Ingénierie des systèmes d'information. Rapelle SQL, modèle relationnel.

Sergey Kirgizov

ESIREM, LIB

2021

Rappel : qualité de l'information

- **Disponibilité :** disponible toujours quand elle est nécessaire et non périmée si disponible
- **Complétude :** comprend tout ce que l'utilisateur doit savoir sur la situation dans laquelle elle est utilisée
- **Brièveté :** ne comprend pas d'éléments non nécessaires
- **Importance :** comprend uniquement ce qui est important dans la situation en cours
- **Exactitude :** correspond à la réalité qu'elle représente, ne comporte pas d'erreurs
- **Précision :** offre une information quantitative avec le degré d'exactitude approprié aux données correspondantes
- **Forme :** Comporte le niveau de détails (affichage tabulaire ou graphique, forme quantitative ou qualitative) en fonction de la situation
- ...

Systèmes d'information de bonne qualité

Leurs propriétés

- **Disponibilité :** vitesse de chargement, la latence...
- **Cohérence :** pas des réponses contradictoires...
- **Robustesse :** sauvegarde, archivage, lutte contre les fausses manips...
- **Confidentialité, sécurité :** cryptage, signatures, chiffrement homomorphe, RGPD
- **Ergonomique et esthétique :** interface conviviale Homme-Machine.

Systèmes d'information de bonne qualité

Bons propriétés parfois sont mutuellement exclusives...

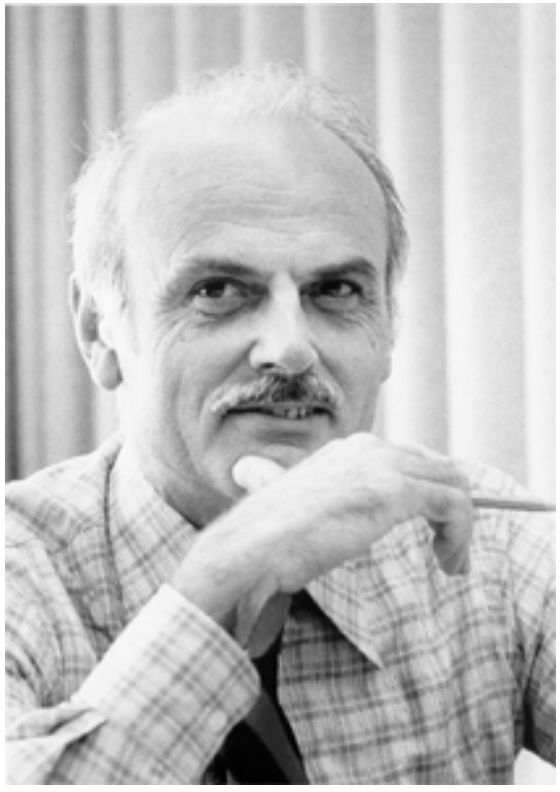
Exemple : archivage vs droit à l'oubli.

Il faut trouver des bons compromis !

Plan

Modèle relationnel

- Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks, 1969
- A Relational Model of Data for Large Shared Data Banks, 1970



Edgar Frank “Ted” Codd (1923-2003)
Programmeur mathématique, pilote pendant la Seconde Guerre.
Inventeur du modèle relationnel des SGBDR.

Modèle relationnel

Les 12 règles de Codd

`https://fr.wikipedia.org/wiki/12_règles_de_Codd`

1. **Unicité**
   Toute l'information dans la base de données est représentée d'une et une seule manière, à savoir par des valeurs dans des champs de colonnes de tables.

Modèle relationnel

Les 12 règles de Codd

`https://fr.wikipedia.org/wiki/12_règles_de_Codd`

1. **Unicité**
   Toute l'information dans la base de données est représentée d'une et une seule manière, à savoir par des valeurs dans des champs de colonnes de tables.

2. **Garantie d'accès**
   Toutes les données doivent être accessibles sans ambiguïté.
   Table → clés primaire → nom de la colonne → valeur.

Modèle relationnel

Les 12 règles de Codd

`https://fr.wikipedia.org/wiki/12_règles_de_Codd`

1. **Unicité**
   Toute l'information dans la base de données est représentée d'une et une seule manière, à savoir par des valeurs dans des champs de colonnes de tables.

2. **Garantie d'accès**
   Toutes les données doivent être accessibles sans ambiguïté.
   Table → clés primaire → nom de la colonne → valeur.

3. **Traitement des valeurs nulles**

   Valeur NULL représenté l'information manquante ou d'information inapplicable.

   Pour tout type des données, systématiquement, exemples :
   - Nombres : NULL != 0
   - Chaînes de caractères : NULL != ""
   - Logique ternaire : NULL != False, NULL != True

Modèle relationnel

Les 12 règles de Codd

`https://fr.wikipedia.org/wiki/12_règles_de_Codd`

1. **Unicité**
   Toute l'information dans la base de données est représentée d'une et une seule manière, à savoir par des valeurs dans des champs de colonnes de tables.

2. **Garantie d'accès**
   Toutes les données doivent être accessibles sans ambiguïté.
   Table → clés primaire → nom de la colonne → valeur.

3. **Traitement des valeurs nulles**

   Valeur NULL représenté l'information manquante ou d'information inapplicable.

   Pour tout type des données, systématiquement, exemples :
   - Nombres : NULL != 0
   - Chaînes de caractères : NULL != ""
   - Logique ternaire : NULL != False, NULL != True

4. **Catalogue lui-même relationnel**
   Le système doit supporter un catalogue en ligne, intégré, relationnel, accessible aux utilisateurs autorisés. Les utilisateurs doivent donc pouvoir accéder à la structure de la base de données (catalogue) employant le même langage d'interrogation qu'ils emploient pour accéder aux données de la base de données.

Modèle relationnel

Les 12 règles de Codd

5. **Sous-langage de données**
   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de
   - définition de structures des données,
   - définition de vues,

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),

Modèle relationnel

Les 12 règles de Codd

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),
   - manipulation de contraintes d'intégrité,

Modèle relationnel

Les 12 règles de Codd

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),
   - manipulation de contraintes d'intégrité,
   - manipulation de contraintesde sécurité (rôles, autorisations),

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),
   - manipulation de contraintes d'intégrité,
   - manipulation de contraintesde sécurité (rôles, autorisations),
   - de gestion de transaction (commencer, valider et annuler une transaction).

Modèle relationnel

Les 12 règles de Codd

5. **Sous-langage de données**

   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),
   - manipulation de contraintes d'intégrité,
   - manipulation de contraintesde sécurité (rôles, autorisations),
   - de gestion de transaction (commencer, valider et annuler une transaction).

Structured Query Language, SQL
Papier "SEQUEL : A Structured English Query Language", 1974, IBM
par Donald D. Chamberlin et Raymond F. Boyce.

6. **Mise à jour des vues**
   Toutes les vues pouvant théoriquement être mises à jour doivent pouvoir l'être par le système.

Modèle relationnel

Les 12 règles de Codd

6. **Mise à jour des vues**
   Toutes les vues pouvant théoriquement être mises à jour doivent pouvoir l'être par le système.

7. **Insertion, mise à jour, et effacement de haut niveau**
   Le système doit supporter les opérations par lot d'insertion, de mise à jour et de suppression. Ceci signifie que des données peuvent être extraites d'une base de données relationnelle dans des ensembles constitués par des données issues de plusieurs tuples et/ou de multiples table. Cette règle explique que l'insertion, la mise à jour, et les opérations d'effacement devraient être supportées aussi bien pour des lots de tuples issues de plusieurs tables que juste pour un tuple unique issu d'une table unique.

8. **Indépendance physique** Les modifications au niveau physique (comment les données sont stockées, si dans les rangées ou les listes liées, etc.) ne nécessitent pas un changement d'une application basée sur les structures.

Modèle relationnel

Les 12 règles de Codd

8. **Indépendance physique** Les modifications au niveau physique (comment les données sont stockées, si dans les rangées ou les listes liées, etc.) ne nécessitent pas un changement d'une application basée sur les structures.

9. **Indépendance logique** Les changements au niveau logique (tables, colonnes, rangées, etc) ne doivent pas exiger un changement dans l'application basée sur les structures. L'indépendance de données logiques est plus difficile à atteindre que l'indépendance de donnée physique.

Modèle relationnel

Les 12 règles de Codd

8. **Indépendance physique** Les modifications au niveau physique (comment les données sont stockées, si dans les rangées ou les listes liées, etc.) ne nécessitent pas un changement d'une application basée sur les structures.

9. **Indépendance logique** Les changements au niveau logique (tables, colonnes, rangées, etc) ne doivent pas exiger un changement dans l'application basée sur les structures. L'indépendance de données logiques est plus difficile à atteindre que l'indépendance de donnée physique.

10. **Indépendance d'intégrité**
    Des contraintes d'intégrité doivent être indiquées séparément des programmes d'application et être stockées dans le catalogue. Il doit être possible de changer de telles contraintes au fur et à mesure sans affecter inutilement les applications existantes.

Modèle relationnel

Les 12 règles de Codd

8. **Indépendance physique** Les modifications au niveau physique (comment les données sont stockées, si dans les rangées ou les listes liées, etc.) ne nécessitent pas un changement d'une application basée sur les structures.

9. **Indépendance logique** Les changements au niveau logique (tables, colonnes, rangées, etc) ne doivent pas exiger un changement dans l'application basée sur les structures. L'indépendance de données logiques est plus difficile à atteindre que l'indépendance de donnée physique.

10. **Indépendance d'intégrité**
    Des contraintes d'intégrité doivent être indiquées séparément des programmes d'application et être stockées dans le catalogue. Il doit être possible de changer de telles contraintes au fur et à mesure sans affecter inutilement les applications existantes.

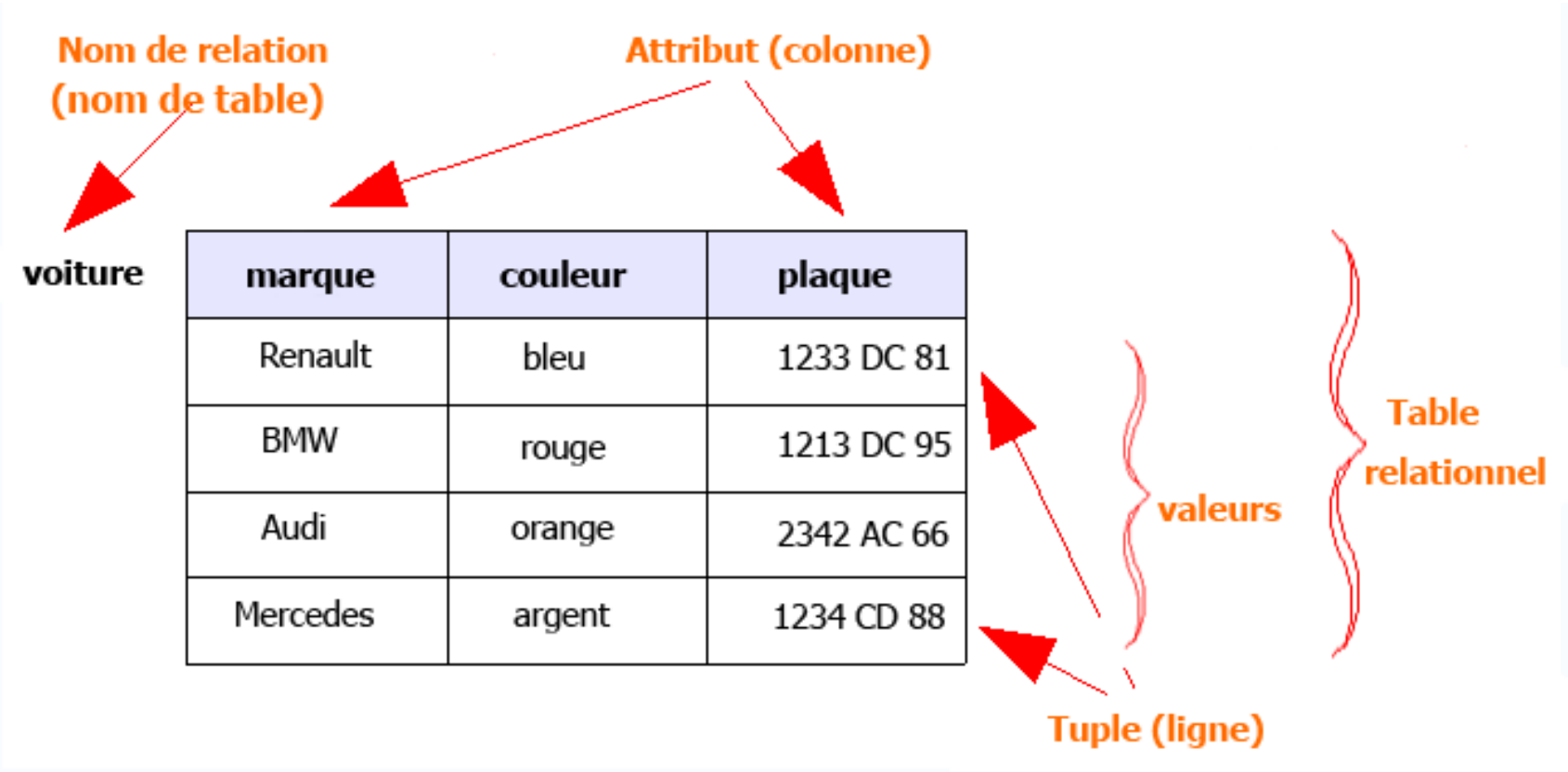
11. **Indépendance de distribution**
    La distribution des parties de la base de données à diverses localisations doit être invisible aux utilisateurs de la base de données. Les applications existantes doivent continuer à fonctionner avec succès quand des données existantes sont redistribués dans le système.
    C'est très important aujourd'hui !

12. **Règle de non-subversion, encapsulation**

    Si le système fournit une interface de bas niveau, cette interface ne doit pas permettre de contourner le système (par exemple une contrainte relationnelle de sécurité ou d'intégrité).

Représentation des données

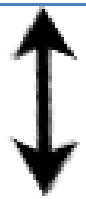
Table relationnel



src : Puerto01 @ Wikipedia.fr

Liens entre deux tables

ID voiture	ID carburant	marque	plaque
1	1	Renault	1233 DC 81
2	2	BMW	1213 DC 95
2	1	Audi	2342 AC 66



ID carburant (*)	type
1	pétrole
2	gas-oil
3	gas

src : Puerto01 @ Wikipedia.fr

Normalisation

https://en.wikipedia.org/wiki/Database_normalization

Normalisation

https://en.wikipedia.org/wiki/Database_normalization

C'est pour la cohérence !

L'idée générale

Respecter l'unicité.

Éliminer les répétitions.

~~Éliminer les répétitions.~~

0NF

Une grande table

Title	Author	Author Nationality	Format	Price	Subject	Pages	Thickness	Publisher	Publisher Country	Publication Type	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL, Database, Design	520	Thick	Apress	USA	E-book	1	Tutorial

1NF

- Valeurs sémantiquement atomiques.
- Les lignes ne se répètent pas.

Title	Format	Author	Author Nationality	Price	Subject 1	Subject 2	Subject 3	Pages	Thickness	Publisher	Publisher country	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	MySQL	Database	Design	520	Thick	Apress	USA	1	Tutorial

Dépendances fonctionnelles

Folklore ? Grèce antique ? ! Mathématiques ?

XIIe siècle, Sharaf al-Din al-Tusi, Iran
XIVe siècle, Nicole Oresme, Normandie

XVIIe siècle, Gottfried Leibniz, Allemagne
XIX-XXe siècle, Boole, Cantor, Russel, Bourbaki...
XXe siècle, SQL. **Claude Delobel et Jorma Rissanen "Decomposition of files, a basis for data storage and retrieval", California, IBM, 1971**

Table $\approx$ fonction.

Dépendances fonctionnelles

Table $\approx$ fonction.

Dépendances fonctionnelles

Table $\approx$ fonction.
Où est l'argument et où est la valeur de la fonction ?

Dépendances fonctionnelles

Table $\approx$ fonction.
Où est l'argument et où est la valeur de la fonction ?

Exemple
On a une table avec les colonnes
COURS, HEURE, JOUR, SALLE, ÉTAGE

Dépendances fonctionnelles :

- (HEURE, JOUR, SALLE) $\rightarrow$ COURS
- (SALLE) $\rightarrow$ ÉTAGE

Dépendances fonctionnelles

Table $\approx$ fonction.
Où est l'argument et où est la valeur de la fonction ?

Exemple
On a une table avec les colonnes
COURS, HEURE, JOUR, SALLE, ÉTAGE

Dépendances fonctionnelles :

- (HEURE, JOUR, SALLE) $\rightarrow$ COURS
- (SALLE) $\rightarrow$ ÉTAGE

Clé :
La donnée qui permet d'identifier de manière unique un enregistrement dans une table.

- (HEURE, JOUR, SALLE) -> (COURS, ÉTAGE)

Plusieurs clés sont possibles, **clé candidates.**
Une **clé primaire** est choisie parmi les clés candidates.

2nf

Un attribut **non clé** ne dépend pas d'une partie de la clé mais de toute la clé.

Manufacturer	**Model**	**Model full name**	**Manufacturer country**
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Redondance –> risque d'erreurs

`https://en.wikipedia.org/wiki/Second_normal_form`

2nf

Un attribut **non clé** ne dépend pas d'une partie de la clé mais de toute la clé.

Manufacturer	**Manufacturer country**
Forte	Italy
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japan
Hoch	Germany

Manufacturer	**Model**	**Model full name**
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Brushmaster	SuperBrush	Brushmaster SuperBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

`https://en.wikipedia.org/wiki/Second_normal_form`

3nf

Tous les attributs **non clé** doivent dépendre directement de la clé.

Clé : Book
Dépendance fonctionnelles :

- Book → Genre
- Book → Author
- Book → Author Nationality
- **Author → Author Nationality ? ? ! !**

Book	Genre	Author	Author Nationality
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne	French
Journey to the Center of the Earth	Science Fiction	Jules Verne	French
Leaves of Grass	Poetry	Walt Whitman	American
Anna Karenina	Literary Fiction	Leo Tolstoy	Russian
A Confession	Religious Autobiography	Leo Tolstoy	Russian

src : `https://en.wikipedia.org/wiki/Transitive_dependency`

Tous les attributs non clé ne sont pas source de dépendance fonctionnelle vers une partie de la clé.

Exemple

Une table avec les Colonnes (A, B, C, D, F)

Dépendances

$A \rightarrow (B, C, D, F)$

$(B, F) \rightarrow (A, C, D)$

$D \rightarrow F$

Clés possibles (candidates) A et (B, F)

Tous les attributs non clé ne sont pas source de dépendance fonctionnelle vers une partie de la clé.

Exemple

Une table avec les Colonnes (A, B, C, D, F)
Dépendances
$A \rightarrow (B, C, D, F)$
$(B, F) \rightarrow (A, C, D)$
$D \rightarrow F$
Clés possibles (candidates) A et (B, F)
Décomposer en 2 tables : (A, B, C, D) et D, F

3.5nf

Forme normale de Boyce-Codd

Tous les attributs non clé ne sont pas source de dépendance fonctionnelle vers une partie de la clé.

Exemple

Une table avec les Colonnes (A, B, C, D, F)
Dépendances
$A \rightarrow (B, C, D, F)$
$(B, F) \rightarrow (A, C, D)$
$D \rightarrow F$
Clés possibles (candidates) A et (B, F)
Décomposer en 2 tables : (A, B, C, D) et D, F

Mais ! Ce n'est pas toujours possible, exemple
$(A, B) \rightarrow C$, $C \rightarrow B$.
Beeri et Bernstein,"Computational problems related to the design of normal form relational schemas", 1979

4nf

Il ne doit pas être possible de reconstituer une table par une jointure de **deux** tables plus petites.

5nf

Il ne doit pas être possible de reconstituer une table par une jointure de **plusieures** tables plus petites.

Autres formes normales existent.

Pourquoi normaliser ?

- **Cohérence**
- **Disponibilité**

Avantages :

- Limiter les redondances
- Diminuer la volumétrie
- Limiter le nombre des mises à jour

Pourquoi normaliser ?

- **Cohérence**
- **Disponibilité**

Avantages :

- Limiter les redondances
- Diminuer la volumétrie
- Limiter le nombre des mises à jour

Inconvénients :

- Beaucoup des tables
- Temps d'accès potentiellement plus longs, nombreuses jointures.

Pourquoi normaliser ?

- **Cohérence**
- **Disponibilité**

Avantages :

- Limiter les redondances
- Diminuer la volumétrie
- Limiter le nombre des mises à jour

Inconvénients :

- Beaucoup des tables
- Temps d'accès potentiellement plus longs, nombreuses jointures.
  Solutions possibles
  - dénormalisation...
  - indices...

SQL

Modèle relationnel

Les 12 règles de Codd

5. **Sous-langage de données**

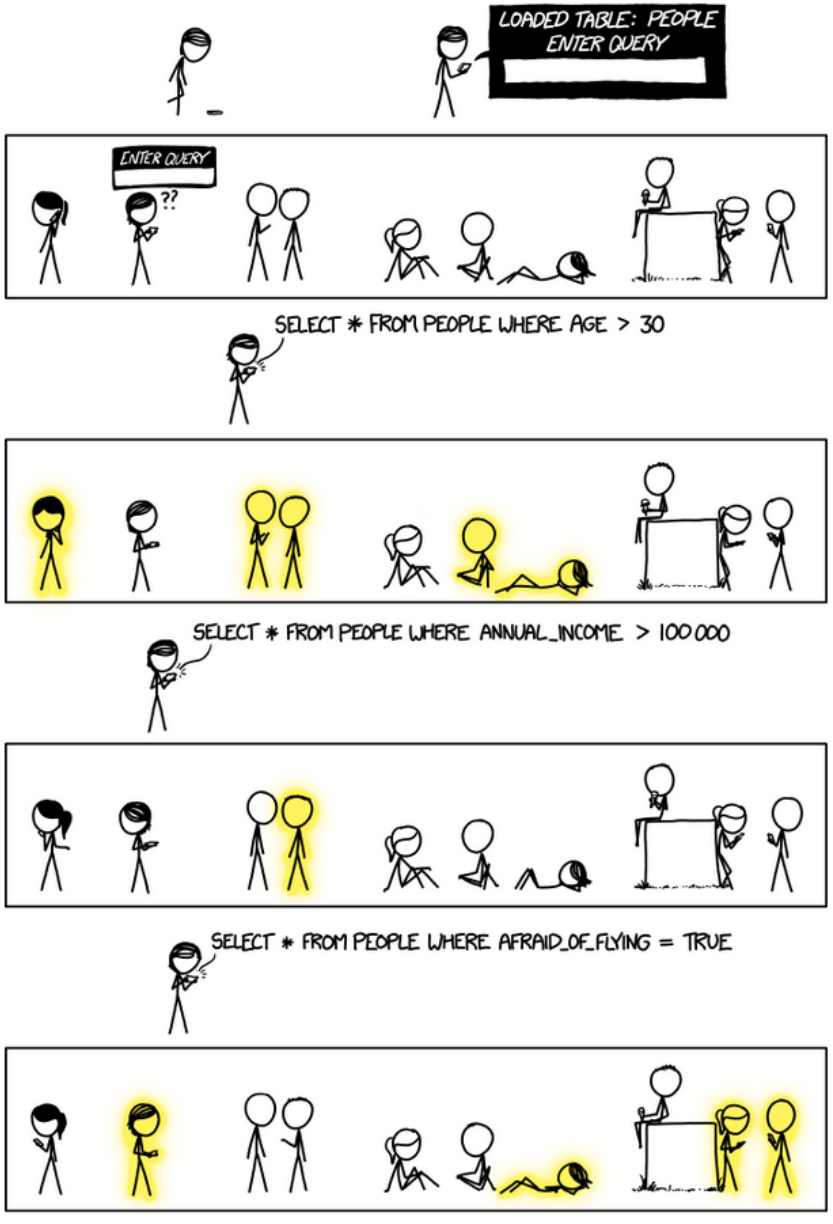
   Au moins un langage qui peut être employé interactivement et dans des programmes d'application, supportant des opérations de

   - définition de structures des données,
   - définition de vues,
   - manipulation de données (CRUD : create, read, update, delete),
   - manipulation de contraintes d'intégrité,
   - manipulation de contraintesde sécurité (rôles, autorisations),
   - de gestion de transaction (commencer, valider et annuler une transaction).

Structured Query Language, SQL
Papier "SEQUEL : A Structured English Query Language", 1974, IBM
par Donald D. Chamberlin et Raymond F. Boyce.

SQL

- Ergonomie
- Cohérence



SQL littérature

- Livre **“Learn SQL the Hard Way”** de Zed A. Shaw
- `https://en.wikipedia.org/wiki/SQL_syntax`
- `https://www.sqlite.org/lang.html`

1970 Modèle relationnelle
1974 SQL

.

.

.

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision that added integrity constraints, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types (arrays), and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice versa (SQL/JRT).
2003	SQL:2003		Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways that SQL can be used with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database, and publishing both XML and conventional SQL-data in XML form. In addition, it lets applications integrate queries into their SQL code with XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[34]
2008	SQL:2008		Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers, TRUNCATE statement,[35] FETCH clause.
2011	SQL:2011		Adds temporal data (PERIOD FOR)[36] (more information at: Temporal database#History). Enhancements for *window functions* and FETCH clause.[37]
2016	SQL:2016		Adds row pattern matching, polymorphic table functions, JSON.
2019	SQL:2019		Adds Part 15, multidimensional arrays (MDarray type and operators).

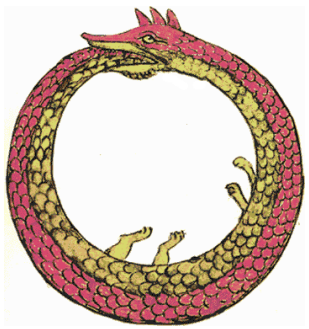
Puissance de SQL

- **Exercise 1**
  Concevoir une requête SQL qui retourne son propre code source.

Puissance de SQL

- **Exercise 1**
  Concevoir une requête SQL qui retourne son propre code source.

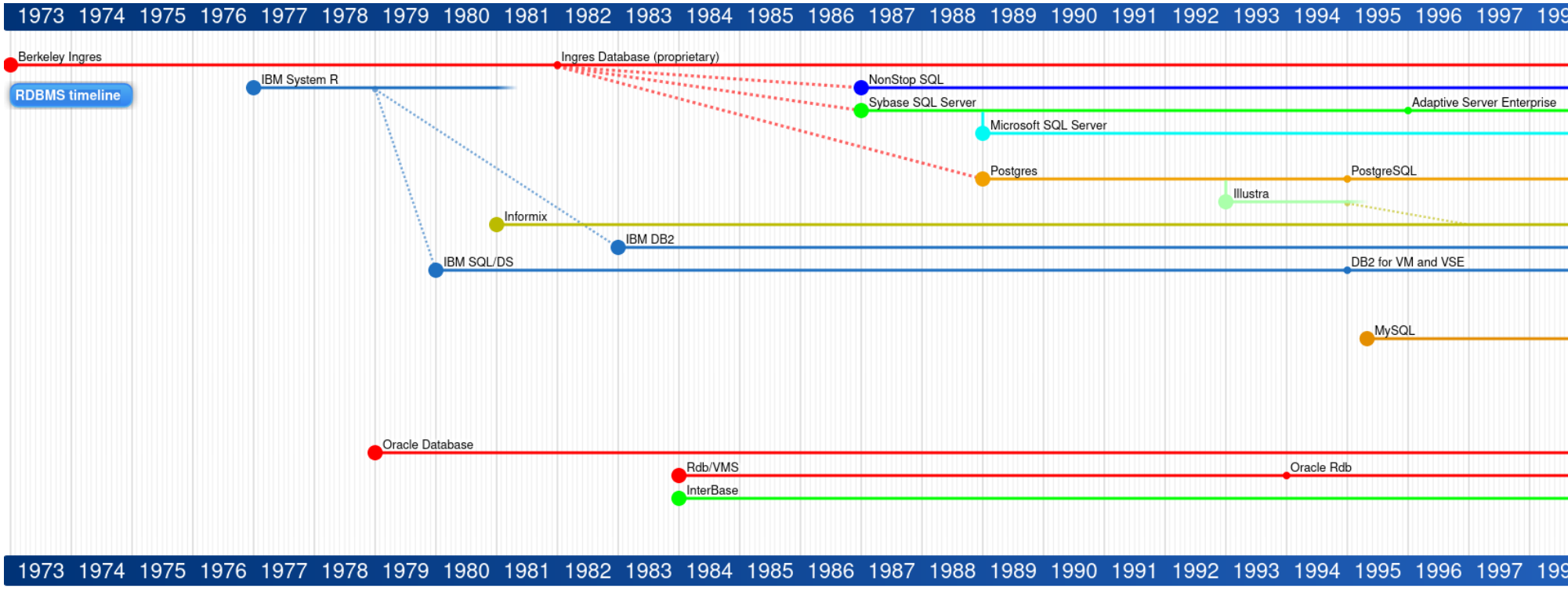
- **Exercise 2**
  Concevoir une requête SQL qui retourne son propre code source, sans utiliser ni tables, ni fichiers.



Note historique

History of Databases

https://www.youtube.com/watch?v=KG-mqHoXOXY



Full `https://en.wikipedia.org/wiki/User:Intgr/RDBMS_timeline`

Les transactions

Transactions

- **Cohérence**
- **Robustesse**

- ACD 1981, Jim Gray
  "The Transaction Concept : Virtues and Limitations"

INTRODUCTION: What is a transaction?

The transaction concept derives from contract law. In making a contract, two or more parties negotiate for a while and then make a deal. The deal is made binding by the joint signature of a document or by some other act (as simple as a handshake or nod). If the parties are rather suspicious of one another or just want to be safe, they appoint an intermediary (usually called an escrow officer to coordinate the commitment of the transaction.

The Christian wedding ceremony gives a good example of such a contract. The bride and groom "negotiate" for days or years and then appoint a minister to conduct the marriage ceremony. The minister first asks if anyone has any objections to the marriage; he then asks the bride and groom if they agree to the marriage. If they both say, "I do", he pronounces them man and wife.

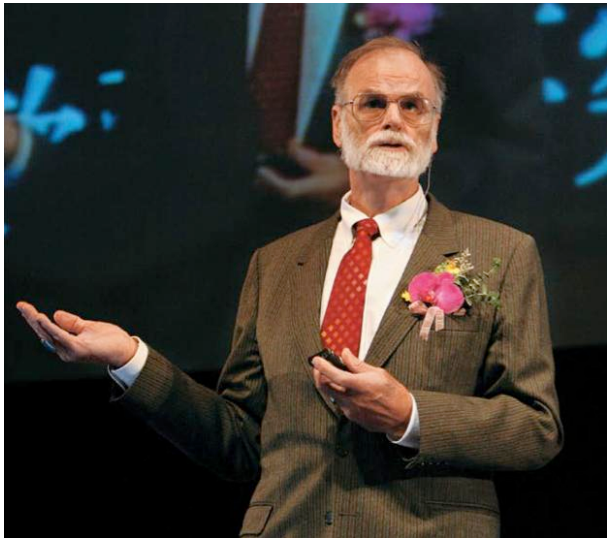
Of course, a contract is simply an agreement. Individuals can violate if they are willing to break the law. But legally, a contract (transaction) can only be annulled if it was illegal in the first place. Adjustment of a bad transaction is done via further compensating transactions (including legal redress).

- ACD 1981, Jim Gray
  “The Transaction Concept : Virtues and Limitations”

Consistency : the transaction must obey legal protocols.

Atomicity : it either happens or it does not ; either all are bound by the contract or none are.

Durability : once a transaction is committed, it cannot be abrogated.

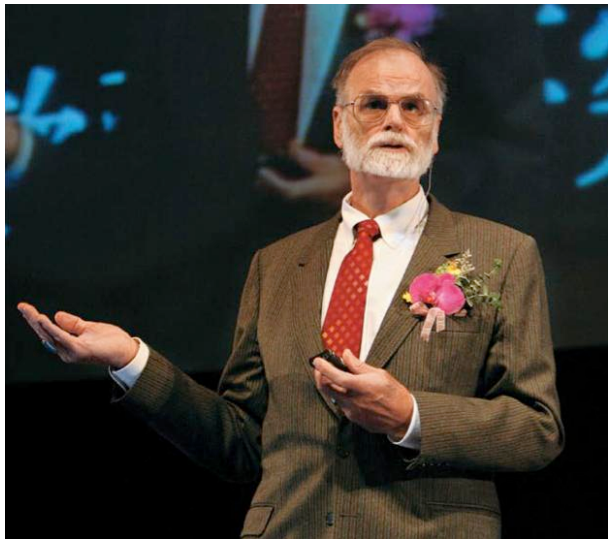


- ACD 1981, Jim Gray
  “The Transaction Concept : Virtues and Limitations”

  Consistency : the transaction must obey legal protocols.

  Atomicity : it either happens or it does not ; either all are bound by the contract or none are.

  Durability : once a transaction is committed, it cannot be abrogated.



- ACID - 1983, Andreas Reuter, Theo Härder
  “Principles of transaction-oriented database recovery”

  Isolation : events within a transaction must be hidden from other transactions running concurrently.

Transactions

- **Cohérence**
- **Robustesse**

Atomicité : exécuter tout our rien !
Revenir à l'état de départ en cas d'erreur.

Cohérence : Une transaction transforme le système d'un état cohérent à un autre état cohérent.

Isolation : Si les transactions s'exécutent simultanément, alors chacune doit demeurer indépendante de l'autre.

Durabilité : une fois confirmé, la transaction demeure enregistrée même en cas d'une panne d'électricité, d'un autre problème matériel.

Mémoire transactionnelle

Programmation concurrente, threads...

- **Cohérence**
- **Robustesse**

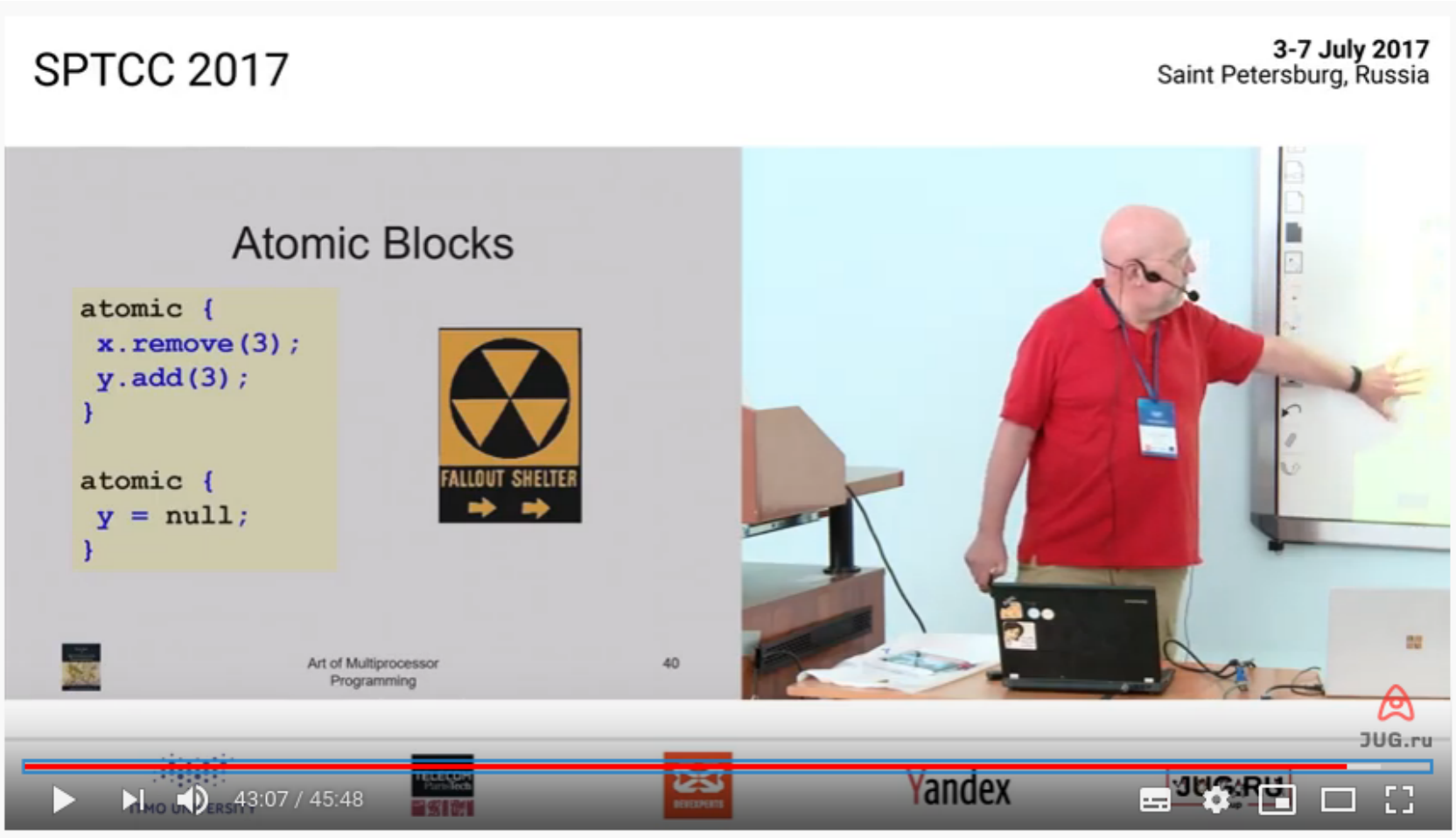
Mémoire transactionnelle matérielle

- Blue Gene/Q, processeur d'IBM
- IBM zEnterprise EC12
- Intel's Transactional Synchronization Extensions (TSX)
- IBM POWER8...

Mémoire transactionnelle logicielle

- STM pour Haskell.
- Mnesia pour Erlang.
- Lightweight Transaction Library pour C.
- AtomJava
- AtomizeJs
- coThreads pour OCaml
- Bibliothèque pour autres languages Scale, Smalltalk, Lisp(s), Groovy, C++, C#, F#,... `https://en.wikipedia.org/wiki/Software_transactional_memory#Implementations`

Leur performance, c'est une question de recherche.



Maurice Herlihy – Transactional Memory (Part 1)

https://www.youtube.com/watch?v=ZkUrl8BZHjk Part 1
https://www.youtube.com/watch?v=FhgkyhXSDm0 Part 2
https://www.youtube.com/watch?v=VH5pTjK54Q4 Part 3
https://www.youtube.com/watch?v=3TfCXEtDKvk Part 4

Contraintes

- **Cohérence**
- **Robustesse**

Les contraintes peuvent être spécifiées pour restreindre les valeurs suite, par exemple, à certaines règles commerciales.

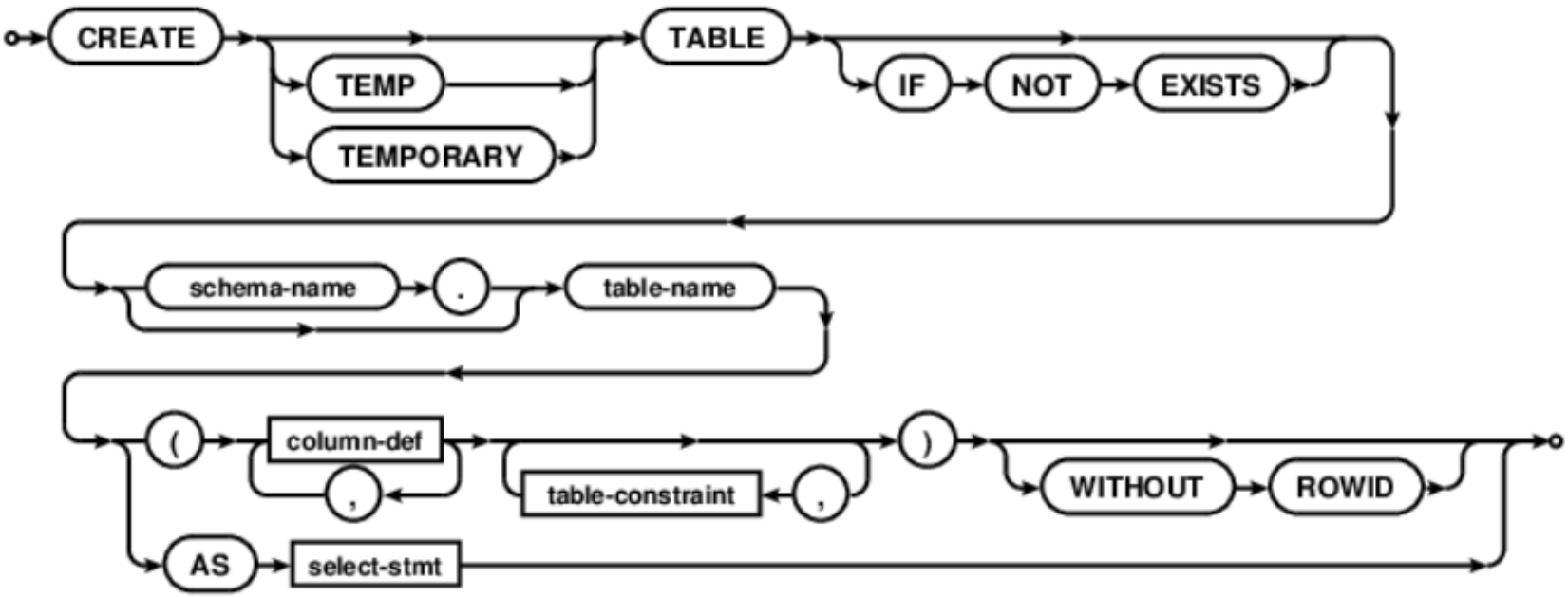
Types de contraintes :

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

Syntaxe de contraintes

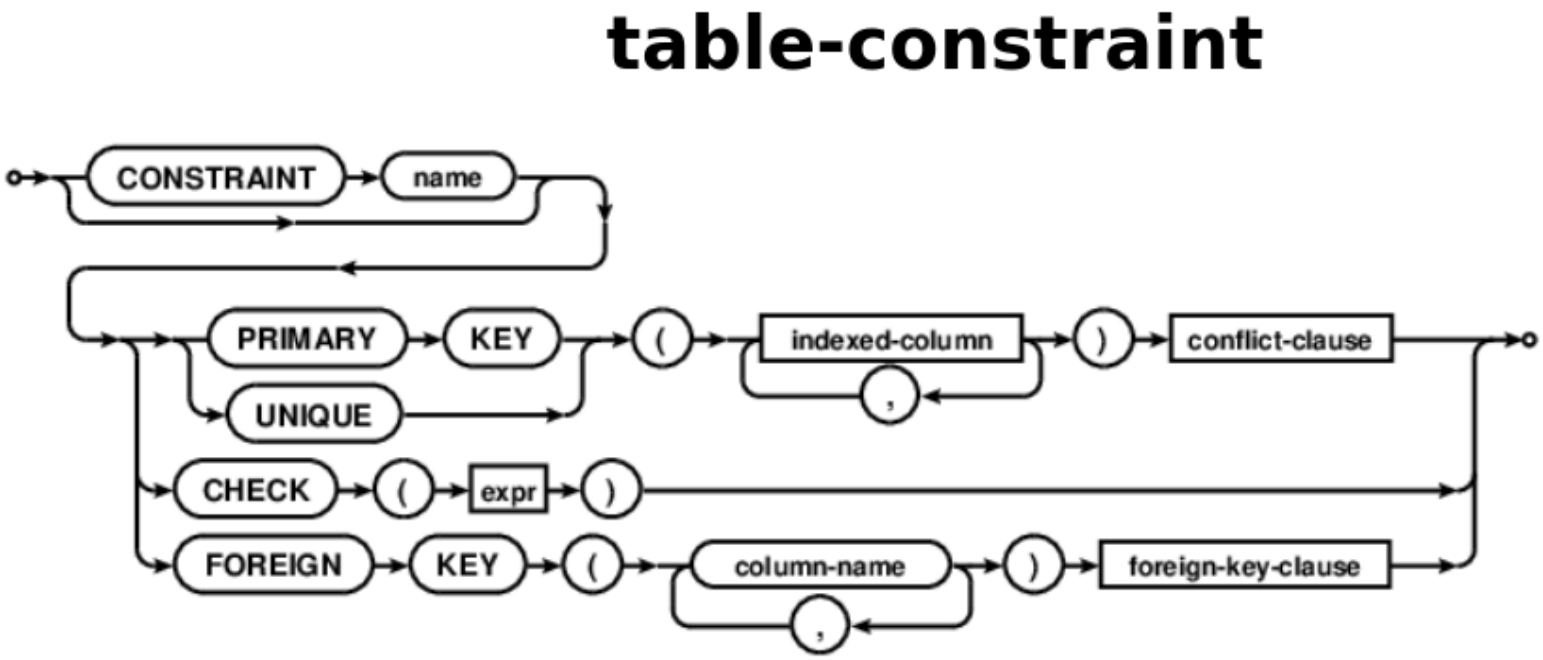
CREATE TABLE

create-table-stmt: hide



src : SQL As Understood By SQLite
`https://www.sqlite.org/lang_createtable.html`

Syntaxe de contraintes



Il ixiste des standards SQL. Mais chaque SGBD peut avoir ses propres nuances syntaxiques.

Triggers $\approx$
réactions algorithmiques
aux différents événements

Triggers

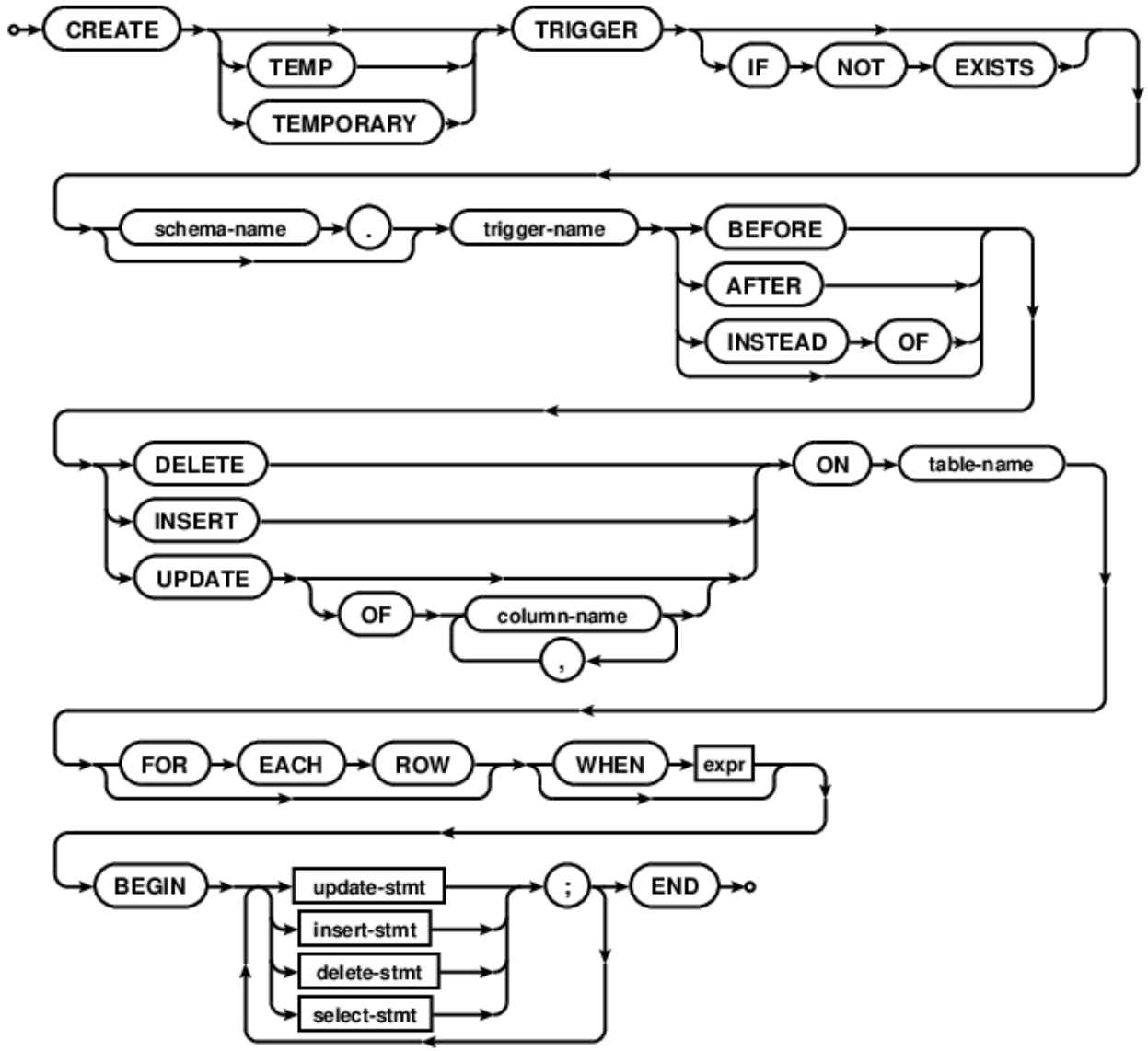
- **Cohérence**
- **Robustesse**

Le code s'éxécutant lors de certains événements :

- Après la création
- Avant la modification
- Après la modification
- Avant la suppression
- Après la suppression
- ...

CREATE TRIGGER

create-trigger-stmt: hide



https://www.sqlite.org/lang_createtrigger.html

SQL++ pour les triggers ?

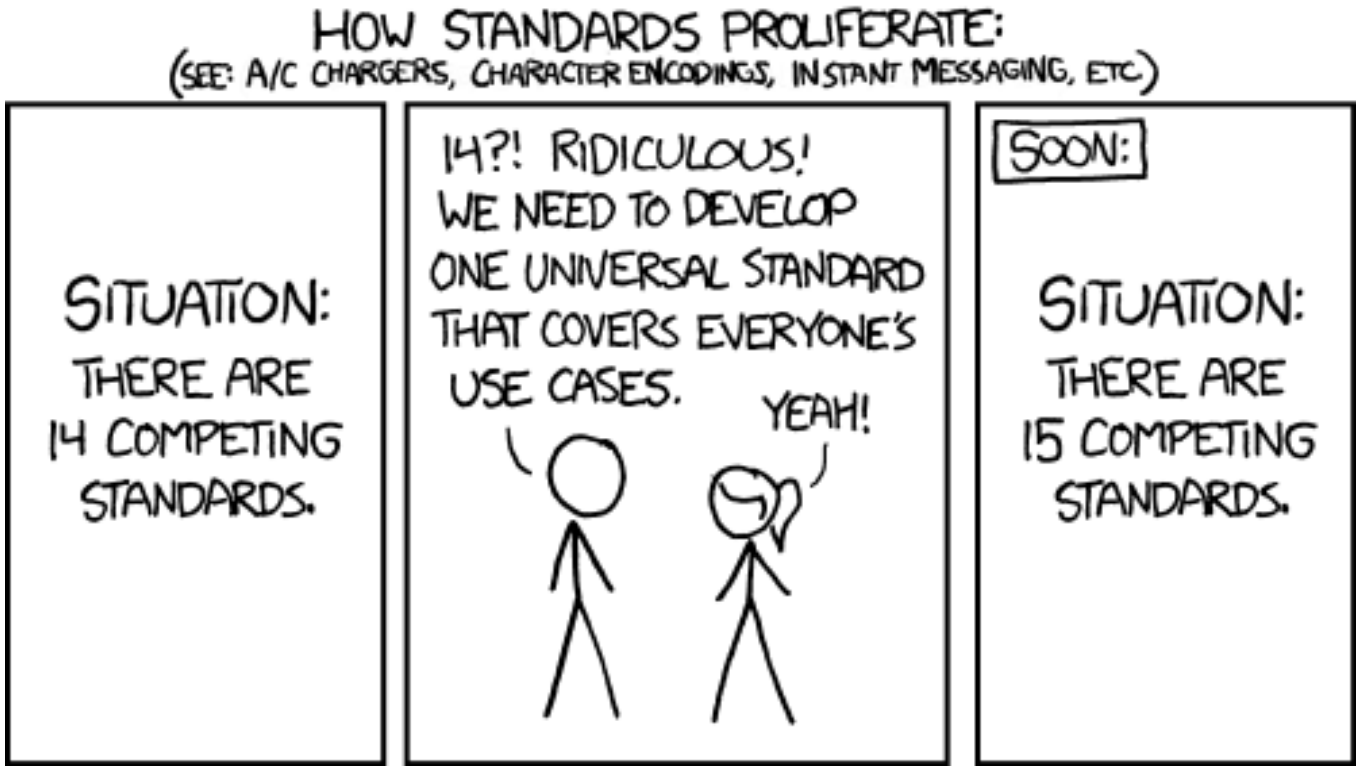
Languages dédiés au triggers

Standard ISO : SQL/Persistent Stored Modules, 1996, SQL :2016

Langages dédiés :

- PL/pgSQL pour PostgreSQL
- PL/SQL pour Oracle
- Transact-SQL pour Microsoft SQL Server
- ...

On regardera SQLite en TP...
`https://www.sqlite.org/c3ref/create_function.html`



https://xkcd.com/927/

Indices

src Dr. Marcus Gossler @ Wiki

Indices

- **Disponibilité**
- **Cohérence**

Pourquoi ?

- accélère les opérations de recherche
- vérifier rapidement les propriétés des données,
- ... de tri
- ... de jointure

Technologies sous-jacentes :

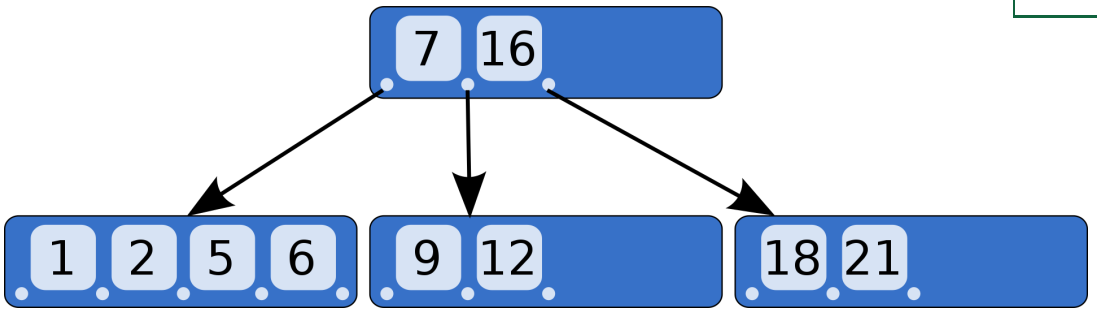
- **La famille d'arbres B.**
- bitmaps
- tables de hachage

Il y an impact sur les performances en modification/création.

B-arbre (dans le cours suivant)

B-tree

- **Disponibilité**
- **Cohérence**



Inventé par : Rudolf Bayer, Edward M. McCreight en **1970** dans BOEING Research Labs.

Applications :

- Base de données : CouchDB, MySQL, Oracle, PostgreSQL, LMDB, SQLite, ...
- Systèmes de fichiers : HFS+, NTFS, jfs2, btrfs, Ext4, ...

Variantes : B+-tree et B*-tree

Merci.