Databases

Conjunctive Queries

Pierre Senellart

5 March 2025

Plan

Conjunctive queries

Calculus: relational calculus query without $\vee$, $\neg$, $\forall$ (so, using only $\exists$, $\wedge$, and relation or equality predicates)

Algebra: algebra query without $\cup$, $\setminus$ (so, using only ρ, π, σ, $\times$ and relation symbols; $\bowtie$ are possible)

SQL: `SELECT ... FROM ... WHERE ...` (no set operators, no aggregation, etc.)

Conjunctive queries: Representation

- We represent conjunctive queries in the form:

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$

  where each $\mathbf{z}_i$ is a tuple of variables among $\mathbf{x}$ and $\mathbf{y}$, and where each x_j appears at least in one $\mathbf{z}_j$

- Set semantics: for all database D, $q(D)$ is a finite set of tuples

Why are they interesting?

- Reasonable (and fairly common!) fragments of the relational calculus, the relational algebra, SQL
- In some sense, simplest query language that is not completely trivial
- Interesting object of studies (structure, complexity properties, etc.)
- Basis to define more elaborate query languages (UCQs, Datalog, conjunctive regular path queries, etc.)

Plan

Data complexity

Theorem

Boolean CQ evaluation is AC^0 *in data complexity.*

Data complexity

Theorem

Boolean CQ evaluation is AC^0 in data complexity.

Proof.

CQs are a particular case of relational calculus queries. □

Combined complexity

Theorem

Boolean CQ evaluation is NP-complete in combined complexity.

Combined complexity

Theorem

Boolean CQ evaluation is NP-complete in combined complexity.

Proof.

Membership in NP is easy. Hardness for NP can be proved by reduction from graph 3-colorability. □

Plan

Hypergraph

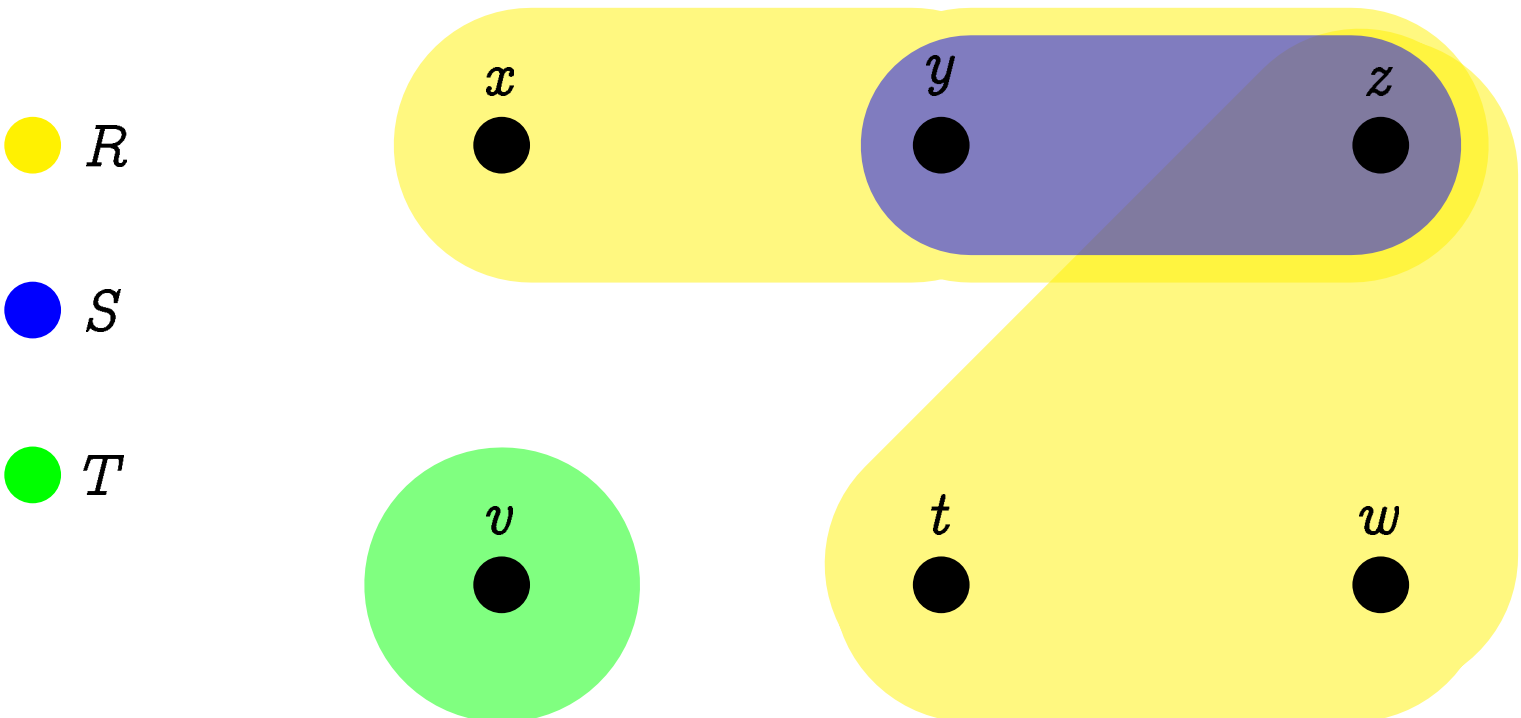
- A hypergraph $\mathcal{H}$ is a pair (V, E) where V is a finite set of vertices, $E \subseteq 2^V$ is a set of hyperedges
- Often useful to considered labeled hypergraphs, where each hyperedge is associated with a label in $\mathcal{L}$
- Sometimes, we allow a multiset of labels instead of a single label
- Generalization of undirected graphs (undirected graphs are hypergraphs where each hyperedge has cardinality 2; labeled hypergraphs are generalizations of labeled graphs; hypergraphs with multiset of labels are generalizations of multigraphs)

Conjunctive query as hypergraph

- Natural representation of a conjunctive query as a labeled hypergraph:
  - variables (and possibly constants) become vertices
  - relation atoms over these variables become hyperedges
  - relation names become labels
- Some information is lost: the order of attributes, which variables are free and which are projected out
- Useful to identify some restrictions of conjunctive queries that are easier to process than others

Example hypergraph of a CQ

$$\exists x \exists y \exists z \exists t \exists v \exists w \; R(x, y, z) \wedge R(z, t, w) \wedge S(y, z) \wedge T(v)$$



Connected components

- Two hyperedges are adjacent if they share a vertex
- Easy to define a notion of connected component of a hypergraph: two vertices are in the same connected component if there exists a path (a sequence of adjacent hyperedge) that connects the two nodes
- Impact on conjunctive query evaluation: the parts of the query corresponding to disconnected components can be evaluated independently
- So we mostly focus on CQs with connected hypergraphs

Cyclicity

- Cyclicity in graphs is easy to define
- Cyclicity in hypergraphs is more complex

Cyclicity

- Cyclicity in graphs is easy to define
- Cyclicity in hypergraphs is more complex
- First proposal: the bipartite graph of which vertex belongs to which hyperedge is cyclic

Cyclicity

- Cyclicity in graphs is easy to define
- Cyclicity in hypergraphs is more complex
- First proposal: the bipartite graph of which vertex belongs to which hyperedge is cyclic $\rightarrow$ Berge-acyclicity
- Intuitive, but too restrictive (cf. previous example of hypergraph)

α-acyclic query

- A hypergraph $\mathcal{H}$ has a join tree if one can find a tree whose nodes are labeled by the hyperedges of $\mathcal{H}$ and such that:
  - every hyperedge of $\mathcal{H}$ appears as the label of one node of the tree;
  - for every vertex x of $\mathcal{H}$, the set of tree nodes labeled by a hyperedge referring to x is a connected subtree
- A query is α-acyclic if its hypergraph has a join tree
- A join tree can be obtained in linear time if it exists [Tarjan and Yannakakis, 1984]

Yannakakis's algorithm [Yannakakis, 1981]

- Algorithm to evaluate acyclic queries (non-necessarily Boolean):
  1. Construct the join tree
  2. Eliminate all useless tuples of a relation with the semijoin operator $\ltimes$: $R \ltimes S = \Pi_{R.*}(R \bowtie S)$ by navigating twice in the join tree: from bottom up, then from top down
  3. Evaluate the query bottom up, by computing joins following the tree and by projecting useless variables out as you go
- Polynomial complexity in the size of the query, the input, and the output (combined complexity)

Yannakakis's algorithm [Yannakakis, 1981]

- Algorithm to evaluate acyclic queries (non-necessarily Boolean):
  1. Construct the join tree
  2. Eliminate all useless tuples of a relation with the semijoin operator $\ltimes$: $R \ltimes S = \Pi_{R.*}(R \bowtie S)$ by navigating twice in the join tree: from bottom up, then from top down
  3. Evaluate the query bottom up, by computing joins following the tree and by projecting useless variables out as you go
- Polynomial complexity in the size of the query, the input, and the output (combined complexity)
- Linear complexity for Boolean queries (and also for free-connex queries, where an extra hyperedge with all free variables are added)

Plan

Homomorphism

Definition

A homomorphism from a CQ q to a CQ q' is a function φ from the variables $\mathbf{x}, \mathbf{y}$ of q fo the variables $\mathbf{x}', \mathbf{y}'$ of q' such that:

- $\varphi(\mathbf{x}) = \mathbf{x}'$
- for every atom $R(\mathbf{z}_i)$ of q, there exists an atom $R(\mathbf{z}'_{i'})$ of q' such that $\varphi(\mathbf{z}_i) = \mathbf{z}'_{i'}$

Definition

A homomorphism is an isomorphism if it is one-to-one and its converse is a homomorphism.

Instance associated to a query

Definition

For all conjunctive query

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$

one can construct the instance associated to q, denoted I_q, where:

- the active domain is $\{ a_z \mid z \in \mathbf{x} \cup \mathbf{y} \}$
- the n tuples of I_q are of the form $R(a_{z_{i1},\ldots,z_{ik}})$ for $R(z_{i1}, \ldots, z_{ik})$ each atom of q

Homomorphism test

Proposition

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$
$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

there exists a homomorphism from q to q' iff $(a_{x'_1}, \ldots, a_{x'_j}) \in q(I_{q'})$.

Proof.

Homomorphism test

Proposition

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$
$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

there exists a homomorphism from q to q' iff $(a_{x'_1}, \ldots, a_{x'_j}) \in q(I_{q'})$.

Proof.

$\Leftarrow$ Suppose $(a_{x'_1}, \ldots, a_{x'_j}) \in q(I_{q'})$.
Then there exists $\psi : \mathbf{x} \cup \mathbf{y} \to \{a_z \mid z \in \mathbf{x}' \cup \mathbf{y}'\}$ such that:

- $\psi(\mathbf{x}) = (a_{x'_1}, \ldots, a_{x'_j})$
- $\forall 1 \leq i \leq n,\ R_i(\psi(\mathbf{z}_i)) \in I_{q'}$

Let $f : \{a_z \mid z \in \mathbf{x}' \cup \mathbf{y}'\} \to \mathbf{x}' \cup \mathbf{y}'$ defined by $f(a_z) = z$. Then $f \circ \psi$ is a homomorphism from q to q'.

Homomorphism test

Proposition

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$
$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

there exists a homomorphism from q to q' iff
$(a_{x'_1}, \ldots, a_{x'_j}) \in q(I_{q'})$.

Proof.

$\Rightarrow$ Suppose there exists $\varphi : \mathbf{x} \cup \mathbf{y} \to \mathbf{x}' \cup \mathbf{y}'$ such that:

- $\varphi(\mathbf{x}) = (x'_1, \ldots, x'_j)$
- $\forall 1 \leq i \leq n$, $R_i(\varphi(\mathbf{z}_i))$ is one of the atoms of q'

Let $f' : \mathbf{x}' \cup \mathbf{y}' \to \{a_z \mid z \in \mathbf{x}' \cup \mathbf{y}'\}$ defined by $f'(z) = a_z$. Then $f' \circ \varphi : \mathbf{x} \cup \mathbf{y} \to \{a_z \mid z \in \mathbf{x}' \cup \mathbf{y}'\}$ witnesses that $(a_{x'_1}, \ldots, a_{x'_j}) \in q(I_{q'})$. □

Homomorphism theorem

Theorem ([Chandra and Merlin, 1977])

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$

$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

$q \sqsubseteq q'$ iff there exists a homomorphism from q' to q.

Homomorphism theorem

Theorem ([Chandra and Merlin, 1977])

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$

$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

$q \sqsubseteq q'$ iff there exists a homomorphism from q' to q.

Proof.

$\Leftarrow$ Let h be a homomorphism from q' to q. Let D be an arbitrary database with active domain A and $t \in q(D)$.

Let $\psi : \mathbf{x} \cup \mathbf{y} \to A$ the corresponding valuation.
Then $\psi \circ h : \mathbf{x}' \cup \mathbf{y}' \to A$ witnesses that $t \in q'(D)$.

Homomorphism theorem

Theorem ([Chandra and Merlin, 1977])

For all CQs

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} : R_1(\mathbf{z}_1) \wedge \cdots \wedge R_n(\mathbf{z}_n)$$

$$q'(\mathbf{x}') \leftarrow \exists \mathbf{y}' : R'_1(\mathbf{z}'_1) \wedge \cdots \wedge R'_{n'}(\mathbf{z}'_{n'})$$

$q \sqsubseteq q'$ iff there exists a homomorphism from q' to q.

Proof.

$\Rightarrow$ According to the homomorphism test, one just needs to show that $(a_{x_1}, \ldots, a_{x_j}) \in q'(I_q)$ where $\mathbf{x} = (x_1, \ldots, x_j)$.

By construction of I_q, $(a_{x_1}, \ldots, a_{x_j}) \in q(I_q)$. As $q \sqsubseteq q'$, we also have $(a_{x_1}, \ldots, a_{x_j}) \in q'(I_q)$. $\square$

Minimal query

Definition

A conjunctive query is minimal if it has a minimal number of atoms among all equivalent conjunctive queries.

Minimal query

Definition

A conjunctive query is minimal if it has a minimal number of atoms among all equivalent conjunctive queries.

- Translation of a CQ to an algebra query: if there are n atoms, we obtain $\leq n - 1$ joins
- Joins are the most costly operations of the relational algebra (bar cross products)
- Finding a minimal query amounts to global optimization

Procédure d'obtention d'une requête minimale

Proposition ([Chandra and Merlin, 1977])

Soit q une CQ. Alors il existe une requête q' équivalente à q obtenue en enlevant des atomes à q qui est minimale.

Procédure d'obtention d'une requête minimale

Proposition ([Chandra and Merlin, 1977])

Soit q une CQ. Alors il existe une requête q' équivalente à q obtenue en enlevant des atomes à q qui est minimale.

Proof.

Soit q'' une requête minimale équivalente à q.

Procédure d'obtention d'une requête minimale

Proposition ([Chandra and Merlin, 1977])

Soit q une CQ. Alors il existe une requête q' équivalente à q obtenue en enlevant des atomes à q qui est minimale.

Proof.

Soit q'' une requête minimale équivalente à q.
Comme $q'' \sqsubseteq q$ et $q \sqsubseteq q''$, par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q'' et ψ de q'' dans q.

Procédure d'obtention d'une requête minimale

Proposition ([Chandra and Merlin, 1977])

Soit q une CQ. Alors il existe une requête q' équivalente à q obtenue en enlevant des atomes à q qui est minimale.

Proof.

Soit q'' une requête minimale équivalente à q.
Comme $q'' \sqsubseteq q$ et $q \sqsubseteq q''$, par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q'' et ψ de q'' dans q.
Alors $h = \psi \circ \varphi$ est un homomorphisme de q dans lui-même.
Soit q' l'ensemble des atomes de q qui sont images par h d'un atome de q.

Procédure d'obtention d'une requête minimale

Proposition ([Chandra and Merlin, 1977])

Soit q une CQ. Alors il existe une requête q' équivalente à q obtenue en enlevant des atomes à q qui est minimale.

Proof.

Soit q'' une requête minimale équivalente à q.
Comme $q'' \sqsubseteq q$ et $q \sqsubseteq q''$, par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q'' et ψ de q'' dans q.
Alors $h = \psi \circ \varphi$ est un homomorphisme de q dans lui-même.
Soit q' l'ensemble des atomes de q qui sont images par h d'un atome de q.
Comme ψ est un homomorphisme de q'' dans q', on a $|q''| \geq |q'|$ où $|\cdot|$ dénote le nombre d'atomes d'une requête. Comme q'' est minimale, q' est également minimale. □

Unicité de la requête minimale

Proposition ([Chandra and Merlin, 1977])

Soient q, q' deux CQ minimales équivalentes. Alors il existe un isomorphisme de q dans q'.

Unicité de la requête minimale: Preuve

Par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q' et ψ de q' dans q.

Unicité de la requête minimale: Preuve

Par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q' et ψ de q' dans q.
q et q' étant toutes deux minimales, tous les atomes de q' sont images par φ d'un atome de q, et réciproquement.

Unicité de la requête minimale: Preuve

Par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q' et ψ de q' dans q.
q et q' étant toutes deux minimales, tous les atomes de q' sont images par φ d'un atome de q, et réciproquement.
$h = \psi \circ \varphi$ est un homomorphisme de q vers lui-même, et tous les atomes de q sont images par h d'un atome de q; en particulier, toutes les variables de q sont dans l'image de h, et h est donc bijectif. h^{-1} étant également un homomorphisme, h est un isomorphisme de q dans q.

Unicité de la requête minimale: Preuve

Par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q' et ψ de q' dans q.
q et q' étant toutes deux minimales, tous les atomes de q' sont images par φ d'un atome de q, et réciproquement.
$h = \psi \circ \varphi$ est un homomorphisme de q vers lui-même, et tous les atomes de q sont images par h d'un atome de q; en particulier, toutes les variables de q sont dans l'image de h, et h est donc bijectif. h^{-1} étant également un homomorphisme, h est un isomorphisme de q dans q.
Comme h est bijective, φ est injective et ψ est surjective; φ et ψ jouant des rôles analogues, on en déduit que les deux fonctions sont bijectives.

Unicité de la requête minimale: Preuve

Par le théorème d'homomorphisme, il existe deux homomorphismes φ de q dans q' et ψ de q' dans q.
q et q' étant toutes deux minimales, tous les atomes de q' sont images par φ d'un atome de q, et réciproquement.
$h = \psi \circ \varphi$ est un homomorphisme de q vers lui-même, et tous les atomes de q sont images par h d'un atome de q; en particulier, toutes les variables de q sont dans l'image de h, et h est donc bijectif. h^{-1} étant également un homomorphisme, h est un isomorphisme de q dans q.
Comme h est bijective, φ est injective et ψ est surjective; φ et ψ jouant des rôles analogues, on en déduit que les deux fonctions sont bijectives.
φ est donc un homomorphisme bijectif de q dans q', et $\varphi^{-1} = h^{-1} \circ \psi$ est un homomorphisme comme composition de deux homomorphismes. Donc φ est un isomorphisme. □

Minimization algorithm

Apply the following procedure to minimize a query:

> *For every atom of the query, test if there exists an equivalent query not containing this atom, and thus if there exists a homomorphism sending this atom to another atom of the query. If so, delete it, and continue until obtaining an equivalent minimal query.*

Aspects de complexité

Proposition

Soient q, q' deux CQ. Ces problèmes sont NP-complets:

(a) déterminer si $q \sqsubseteq q'$

(b) déterminer si $q \equiv q'$

(c) déterminer si q est non minimale

Proof.

- Réduction PTIME de (a) vers (b) ($q \sqsubseteq q'$ ssi $q \equiv q \wedge q'$)
- Réduction PTIME de (b) vers (a) (pour tester si $q \sqsubseteq q'$ et $q' \sqsubseteq q$, si q et q' sont connectées et non booléennes, on peut le faire avec un seul test $q \wedge \bar{q}' \sqsubseteq q' \wedge \bar{q}$ où $\bar{q}$ et $\bar{q}'$ sont des copies de q, q' avec de nouvelles variables; généralisation au cas non-connecté ou booléen en exercice)
- (c) : $|q|$ tests de (b)

Il suffit donc de montrer (a) dans NP et (a) et (c) NP-difficiles.

Inclusion dans NP: Preuve

Soient deux CQ q et q'.

Pour déterminer si $q \sqsubseteq q'$:

- Proposer de manière non-déterministe une fonction des variables de q' vers q (proposition de taille polynomiale en la taille de q, q').
- Vérifier en temps polynomial que c'est un homomorphisme (ce qui implique que $q \sqsubseteq q'$ par le théorème d'homomorphisme); si oui, accepter.

Cette procédure est un algorithme non-déterministe en temps polynomial.

Inclusion / non-minimalité NP-difficile : Preuve

On réduit depuis 3-coloriabilité, un problème NP-difficile.

Inclusion / non-minimalité NP-difficile : Preuve

On réduit depuis 3-coloriabilité, un problème NP-difficile.

Soit un graphe non-orienté connexe $G = (N, E)$ avec $N = \{x_1 \dots x_n\}$. On construit une requête booléenne q_G avec variables $N \cup \{r, v, b\}$ dont les atomes sont les suivants:

- $E(x, y)$ pour chaque $\{x, y\} \in E$
- $N_i(x_i)$ pour chaque $x_i \in N$
- $R(r)$, $V(v)$, $B(b)$
- $E(r, v), E(v, r), E(r, b), E(b, r), E(b, v), E(v, b)$
- $N_i(r)$, $N_i(v)$, $N_i(b)$ pour chaque $1 \leq i \leq n$

Alors $q_G \setminus \{N_1(x_1)\} \sqsubseteq q_G$ ssi q_G est non-minimal ssi G est 3-coloriable.

Cette réduction est PTIME. □

Remarque sur la complexité

NP-difficile... en les requêtes. Les requêtes peuvent être suffisamment petites pour qu'un algorithme exponentiel ne soit pas un problème.

Bag semantics [Chaudhuri and Vardi, 1993]

- In practice, RDBMSs implement a bag semantics
- Two queries in bag semantics are equivalent if and only if they are isomorphic (intuitively, because two similar but non isomorphic queries can introduce a different number of results)
- Query containment: Π_2^P-hard claimed in [Chaudhuri and Vardi, 1993] (but not proved, and claim retracted since!). Decidability (and precise complexity if decidable): open!

En pratique dans les SGBD

- L'algorithme de minimisation n'est pas implémenté, pour de nombreuses (plus ou moins bonnes) raisons:
  - La plupart des requêtes ont une sémantique multi-ensembliste
  - Algorithme exponentiel
  - L'algorithme de minimisation ne marche que pour les CQ (donc pas de négation, d'agrégation, d'union...)
  - Les SGBD sont très conservateurs dans leur approche de l'optimisation de requêtes, ne veulent pas causer de régression
- À la place, les SGBD utilisent des optimisations locales des plans d'exécution basées sur des statistiques (cf. cours ultérieur)
- Exemple assez frappant du fossé entre théorie des BD et BD systèmes

Plan

Conclusions

- Conjunctive queries: A nice fragment of the relational calculus, with corresponding fragments of the relational algebra and SQL
- Query evaluation can be even more efficient when queries are acyclic!
- Static analysis "only" (co)NP-complete, reasonable to implement

Bibliography I

Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90, 1977. doi: 10.1145/800105.803397. URL `http://doi.acm.org/10.1145/800105.803397`.

Surajit Chaudhuri and Moshe Y. Vardi. Optimization of *Real* conjunctive queries. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*, pages 59–70, 1993. doi: 10.1145/153850.153856. URL `http://doi.acm.org/10.1145/153850.153856`.

Bibliography II

Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984. doi: 10.1137/0213035. URL `http://dx.doi.org/10.1137/0213035`.

Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.