Le modèle relationnel

Notions de base

- Table = relation
- Noms des colonnes = *attributs*
- Ligne = *n-uplet*

`Bieres`	nom	fabriquant
	1664	Kronenbourg
	mort subite	Jeanlain
	...	...

Notions de base

- *Schéma d'une relation* = nom(attributs).
  exemple : `Bieres(nom, fabriquant)`
  - L'ordre des attributs est indifférent.
  - *Instance d'une relation* c'est l'ensemble courant de lignes pour un schéma de relation donné.

- *Schéma de la base* = l'ensemble des schémas de relations.

Clefs

Un attribut ou un ensemble d'attributs K est une clef pour une relation R si nous nous attendons à ce qu'aucune instance de R soit telle que deux n-uplets différents aient les mêmes valeurs pour les attributs de K.

- On indique une clef en soulignant les attributs qui la composent.

- Exemple : si `nom` est une clef pour `Bieres`

  Bieres(<u>nom</u>, fabriquant)

Pourquoi des relations ?

- Modèle très simple.

- *Souvent* constitue une bonne adéquation avec la façon que nous avons de percevoir les données.

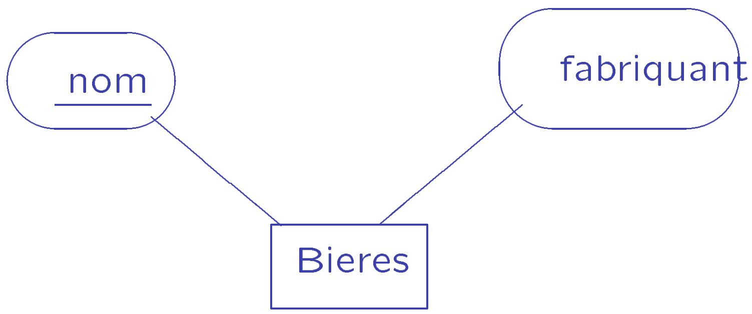
- C'est le modèle théorique sous-jacent à SQL le plus important langage pour bases de données à ce jour.

Relationnel : conception

- Les relations sont plus proches des structures de stockage que les concepts d'Entité-Associations (E/R).
  - Donc le passage de modèles E/R au relationnel requiert un effort intellectuel supplémentaire.

Cas le plus simple : Ens d'entités → Relation

Les attributs de l'ensemble d'entités deviennent des attributs de la relation.



Devient :

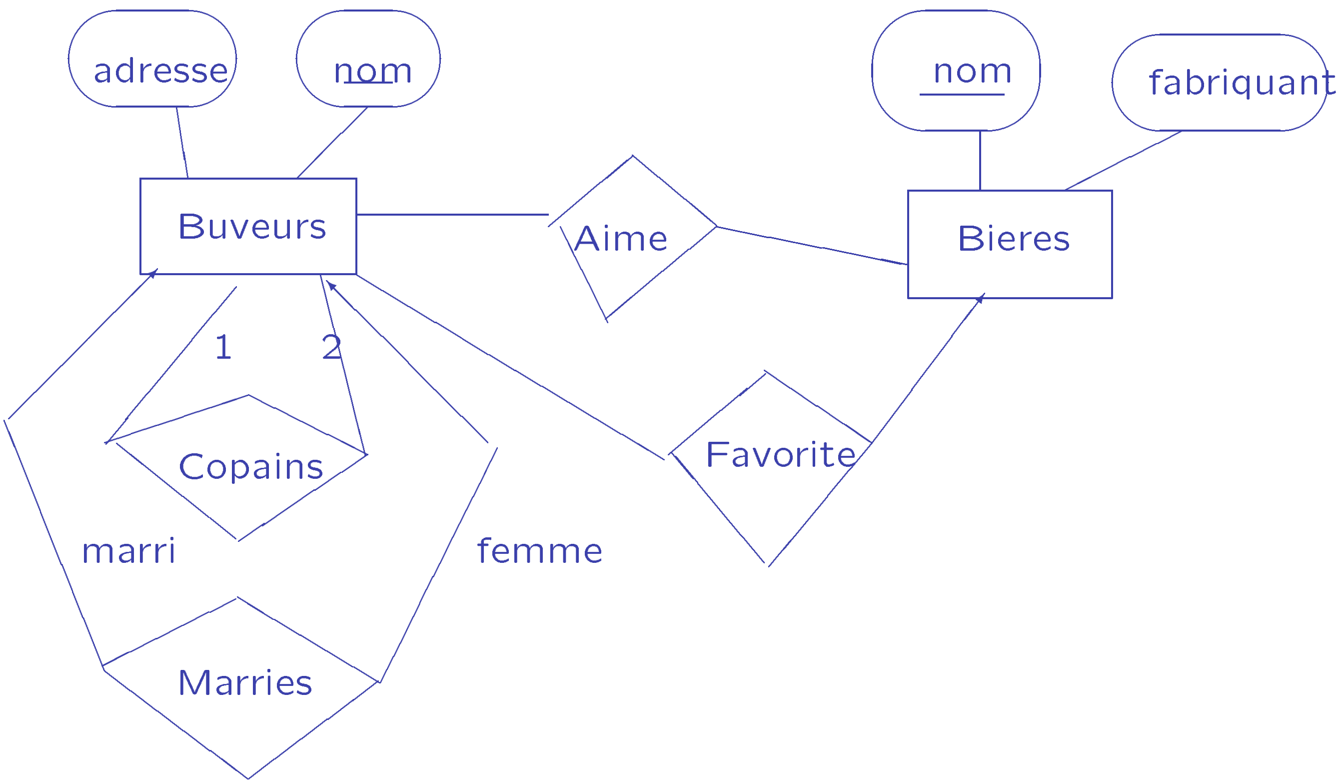
`Bieres(nom, fabriquant)`

E/R Associations → Relations

A chaque association correspond une relation. Cette dernière a pour clef les attributs clefs de chaque ensemble d'entités participant à l'association.

- Si l'association est N-1, la clef se compose uniquement des attributs clefs de la partie "N".
- Si l'association est 1-1, on peut choisir arbitrairement comme clef les attributs clef de l'un ou l'autre des ensembles d'entités reliés.
- Le renommage d'attributs est autorisé et même essentiel en présence de rôles.

Exemple



Aime(<u>buveur</u>, <u>biere</u>)

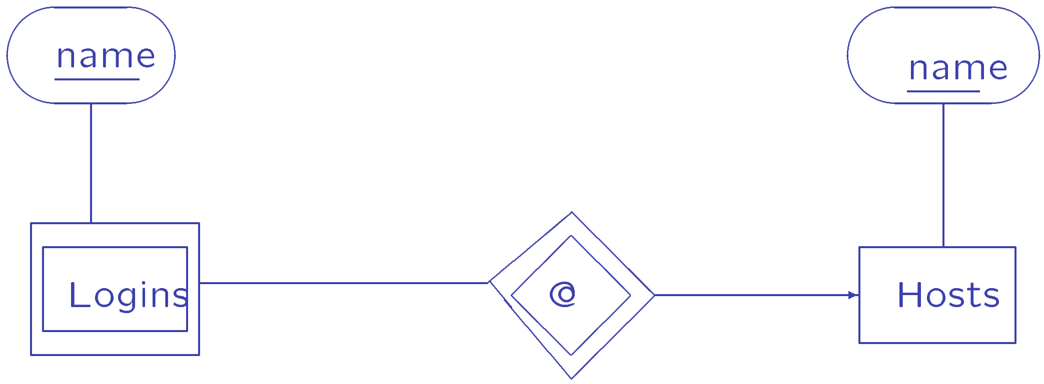
Favorite(<u>buveur</u>, biere)

Marries(mari, <u>femme</u>)

Copains(<u>nom1</u>, <u>nom2</u>)

Pour l'association Marries, nous pouvons choisir mari ou femme comme clef.

Ensembles d'entités faibles → Relations



Hosts(<u>hostname</u>)

Logins(<u>loginname</u>, <u>hostname</u>)

At(<u>loginname</u>, <u>hostname</u>, hostname2)

- Dans `At`, `hostname` et `hostname2` doivent être le même host ⇒ en détruire un.

- Donc `Logins` et `At` deviennent la même relation ⇒ en détruire une.

Ensembles d'entités faibles → Relations

- Dans ce cas, le schéma de `Hosts` est un sous-ensemble du schéma de `Logins`. Faut-il détruire `Hosts`?

- Règle générale : détruire la relation qui provient d'une association N-1 comportant un ensemble d'entités faibles.

Sous-classes → Relations

Deux approches :

1. Créer une relation pour chaque ensemble d'entités.

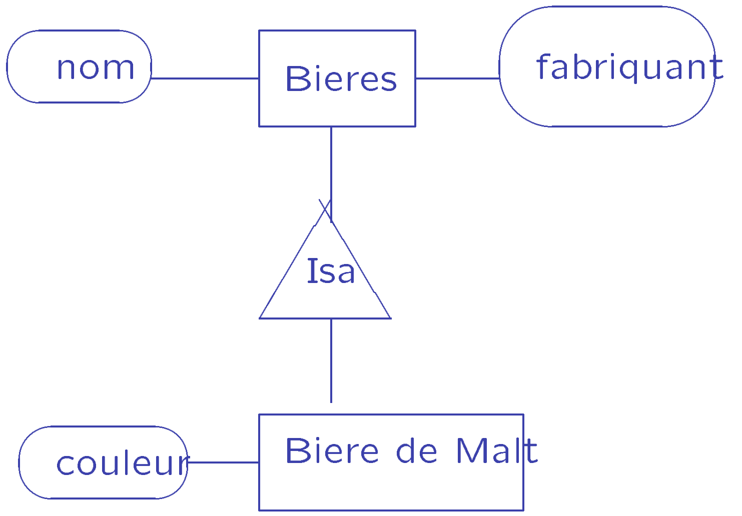
   - Une entité est représentée dans la relation pour chaque sous-classe à laquelle elle appartient.

   - La relation a seulement les attributs correspondant à l'ensemble d'entité + clef.

2. Utiliser des valeurs nulles. Créer une relation pour la classe "racine" avec tous les attributs des sous-classes.

   - Mettre `NULL` dans les attributs n'existant pas pour une entité.

Exemple



Exemple

1. sans valeurs nulles

Bieres	nom	fabriquant
	Budweiser	A.B.
	Pelforth	les brasseurs

Bieres de Malt	nom	couleur
	Pelforth	brune

2. avec valeurs nulles

Bieres	nom	fabriquant	couleur
	Budweiser	A.B.	NULL
	Pelforth	les brasseurs	brune

Dépendences Fonctionnelles

Notée :

$$X \rightarrow A$$

Assertion sur une relation R qui établi que si deux n-uplets ont les mêmes valeurs sur les attributs de X alors ils doivent avoir les mêmes valeurs sur l'attribut A.

- Important car représentent des contraintes sur les données qui sont stockées dans une relation. (meilleure adéquation entre le monde réel et sa modélisation.
  - Contrôle des données au niveau du schéma.
- Outil mathématique (formel) pour expliquer le processus de "normalisation" qui est vital pour re-concevoir des schémas lorsque la conception initiale a des défauts.

Exemple

`Buveurs (nom, adresse, bieresaimees, fabriquant, bierefav)`

nom	adresse	bieresaimees	fabriquant	bierefav
marcel	paris	1664	kro	mort-subite
marcel	paris	mort-subite	jeanlain	mort-subite
jules	marseille	1664	kro	1664

- DF raisonnables :

  1. `nom` $\longrightarrow$ `adresse`

  2. `nom` $\longrightarrow$ `bierefav`

  3. `bieresaimees` $\longrightarrow$ `fabriquant`

Exemple

- Nota bene : Impliquent que `nom` et `bieresaimees` soient des clefs, mais les DF donnent plus de détails que le simple fait de déclarer une clef.

Dépendences Fonctionnelles

- Les Clefs (en général) déterminent fonctionnellement tous les autres attributs du schéma. Pour notre exemple :

$$\texttt{nom, bieresaimees} \rightarrow \texttt{adresse, bierefav, fabriquant}$$

- Notation raccourcie : fusionner les DF ayant la même partie gauche en concaténant leurs parties droites.

Dépendences Fonctionnelles

- Lorsque des DF *ne sont pas* de la forme *Clef* → autres attributs, cela signifie que l'on surcharge trop une seule relation.

- Quelquefois, plusieurs attributs déterminent conjointement un autre, bien qu'aucun des deux pris séparément ne le fassent. Exemple :

  `biere, bar` → `prix`

Notion Formelle de Clef

K est une clef pour (le schéma de) la relation R si :

1. $K \rightarrow$ tous les attributs de R.

2. $\forall$ $K' \subset K$; $\exists$ A tel que ($K' \not\rightarrow A$)

- Si K satisfait seulement (1) on dit que K est une *Super-clef ou surclef*.

Conventions de notation

- X, etc représentent des ensembles d'attributs ; A etc, un seul attribut.

- On omet les accolades dans les DF, ainsi on écrira ABC au lieu de $\{A, B, C\}$

Exemple

`Buveurs (`<u>`nom`</u>`, adresse, `<u>`bieresaimees`</u>`, fabriquant, bierefav)`

- {`nom, bieresaimees`} détermine fonctionnellement tous les attributs comme nous l'avons vu.
  - Question : Est-ce une surclef?
- `nom` $\rightarrow$ `bieresaimees` est fausse donc `noms` n'est pas une clef.
- `bieresaimees` $\rightarrow$ `nom` est aussi fausse donc `bieresaimees` n'est pas une clef.

Exemple

- Par conséquent, {`nom, bieresaimees`} est une clef.

- Pas d'autres clefs sur cet exemple.
  - Ni `nom` ni `bieresaimees` n'apparaissent en partie droite de DF donc elles doivent composer n'importe qu'elle surclef.

Qui détermine les clefs et les DF?

- On peut définir un schéma de la relation R simplement en donnant une unique clef K
  - Conséquence les seules DF sont $K \to A$ pour tout A du schéma de R.
  - Pas de surprise K est l'unique clef pour ces DF conformément à la définition formelle de clef.

Qui détermine les clefs et les DF?

- Ou bien, on peut énoncer un certain nombre de DF et *déduire* une ou plusieurs clefs à partir de la définition formelle.

- Règle de bon sens : les DF proviennent de la réalité modélisée, d'associations N-1 ou bien sûr de la notion de clef.
  - "A un horaire donné, pour une salle donnée un seul cours peut avoir lieu"
    `salle horaire` → `cours`.

Normalisation

Objectif = BCNF (Boyce Codd Normal Form) = Toutes les DF sont conséquences du fait que "clef $\rightarrow$ tout le reste".

- Formellement, R est en BCNF si toute DF non triviale pour R, disons $X \rightarrow A$, X est une surclef.
  - "non triviale" = partie droite de la DF n'apparait pas en partie gauche.

Pourquoi normaliser ?

1. Garantie qu 'il n'y a pas de redondances dues aux DF.

2. Garantie qu'il n'y aura pas *d'anomalies de mises à jour* c.a.d une occurence d'un fait est modifiée pas toutes.

3. garantie qu'il n'y aura pas d'anomalies de destruction c.a.d un fait valide est perdu lorsqu'un n-uplet est détruit.

Exemples de problèmes

`Buveurs (nom, adresse, bieresaimees, fabriquant, bierefav)`

nom	adresse	bieresaimees	fabriquant	bierefav
marcel	paris	1664	kro	mort-subite
marcel	???	mort-subite	jeanlain	???e
jules	marseille	1664	???	1664

DF

1. nom $\longrightarrow$ adresse
2. nom $\longrightarrow$ bierefav
3. bieresaimees $\longrightarrow$ fabriquant

Problèmes

- ??? sont redondants car on peut les inférer à partir des DF.
- Anomalies de mises à jour : Si Marcel déménage à toulouse il faudra modifier son adresse partout.
- Anomalies de destruction : Si personne n'a pour bière favorite la "1664" on pert trace de son fabriquant (Kro)

Exemple suite

En fait, chacune des DF viole la condition de BCNF :

- Clef = $\{\texttt{nom, bieresaimees}\}$
  - Chaque DF a en partie gauche un sous ensemble strict de la clef.

Autre Exemple

`Bieres(nom, fabriquant, adressfab).`

- DF $=$ `nom` $\rightarrow$ `fabriquant`, `fabriquant` $\rightarrow$ `adressfab`
- La seule clef est `nom`
  - Or `fabriquant` $\rightarrow$ `adressfab` viole la condition de BCNF car sa partie gauche n'a aucun lien avec la clef.

Déduction de DF

Important car :

- Quand on souhaite améliorer la conception, souvent on se pose la question de savoir si telle ou telle DF est vraie.

Etant données des DF $X1 \rightarrow A1$, $X2 \rightarrow A2$, $\ldots$, $Xn \rightarrow An$, est-ce que $Y \rightarrow B$ est nécessairement vraie pour la même relation?

- Commencer par supposer que deux n-uplets ont la même valeur sur Y. Utiliser les DF pour inférer les autres attributs sur lesquels ils doivent coincider. Si B est parmi ces attributs alors oui, sinon non.

Algorithme

Calcul de $Y^+ =$ *la fermeture* de Y :

- Base : $Y^+ := Y$
- Induction : Si $X \subseteq Y^+$ et si $X \rightarrow A$ est une DF donnée, alors ajouter A à Y^+.
- Fin quand Y^+ ne peut plus être changé. Comme conséquence : $Y \rightarrow Y^+$ et seulement Y^+.

Exemple

$A \rightarrow B,\ BC \rightarrow D$

- $A^+ = AB$
- $C^+ = C$
- $(AC)^+ = ABCD.$
- Donc AC est une clef.

Déduire toutes les DF

Motivation : Supposons que nous ayons une relation $ABCD$ ainsi qu'un ensemble donné F de DF. Si nous décidons de décomposer $ABCD$ en ABC et AD, quelles sont les DF pour ABC et AD?

- Exemple : $F = \{AB \to C, C \to D, D \to A\}$. En apparence, seule $AB \to C$ s'applique à ABC, mais en fait $C \to A$ peut être déduite de F et vaut aussi sur ABC.

- Le problème est de complexité exponentielle dans le pire des cas.

Algorithme

Pour chaque ensemble d'attributs X calculer X^+.

- Ajouter $X \rightarrow A$ pour chaque A dans X^+ - X.

Algorithme suite

- Ignorer ou enlever les DF "évidentes".

  1. DF *triviales* : Partie droite sous ensemble de la partie gauche.
     - Conséquence pas de calcul de $\emptyset^+$ ou la fermeture de tous les attributs (absurde).
  2. Enlever $XY \to Z$ si $X \to Z$ est une DF.
     - Conséquence si X^+ consiste en tous les attributs, alors pas de sens de calculer la fermeture de sur-ensembles de X.
  3. Ignorer les DF dont les parties droites ne consistent pas en un seul attribut.

Algorithme fin

- Notez qu'après avoir projeté les DF découvertes sur une relation, les DF éliminées par les règles 1, 2 et 3 peuvent être inférées *dans la relation projetée*.

Exemple

Exemple : $F = \{AB \rightarrow C,\ C \rightarrow D,\ D \rightarrow A\}$

Quelles sont les DF déductibles?

- $A^+ = A$; $B^+ = B$ (rien).
- $C^+ = ACD$ (ajouter $C \rightarrow A$) .
- $D^+ = AD$ (rien de nouveau).
- $(AB)^+ = ABCD$ (ajouter $AB \rightarrow D$; ignorer tous les sur-ensembles de AB)
- $BC^+ = ABCD$ (rien de nouveau ; ignorer tous les sur-ensembles de BD)

Exemple

- $BD^+ = ABCD$ (ajouter $BD \to C$; igmorer tous les sur-ensembles de BD)

- $AC^+ = ACD$; $AD^+ = AD$; $CD^+ = ACD$ (rien de nouveau).

- $ACD^+ = ACD$ (rien de nouveau).

- Tous les autres ensembles contiennent AB, BC, ou BD donc ignorer.

- Les seules DF interessantes qui découlent de F sont $\{C \to A,\ AB \to D,\ BD \to C\}$.

Décomposition pour atteindre la forme BCNF

Cadre de travail : un schéma de relation R, un ensemble donné F de DF. Supposons que R viole la condition de BCNF par la DF $X \rightarrow A$.

- Notez que nous ne devons considérer que les seules DF de F, car n'importe quelle DF non triviale qui découle de F doit contenir une de leurs parties gauche dans sa propre partie gauche.
  - Donc, toute FD déduite qui ne possèderait pas de surclef comme partie gauche doit avoir une DF de F avec la même propriété.

Décomposition BCNF

1. Etendre la partie droite de la DF $X \rightarrow A$ afin d'inclure X^+.
   - Ne peut pas se ramener à tous les attributs - pourquoi ?

2. Decomposer R en X^+ et (R - X^+) $\cup$ X.

3. Trouver les DF pour les relations ainsi décomposées.
   - Projeter les DF de F = calculer toutes les conséquences de F qui mettent en jeu les seuls attributs de X^+ ou de (R - X^+) $\cup$ X.

Exemple

$R =$ Buveurs (<u>nom</u>, adresse, <u>bieresaimees</u>, fabriquant, bierefav) $F =$

1. nom $\rightarrow$ adresse
2. nom $\rightarrow$ bierefav
3. bieresaimees $\rightarrow$ fabriquant

Choix d'une DF violant BCNF nom $\rightarrow$ adresse.

- Extension de la partie droite: nom $\rightarrow$ adresse, bierefav.

Exemple suite

- Relations décomposées :
  `Buveurs1(`<u>`nom`</u>`, adresse, bierefav)`
  `Buveurs2(`<u>`nom`</u>`, `<u>`bieresaimees`</u>`, fabriquant)`

- DF projetées (en sautant une importante partie du travail qui ne mène nulle part) :
  - Pour `Buveurs1` : `nom` $\rightarrow$ `adresse` et `nom` $\rightarrow$ `bierefav`.
  - Pour `Buveurs2` : `bieresaimees` $\rightarrow$ `fabriquant`.

Exemple suite

- Nouvelles violations BCNF ?
  - Pour `Buveurs1`, `nom` est clef et toutes les parties gauches des DF sont des sur-clefs.
  - Pour `Buveurs2` {`nom`, `bieresaimees`} est la clef et `bieresaimees` $\rightarrow$ `fabriquant` viole BCNF.

Exemple suite et fin

Décomposition de `Buveurs2`

- Extensions de `bieresaimees` → `fabriquant`: rien.

- Décomposition :
  Buveurs3(<u>bieresaimees</u>, fabriquant)
  Buveurs4(<u>nom</u>, <u>bieresaimees</u>)

- Les relations résultant du processus sont toutes en BCNF:

  Buveurs1(<u>nom</u>, adresse, bierefav)
  Buveurs3(<u>bieresaimees</u>, fabriquant)
  Buveurs4(<u>nom</u>, <u>bieresaimees</u>)