Databases

Conjunctive Queries

Pierre Senellart

18 February 2026

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

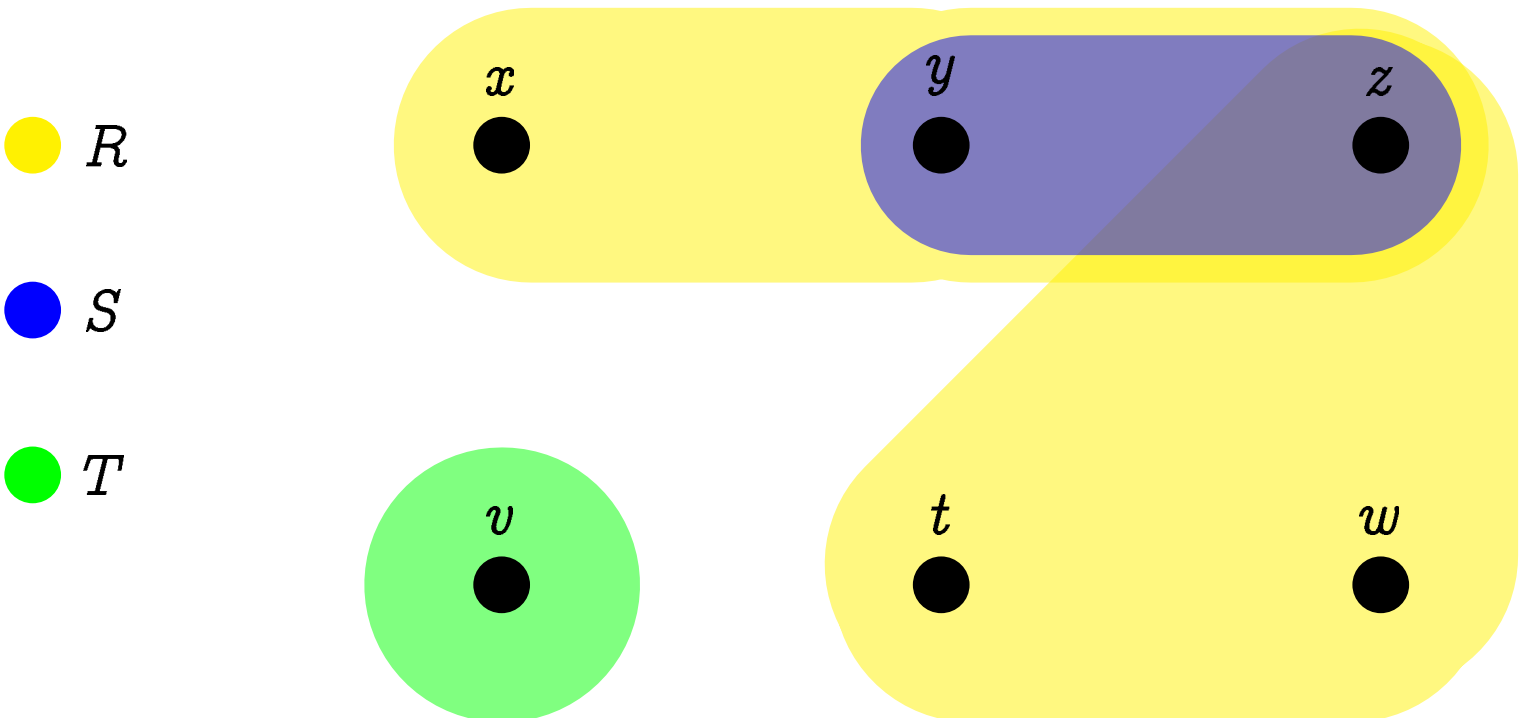
- A hypergraph $\mathcal{H}$ is a pair (V, E) where V is a finite set of vertices, $E \subseteq 2^V$ is a set of hyperedges
- Often useful to consider labeled hypergraphs, where each hyperedge is associated with a label in $\mathcal{L}$
- Sometimes, we allow a multiset of labels instead of a single label
- 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
- If the query includes constants, special care needs to be given (simplest: each constant can be encoded by a fresh unary predicate $C_c(\cdot)$ with fact $C_c(c)$)
- 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 hyperedges) that connects the two vertices
- 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 (not 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 (not 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)

Beyond acyclicity: bounded width

- Acyclic CQs admit a join tree $\Rightarrow$ very efficient evaluation (Yannakakis).
- More generally, if the query has bounded width (various notions of width, e.g., bounded treewidth of the primal graph), evaluation can be done by dynamic programming on a decomposition
- This yields fixed-parameter tractability in the width parameter
- Connects to structural parameters studied in graph theory and graph algorithms

Plan

Homomorphism

Definition

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

- $\varphi(\mathbf{x}_i) = \mathbf{x}'_i$ for all i
- 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 every 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 $R(a_{z_{i1}}, \ldots, a_{z_{ik}})$ corresponding to each atom $R(z_{i1}, \ldots, z_{ik})$ of q

Example: instance associated to a query

Consider the Boolean CQ:

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

The associated instance I_q is defined as follows:

- Active domain:

$$\{a_x, a_y, a_z, a_t, a_v, a_w\}$$

- Facts:

$$R(a_x, a_y, a_z)$$
$$R(a_z, a_t, a_w)$$
$$S(a_y, a_z)$$
$$T(a_v)$$

Each variable becomes a fresh constant.

Homomorphism test

Proposition

For all CQs of arity j

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

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 of arity j

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

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 $\psi : \mathbf{x} \cup \mathbf{y} \to \{a_z \mid z \in \mathbf{x}' \cup \mathbf{y}'\}$:

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

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 of arity j

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

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:

- $\varphi(\mathbf{x}) = (x'_1, \ldots, x'_j)$
- $\forall 1 \leqslant i \leqslant n$, $R_i(\varphi(\mathbf{z}_i))$ 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 of arity j

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

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

Homomorphism theorem

Theorem ([Chandra and Merlin, 1977])

For all CQs of arity j

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

$q \sqsubseteq q'$ 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 of arity j

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

$q \sqsubseteq q'$ 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 $\leqslant n - 1$ joins
- Joins are the most costly operations of the relational algebra (except cross products)
- Finding a minimal query amounts to global optimization

Procedure to obtain a minimal query

Proposition ([Chandra and Merlin, 1977])

Let q be a CQ. Then there exists a query q' equivalent to q obtained by removing atoms from q that is minimal.

Procedure to obtain a minimal query

Proposition ([Chandra and Merlin, 1977])

Let q be a CQ. Then there exists a query q' equivalent to q obtained by removing atoms from q that is minimal.

Proof.

Let q'' be a minimal query equivalent to q.

Procedure to obtain a minimal query

Proposition ([Chandra and Merlin, 1977])

Let q be a CQ. Then there exists a query q' equivalent to q obtained by removing atoms from q that is minimal.

Proof.

Let q'' be a minimal query equivalent to q.
Since $q'' \sqsubseteq q$ and $q \sqsubseteq q''$, by the homomorphism theorem, there exist two homomorphisms φ from q to q'' and ψ from q'' to q.

Procedure to obtain a minimal query

Proposition ([Chandra and Merlin, 1977])

Let q be a CQ. Then there exists a query q' equivalent to q obtained by removing atoms from q that is minimal.

Proof.

Let q'' be a minimal query equivalent to q.
Since $q'' \sqsubseteq q$ and $q \sqsubseteq q''$, by the homomorphism theorem, there exist two homomorphisms φ from q to q'' and ψ from q'' to q.
Then $h = \psi \circ \varphi$ is a homomorphism from q to itself. Let q' be the set of atoms of q that are images under h of an atom of q.

Procedure to obtain a minimal query

Proposition ([Chandra and Merlin, 1977])

Let q be a CQ. Then there exists a query q' equivalent to q obtained by removing atoms from q that is minimal.

Proof.

Let q'' be a minimal query equivalent to q.
Since $q'' \sqsubseteq q$ and $q \sqsubseteq q''$, by the homomorphism theorem, there exist two homomorphisms φ from q to q'' and ψ from q'' to q. Then $h = \psi \circ \varphi$ is a homomorphism from q to itself. Let q' be the set of atoms of q that are images under h of an atom of q. Since ψ is a homomorphism from q'' to q', we have $|q''| \geqslant |q'|$, where $|\cdot|$ denotes the number of atoms of a query. As q'' is minimal, q' is also minimal. □

Uniqueness of the minimal query

Proposition ([Chandra and Merlin, 1977])

Let q and q' be two equivalent minimal CQs. Then there exists an isomorphism from q to q'.

Uniqueness of the minimal query: Proof

By the homomorphism theorem, there exist two homomorphisms φ from q to q' and ψ from q' to q.

Uniqueness of the minimal query: Proof

By the homomorphism theorem, there exist two homomorphisms φ from q to q' and ψ from q' to q.
Since q and q' are both minimal, all atoms of q' are images under φ of an atom of q, and conversely.

Uniqueness of the minimal query: Proof

By the homomorphism theorem, there exist two homomorphisms φ from q to q' and ψ from q' to q.
Since q and q' are both minimal, all atoms of q' are images under φ of an atom of q, and conversely.
$h = \psi \circ \varphi$ is a homomorphism from q to itself, and all atoms of q are images under h of an atom of q; in particular, all variables of q are in the image of h, hence h is bijective. Since h^{-1} is also a homomorphism, h is an isomorphism of q to itself.

Uniqueness of the minimal query: Proof

By the homomorphism theorem, there exist two homomorphisms φ from q to q' and ψ from q' to q.
Since q and q' are both minimal, all atoms of q' are images under φ of an atom of q, and conversely.
$h = \psi \circ \varphi$ is a homomorphism from q to itself, and all atoms of q are images under h of an atom of q; in particular, all variables of q are in the image of h, hence h is bijective. Since h^{-1} is also a homomorphism, h is an isomorphism of q to itself.
As h is bijective, φ is injective and ψ is surjective; since φ and ψ play symmetric roles, both functions are bijective.

Uniqueness of the minimal query: Proof

By the homomorphism theorem, there exist two homomorphisms φ from q to q' and ψ from q' to q.
Since q and q' are both minimal, all atoms of q' are images under φ of an atom of q, and conversely.
$h = \psi \circ \varphi$ is a homomorphism from q to itself, and all atoms of q are images under h of an atom of q; in particular, all variables of q are in the image of h, hence h is bijective. Since h^{-1} is also a homomorphism, h is an isomorphism of q to itself.
As h is bijective, φ is injective and ψ is surjective; since φ and ψ play symmetric roles, both functions are bijective.
Thus φ is a bijective homomorphism from q to q', and $\varphi^{-1} = h^{-1} \circ \psi^{-1}$ is a homomorphism as a composition of two homomorphisms. Therefore φ is an isomorphism. □

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.*

Complexity aspects

Proposition

Let q, q' be two CQs. The following problems are NP-complete:

(a) deciding whether $q \sqsubseteq q'$

(b) deciding whether $q \equiv q'$

(c) deciding whether q is non-minimal

Proof.

- PTIME reduction from (a) to (b) ($q \sqsubseteq q'$ iff $q \equiv q \wedge q'$)
- PTIME reduction from (b) to (a) (to test whether $q \sqsubseteq q'$ and $q' \sqsubseteq q$; if q and q' are connected non-Boolean, this can be done by testing $q \wedge \bar{q}' \sqsubseteq q' \wedge \bar{q}$, where $\bar{q}$ and $\bar{q}'$ are copies of q, q' with fresh variables; generalization left as exercise)
- (c): perform $|q|$ tests of (b)
- Suffices to show: (a) is in NP and (a), (c) are NP-hard. □

Inclusion in NP: Proof

Let q and q' be two CQs.

To determine whether $q \sqsubseteq q'$:

- Nondeterministically guess a function from the variables of q' to those of q (description size polynomial in the size of q and q').
- Verify in polynomial time that it is a homomorphism (which implies $q \sqsubseteq q'$ by the homomorphism theorem); if so, accept.

This is a nondeterministic polynomial-time algorithm.

NP-hardness of inclusion / non-minimality: Proof

We reduce from 3-colorability, an NP-hard problem.

NP-hardness of inclusion / non-minimality: Proof

We reduce from 3-colorability, an NP-hard problem.

Let $G = (N, E)$ be a connected undirected graph with $N = \{x_1, \ldots, x_n\}$. We construct a Boolean query q_G with variables $N \cup \{r, v, b\}$ whose atoms are:

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

Then

$$q_G \setminus \{N_1(x_1)\} \sqsubseteq q_G$$

iff q_G is non-minimal iff G is 3-colorable.

This reduction is computable in PTIME. □

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!

Remark on complexity

NP-hard... in the query size. Queries are often sufficiently small in practice that an exponential algorithm is not necessarily problematic.

In practice in DBMSs

- The minimization algorithm is not implemented, for many (more or less good) reasons:
  - Most queries use bag semantics
  - The algorithm is exponential
  - It applies only to CQs (no negation, aggregation, union, ...)
  - DBMSs are very conservative in query optimization to avoid regressions
- Instead, DBMSs rely on local optimizations of execution plans based on statistics (see later lecture)
- A striking example of the gap between database theory and database systems

Plan

Conclusions

- Conjunctive queries: A nice fragment of the relational calculus, with corresponding fragments of the relational algebra and SQL
- Boolean CQ evaluation: existence of a homomorphism from the query to the instance
- 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.