Query Compilation and Plan Synthesis

Grant Weddell

Cheriton School of CS

Spring 2012
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.

- 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 \( S_L \).

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

1. Both \( \mathcal{I}_1 \) and \( \mathcal{I}_2 \) satisfy \( \Sigma \).
2. \( (R)^{\mathcal{I}_1} = (R)^{\mathcal{I}_2} \) for all non-logical parameters \( R/m/n \in S_A \).
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 $\Sigma$ 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 $\Sigma^*$ and $Q^*$ are respective copies of $\Sigma$ 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 $R^*$.)

$Q$ 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.
Recall our basic physical design for PAYROLL.

\[
\Sigma = \{ \forall x, y, z. (\text{employee}(x, y, z) \rightarrow \text{emp-array}0(z, x, y)), \\
\forall x, y, z. (\text{emp-array}0(z, x, y) \rightarrow \text{employee}(x, y, z)) \}\ 
\]

\[
S_A = \{ \text{emp-array}0/3/0 \}
\]

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

\[
(\Sigma \cup \Sigma^*) = \{ \forall x, y, z. (\text{employee}(x, y, z) \rightarrow \text{emp-array}0(z, x, y)), \\
\forall x, y, z. (\text{emp-array}0(z, x, y) \rightarrow \text{employee}(x, y, z)), \\
\forall x, y, z. (\text{employee}^*(x, y, z) \rightarrow \text{emp-array}0(z, x, y)), \\
\forall x, y, z. (\text{emp-array}0(z, x, y) \rightarrow \text{employee}^*(x, y, z)) \}\ 
\]

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

\[
(\Sigma \cup \Sigma^*) \models \text{employee}(x, y, z) \rightarrow \text{employee}^*(x, y, z)
\]
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 $\phi \in \text{WFF}$ is a formula, $x$ is a variable and $t$ is a term such that $\text{Fv}(t)$ does not contain variables quantified in $\phi$.

A substitution of $t$ for $x$ in $\phi$ is the WFF obtained from $\phi$ 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 $\theta$, for multiple variables in a formula.

The simultaneous application of all substitutions in $\theta$ to the formula $\phi$ is written in the same way.

$$\phi\theta$$
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 $\Sigma$ 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 $\Sigma$.

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

3. A substitution $\theta$ such that, for each atom $\phi_i$ in $\phi$, $\phi_i\theta$ is an atom in $\varphi$.

Then the following holds.

$$\Sigma \models (\exists x_1. \cdots . \exists x_m. \varphi) \equiv (\exists x_1. \cdots . \exists x_m. \varphi \land (\exists x_{j+1}. \cdots . \exists x_k. \psi)\theta)$$
Recall the formula resulting from a chase step.

\[ \exists x_1 \cdots \exists x_m. \varphi \land (\exists x_{j+1} \cdots \exists x_k. \psi) \theta \]

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.

\[ \{ \text{employee}(x, y, z) \} \]

Two applications of a chase step obtains the following.

\[ \{ \text{employee}(x, y, z), \text{employee}0(z, x, y), \text{employee}^*(x, y, z) \} \]
Assume a set of TGDs is given by $\Sigma$, and that $Q$ is a conjunctive user query given in the alternative syntax.

A repeated application of chase steps over all dependencies in $\Sigma$ is called the *chase of $Q$ with $\Sigma$*, 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 $\Sigma$ 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. Also assume $\{\psi_1, \ldots, \psi_n\}$ 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 $S_A$ 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 $Qp(\cdot)$, for obtaining a query plan from $(\psi_1, \ldots, \psi_n)$ where $\Sigma \models Q \triangleleft Qp((\psi_1, \ldots, \psi_n))$.\(^1\)

\(^1\)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 \( \Sigma \) of TGDs.

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

\[
\Sigma \models Q \triangleleft Qp((\psi_1, \ldots, \psi_\ell))
\]

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

1. Initialize: \( S \leftarrow () \); \( G \leftarrow \text{Chase}_\Sigma(S) \); \( n \leftarrow 0 \); \( \text{success} \leftarrow \text{false} \).
2. If there exists \( \psi \in G \) for which \( S \models (\psi) \) satisfies binding pattern requirements, then \( S \leftarrow S \UPARROW (\psi) \).
3. If there exists \( \theta \) over the existential variables of \( Q \) for which \( Q\theta \subseteq (\text{Chase}_\Sigma(\text{Setof}(S)) \cap G) \), then \( \text{success} \leftarrow \text{true} \). Otherwise \( n \leftarrow n + 1 \).
4. Resume at Step 2.
Assuming \((\psi_1, \ldots, \psi_\ell)\) satisfies binding pattern requirements.

\[
\varphi_1, \ldots, \varphi_k \\
\vdots \\
\psi_1, \ldots, \psi_\ell \\
\vdots \\
\varphi_1\theta, \ldots, \varphi_k\theta
\]

\[
\text{Chase}_\Sigma(Q) \\
\text{Chase}_\Sigma(Q') \\
\text{Chase}_\Sigma(Q)
\]
ACME Case: A Plan Using $\text{emp-array0}$

Assume our basic physical design for PAYROLL.

\[
\Sigma = \{ \forall x, y, z. (\text{employee}(x, y, z) \rightarrow \text{emp-array0}(z, x, y)), \\
\forall x, y, z. (\text{emp-array0}(z, x, y) \rightarrow \text{employee}(x, y, z)) \} 
\]

\[
S_A = \{ \text{emp-array0}/3/0 \} 
\]

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

\[
\{ \text{employee}(x, y, z) \} 
\]
An execution of the chase procedure is successful where \( n = 1 \) and with other results as follows.

\[
Q = \{\text{employee}(x, y, z)\} \\
S = (\text{employee0}(z, x, y)) \\
G = \{\text{employee}(x, y, z), \text{employee0}(z, x, y)\}
\]

The successful result is a consequence of the following.

1. \((\text{Chase}_\Sigma(\text{Setof}(S)) \cap G) = \{\text{employee}(x, y, z), \text{employee0}(z, x, y)\}\).
2. \(Q\theta \subseteq (\text{Chase}_\Sigma(\text{Setof}(S)) \cap G)\), where \(\theta = [y/y]\).
Assume a set of TGDs is given by \( \Sigma \), 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.

\[ \Sigma \models Q \triangleleft Q_p((\psi_1, \ldots, \psi_\ell)). \]

\textit{(plan synthesis)}

\textit{(backchase)} If

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

then \( \Sigma \models Q \triangleleft Q_p((\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots \psi_\ell)). \)
Equality Generating Dependencies (EGDs)

Recall that an EGD has the following form.

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

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 $Qp(\cdot)$ 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 $\langle S_L \cup S_P, \Sigma \rangle$ 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.

$$\Sigma \models Q \triangleleft Q_1 \text{ iff } \Sigma \models Q \triangleleft Q_2$$
Assume $Q$ is a query plan that contains a subplan $Q_1$. Write $Q[Q_1]$ to denote this and call $Q[]$, in which $Q_1$ in $Q$ has been replaced by a placeholder “[]”, a context.

**Observation:** Contexts can be composed: if $Q[]$ and $Q'[[]]$ are contexts and $Q''$ a query plan such that $Q[Q'[Q'']]$ is also a query plan, then $Q[Q'[[]]]$ is a context.

Given a context $Q[]$, a user query $Uq_{[]} (Q[])$ abstracting properties of variables *within the context* is defined as follows.

\[
Uq_{[]} (Q[]) \equiv \begin{cases} 
\top & Q[] = "[]" \\
Uq(Q_2) \land Uq_{[]} (Q_1[]) & Q[] = "Q_1[Q_2 \land []]\” \text{ or } "Q_1[[] \land Q_2]\” \\
\exists x. Uq_{[]} (Q_1[]) & Q[] = "Q_1[\exists x.[]]\” \\
Uq_{[]} (Q_1[]) & Q[] = "Q_1[[]]\” \text{ or } "Q_1[[] \lor Q_2]\”
\end{cases}
\]
Assume \( \langle S_L \cup S_P, \Sigma \rangle \) is a physical design and \( Q \) a query plan. Then the following rewrite rules hold.

\[
\begin{align*}
Q[\{ R(x_1, \ldots, x_k) \}] & \iff Q[R(x_1, \ldots, x_k)] \\
Q[\{ Q_1 \land Q_2 \}] & \iff Q[\{ Q_1 \} \land \{ Q_2 \}] \\
Q[\{ \exists x.Q_1 \}] & \iff Q[\exists x.\{ Q_1 \}] \\
Q[\{ \neg Q_1 \}] & \iff Q[\neg Q_1] \\
Q[\{ \neg\{ Q_1 \} \}] & \iff Q[\neg\{ Q_1 \}] \\
Q[\{ Q_1 \lor Q_2 \}] & \iff Q[\{ Q_1 \} \lor \{ Q_2 \}] \\
\end{align*}
\]

\( C_1 \) and \( C_2 \) correspond to the following respective conditions, where \( y_1 \) and \( y_2 \) in the former are fresh variable names not occurring in \( Q \) or \( Q_1 \).

\[
\Sigma \cup \{ Uq[i](Q[]) \} \land Uq(Q_1)[y_1/x] \land Uq(Q_1)[y_2/x] \models y_1 \approx y_2
\]

\[
\Sigma \cup \{ Uq[i](Q[]) \} \models (Q_1 \land Q_2) \rightarrow \bot
\]
Incremental Query Context

Given a context $Q[]$, a user query $Uqinc_{[\]}(Q[\])$ abstracting incremental properties of variables within the context is defined as follows.

$$\begin{align*}
Uqinc_{[\]}(Q[\]) &\equiv \\
T &\implies Q[\] = "[]"
\Uq(Q_2) \land Uqinc_{[\]}(Q_1[\]) &\implies Q[\] = "Q_1[Q_2 \land []]"
\exists x. Uqinc_{[\]}(Q_1[\]) &\implies Q[\] = "Q_1[\exists x.[]]"
Uqinc_{[\]}(Q_1[\]) &\implies Q[\] = "Q_1[{}[]]", "Q_1[\neg[]]", "Q_1[Q_2 \lor []]", "Q_1[][\lor Q_2]" \text{ or } "Q_1[][\land Q_2]"
\end{align*}$$
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 $\langle S_L \cup S_P, \Sigma \rangle$ is a physical design and $Q$ a query plan. Then the following rewrite rule holds.

$$Q[\{Q_1\} \land Q_2] \quad \overset{C_1}{\leftrightarrow} \quad Q[[Q_1]_\ell \land (Q_2 \land !\ell)]$$

$C_1$ corresponds to the following condition, where $\text{Out}(Q_1) = \{x_1, \ldots, x_k\}$ and where each $y_i$ and $z_j$ are fresh variable names not occurring in $Q$, $Q_1$ or $Q_2$.

$$\Sigma \cup \{\text{Uq}(Q) \land \text{Uq}((Q_1 \land Q_2)[y_1/x_1, \ldots, y_k/x_k]) \land \text{Uq}((Q_1 \land Q_2)[z_1/x_1, \ldots, z_k/x_k])\} \models (y_1 \approx z_1) \land \cdots \land (y_k \approx z_k)$$
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.

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

**Coverage Dependencies**

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

**Denial Dependencies**

\[ \forall x_1 \ldots \forall x_k.(\exists x_{k+1} \ldots \exists x_n.\phi) \rightarrow \text{false} \]

where **false** stands for an unsatisfiable formula, e.g., \( p \land \neg p \).
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 \texttt{false} appears are then deleted.

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

$S_L \equiv \{ R/2 \}$

$S_P(= S_A) \equiv \{ V_1/2/0, V_2/2/0, V_3/2/0 \}$

$\Sigma \equiv \{ \forall x, y. (V_1(x, y) \equiv \exists u, w. (R(u, x) \land R(u, w) \land R(w, y))),$ 

$\forall x, y. (V_2(x, y) \equiv \exists u, w. (R(x, u) \land R(u, w) \land R(w, y))),$ 

$\forall x, y. (V_3(x, y) \equiv \exists u. (R(x, u) \land R(u, y))) \}$

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

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

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

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

\[
\exists u, v. (V_1(x, u) \land V_3(v, u) \land V_2(v, y) \land \forall w. (V_3(w, u) \rightarrow V_2(w, y)))
\]

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

\[
\exists u, v. \{ (V_1(x, u) \land V_3(v, u) \land V_2(v, y) \land \neg \exists w. (V_3(w, u) \land \neg V_2(w, y))) \}
\]

In general, it is known that there does not exist a \textit{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 $\langle S_L \cup S_P, \Sigma \rangle$ is a physical design and $Q$ a user query.

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

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

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

$$((\bigwedge \Sigma) \land Q) \rightarrow ((\bigwedge \Sigma^*) \rightarrow Q^*)$$

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

¹Recall that $\Sigma^*$ and $Q^*$ are respective copies of $\Sigma$ and $Q$ in which all non-logical parameters *not present* in $S_A$ are uniformly renamed.
A Revised Chase Procedure for Plan Synthesis

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

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

$$\Sigma \models Q \triangleleft Q_p((\psi_1, \ldots, \psi_{\ell}))$$

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

1. Initialize: $S \leftarrow (\); G \leftarrow \text{Chase}_\Sigma(S); n \leftarrow 0; success \leftarrow \text{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

$$\left(\bigwedge \Sigma\right) \land \left(\bigwedge \Sigma^*\right) \land \psi_1 \land \cdots \land \psi_{\ell} \land \neg Q^*$$

is not satisfiable, then $success \leftarrow \text{true}$. Otherwise $n \leftarrow n + 1$.
4. Resume at Step 2.
**Input:** A user query $Q$ and a set $\Sigma$ of constraints.

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

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

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

The Nash case can be solved by this procedure.