S = ((\textbf{filiere}, \{\text{MPSI,PCSI,...}\}), (\textbf{numéro}, \mathbb{N}^*))$$

Schéma relationnel

Définition

Soit $\mathcal{A}$ un ensemble d'attributs et dom une application qui associe un domaine à chaque attribut.
Un *Schéma relationnel* est un tuple $S = (A_1, A_2, \ldots A_n) \in \mathcal{A}^n$ où les A_i sont distincts deux à deux (mais peuvent avoir les mêmes domaines).

Notation

- On écrit $B \in S$ si $B \subset \{A_1, \ldots, A_n\}$.
- Si $X = \{B_1, \ldots, B_m\}$ est un ensemble d'attributs (distincts), on écrit $X \subset S$ si tous les B_i sont dans $\{A_1, \ldots, A_n\}$.
- On s'autorise aussi des notations de la forme :

$$(\textbf{nom,ville}) \subset (\textbf{téléphone,nom,ville,classe})$$

Table

Définition

On appelle *relation* ou *table* associée à un schéma relationnel $(A_1, A_2, \ldots, A_n)$ tout ensemble fini de tuples de $\mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \ldots \mathrm{dom}(A_n)$.

Notation

- Les relations sont souvent notées sous la forme $R(S)$ (pour indiquer que R est associé au schéma S).
- Explication : dans la colonne i d'une table de schéma S, les valeurs sont obligatoirement dans le domaine $\mathrm{dom}(A_i)$. C'est ce qu'on appelle une *contrainte d'intégrité* (ici, on parle d'*intégrité de domaine*).

Table

Définition

On appelle *relation* ou *table* associée à un schéma relationnel $(A_1, A_2, \ldots, A_n)$ tout ensemble fini de tuples de $\mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \ldots \mathrm{dom}(A_n)$.

Exemple

Si la table **classe** est finie on peut la représenter par un tableau :

classe(filière,numéro)=

Filière	Numéro
MPSI	1
PC	3
PCSI	2
PCSI	1

Table

Définition

On appelle *relation* ou *table* associée à un schéma relationnel $(A_1, A_2, \ldots, A_n)$ tout ensemble fini de tuples de $\mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \ldots \mathrm{dom}(A_n)$.

Exemple

L'ordre des attributs et des tuples n'a pas d'importance. On a aussi :

classe(filière,numéro)=

Numéro	Filière
3	PC
1	PCSI
1	MPSI
2	PCSI

Représentation des schémas relationnels

Notation

Nom du schéma	
Attribut 1	`type 1`
Attribut 2	`type 2`

Remarque

Deux relations distinctes peuvent avoir le même schéma.

Représentation des schémas relationnels

Notation

Nom du schéma	
Attribut 1	`type 1`
Attribut 2	`type 2`

Exemple

Le shéma

élève	
Nom	`string`
Année de naissance	`int`

possède les instances

Nom	Année de naissance
Hoareau	1996
Grondin	1995

Nom	Année de naissance
Nativel	1998
Hoareau	1996
Grondin	1997

Multi-ensemble

Un *multi-ensemble* est une sorte d'ensemble dans lequel un même élément peut apparaître plusieurs fois comme dans $\{1, 2, 3, 2\}$.

Remarque

- Notion à mi-chemin des ensembles et des listes.

Multi-ensemble

Un *multi-ensemble* est une sorte d'ensemble dans lequel un même élément peut apparaître plusieurs fois comme dans $\{1, 2, 3, 2\}$.

Remarque

- Notion à mi-chemin des ensembles et des listes.
- On peut voir les multi-ensembles comme des listes quotientées par les permutations, *i.e.* des listes commutatives.

Multi-ensemble

Un *multi-ensemble* est une sorte d'ensemble dans lequel un même élément peut apparaître plusieurs fois comme dans $\{1, 2, 3, 2\}$.

Remarque

- Notion à mi-chemin des ensembles et des listes.
- On peut voir les multi-ensembles comme des listes quotientées par les permutations, *i.e.* des listes commutatives.
- Le multi-ensemble $\{1, 2, 2\}$ est égal à $\{2, 1, 2\}$.

Multi-ensemble

Un *multi-ensemble* est une sorte d'ensemble dans lequel un même élément peut apparaître plusieurs fois comme dans $\{1, 2, 3, 2\}$.

Remarque

- Notion à mi-chemin des ensembles et des listes.
- On peut voir les multi-ensembles comme des listes quotientées par les permutations, *i.e.* des listes commutatives.
- Le multi-ensemble $\{1, 2, 2\}$ est égal à $\{2, 1, 2\}$.
- Les relations du modèle relationnel sont en fait des multi-ensembles.

Multi-ensemble

Un *multi-ensemble* est une sorte d'ensemble dans lequel un même élément peut apparaître plusieurs fois comme dans $\{1, 2, 3, 2\}$.

Remarque

- Notion à mi-chemin des ensembles et des listes.
- On peut voir les multi-ensembles comme des listes quotientées par les permutations, *i.e.* des listes commutatives.
- Le multi-ensemble $\{1, 2, 2\}$ est égal à $\{2, 1, 2\}$.
- Les relations du modèle relationnel sont en fait des multi-ensembles.

Notation objet

Notation

Soit $R(S)$ une relation, $e \in R(S)$ un enregistrement et $A \in S$. On note $e.A$ la composante du tuple e associée à l'attribut A.
Si $K \subset S$, on note $e.K$ le sous-tuple de e constitué des composantes associées aux éléments de K. Il s'agit de la projection de e sur les attributs de K.

Notation objet

Notation

Soit $R(S)$ une relation, $e \in R(S)$ un enregistrement et $A \in S$. On note $e.A$ la composante du tuple e associée à l'attribut A.
Si $K \subset S$, on note $e.K$ le sous-tuple de e constitué des composantes associées aux éléments de K. Il s'agit de la projection de e sur les attributs de K.

Exemple

classe(**filière**,**numéro**,**Salle**)=

Filière	Numéro	Salle
MPSI	1	B.10
MPSI	2	C.34
PCSI	2	B.1

Si $e = (\text{PCSI}, 2, B.1)$, alors $e.\text{Numéro} = 2$ et $e.(\text{Numéro}, \text{Salle}) = (2, B.1)$.
On dit que $e.A$ est *la projection de e sur l'attribut A*. $e.(A_1, \ldots, A_n)$ est la *projection de e sur* $A_1 \times \cdots \times A_n$.

Clé unique

Définition

Soit $R(S)$ une relation de schéma S. On dit que $K \subset S$ est une *clé unique pour* R si et seulement si

$$\forall (t_1, t_2) \in R^2,\ t_1.K = t_2.K \iff t_1 = t_2.$$

Remarque

- La connaissance des attributs dans K suffit à distinguer deux éléments.

Clé unique

Définition

Soit $R(S)$ une relation de schéma S. On dit que $K \subset S$ est une *clé unique pour* R si et seulement si

$$\forall (t_1, t_2) \in R^2,\ t_1.K = t_2.K \iff t_1 = t_2.$$

Remarque

- La connaissance des attributs dans K suffit à distinguer deux éléments.
- K est une clé unique si et seulement si la projection sur K est injective.

Clé unique

Définition

Soit $R(S)$ une relation de schéma S. On dit que $K \subset S$ est une *clé unique pour* R si et seulement si

$$\forall (t_1, t_2) \in R^2,\ t_1.K = t_2.K \iff t_1 = t_2.$$

Remarque

- La connaissance des attributs dans K suffit à distinguer deux éléments.
- K est une clé unique si et seulement si la projection sur K est injective.
- Lorsqu'il y a une clé unique, la table ne contient pas de doublon de lignes.

Clé unique

Définition

Soit $R(S)$ une relation de schéma S. On dit que $K \subset S$ est une *clé unique pour* R si et seulement si

$$\forall (t_1, t_2) \in R^2,\ t_1.K = t_2.K \iff t_1 = t_2.$$

Remarque

- La connaissance des attributs dans K suffit à distinguer deux éléments.
- K est une clé unique si et seulement si la projection sur K est injective.
- Lorsqu'il y a une clé unique, la table ne contient pas de doublon de lignes.
- Souvent, on impose que K soit de cardinal minimum. C'est de bon sens : nous nous en tenons à cette pratique.

Clé unique

Définition

Soit $R(S)$ une relation de schéma S. On dit que $K \subset S$ est une *clé unique pour* R si et seulement si

$$\forall (t_1, t_2) \in R^2,\ t_1.K = t_2.K \iff t_1 = t_2.$$

Exemple

élève(**Nom**,**Prénom**,**Année de naissance**)=

Nom	Prénom	Année de naissance
Hoareau	Patrice	1996
Hoareau	Patrice	1995
Dupont	Marie	1997
Grondin	Patrice	1996

Dans cette relation (**Nom**,**Prénom**) n'est pas une clé unique, ni (**Prénom**,**Année**) mais (**Nom**,**Année**) est une clé unique.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.
- Le terme *clé unique* est trompeur : il peut y en avoir plusieurs !
  Exemple : dans la table **Etudiant(id,nom, prénom,num. de sécu)**
  il y a deux clés uniques possibles :

  Les clés sont choisies par le développeur au moment d ela conception.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.
- Le terme *clé unique* est trompeur : il peut y en avoir plusieurs !
  Exemple : dans la table **Etudiant(id,nom, prénom,num. de sécu)** il y a deux clés uniques possibles :
  - **id** (le numéro d'étudiant).

  Les clés sont choisies par le développeur au moment d ela conception.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.
- Le terme *clé unique* est trompeur : il peut y en avoir plusieurs !
  Exemple : dans la table **Etudiant(id,nom, prénom,num. de sécu)** il y a deux clés uniques possibles :
  - **id** (le numéro d'étudiant).
  - **num. de sécu**

  Les clés sont choisies par le développeur au moment d ela conception.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.
- Le terme *clé unique* est trompeur : il peut y en avoir plusieurs !
  Exemple : dans la table **Etudiant(id,nom, prénom,num. de sécu)** il y a deux clés uniques possibles :
  - **id** (le numéro d'étudiant).
  - **num. de sécu**

  Les clés sont choisies par le développeur au moment d ela conception.
- Une clé unique peut porter sur plusieurs attributs : il peut très bien ne pas y avoir de clé à un seul élément.

Clé unique

Soit $R(S)$ une relation de schéma S.

- Souvent, on cherche à limiter la clé unique à un seul attribut.
- Le terme *clé unique* est trompeur : il peut y en avoir plusieurs ! Exemple : dans la table **Etudiant(id,nom, prénom,num. de sécu)** il y a deux clés uniques possibles :
  - **id** (le numéro d'étudiant).
  - **num. de sécu**

  Les clés sont choisies par le développeur au moment d ela conception.
- Une clé unique peut porter sur plusieurs attributs : il peut très bien ne pas y avoir de clé à un seul élément.
- Et d'ailleurs, il est possible qu'il n'y ait pas de clé unique (si la table possède des doublons de lignes). Mais on décourage d'utiliser de telles tables.

Clé primaire

MySQL fait la distinction entre les notions de *clé unique* et *clé primaire*.

Définition

Une *clé primaire* est une clé unique particulière associée à un *index*.

Remarque

- On peut voir l'index comme une table des matières facilitant un accès rapide aux enregistrement d'une table ayant une clé primaire.

Clé primaire

MySQL fait la distinction entre les notions de *clé unique* et *clé primaire*.

Définition

Une *clé primaire* est une clé unique particulière associée à un *index*.

Remarque

- On peut voir l'index comme une table des matières facilitant un accès rapide aux enregistrement d'une table ayant une clé primaire.
- En particulier, la complexité des *jointures* est grandement diminuée par l'usage d'une clé primaire ; les valeurs possibles étant triées dans l'index.

Clé primaire

MySQL fait la distinction entre les notions de *clé unique* et *clé primaire*.

Définition

Une *clé primaire* est une clé unique particulière associée à un *index*.

Remarque

- On peut voir l'index comme une table des matières facilitant un accès rapide aux enregistrement d'une table ayant une clé primaire.
- En particulier, la complexité des *jointures* est grandement diminuée par l'usage d'une clé primaire ; les valeurs possibles étant triées dans l'index.
- Il peut y avoir plusieurs clés unique par table mais une seule clé primaire.

Clé primaire

MySQL fait la distinction entre les notions de *clé unique* et *clé primaire*.

Définition

Une *clé primaire* est une clé unique particulière associée à un *index*.

Remarque

- On peut voir l'index comme une table des matières facilitant un accès rapide aux enregistrement d'une table ayant une clé primaire.
- En particulier, la complexité des *jointures* est grandement diminuée par l'usage d'une clé primaire ; les valeurs possibles étant triées dans l'index.
- Il peut y avoir plusieurs clés unique par table mais une seule clé primaire.
- Une clé unique peut prendre la valeur **NULL** (case vide, équivalent Python de **None**) pas la clé primaire.

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$
- On commence par trier (une fois pour toute) les enregistrements de T selon la clé K.

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$
- On commence par trier (une fois pour toute) les enregistrements de T selon la clé K.
- L'index est alors une structure de données permettant de dire que, pour les éléments de la colonne K :

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$
- On commence par trier (une fois pour toute) les enregistrements de T selon la clé K.
- L'index est alors une structure de données permettant de dire que, pour les éléments de la colonne K :
  - ceux qui commencent par un **'A'** se trouvent au début de T ;

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$
- On commence par trier (une fois pour toute) les enregistrements de T selon la clé K.
- L'index est alors une structure de données permettant de dire que, pour les éléments de la colonne K :
  - ceux qui commencent par un **'A'** se trouvent au début de T ;
  - ceux commençant par un **'B'** se trouvent à partir du 20ème enregistrement, ..., ceux commençant par 'V' à partir de 250ème, **'W'** à partir du 300ème enregistrement etc.

Notion d'index

Notion hors programme.

- Soit K la clé primaire de la table T. Pour simplifier supposons $|K| = 1$
- On commence par trier (une fois pour toute) les enregistrements de T selon la clé K.
- L'index est alors une structure de données permettant de dire que, pour les éléments de la colonne K :
  - ceux qui commencent par un **'A'** se trouvent au début de T ;
  - ceux commençant par un **'B'** se trouvent à partir du 20ème enregistrement, ..., ceux commençant par 'V' à partir de 250ème, **'W'** à partir du 300ème enregistrement etc.
- Si on cherche *Voiture*, on se place donc entre les positions 250 et 249 puis on procède par dichotomie.

Clé unique VS clé primaire

- Conformément au programme nous ferons désormais la confusion : nous n'emploirons plus que l'expression « clé primaire » sans nous soucier de la présence d'un index ou non.

Clé unique VS clé primaire

- Conformément au programme nous ferons désormais la confusion : nous n'emploirons plus que l'expression « clé primaire » sans nous soucier de la présence d'un index ou non.
- Une conséquence est que nous avons au plus une clé primaire par table.

Clé unique VS clé primaire

- Conformément au programme nous ferons désormais la confusion : nous n'emploirons plus que l'expression « clé primaire » sans nous soucier de la présence d'un index ou non.
- Une conséquence est que nous avons au plus une clé primaire par table.
- Une autre est que les cases des colonnes définissant la clé primaire ne sont jamais vide (pas de valeur **NULL**).

Clé primaire

- On indique par un symbole dans le schéma qu'une clé est unique/primaire.

Clé primaire

- On indique par un symbole dans le schéma qu'une clé est unique/primaire.
- Nous signalons les clés uniques en les soulignant. Sous PHPMYADMIN, les clés primaires sont représentées par des clés jaunes, les clés uniques par une clé grise.

élève	
Nom	`string`
<u>Numéro SS</u>	`int`

Clé primaire

- On indique par un symbole dans le schéma qu'une clé est unique/primaire.
- Nous signalons les clés uniques en les soulignant. Sous PHPMYADMIN, les clés primaires sont représentées par des clés jaunes, les clés uniques par une clé grise.

élève	
Nom	`string`
<u>Numéro SS</u>	`int`

- Un mot clé **PRIMARY** indique, au moment de la création de la table dans la plupart des SGBD, qu'une clé est primaire.

Clé primaire

- On indique par un symbole dans le schéma qu'une clé est unique/primaire.
- Nous signalons les clés uniques en les soulignant. Sous PHPMYADMIN, les clés primaires sont représentées par des clés jaunes, les clés uniques par une clé grise.

élève	
Nom	`string`
<u>Numéro SS</u>	`int`

- Un mot clé **PRIMARY** indique, au moment de la création de la table dans la plupart des SGBD, qu'une clé est primaire.
- Si un tuple déjà défini possède une valeur **v** pour la clé primaire de la table **T**, alors le SGBD devrait empêcher l'ajout de tout nouveau tuple à **T** possédant la valeur **v** pour la clé.

Deux schémas

Soit une BDD modélisant une bibliothèque simplifiée avec deux tables

dont les schémas sont :

livre	
titre	`string`
auteur	`string`
année de publication	`int`

et

emprunteur	
Nom	`string`
Livre emprunté	`string`

Deux tables

- Une table **bibliothèque** instanciant **livre** :

titre	auteur	Publication
Harry Potter	J.K. Rowling	1997
Pensées	Pascal	1670
Marseille coquin	Anonyme	2016

Deux tables

- Une table **bibliothèque** instanciant **livre** :

titre	auteur	Publication
Harry Potter	J.K. Rowling	1997
Pensées	Pascal	1670
Marseille coquin	Anonyme	2016

- Une table **Clients** instanciant **emprunteur**

Nom	Livre emprunté
Hoareau	Harry Potter
Grondin	Marseille coquin
Dupont	Maths MP

Deux tables

- Une table **bibliothèque** instanciant **livre** :

titre	auteur	Publication
Harry Potter	J.K. Rowling	1997
Pensées	Pascal	1670
Marseille coquin	Anonyme	2016

- Une table **Clients** instanciant **emprunteur**

Nom	Livre emprunté
Hoareau	Harry Potter
Grondin	Marseille coquin
Dupont	Maths MP

- On en conclut que Hoareau a emprunté ≪ Harry Potter ≫ et que Grondin est un petit coquin !

Deux tables

- Une table **bibliothèque** instanciant **livre** :

titre	auteur	Publication
Harry Potter	J.K. Rowling	1997
Pensées	Pascal	1670
Marseille coquin	Anonyme	2016

- Une table **Clients** instanciant **emprunteur**

Nom	Livre emprunté
Hoareau	Harry Potter
Grondin	Marseille coquin
Dupont	Maths MP

- On en conclut que Hoareau a emprunté ≪ Harry Potter ≫ et que Grondin est un petit coquin !
- Dupont emprunte un ouvrage qui n'existe pas dans la **bibliothèque**, ce qui concerne davantage l'administrateur de BDD que la vie privée de Grondin.

Clé étrangère

Définition

Une clé étrangère (représentée dans ce cours par un #) est un attribut qui est la clé primaire d'une autre relation. Elle permet d'établir le lien entre plusieurs relations. Elle met en évidence les dépendances fonctionnelles entre 2 tables.

Remarque

En SQL, on déclare une clé étrangère avec les mots clés **FOREIGN KEY**.

Clé étrangère

- On peut représenter les liens entre deux relations dans un diagramme par une flèche depuis l'attribut vers la clé primaire.

Clé étrangère

- On peut représenter les liens entre deux relations dans un diagramme par une flèche depuis l'attribut vers la clé primaire.
- Si on impose que le domaine de **livre emprunté** est constitué exactement des livres apparaissant dans la relation **bibliothèque**, le schéma de **emprunteur** devient

Shéma référençant

emprunteur	
Nom	`string`
Livre emprunté#	`titre`

$\rightarrow$

Schéma référencé

livre	
<u>titre</u>	`string`
auteur	`string`
année de publication	`int`

Clé étrangère

- On peut représenter les liens entre deux relations dans un diagramme par une flèche depuis l'attribut vers la clé primaire.
- Si on impose que le domaine de **livre emprunté** est constitué exactement des livres apparaissant dans la relation **bibliothèque**, le schéma de **emprunteur** devient

Shéma référençant

emprunteur	
Nom	`string`
Livre emprunté#	`titre`

$\rightarrow$

Schéma référencé

livre	
titre	`string`
auteur	`string`
année de publication	`int`

- Le tuple **(Dupont,Maths MP)** ne peut plus être un enregistrement de **Client** car **Maths MP** n'est pas dans la colonne ≪ titre ≫ de **Bibliothèque**.

La contrainte de clé étrangère est gérée par la plupart des SGBD : **Oracle, Microsoft SQL Server, PostgreSQL, SQLite**, etc.

Mode de communication

- Environnement *client-serveur* : mode de communication à travers un réseau entre plusieurs programmes ou logiciels

Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur.

Mode de communication

- Environnement *client-serveur* : mode de communication à travers un réseau entre plusieurs programmes ou logiciels
  1. le premier, le *client*, envoie des requêtes ;

Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur.

Mode de communication

- Environnement *client-serveur* : mode de communication à travers un réseau entre plusieurs programmes ou logiciels
  1. le premier, le *client*, envoie des requêtes ;
  2. l'autre ou les autres, les *serveurs*, attendent les requêtes des clients et y répondent.

  Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur.

Vocabulaire

- Serveurs : souvent des ordinateurs dédiés au logiciel serveur qu'ils abritent (ex : serveur Web, serveur de bases de données, d'impression ...). Ils sont dotés de capacités supérieures à celles des ordinateurs personnels en termes de puissance de calcul, d'entrées-sorties et de connexions réseau.

Vocabulaire

- Serveurs : souvent des ordinateurs dédiés au logiciel serveur qu'ils abritent (ex : serveur Web, serveur de bases de données, d'impression ...). Ils sont dotés de capacités supérieures à celles des ordinateurs personnels en termes de puissance de calcul, d'entrées-sorties et de connexions réseau.
- Clients : souvent des ordinateurs personnels ou des appareils individuels (téléphone, tablette), mais pas systématiquement.

Vocabulaire

- Serveurs : souvent des ordinateurs dédiés au logiciel serveur qu'ils abritent (ex : serveur Web, serveur de bases de données, d'impression ...). Ils sont dotés de capacités supérieures à celles des ordinateurs personnels en termes de puissance de calcul, d'entrées-sorties et de connexions réseau.
- Clients : souvent des ordinateurs personnels ou des appareils individuels (téléphone, tablette), mais pas systématiquement.
- Nombre de clients. Un serveur peut répondre aux requêtes d'un grand nombre de clients.

Exemples

Grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir :

- un serveur web publie des pages web demandées par des navigateurs web ;

Exemples

Grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir :

- un serveur web publie des pages web demandées par des navigateurs web ;
- un serveur de messagerie électronique envoie des mails à des clients de messagerie ;

Exemples

Grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir :

- un serveur web publie des pages web demandées par des navigateurs web ;
- un serveur de messagerie électronique envoie des mails à des clients de messagerie ;
- un serveur de fichiers permet de stocker et consulter des fichiers sur le réseau ;

Exemples

Grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir :

- un serveur web publie des pages web demandées par des navigateurs web ;
- un serveur de messagerie électronique envoie des mails à des clients de messagerie ;
- un serveur de fichiers permet de stocker et consulter des fichiers sur le réseau ;
- un serveur de données à communiquer des données stockées dans une base de données, etc...

Notion de port

- La notion de *port logiciel* permet, sur un ordinateur donné, de distinguer différents interlocuteurs. Ces interlocuteurs sont des programmes informatiques qui, selon les cas, écoutent ou émettent des informations sur ces ports. Un port est distingué par son numéro.

Notion de port

- La notion de *port logiciel* permet, sur un ordinateur donné, de distinguer différents interlocuteurs. Ces interlocuteurs sont des programmes informatiques qui, selon les cas, écoutent ou émettent des informations sur ces ports. Un port est distingué par son numéro.
- Image : PORT = PORTE donnant accès au système d'exploitation. Pour fonctionner, un programme ouvre des portes pour accéder aux services de l'OS. Quand on ferme le programme, la porte n'a plus besoin d'être ouverte.

Notion de port

- La notion de *port logiciel* permet, sur un ordinateur donné, de distinguer différents interlocuteurs. Ces interlocuteurs sont des programmes informatiques qui, selon les cas, écoutent ou émettent des informations sur ces ports. Un port est distingué par son numéro.
- Image : PORT = PORTE donnant accès au système d'exploitation. Pour fonctionner, un programme ouvre des portes pour accéder aux services de l'OS. Quand on ferme le programme, la porte n'a plus besoin d'être ouverte.
- Lorsqu'un logiciel client veut dialoguer avec un logiciel serveur (le *service*), il a besoin de connaître le port écouté par ce dernier. Par exemple port 80 pour un serveur web **HTTP** ; port 3306 serveur de bases de données **MySQL** ; port 8888 pour jupyter...

Notion de port

- La notion de *port logiciel* permet, sur un ordinateur donné, de distinguer différents interlocuteurs. Ces interlocuteurs sont des programmes informatiques qui, selon les cas, écoutent ou émettent des informations sur ces ports. Un port est distingué par son numéro.
- Image : PORT = PORTE donnant accès au système d'exploitation. Pour fonctionner, un programme ouvre des portes pour accéder aux services de l'OS. Quand on ferme le programme, la porte n'a plus besoin d'être ouverte.
- Lorsqu'un logiciel client veut dialoguer avec un logiciel serveur (le *service*), il a besoin de connaître le port écouté par ce dernier. Par exemple port 80 pour un serveur web **HTTP** ; port 3306 serveur de bases de données **MySQL** ; port 8888 pour jupyter...
- **cat /etc/services** pour avoir la liste des services bien connus. Ou **sudo netstat -antup — grep LISTEN** pour la liste des ports en écoute.

Caractéristiques d'un processus serveur

- Attend une connexion entrante sur un ou plusieurs *ports* réseaux.

Caractéristiques d'un processus serveur

- Attend une connexion entrante sur un ou plusieurs *ports* réseaux.
- à la connexion d'un client sur le port en écoute, ouvre un *socket local* (interface de comunication) avec le système d'exploitation ;

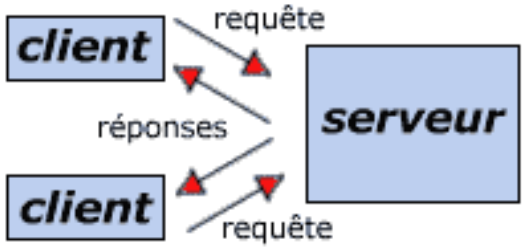
Caractéristiques d'un processus serveur

- Attend une connexion entrante sur un ou plusieurs *ports* réseaux.
- à la connexion d'un client sur le port en écoute, ouvre un *socket local* (interface de comunication) avec le système d'exploitation ;
- suite à la connexion, le processus serveur communique avec le client suivant le protocole prévu par la couche **application** du modèle OSI.

Caractéristiques d'un processus client

- établit la connexion au serveur grâce à son adresse IP et le port, qui désigne un service particulier du serveur. Un socket est créé côté client.
- lorsque la connexion est acceptée par le serveur, les deux côtés communiquent via les sockets comme le prévoit la couche **application** du modèle OSI.

FIGURE – Architecture client-serveur



La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.

La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.
- Le client insère des pièces dans l'automate, sélectionne sa boisson et attend que la machine remplisse son gobelet.

La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.
- Le client insère des pièces dans l'automate, sélectionne sa boisson et attend que la machine remplisse son gobelet.
- Le serveur est la machine à café. Le couple (client, automate) est une architecture client-serveur.

La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.
- Le client insère des pièces dans l'automate, sélectionne sa boisson et attend que la machine remplisse son gobelet.
- Le serveur est la machine à café. Le couple (client, automate) est une architecture client-serveur.
- Le client accède directement à la ressource.

La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.
- Le client insère des pièces dans l'automate, sélectionne sa boisson et attend que la machine remplisse son gobelet.
- Le serveur est la machine à café. Le couple (client, automate) est une architecture client-serveur.
- Le client accède directement à la ressource.
  - Si le serveur est en panne, c'est au client d'en trouver un autre (pb de maintenance)

La machine à café

- Dans une cafétéria, les cafés sont délivrés par un automate.
- Le client insère des pièces dans l'automate, sélectionne sa boisson et attend que la machine remplisse son gobelet.
- Le serveur est la machine à café. Le couple (client, automate) est une architecture client-serveur.
- Le client accède directement à la ressource.
  - Si le serveur est en panne, c'est au client d'en trouver un autre (pb de maintenance)
  - Si le client est malhonnête, il peut tenter d'insérer de fausses pièces (il ne court aucun risque).

Une brasserie

- Dans une brasserie, les garçons de café ont accès directement au percolateur.

Une brasserie

- Dans une brasserie, les garçons de café ont accès directement au percolateur.
- Le client (couche présentation) s'assied à une table, attend que le garçon (couche métier) vienne prendre sa commande.

Une brasserie

- Dans une brasserie, les garçons de café ont accès directement au percolateur.
- Le client (couche présentation) s'assied à une table, attend que le garçon (couche métier) vienne prendre sa commande.
- Une fois que le garçon a pris la commande, il accède au percolateur (couche accès aux données) derrière le comptoir, prépare l'expresso et le ramène au client.

Une brasserie

- Dans une brasserie, les garçons de café ont accès directement au percolateur.
- Le client (couche présentation) s'assied à une table, attend que le garçon (couche métier) vienne prendre sa commande.
- Une fois que le garçon a pris la commande, il accède au percolateur (couche accès aux données) derrière le comptoir, prépare l'expresso et le ramène au client.
- Le triplet (client,garçon,percolateur) est une architecture trois-tiers (ou trois couches)

Une brasserie

Le client accède n'accède plus directement à la ressource.

- Si le percolateur est en panne, c'est au garçon et pas au client d'en trouver un autre (maintenance facilitée, on peut imaginer un percolateur d'appoint en attendant la réparation du principal)

Une brasserie

Le client accède n'accède plus directement à la ressource.

- Si le percolateur est en panne, c'est au garçon et pas au client d'en trouver un autre (maintenance facilitée, on peut imaginer un percolateur d'appoint en attendant la réparation du principal)
- Si le client est malhonnête, il lui est plus difficile d'accéder au percolateur pour se servir gratuitement (sécurité renforcée).

Une brasserie

Le client accède n'accède plus directement à la ressource.

- Si le percolateur est en panne, c'est au garçon et pas au client d'en trouver un autre (maintenance facilitée, on peut imaginer un percolateur d'appoint en attendant la réparation du principal)
- Si le client est malhonnête, il lui est plus difficile d'accéder au percolateur pour se servir gratuitement (sécurité renforcée).
- Bien sûr, le client pourrait attendre que le garçon prenne la commande d'une autre personne pour accéder en cachette au percolateur. Il suffirait alors de mettre quelqu'un en permanence derrière le bar (le patron) et ce problème serait résolu (mais on passerait en architecture 4 couches).

Architecture trois-tiers

Définition

Le mot *tier* signifie étage ou niveau en anglais. On dit aussi *couche*.

- Une *application* est composée de 3 couches indépendantes :

Architecture trois-tiers

Définition

Le mot *tier* signifie étage ou niveau en anglais. On dit aussi *couche*.

- Une *application* est composée de 3 couches indépendantes :
  1. couche *présentation*,

Architecture trois-tiers

Définition

Le mot *tier* signifie étage ou niveau en anglais. On dit aussi *couche*.

- Une *application* est composée de 3 couches indépendantes :
  1. couche *présentation*,
  2. couche *traitements* (on dit aussi *métier* ou *application*)

Architecture trois-tiers

Définition

Le mot *tier* signifie étage ou niveau en anglais. On dit aussi *couche*.

- Une *application* est composée de 3 couches indépendantes :
  1. couche *présentation*,
  2. couche *traitements* (on dit aussi *métier* ou *application*)
  3. couche d'*accès aux données*.

Architecture trois-tiers

Définition

Le mot *tier* signifie étage ou niveau en anglais. On dit aussi *couche*.

- Une *application* est composée de 3 couches indépendantes :
  1. couche *présentation*,
  2. couche *traitements* (on dit aussi *métier* ou *application*)
  3. couche d'*accès aux données*.
- Ces 3 couches communiqueront entre elles à l'aide de fonctions spécifiques (des API : *Application Programming Interface* ou *Interfaces de programmation*).

SGBD et architecture trois-tiers

On répartit les rôles entre

- Un serveur contenant la base données (non accessible par les clients)

SGBD et architecture trois-tiers

On répartit les rôles entre

- Un serveur contenant la base données (non accessible par les clients)
- Un *système de gestion de base données* : une interface souvent graphique entre les clients et la base.

SGBD et architecture trois-tiers

On répartit les rôles entre

- Un serveur contenant la base données (non accessible par les clients)
- Un *système de gestion de base données* : une interface souvent graphique entre les clients et la base.
  - Elle transmet la demande (requête) du client au serveur de données.

SGBD et architecture trois-tiers

On répartit les rôles entre

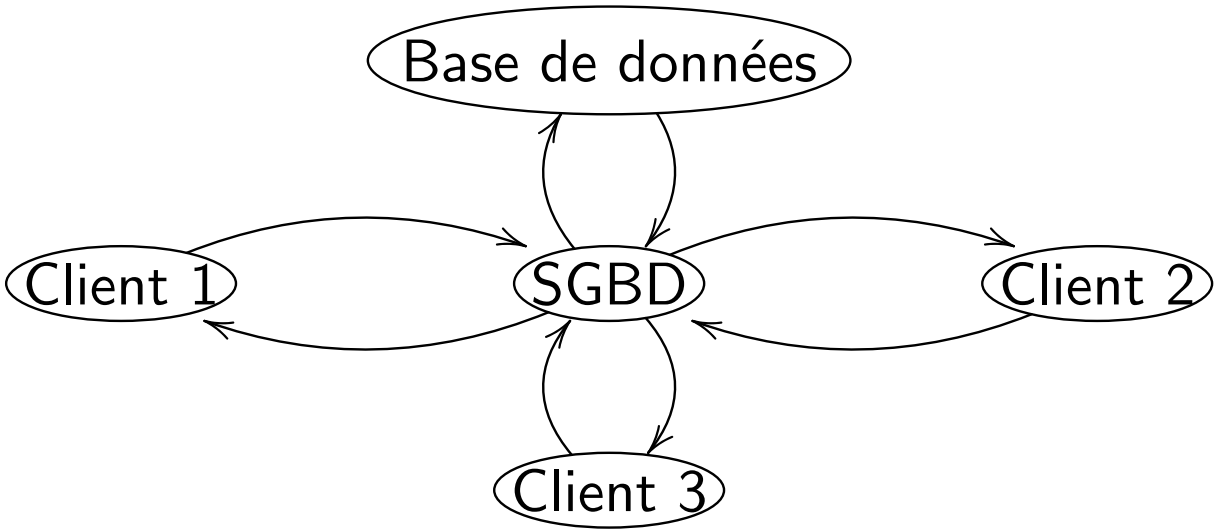
- Un serveur contenant la base données (non accessible par les clients)
- Un *système de gestion de base données* : une interface souvent graphique entre les clients et la base.
  - Elle transmet la demande (requête) du client au serveur de données.
  - Elle récupère la réponse du serveur de données et la transmet au client.

SGBD et architecture trois-tiers

On répartit les rôles entre

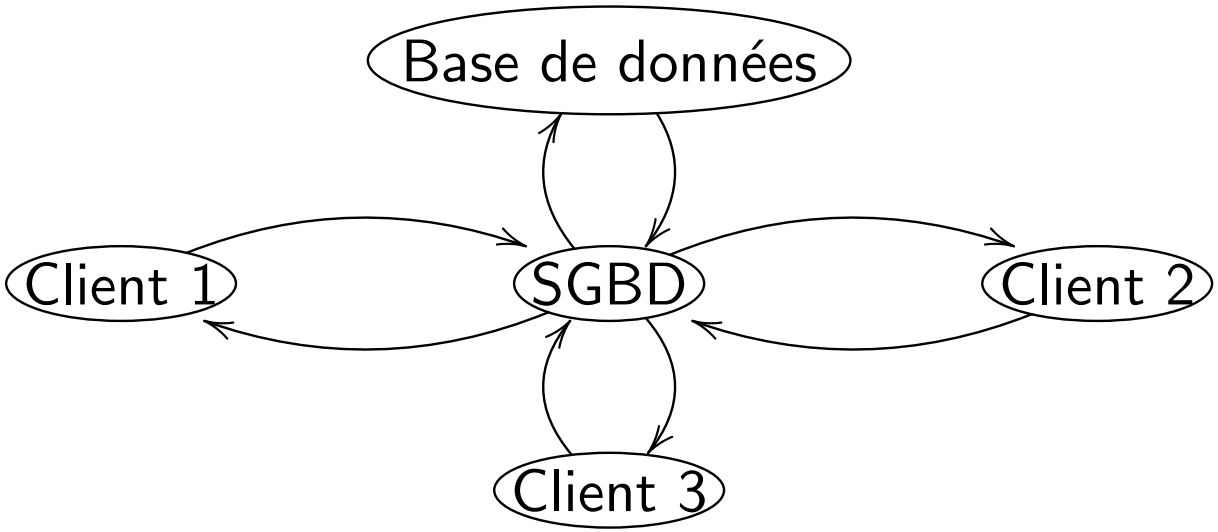
- Un serveur contenant la base données (non accessible par les clients)
- Un *système de gestion de base données* : une interface souvent graphique entre les clients et la base.
  - Elle transmet la demande (requête) du client au serveur de données.
  - Elle récupère la réponse du serveur de données et la transmet au client.
- Le point important : le client ne communique jamais directement avec le serveur de données.

SGBD et architecture trois-tiers



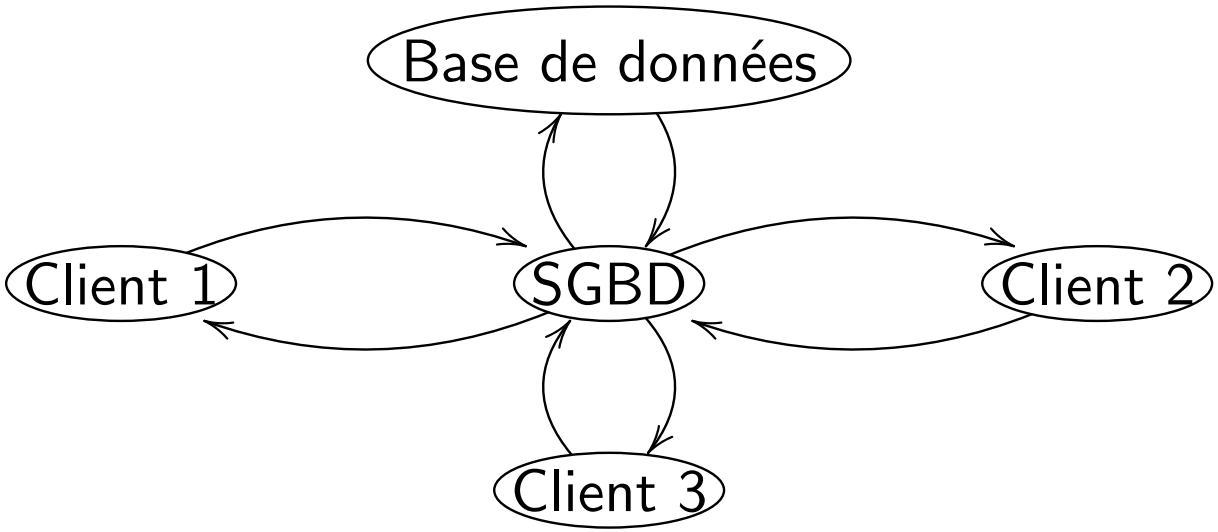
- Seul le SGBD peut accéder aux données et les modifier.

SGBD et architecture trois-tiers



- Seul le SGBD peut accéder aux données et les modifier.
- Le client n'a pas besoin de connaître le SQL : souvent une interface graphique avec des cases à cliquer lui évite de le faire.

SGBD et architecture trois-tiers



- Seul le SGBD peut accéder aux données et les modifier.
- Le client n'a pas besoin de connaître le SQL : souvent une interface graphique avec des cases à cliquer lui évite de le faire.
- Le client n'a pas besoin d'installer de logiciel : un navigateur Web lui suffit