Query Processing for Non-traditional Applications

CS848 Spring 2013

Cheriton School of CS

The Chase and Duplicates

Outline

Consider how to find query plans for user queries.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.
- Conjunctive queries and the chase.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.
- Conjunctive queries and the chase.
- Post-processing.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.
- Conjunctive queries and the chase.
- Post-processing.
- Positive queries and the chase.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.
- Conjunctive queries and the chase.
- Post-processing.
- Positive queries and the chase.
- First-order queries and interpolation.

Outline

Consider how to find query plans for user queries.

One can find query plans by doing the following.

1. Finding a rewriting over access paths of the user query that satisfies binding pattern requirements.
2. Post-processing to consider assignments and comparisons, duplicate elimination, and cut insertion.

Topics

- Beth definability.
- Conjunctive queries and the chase.
- Post-processing.
- Positive queries and the chase.
- First-order queries and interpolation.
- Examples relating to ACME's PAYROLL system.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Assume $\langle S_L \cup S_P, \Sigma \rangle$ is a physical design with access paths S_A, and Q is a user query over S_L.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design with access paths $\mathsf{S_A}$, and Q is a user query over $\mathsf{S_L}$.

Q is *(Beth) definable in* $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ if the following condition is satisfied:

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design with access paths $\mathsf{S_A}$, and Q is a user query over $\mathsf{S_L}$.

Q is *(Beth) definable in* $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ if the following condition is satisfied:

$Q(\mathcal{I}_1) = Q(\mathcal{I}_2)$ for all interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ for which the following hold.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design with access paths $\mathsf{S_A}$, and Q is a user query over $\mathsf{S_L}$.

Q is *(Beth) definable in* $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ if the following condition is satisfied:

$Q(\mathcal{I}_1) = Q(\mathcal{I}_2)$ for all interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ for which the following hold.

1. Both $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy Σ.

Existence of Query Plans

A necessary condition for the *existence* of a query plan underlies the following (equivalent) problems.

- Is the data represented by the available access paths *sufficient* to answer the user query?
- Is the answer to the query *entirely determined* by the interpretation of the available access paths?

The problems correspond to a determination of *Beth definability* if interpretations of access paths can be infinite.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design with access paths $\mathsf{S_A}$, and Q is a user query over $\mathsf{S_L}$.

Q is *(Beth) definable in* $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ if the following condition is satisfied:

$Q(\mathcal{I}_1) = Q(\mathcal{I}_2)$ for all interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ for which the following hold.

1. Both $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy Σ.
2. $(R)^{\mathcal{I}_1} = (R)^{\mathcal{I}_2}$ for all non-logical parameters $R/m/n \in \mathsf{S_A}$.

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, S_A is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, S_A is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Also assume Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in S_A are uniformly renamed.

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, S_A is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Also assume Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in S_A are uniformly renamed. (E.g., each $R \in S - S_A$ is replaced by R^*.)

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, $\mathsf{S_A}$ is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Also assume Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in $\mathsf{S_A}$ are uniformly renamed. (E.g., each $R \in \mathsf{S} - \mathsf{S_A}$ is replaced by R^*.)

Q is Beth definable if and only if $(\Sigma \cup \Sigma^*) \models (Q \rightarrow Q^*)$.

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, $\mathsf{S_A}$ is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Also assume Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in $\mathsf{S_A}$ are uniformly renamed. (E.g., each $R \in \mathsf{S} - \mathsf{S_A}$ is replaced by R^*.)

Q is Beth definable if and only if $(\Sigma \cup \Sigma^*) \models (Q \rightarrow Q^*)$.

Definability can therefore serve as an approximate test to determine if the data *stored* in instances of access paths in a given physical design is sufficient, in principle, to answer the user query.

Existence of Query Plans

Definability of user queries with respect to a physical design can be tested syntactically in an important case.

Assume Q is a user query, S_A is a set of access paths, and Σ is a set of constraints for which each $\psi \in \Sigma$ is domain independent when considered to be user query with no parameters.

Also assume Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in S_A are uniformly renamed. (E.g., each $R \in \mathsf{S} - \mathsf{S}_\mathsf{A}$ is replaced by R^*.)

Q is Beth definable if and only if $(\Sigma \cup \Sigma^*) \models (Q \rightarrow Q^*)$.

Definability can therefore serve as an approximate test to determine if the data *stored* in instances of access paths in a given physical design is sufficient, in principle, to answer the user query.

Failing this condition is strong evidence that the physical design has insufficient material capability to answer the user query.

ACME Case: Compiling with `emp-array0`

Recall our basic physical design for PAYROLL.

$$\Sigma = \{ \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)), \\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z)) \}$$

$$\mathrm{S_A} = \{ \texttt{emp-array0}/3/0 \}$$

ACME Case: Compiling with `emp-array0`

Recall our basic physical design for PAYROLL.

$$\Sigma = \{\ \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)),\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z))\ \}$$

$$\mathrm{S_A} = \{\ \texttt{emp-array0}/3/0\}$$

The left-hand-side of our syntactic test is therefore as follows.

$$\begin{aligned}(\Sigma \cup \Sigma^*) = \{\ & \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)),\\ & \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z)),\\ & \forall x, y, z.(\texttt{employee}^*(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)),\\ & \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}^*(x, y, z))\ \}\end{aligned}$$

ACME Case: Compiling with emp-array0

Recall our basic physical design for PAYROLL.

$$\Sigma = \{\, \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)), \\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z)) \,\}$$

$$\mathrm{S_A} = \{\, \texttt{emp-array0}/3/0 \}$$

The left-hand-side of our syntactic test is therefore as follows.

$$(\Sigma \cup \Sigma^*) = \{\, \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)), \\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z)), \\ \forall x, y, z.(\texttt{employee}^*(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)), \\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}^*(x, y, z)) \,\}$$

... and, for the user query $\texttt{employee}(x, y, z)$, the following holds.

$$(\Sigma \cup \Sigma^*) \models \texttt{employee}(x, y, z) \rightarrow \texttt{employee}^*(x, y, z)$$

ACME Case: Compiling with `emp-array2`

Conversely, Beth definability does not take into account the *binding pattern restrictions* on access paths in terms of mandatory input parameters. Hence, a plan may still not exist.

The same reasoning leads to the conclusion that the following query plan is another candidate plan for user query $\texttt{employee}(x, y, z)$.

$$\texttt{emp-array2}(z, x, y)$$

ACME Case: Compiling with `emp-array2`

Conversely, Beth definability does not take into account the *binding pattern restrictions* on access paths in terms of mandatory input parameters. Hence, a plan may still not exist.

The same reasoning leads to the conclusion that the following query plan is another candidate plan for user query $\texttt{employee}(x, y, z)$.

$$\texttt{emp-array2}(z, x, y)$$

In this case, the input variables of the plan do not correspond to the user query parameters (and the plan therefore fails to qualify as an implementation of the user query).

ACME Case: Compiling with `emp-array2`

Conversely, Beth definability does not take into account the *binding pattern restrictions* on access paths in terms of mandatory input parameters. Hence, a plan may still not exist.

The same reasoning leads to the conclusion that the following query plan is another candidate plan for user query $\texttt{employee}(x, y, z)$.

$$\texttt{emp-array2}(z, x, y)$$

In this case, the input variables of the plan do not correspond to the user query parameters (and the plan therefore fails to qualify as an implementation of the user query).

The opposite holds for the following user query in which the plan input variables now match the query parameters.

$$\texttt{employee}(x, y, z)\{x, z\}$$

Substitution in FOL

Assume $\phi \in$ WFF is a formula, x is a variable and t is a term such that $\mathrm{Fv}(t)$ does not contain variables quantified in ϕ.

Substitution in FOL

Assume $\phi \in$ WFF is a formula, x is a variable and t is a term such that Fv(t) does not contain variables quantified in ϕ.

A *substitution of t for x in ϕ* is the WFF obtained from ϕ by syntactically replacing all free occurrences of x by t

Substitution in FOL

Assume $\phi \in$ WFF is a formula, x is a variable and t is a term such that Fv(t) does not contain variables quantified in ϕ.

A *substitution of t for x in ϕ* is the WFF obtained from ϕ by syntactically replacing all free occurrences of x by t and is written as follows.

$$\phi[t/x]$$

Substitution in FOL

Assume $\phi \in$ WFF is a formula, x is a variable and t is a term such that Fv(t) does not contain variables quantified in ϕ.

A *substitution of t for x in ϕ* is the WFF obtained from ϕ by syntactically replacing all free occurrences of x by t and is written as follows.

$$\phi[t/x]$$

Substitutions can be composed yielding *simultaneous* substitutions, denoted θ, for multiple variables in a formula.

Substitution in FOL

Assume $\phi \in$ WFF is a formula, x is a variable and t is a term such that Fv(t) does not contain variables quantified in ϕ.

A *substitution of* t *for* x *in* ϕ is the WFF obtained from ϕ by syntactically replacing all free occurrences of x by t and is written as follows.

$$\phi[t/x]$$

Substitutions can be composed yielding *simultaneous* substitutions, denoted θ, for multiple variables in a formula.

The *simultaneous application* of all substitutions in θ to the formula ϕ is written in the same way.

$$\phi\theta$$

A Chase Step

Chase procedures are an algorithmic technique for synthesizing query plans for conjunctive queries.

A Chase Step

Chase procedures are an algorithmic technique for synthesizing query plans for conjunctive queries. The main idea is given by the following theorem.

Assume Σ is a given theory in FOL, and also that we are given the following.

1. A conjunctive query free of equality atoms.

$$\exists x_1. \cdots .\exists x_m.\varphi$$

A Chase Step

Chase procedures are an algorithmic technique for synthesizing query plans for conjunctive queries. The main idea is given by the following theorem.

Assume Σ is a given theory in FOL, and also that we are given the following.

1. A conjunctive query free of equality atoms.

$$\exists x_1. \cdots .\exists x_m.\varphi$$

2. A tuple generating dependency (TGD) in Σ.

$$\forall x_1. \cdots .\forall x_i.(\exists x_{i+1}. \cdots .\exists x_j.\phi \rightarrow \exists x_{j+1}. \cdots .\exists x_k.\psi)$$

A Chase Step

Chase procedures are an algorithmic technique for synthesizing query plans for conjunctive queries. The main idea is given by the following theorem.

Assume Σ is a given theory in FOL, and also that we are given the following.

1. A conjunctive query free of equality atoms.

$$\exists x_1.\cdots.\exists x_m.\varphi$$

2. A tuple generating dependency (TGD) in Σ.

$$\forall x_1.\cdots.\forall x_i.(\exists x_{i+1}.\cdots.\exists x_j.\phi \rightarrow \exists x_{j+1}.\cdots.\exists x_k.\psi)$$

3. A substitution θ such that, for each atom ϕ_i in ϕ, $\phi_i\theta$ is an atom in φ.

A Chase Step

Chase procedures are an algorithmic technique for synthesizing query plans for conjunctive queries. The main idea is given by the following theorem.

Assume Σ is a given theory in FOL, and also that we are given the following.

1. A conjunctive query free of equality atoms.

$$\exists x_1.\cdots.\exists x_m.\varphi$$

2. A tuple generating dependency (TGD) in Σ.

$$\forall x_1.\cdots.\forall x_i.(\exists x_{i+1}.\cdots.\exists x_j.\phi \to \exists x_{j+1}.\cdots.\exists x_k.\psi)$$

3. A substitution θ such that, for each atom ϕ_i in ϕ, $\phi_i\theta$ is an atom in φ.

Then the following holds.

$$\Sigma \models (\exists x_1.\cdots.\exists x_m.\varphi) \equiv (\exists x_1.\cdots.\exists x_m.\varphi \wedge (\exists x_{j+1}.\cdots.\exists x_k.\psi)\theta)$$

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1.\cdots.\exists x_m.\varphi \wedge (\exists x_{j+1}.\cdots.\exists x_k.\psi)\theta$$

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1. \cdots .\exists x_m.\varphi \wedge (\exists x_{j+1}. \cdots .\exists x_k.\psi)\theta$$

The formula is easily converted to a conjunctive query.

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1.\cdots.\exists x_m.\varphi \wedge (\exists x_{j+1}.\cdots.\exists x_k.\psi)\theta$$

The formula is easily converted to a conjunctive query. (Use a standard equivalence for FO formulae that allows existential quantification to commute with conjunction after renaming the bound variable if necessary.)

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1. \cdots .\exists x_m.\varphi \wedge (\exists x_{j+1}. \cdots .\exists x_k.\psi)\theta$$

The formula is easily converted to a conjunctive query. (Use a standard equivalence for FO formulae that allows existential quantification to commute with conjunction after renaming the bound variable if necessary.)

This allows syntactically equating conjunctive queries with sets of their atoms in which parameters and remaining free variables are distinguished.

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1. \cdots .\exists x_m.\varphi \wedge (\exists x_{j+1}. \cdots .\exists x_k.\psi)\theta$$

The formula is easily converted to a conjunctive query. (Use a standard equivalence for FO formulae that allows existential quantification to commute with conjunction after renaming the bound variable if necessary.)

This allows syntactically equating conjunctive queries with sets of their atoms in which parameters and remaining free variables are distinguished.

The PAYROLL query to *obtain the employee numbers x of all employees with a given salary z* is given as follows with this alternative syntax.

$$\{\texttt{employee}(x, y, z)\}$$

Conjunctive User Queries as Atom Sets

Recall the formula resulting from a chase step.

$$\exists x_1. \cdots .\exists x_m.\varphi \wedge (\exists x_{j+1}. \cdots .\exists x_k.\psi)\theta$$

The formula is easily converted to a conjunctive query. (Use a standard equivalence for FO formulae that allows existential quantification to commute with conjunction after renaming the bound variable if necessary.)

This allows syntactically equating conjunctive queries with sets of their atoms in which parameters and remaining free variables are distinguished.

The PAYROLL query to *obtain the employee numbers x of all employees with a given salary z* is given as follows with this alternative syntax.

$$\{\texttt{employee}(x, y, z)\}$$

Two applications of a chase step obtains the following.

$$\{\texttt{employee}(x, y, z), \texttt{employee0}(z, x, y), \texttt{employee}^*(x, y, z)\}$$

The Chase

Assume a set of TGDs is given by Σ, and that Q is a conjunctive user query given in the alternative syntax.

The Chase

Assume a set of TGDs is given by Σ, and that Q is a conjunctive user query given in the alternative syntax.

A repeated application of chase steps over all dependencies in Σ is called the *chase of Q with* Σ, and is written $\text{Chase}_\Sigma(Q)$.

The Chase

Assume a set of TGDs is given by Σ, and that Q is a conjunctive user query given in the alternative syntax.

A repeated application of chase steps over all dependencies in Σ is called the *chase of Q with Σ*, and is written $\text{Chase}_{\Sigma}(Q)$.

Observation: The application of chase steps is confluent (up to renaming of variables).

The Chase

Assume a set of TGDs is given by Σ, and that Q is a conjunctive user query given in the alternative syntax.

A repeated application of chase steps over all dependencies in Σ is called the *chase of Q with* Σ, and is written $\text{Chase}_\Sigma(Q)$.

Observation: The application of chase steps is confluent (up to renaming of variables).

This implies any fair sequence of applying the individual chase steps for TGDs in Σ leads to the same (in the limit possibly infinite) expansion of the original conjunctive query.

An Abstraction of Conjunctive Plans

Assume $\langle S_L \cup S_P, \Sigma \rangle$ is a physical design.

.

An Abstraction of Conjunctive Plans

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design. Also assume $\{\psi_{i_1}, \ldots, \psi_{i_n}\}$ is a conjunctive user query Q given in our alternative syntax.

.

An Abstraction of Conjunctive Plans

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design. Also assume $\{\psi_{i_1}, \ldots, \psi_{i_n}\}$ is a conjunctive user query Q given in our alternative syntax.

If there exists a sequence

$$(\psi_1(= P_1(x_{1,1}, \ldots, x_{1,n_1})), \ldots, \psi_n(= P_n(x_{n,1}, \ldots, x_{n,m_n})))$$

for which each P_i occurs in $\mathsf{S_A}$ and either of the following conditions hold for each x_{i_j} occurring in a parameter position:

1. $x_{i,j}$ is a parameter of Q and
2. $x_{i,j}$ occurs in a non-parameter position of some P_k where $k < i$,

.

An Abstraction of Conjunctive Plans

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design. Also assume $\{\psi_{i_1}, \ldots, \psi_{i_n}\}$ is a conjunctive user query Q given in our alternative syntax.

If there exists a sequence

$$(\psi_1(= P_1(x_{1,1}, \ldots, x_{1,n_1})), \ldots, \psi_n(= P_n(x_{n,1}, \ldots, x_{n,m_n})))$$

for which each P_i occurs in $\mathsf{S_A}$ and either of the following conditions hold for each x_{i_j} occurring in a parameter position:

1. $x_{i,j}$ is a parameter of Q and
2. $x_{i,j}$ occurs in a non-parameter position of some P_k where $k < i$,

then there is a procedure, denoted $\mathsf{Qp}(\cdot)$, for obtaining a query plan from $(\psi_1, \ldots, \psi_n)$ where $\Sigma \models Q \lhd \mathsf{Qp}((\psi_1, \ldots, \psi_n))$.

An Abstraction of Conjunctive Plans

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design. Also assume $\{\psi_{i_1}, \ldots, \psi_{i_n}\}$ is a conjunctive user query Q given in our alternative syntax.

If there exists a sequence

$$(\psi_1(= P_1(x_{1,1}, \ldots, x_{1,n_1})), \ldots, \psi_n(= P_n(x_{n,1}, \ldots, x_{n,m_n})))$$

for which each P_i occurs in $\mathsf{S_A}$ and either of the following conditions hold for each x_{i_j} occurring in a parameter position:

1. $x_{i,j}$ is a parameter of Q and
2. $x_{i,j}$ occurs in a non-parameter position of some P_k where $k < i$,

then there is a procedure, denoted $\mathrm{Qp}(\cdot)$, for obtaining a query plan from $(\psi_1, \ldots, \psi_n)$ where $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_n))$.[1]

[1] Defining this procedure is an easy but worthwhile exercise.

A Chase Procedure for Plan Synthesis

Input: A conjunctive user query $Q(= \{\varphi_1, \ldots, \varphi_k\})$ and a set Σ of TGDs.

Result: A (possibly infinite) sequence $S = (\psi_1, \psi_2, \ldots)$ satisfying binding pattern requirements, and such that

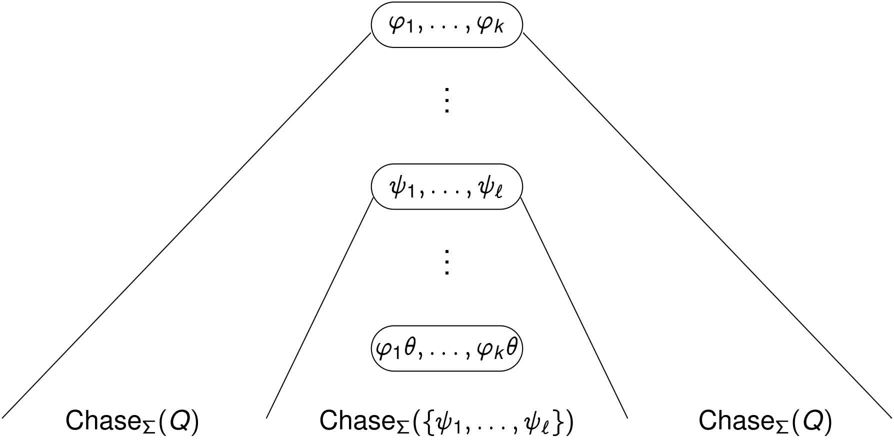
$$\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$$

for all finite prefixes $(\psi_1, \ldots, \psi_\ell)$ of S where $n \leq \ell$ if *success*.

1. Initialize: $S \leftarrow (\,)$; $G \leftarrow \mathrm{Chase}_\Sigma(S)$; $n \leftarrow 0$; *success* $\leftarrow$ false.
2. If there exists $\psi \in G$ for which $S \mid (\psi)$ satisfies binding pattern requirements, then $S \leftarrow S \mid (\psi)$.
3. If there exists θ over the existential variables of Q for which $Q\theta \subseteq (\mathrm{Chase}_\Sigma(\mathrm{Setof}(S)) \cap G)$, then *success* $\leftarrow$ true. Otherwise $n \leftarrow n + 1$.
4. Resume at Step 2.

A Chase Procedure for Plan Synthesis (cont'd)

Assuming $(\psi_1, \ldots, \psi_\ell)$ satisfies binding pattern requirements.



ACME Case: A Plan Using `emp-array0`

Assume our basic physical design for PAYROLL.

$$\Sigma = \{\ \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)),\\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z))\ \}$$

$$\mathsf{S_A} = \{\ \texttt{emp-array0}/3/0\}$$

ACME Case: A Plan Using `emp-array0`

Assume our basic physical design for PAYROLL.

$$\Sigma = \{ \forall x, y, z.(\texttt{employee}(x, y, z) \rightarrow \texttt{emp-array0}(z, x, y)), \\ \forall x, y, z.(\texttt{emp-array0}(z, x, y) \rightarrow \texttt{employee}(x, y, z)) \ \}$$

$$\mathsf{S_A} = \{ \texttt{emp-array0}/3/0 \}$$

Also assume Q is the above user query to *obtain the employee numbers x of all employees with a given salary z*.

$$\{\texttt{employee}(\color{red}{x}, y, \color{blue}{z})\}$$

ACME Case: A Plan Using `emp-array0` (cont'd)

An execution of the chase procedure is successful where $n = 1$ and with other results as follows.

$$Q = \{\mathtt{employee}(x, y, z)\}$$

$$S = (\mathtt{employee0}(z, x, y))$$

$$G = \{\mathtt{employee}(x, y, z), \mathtt{employee0}(z, x, y)\}$$

ACME Case: A Plan Using `emp-array0` (cont'd)

An execution of the chase procedure is successful where $n = 1$ and with other results as follows.

$$Q = \{\texttt{employee}(x, y, z)\}$$

$$S = (\texttt{employee0}(z, x, y))$$

$$G = \{\texttt{employee}(x, y, z), \texttt{employee0}(z, x, y)\}$$

The successful result is a consequence of the following.

1. $(\text{Chase}_{\Sigma}(\text{Setof}(S)) \cap G) = \{\texttt{employee}(x, y, z), \texttt{employee0}(z, x, y)\}$.

ACME Case: A Plan Using `emp-array0` (cont'd)

An execution of the chase procedure is successful where $n = 1$ and with other results as follows.

$$Q = \{\mathtt{employee}(x, y, z)\}$$

$$S = (\mathtt{employee0}(z, x, y))$$

$$G = \{\mathtt{employee}(x, y, z), \mathtt{employee0}(z, x, y)\}$$

The successful result is a consequence of the following.

1. $(\mathrm{Chase}_\Sigma(\mathrm{Setof}(S)) \cap G) = \{\mathtt{employee}(x, y, z), \mathtt{employee0}(z, x, y)\}$.
2. $Q\theta \subseteq (\mathrm{Chase}_\Sigma(\mathrm{Setof}(S)) \cap G)$, where $\theta = [y/y]$.

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

(*plan synthesis*) $\Sigma \models Q \lhd \text{Qp}((\psi_1, \ldots, \psi_\ell))$.

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

(*plan synthesis*) $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$.

(*backchase*) If

1. $\mathrm{Fv}(Q) = \mathrm{Fv}(\{\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell\})$

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

(*plan synthesis*) $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$.

(*backchase*) If

1. $\mathrm{Fv}(Q) = \mathrm{Fv}(\{\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell\})$,
2. $(\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell)$ satisfies binding pattern requirements

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

(*plan synthesis*) $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$.

(*backchase*) If

1. $\mathrm{Fv}(Q) = \mathrm{Fv}(\{\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell\})$,
2. $(\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell)$ satisfies binding pattern requirements, and
3. $\Sigma \models (\psi_1 \wedge \cdots \wedge \psi_{i-1} \wedge \psi_{i+1} \wedge \cdots \wedge \psi_\ell) \rightarrow \psi_i$

Successful Chase and Backchase

Assume a set of TGDs is given by Σ, that Q is a conjunctive user query, and that $S = (\psi_1, \psi_2, \ldots)$ is a sequence computed by the chase procedure.

The following hold for any $\ell \geq n$ if the procedure is successful.

(*plan synthesis*) $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$.

(*backchase*) If

1. $\mathrm{Fv}(Q) = \mathrm{Fv}(\{\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell\})$,
2. $(\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell)$ satisfies binding pattern requirements, and
3. $\Sigma \models (\psi_1 \wedge \cdots \wedge \psi_{i-1} \wedge \psi_{i+1} \wedge \cdots \wedge \psi_\ell) \to \psi_i$

then $\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell))$.

Equality Generating Dependencies (EGDs)

Recall that an EGD has the following form.

$$\forall x_1 . \cdots . \forall x_k . \phi \to x_i \approx x_j$$

Such a dependency can also be used in a chase step provided the formula Q resulting from such a step is immediately transformed as follows.

For all equality atoms of the form $x_i \approx x_j$ in Q, repeatedly perform the following steps until none changes Q.

1. If x_j occurs prior to x_i in a lexicographic ordering, then replace $x_i \approx x_j$ by $x_j \approx x_i$ in Q.
2. If x_j is bound in Q, replace x_j by x_i in Q and remove both the equality atom and the existential quantifier for x_j from Q.
3. If x_i is bound in Q, replace x_i by x_j in Q and remove both the equality atom and the existential quantifier for x_i from Q.
4. If both x_i and x_j are free in Q, replace x_j by x_i in Q except in the equality atom $x_i \approx x_j$, and keep this atom in Q if not already there.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Q' will in general require a top-level *duplicate elimination* operation to ensure that the query plan implements Q.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Q' will in general require a top-level *duplicate elimination* operation to ensure that the query plan implements Q.

Adding duplicate elimination unconditionally to any query plan is clearly unacceptable on performance grounds.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Q' will in general require a top-level *duplicate elimination* operation to ensure that the query plan implements Q.

Adding duplicate elimination unconditionally to any query plan is clearly unacceptable on performance grounds. We now consider how to rewrite query plans to reduce and possibly avoid the overhead that this entails.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Q' will in general require a top-level *duplicate elimination* operation to ensure that the query plan implements Q.

Adding duplicate elimination unconditionally to any query plan is clearly unacceptable on performance grounds. We now consider how to rewrite query plans to reduce and possibly avoid the overhead that this entails.

Assume $\langle \mathrm{S_L} \cup \mathrm{S_P}, \Sigma \rangle$ is a physical design, and that Q_1 and Q_2 are a pair of query plans over the design.

Eliminating Duplicate Elimination

Consider a query plan Q' obtained by composing $\mathrm{Qp}(\cdot)$ with the result of a successful chase of a user query Q.

Q' will in general require a top-level *duplicate elimination* operation to ensure that the query plan implements Q.

Adding duplicate elimination unconditionally to any query plan is clearly unacceptable on performance grounds. We now consider how to rewrite query plans to reduce and possibly avoid the overhead that this entails.

Assume $\langle \mathrm{S_L} \cup \mathrm{S_P}, \Sigma \rangle$ is a physical design, and that Q_1 and Q_2 are a pair of query plans over the design.

A *rewrite rule* is written as $Q_1 \leftrightarrow Q_2$ and is correct if the following holds for all user queries Q over the design.

$$\Sigma \models Q \lhd Q_1 \ \text{ iff } \ \Sigma \models Q \lhd Q_2$$

Query Context

Assume Q_1 is a query plan that contains a subplan Q_2.

Query Context

Assume Q_1 is a query plan that contains a subplan Q_2. Write Q_1^c to denote a *query context* in which Q_2 has been replace by a placeholder "[ ]". Also write $Q^c[Q']$ to denote a substitution of the placeholder "[ ]" in query context Q^c with Q' (either a query or a query context).

Query Context

Assume Q_1 is a query plan that contains a subplan Q_2. Write Q_1^c to denote a *query context* in which Q_2 has been replace by a placeholder "[ ]". Also write $Q^c[Q']$ to denote a substitution of the placeholder "[ ]" in query context Q^c with Q' (either a query or a query context).

Observation: Contexts can be composed: if Q_1^c and Q_2^c are contexts, then $Q_1^c[Q_2^c]$ is a context.

Query Context

Assume Q_1 is a query plan that contains a subplan Q_2. Write Q_1^c to denote a *query context* in which Q_2 has been replace by a placeholder "[ ]". Also write $Q^c[Q']$ to denote a substitution of the placeholder "[ ]" in query context Q^c with Q' (either a query or a query context).

Observation: Contexts can be composed: if Q_1^c and Q_2^c are contexts, then $Q_1^c[Q_2^c]$ is a context.

Given a context Q^c, a *user query* $\mathrm{Uq}_p(Q^c)$ abstracting *properties* of variables *within the context* is defined as follows.

$$\mathrm{Uq}_p(Q^c) \equiv \begin{cases} \top & Q^c = \text{“[ ]”} \\ \mathrm{Uq}(Q_2) \wedge \mathrm{Uq}_p(Q_1^c) & Q^c = \text{“}Q_1^c[Q_2 \wedge [\,]]\text{” or “}Q_1^c[[\,] \wedge Q_2]\text{”} \\ \exists x.\, \mathrm{Uq}_p(Q_1^c) & Q^c = \text{“}Q_1^c[\exists x.[\,]]\text{”} \\ \mathrm{Uq}_p(Q_1^c) & Q^c = \text{“}Q_1^c[\{[\,]\}]\text{”, “}Q_1^c[\neg[\,]]\text{”, “}Q_1^c[Q_2 \vee [\,]]\text{”} \\ & \quad \text{or “}Q_1^c[[\,] \vee Q_2]\text{”} \end{cases}$$

Eliminating Duplicate Elimination (cont'd)

Assume $\langle \mathrm{S_L} \cup \mathrm{S_P}, \Sigma \rangle$ is a physical design and $Q^c[Q']$ a query plan. Then the following rewrite rules hold.

$$
\begin{array}{rcl}
Q^c[\{R(x_1,\ldots,x_k)\}] & \leftrightarrow & Q^c[R(x_1,\ldots,x_k)] \\
Q^c[\{Q_1 \wedge Q_2\}] & \leftrightarrow & Q^c[\{Q_1\} \wedge \{Q_2\}] \\
Q^c[\{\exists x.Q_1\}] & \underset{\mathcal{C}_1}{\leftrightarrow} & Q^c[\exists x.\{Q_1\}] \\
Q^c[\{\neg Q_1\}] & \leftrightarrow & Q^c[\neg Q_1] \\
Q^c[\neg\{Q_1\}] & \leftrightarrow & Q^c[\neg Q_1] \\
Q^c[\{Q_1 \vee Q_2\}] & \underset{\mathcal{C}_2}{\leftrightarrow} & Q^c[\{Q_1\} \vee \{Q_2\}]
\end{array}
$$

Eliminating Duplicate Elimination (cont'd)

Assume $\langle \mathrm{S_L} \cup \mathrm{S_P}, \Sigma \rangle$ is a physical design and $Q^c[Q']$ a query plan. Then the following rewrite rules hold.

$$
\begin{aligned}
Q^c[\{R(x_1,\ldots,x_k)\}] &\quad \leftrightarrow \quad Q^c[R(x_1,\ldots,x_k)] \\
Q^c[\{Q_1 \wedge Q_2\}] &\quad \leftrightarrow \quad Q^c[\{Q_1\} \wedge \{Q_2\}] \\
Q^c[\{\exists x.Q_1\}] &\quad \underset{\mathcal{C}_1}{\leftrightarrow} \quad Q^c[\exists x.\{Q_1\}] \\
Q^c[\{\neg Q_1\}] &\quad \leftrightarrow \quad Q^c[\neg Q_1] \\
Q^c[\neg\{Q_1\}] &\quad \leftrightarrow \quad Q^c[\neg Q_1] \\
Q^c[\{Q_1 \vee Q_2\}] &\quad \underset{\mathcal{C}_2}{\leftrightarrow} \quad Q^c[\{Q_1\} \vee \{Q_2\}]
\end{aligned}
$$

$\mathcal{C}_1$ and $\mathcal{C}_2$ correspond to the following respective conditions, where y_1 and y_2 in the former are fresh variable names not occurring in Q or Q_1.

$$\Sigma \cup \{\mathrm{Uq}_p(Q^c) \wedge \mathrm{Uq}(Q_1)[y_1/x] \wedge \mathrm{Uq}(Q_1)[y_2/x]\} \models (y_1 \approx y_2)$$

$$\Sigma \cup \{\mathrm{Uq}_p(Q^c)\} \models (Q_1 \wedge Q_2) \to \bot$$

Incremental Query Context

Given a context Q^c, a *user query* $\mathrm{Uq}_{ip}(Q^c)$ abstracting *incremental properties* of variables within the context is defined as follows.

$$\mathrm{Uq}_{ip}(Q^c) \equiv \begin{cases} \top & Q^c = \text{“}[\,]\text{”} \\ \mathrm{Uq}(Q_2) \wedge \mathrm{Uq}_{ip}(Q_1^c) & Q^c = \text{“}Q_1^c[Q_2 \wedge [\,]]\text{”} \\ \exists x.\, \mathrm{Uq}_{ip}(Q_1^c) & Q^c = \text{“}Q_1^c[\exists x.[\,]]\text{”} \\ \mathrm{Uq}_{ip}(Q_1^c) & Q^c = \text{“}Q_1^c[\{[\,]\}]\text{”}, \text{“}Q_1^c[\neg[\,]]\text{”}, \text{“}Q_1^c[Q_2 \vee [\,]]\text{”}, \\ & \qquad \text{“}Q_1^c[[\,] \vee Q_2]\text{” or “}Q_1^c[[\,] \wedge Q_2]\text{”} \end{cases}$$

Cut Insertion

Observe that the rewrite rules for duplicate elimination are bidirectional, and can therefore determine situations in which such operators can be *added* to a query plan.

Cut Insertion

Observe that the rewrite rules for duplicate elimination are bidirectional, and can therefore determine situations in which such operators can be *added* to a query plan.

This is useful when formulating additional rewrite rules that determine when cut operators can be inserted in query plans without any impact on their ability to implement user queries.

Cut Insertion

Observe that the rewrite rules for duplicate elimination are bidirectional, and can therefore determine situations in which such operators can be *added* to a query plan.

This is useful when formulating additional rewrite rules that determine when cut operators can be inserted in query plans without any impact on their ability to implement user queries.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design and $Q^c[\{Q_1\} \wedge Q_2]$ a query plan. Then the following rewrite rule holds.

$$Q^c[\{Q_1\} \wedge Q_2] \underset{\mathcal{C}}{\leftrightarrow} Q^c[[\{Q_1\}]_\ell \wedge (Q_2 \wedge\, !_\ell)]$$

Cut Insertion

Observe that the rewrite rules for duplicate elimination are bidirectional, and can therefore determine situations in which such operators can be *added* to a query plan.

This is useful when formulating additional rewrite rules that determine when cut operators can be inserted in query plans without any impact on their ability to implement user queries.

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design and $Q^c[\{Q_1\} \wedge Q_2]$ a query plan. Then the following rewrite rule holds.

$$Q^c[\{Q_1\} \wedge Q_2] \underset{\mathcal{C}}{\leftrightarrow} Q^c[[\{Q_1\}]_\ell \wedge (Q_2 \wedge\, !_\ell)]$$

$\mathcal{C}_1$ corresponds to the following condition, where $\mathrm{Out}(Q_1) = \{x_1, \ldots, x_k\}$ and where each y_i and z_j are fresh variable names not occurring in Q^c, Q_1 or Q_2.

$$\Sigma \cup \{\mathrm{Uq}_{ip}(Q^c) \wedge \mathrm{Uq}((Q_1 \wedge Q_2)[y_1/x_1, \ldots, y_k/x_k]) \wedge \mathrm{Uq}((Q_1 \wedge Q_2)[z_1/x_1, \ldots, z_k/x_k])\} \models (y_1 \approx z_1) \wedge \cdots \wedge (y_k \approx z_k)$$

Sets of Conjunctive Queries

It is straightforward to expand the family of dependencies beyond TGDs and EGDs with a straightforward generalization of the chase procedure.

Sets of Conjunctive Queries

It is straightforward to expand the family of dependencies beyond TGDs and EGDs with a straightforward generalization of the chase procedure.

The additional varieties of dependencies that become possible with this generalization are as follows.

Sets of Conjunctive Queries

It is straightforward to expand the family of dependencies beyond TGDs and EGDs with a straightforward generalization of the chase procedure.

The additional varieties of dependencies that become possible with this generalization are as follows.

Coverage Dependencies

$$\forall x_1. \cdots .\forall x_k.(\exists x_{k+1}. \cdots .\exists x_\ell.\phi) \to$$
$$((\exists y_{1,1}. \cdots .\exists y_{1,m_1}.\psi_1) \vee \ldots \vee (\exists y_{n,1}. \cdots .\exists x_{n,m_n}.\psi_n))$$

Sets of Conjunctive Queries

It is straightforward to expand the family of dependencies beyond TGDs and EGDs with a straightforward generalization of the chase procedure.

The additional varieties of dependencies that become possible with this generalization are as follows.

Coverage Dependencies

$$\forall x_1. \cdots .\forall x_k.(\exists x_{k+1}. \cdots .\exists x_\ell.\phi) \rightarrow$$
$$((\exists y_{1,1}. \cdots .\exists y_{1,m_1}.\psi_1) \vee \ldots \vee (\exists y_{n,1}. \cdots .\exists x_{n,m_n}.\psi_n))$$

Denial Dependencies

$$\forall x_1. \cdots .\forall x_k.(\exists x_{k+1}. \cdots .\exists x_n.\phi) \rightarrow \perp$$

where $\perp$ stands for an unsatisfiable formula, e.g., $p \wedge \neg p$.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

This assumes that the sets denote disjunctions of member conjunctive queries, and that the implicit disjunctions are eventually replaced with concatenation operations.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

This assumes that the sets denote disjunctions of member conjunctive queries, and that the implicit disjunctions are eventually replaced with concatenation operations.

The new chase works by incorporating the following.

1. Disjunctions in the result of the chase step are distributed over conjunctions and existential quantifiers in order to obtain a disjunction of conjunctive queries.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

This assumes that the sets denote disjunctions of member conjunctive queries, and that the implicit disjunctions are eventually replaced with concatenation operations.

The new chase works by incorporating the following.

1. Disjunctions in the result of the chase step are distributed over conjunctions and existential quantifiers in order to obtain a disjunction of conjunctive queries.
2. All disjuncts in which the atom **false** appears are then deleted.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

This assumes that the sets denote disjunctions of member conjunctive queries, and that the implicit disjunctions are eventually replaced with concatenation operations.

The new chase works by incorporating the following.

1. Disjunctions in the result of the chase step are distributed over conjunctions and existential quantifiers in order to obtain a disjunction of conjunctive queries.
2. All disjuncts in which the atom **false** appears are then deleted.

Note: These steps predispose the expanded query to further applications of chase on each of the disjuncts individually.

Sets of Conjunctive Queries

A more general chase procedure now maps sets of conjunctive queries to sets of conjunctive queries.

This assumes that the sets denote disjunctions of member conjunctive queries, and that the implicit disjunctions are eventually replaced with concatenation operations.

The new chase works by incorporating the following.

1. Disjunctions in the result of the chase step are distributed over conjunctions and existential quantifiers in order to obtain a disjunction of conjunctive queries.
2. All disjuncts in which the atom **false** appears are then deleted.

Note: These steps predispose the expanded query to further applications of chase on each of the disjuncts individually.

The remainder of the procedure proceeds as in the original chase: each sub-chases must imply the user query, and the resulting subplans are then combined with concatenation operators to obtain a query plan.

Beyond Chasing: the Nash Case

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is the following physical design.

$$
\begin{array}{rcl}
\mathsf{S_L} & \equiv & \{R/2\} \\
\mathsf{S_P}(= \mathsf{S_A}) & \equiv & \{V_1/2/0, V_2/2/0, V_3/2/0\} \\
\Sigma & \equiv & \{ \forall x, y.(V_1(x, y) \equiv \exists u, w.(R(u, x) \wedge R(u, w) \wedge R(w, y))), \\
 & & \ \ \forall x, y.(V_2(x, y) \equiv \exists u, w.(R(x, u) \wedge R(u, w) \wedge R(w, y))), \\
 & & \ \ \forall x, y.(V_3(x, y) \equiv \exists u.(R(x, u) \wedge R(u, y))) \qquad \}
\end{array}
$$

Beyond Chasing: the Nash Case

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is the following physical design.

$$\begin{array}{rcl} \mathsf{S_L} & \equiv & \{R/2\} \\ \mathsf{S_P}(= \mathsf{S_A}) & \equiv & \{V_1/2/0, V_2/2/0, V_3/2/0\} \\ \Sigma & \equiv & \{\, \forall x, y.(V_1(x, y) \equiv \exists u, w.(R(u, x) \wedge R(u, w) \wedge R(w, y))), \\ & & \quad \forall x, y.(V_2(x, y) \equiv \exists u, w.(R(x, u) \wedge R(u, w) \wedge R(w, y))), \\ & & \quad \forall x, y.(V_3(x, y) \equiv \exists u.(R(x, u) \wedge R(u, y))) \qquad \} \end{array}$$

Also let Q denote the following (conjunctive) user query.

$$\exists u, v, w.(R(u, x) \wedge R(u, w) \wedge R(w, v) \wedge R(v, y))$$

Beyond Chasing: the Nash Case

Assume $\langle S_L \cup S_P, \Sigma \rangle$ is the following physical design.

$$
\begin{array}{rcl}
S_L & \equiv & \{R/2\} \\
S_P(= S_A) & \equiv & \{V_1/2/0, V_2/2/0, V_3/2/0\} \\
\Sigma & \equiv & \{ \forall x, y.(V_1(x, y) \equiv \exists u, w.(R(u, x) \wedge R(u, w) \wedge R(w, y))), \\
 & & \quad \forall x, y.(V_2(x, y) \equiv \exists u, w.(R(x, u) \wedge R(u, w) \wedge R(w, y))), \\
 & & \quad \forall x, y.(V_3(x, y) \equiv \exists u.(R(x, u) \wedge R(u, y))) \quad \}
\end{array}
$$

Also let Q denote the following (conjunctive) user query.

$$\exists u, v, w.(R(u, x) \wedge R(u, w) \wedge R(w, v) \wedge R(v, y))$$

An equivalent user query to Q can be formulated in terms of the three predicates that are access paths, V_1, V_2 and V_3.

$$\exists u.(V_1(x, u) \wedge \forall w.(V_3(w, u) \rightarrow V_2(w, y))).$$

Beyond Chasing (cont'd)

Another equivalent user query to Q can be formulated that leads directly to a query plan, again, only with non-logical parameters that are access paths.

$$\exists u, v.(V_1(x, u) \wedge V_3(v, u) \wedge V_2(v, y) \wedge \forall w.(V_3(w, u) \rightarrow V_2(w, y)))$$

Beyond Chasing (cont'd)

Another equivalent user query to Q can be formulated that leads directly to a query plan, again, only with non-logical parameters that are access paths.

$$\exists u, v.(V_1(x, u) \wedge V_3(v, u) \wedge V_2(v, y) \wedge \forall w.(V_3(w, u) \rightarrow V_2(w, y)))$$

In particular, standard equivalences in FOL can then be applied to this to obtain a query plan that implements Q.

$$\exists u, v.\{(V_1(x, u) \wedge V_3(v, u) \wedge V_2(v, y) \wedge \neg \exists w.(V_3(w, u) \wedge \neg V_2(w, y)))\}$$

Beyond Chasing (cont'd)

Another equivalent user query to Q can be formulated that leads directly to a query plan, again, only with non-logical parameters that are access paths.

$$\exists u, v.(V_1(x,u) \wedge V_3(v,u) \wedge V_2(v,y) \wedge \forall w.(V_3(w,u) \rightarrow V_2(w,y)))$$

In particular, standard equivalences in FOL can then be applied to this to obtain a query plan that implements Q.

$$\exists u, v.\{(V_1(x,u) \wedge V_3(v,u) \wedge V_2(v,y) \wedge \neg \exists w.(V_3(w,u) \wedge \neg V_2(w,y)))\}$$

In general, it is known that there does not exist a *positive* user query equivalent to Q, that is, a user query with no occurrences of either negation or universal quantification.

Interpolation and Craig's Theorem

Assume $\langle \mathsf{S}_\mathsf{L} \cup \mathsf{S}_\mathsf{P}, \Sigma \rangle$ is a physical design and Q a user query.

Interpolation and Craig's Theorem

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design and Q a user query.

Recall that Q is Beth definable if and only if the following holds.

$$(\Sigma \cup \Sigma^*) \models (Q \rightarrow Q^*)$$

Interpolation and Craig's Theorem

Assume $\langle S_L \cup S_P, \Sigma \rangle$ is a physical design and Q a user query.

Recall that Q is Beth definable if and only if the following holds.[1]

$$(\Sigma \cup \Sigma^*) \models (Q \to Q^*)$$

[1] Recall that Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in S_A are uniformly renamed.

Interpolation and Craig's Theorem

Assume $\langle \mathrm{S_L} \cup \mathrm{S_P}, \Sigma \rangle$ is a physical design and Q a user query.

Recall that Q is Beth definable if and only if the following holds.[1]

$$(\Sigma \cup \Sigma^*) \models (Q \rightarrow Q^*)$$

The condition can be reformulated to the form "$\models \varphi \rightarrow \psi$".

$$\models ((\bigwedge \Sigma) \wedge Q) \rightarrow ((\bigwedge \Sigma^*) \rightarrow Q^*)$$

[1] Recall that Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in $\mathrm{S_A}$ are uniformly renamed.

Interpolation and Craig's Theorem

Assume $\langle \mathsf{S_L} \cup \mathsf{S_P}, \Sigma \rangle$ is a physical design and Q a user query.

Recall that Q is Beth definable if and only if the following holds.[1]

$$(\Sigma \cup \Sigma^*) \models (Q \to Q^*)$$

The condition can be reformulated to the form "$\models \varphi \to \psi$".

$$\models ((\bigwedge \Sigma) \wedge Q) \to ((\bigwedge \Sigma^*) \to Q^*)$$

(*Craig's Theorem*) Let φ and ψ be WFFs. Then $\models \varphi \to \psi$ implies that there is a WFF ϕ that contains only non-logical symbols common to both φ and ψ, called the *interpolant*, such that $\models \varphi \to (\phi \to \psi)$.

[1] Recall that Σ^* and Q^* are respective copies of Σ and Q in which all non-logical parameters *not present* in $\mathsf{S_A}$ are uniformly renamed.

A Revised Chase Procedure for Plan Synthesis

Input: A conjunctive user query $Q(= \{\varphi_1, \ldots, \varphi_k\})$ and a set Σ of TGDs.

Result: A (possibly infinite) sequence $S = (\psi_1, \psi_2, \ldots)$ satisfying binding pattern requirements, and such that

$$\Sigma \models Q \lhd \mathrm{Qp}((\psi_1, \ldots, \psi_\ell))$$

for all finite prefixes $(\psi_1, \ldots, \psi_\ell)$ of S where $n \leq \ell$ if *success*.

1. Initialize: $S \leftarrow (\,)$; $G \leftarrow \mathrm{Chase}_\Sigma(S)$; $n \leftarrow 0$; *success* $\leftarrow$ false.
2. If there exists $\psi \in G$ for which $S \mid (\psi)$ satisfies binding pattern requirements, then $S \leftarrow S \mid (\psi)$.
3. If
   $$(\bigwedge \Sigma) \wedge (\bigwedge \Sigma^*) \wedge \psi_1 \wedge \cdots \wedge \psi_\ell \wedge (\neg Q^*)$$
   is not satisfiable, then *success* $\leftarrow$ true. Otherwise $n \leftarrow n + 1$.
4. Resume at Step 2.

General Plan Synthesis by Interpolation

Input: A user query Q and a set Σ of constraints.

Result: An enumeration of possible (first order) query plans Q' of Q.

1. Enumerate interpolants Q'' of the following that lead to a query plan Q'.

$$(\bigwedge \Sigma) \wedge (\bigwedge \Sigma^*) \wedge Q \wedge (\neg Q^*)$$

General Plan Synthesis by Interpolation

Input: A user query Q and a set Σ of constraints.

Result: An enumeration of possible (first order) query plans Q' of Q.

1. Enumerate interpolants Q'' of the following that lead to a query plan Q'.

$$(\bigwedge \Sigma) \wedge (\bigwedge \Sigma^*) \wedge Q \wedge (\neg Q^*)$$

The Nash case can be solved by this procedure.