Chase

Alin Deutsch
University of California San Diego

Alan Nash
Tradeworx

SYNONYMS

none

DEFINITION

The chase is a procedure that takes as input a set Σ of constraints and an instance I. The chase does not always terminate, but if it does it produces as output an instance U with the following properties:

1. $U \models \Sigma$; that is, U satisfies Σ.
2. $I \rightarrow U$; that is, there is a homomorphism from I to U.
3. For every instance J (finite or infinite), if $J \models \Sigma$ and $I \rightarrow J$, then $U \rightarrow J$.

In [7], an instance that satisfies (1) and (2) above is called a *model of* Σ *and* I and an instance that satisfies (3) above is called *strongly universal*.
In summary, the chase is a procedure which—whenever it terminates—yields a strongly-universal model

Comments.

1. The set Σ of constraints is usually a set of tuple-generating dependencies (tgds) and equality-generating dependencies (egds) [5], or, equivalently, embedded dependencies [5, 10]. However, the chase has been extended to wider classes of constraints and to universality under functions other than homomorphisms [6, 7, 9]. In this case, the chase often produces a strongly-universal model *set* (see below), instead of a single model.
2. It was noted in [7] that in database applications, *weak universality* (condition 3 above restricted to finite instances) would suffice. Nevertheless, the chase gives strong universality.

HISTORICAL BACKGROUND

The term "chase" was coined in [15], where it was used to test the logical implication of dependencies (i.e. whether all databases satisfying a set Σ of dependencies must also satisfy a given dependency σ). The implication problem was one of the key concerns of dependency theory, with applications to automatic schema design. [15] defined the chase for the classes of functional, join and multivalued dependencies. Related chase formulations for various kinds of dependencies were introduced in [14, 17]. [5] unified the treatment of the implication problem for various dependency classes by introducing and defining the chase for tuple-generating and equality-generating dependencies (sufficiently expressive to capture all prior dependencies).

Ancestors of the chase (introduced as unnamed algorithms) appear in [4, 3, 2]. [4] introduces tableaux, a pattern-based representation for relational queries, and shows how to check the equivalence of tableau queries in the presence of functional dependencies, with applications to query optimization. To this end, the tableaux are modified using an algorithm that coincides with the chase with functional dependencies. [3] uses the same algorithm for minimization of tableaux under functional dependencies. This algorithm is extended in [2] to include also multivalued dependencies, for the purpose of checking whether the join of several relations is lossless

(i.e. the original relations can be retrieved as projections of the join result).

The chase was extended to include disjunction and inequality in [9], and to arbitrary $\forall\exists$-sentences in [6]. Independently, [13] extended the chase to a particular case of disjunctive dependencies incorporating disjunctions of equalities between variables and constants (see also [12]). There are also extensions of the chase to deal with more complex data models beyond relational. [16] extends the chase (and the language of embedded dependencies) to work over complex values and dictionaries. For an excellent survey of the history of the chase prior to 1995, consult [1].

SCIENTIFIC FUNDAMENTALS

A *tuple-generating dependency (tgd)* is a constraint σ of the form

$$\forall \bar{x}, \bar{y}\ (\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \beta(\bar{x}, \bar{z}))$$

where α and β are conjunctions of relational atoms. Furthermore, every variable in $\bar{x}$ appears in both α and β. The $\forall \bar{x}, \bar{y}$ prefix of universal quantifiers is usually omitted. If $\bar{z}$ is empty, then σ is *full*.

An *equality-generating dependency (egd)* is a constraint ϕ of the form

$$\forall x_1, x_2, \bar{y}\ (\alpha(x_1, x_2, \bar{y}) \rightarrow x_1 = x_2)$$

where α is a conjunction of relational atoms.

The chase is used on instances whose active domain consists of constants and *labeled nulls*. A *homomorphism* from A to B is denoted $A \rightarrow B$. It is a mapping h on the constants and nulls in A that (i) preserves constants (i.e. $h(c) = c$ for every constant c) and preserves relationships (i.e. for every tuple $R(x_1, \ldots, x_n) \in A$, we have $R(h(x_1), \ldots, h(x_n)) \in B$). Two instances A and B are *homomorphically equivalent* if $A \rightarrow B$ and $B \rightarrow A$.

The chase is a natural procedure for building strong universal models. Indeed, it turns out that checking for strong universality is undecidable as shown in [7]). In contrast, checking whether an instance is a model can be done efficiently. Therefore, it is natural to define any procedure for constructing strong universal models by steps which always preserve strong universality while attempting to obtain a model and then to check whether a model was indeed obtained. This is precisely what the chase does.

A tgd $\sigma \in \Sigma$ *fails* or *applies* on $A, \bar{a}$ if there is $\bar{b}$ in A such that the premise α of σ satisfies $A \models \alpha(\bar{a}, \bar{b})$, yet there is no $\bar{c}$ in A such that the conclusion β of σ satisfies $A \models \beta(\bar{a}, \bar{c})$. Assume that the instance A' is obtained by adding to A the tuples in $\beta(\bar{a}, \bar{n})$ where $\bar{n}$ is a tuple of new nulls. Then A' is the result of *firing* σ on $A, \bar{a}$. Notice that $A \subseteq A'$ and that σ does not fail on $A', \bar{a}$. It is easy to verify that if A is strongly universal for Σ and I, then so is A' (towards this, it is essential that all the nulls in $\bar{n}$ be new and distinct).

An egd $\sigma \in \Sigma$ *fails* or *applies* on A, a_1, a_2 if there is $\bar{b}$ in A such that the premise α of σ satisfies $A \models \alpha(a_1, a_2, \bar{b})$, yet $a_1 \neq a_2$. If a_2 is a null and we replace a_2 everywhere in A with a_1 to obtain A', then we say that A' is the result of *firing* σ on A, a_1, a_2. Notice that $A \rightarrow A'$ and that σ does not fail on A', a_1, a_1. It is easy to verify that if A is strongly universal for Σ and I, then so is A'. If a_2 is a constant, but a_1 is null, then we can replace a_1 everywhere in A with a_2 instead. However, if both a_1 and a_2 are constants, then it is not possible to satisfy σ and preserve strong universality and the chase *fails*.

The standard chase procedure proceeds as follows.

1. Set $A_0 := I$.
2. Repeat the following:

2a. If A_n is a model of Σ and I, stop and return A_n.

2b. Otherwise, there must be either

(i) a tgd σ and $\bar{a}$ such that σ fails on $A, \bar{a}$, or

(ii) an egd σ' and a_1, a_2 such that σ' fails on A, a_1, a_2.

Obtain A_{n+1} by picking one such σ and $\bar{a}$ and firing σ on $A_n, \bar{a}$, or by picking one such σ' and a_1, a_2 and firing σ' on A, a_1, a_2. (This is one *chase step* of the standard chase.)

Notice that, at every chase step, there may be a choice of σ and $\bar{a}$, respectively σ' and a_1, a_2. How these choices are picked is often left unspecified and in that case the standard chase is non-deterministic. The chase *terminates* if A_n is a model of Σ and I for some n.

The chase of instance I with tgds Σ produces a sequence of instances $I = A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$ such that every A_i is strongly universal for Σ and I. The chase with tgds and egds produces a sequence $I = A_0 \to A_1 \to A_2 \to \ldots$ such that every A_i is strongly universal for Σ and I. In the presence of egds, it is no longer the case that $A_i \subseteq A_j$ for $i \leq j$ and there is the additional complication that a chase step may fail. The chase for tgds and egds is described in more detail in [1].

Example 1 Consider the schema consisting of two relations:

1. employee Emp(ss#, name, dept#), with social security, name, and dept. number, and
2. department Dept(dept#, name, location, mgr#), with dept. number, name, location, and its manager's social security number.

Assume that Σ consists of the constraints

σ_1: dept# is a foreign key in Emp,

σ_2: mgr# is a foreign key in Dept, and

σ_3: every manager manages his own department.

(We omit the constraints that say that ss# is a key for Emp and that dept# is a key for Dept to keep the example simple.) These constraints can be written as follows (where σ_1 and σ_2 are tgds and σ_3 is an egd):

σ_1: $\text{Dept}(d, e, \ell, m) \to \exists n, d' \, \text{Emp}(m, n, d')$,

σ_2: $\text{Emp}(s, n, d) \to \exists e, \ell, m \, \text{Dept}(d, e, \ell, m)$, and

σ_3: $\text{Dept}(d, e, \ell, m), \text{Emp}(m, n, d') \to d = d'$.

Consider the initial instance

$$I_0 = \text{Dept}(1, \text{“HR”}, \text{“somewhere”}, 333 - 33 - 3333)$$

containing a single tuple. Then in the first step of the chase, σ_1 fires, giving

$$I_1 = \{\text{Dept}(1, \text{“HR”}, \text{“somewhere”}, 33), \text{Emp}(33, \alpha, \beta)\}$$

where α and β are labeled nulls. In the second step, both σ_2 and σ_3 apply. If σ_3 fires, then β is set to 1 and yields

$$I_2 = \{\text{Dept}(1, \text{“HR”}, \text{“somewhere”}, 33), \text{Emp}(33, \alpha, 1)\}.$$

Since I_2 satisfies Σ, the chase terminates. However, if instead at the second step σ_2 fires, then it gives

$$I_2' = \{\text{Dept}(1, \text{“HR”}, \text{“somewhere”}, 33), \text{Emp}(33, \alpha, \beta), \text{Dept}(\beta, \gamma, \delta, \epsilon\}$$

where γ, δ, and ϵ are new nulls. In this case, it is possible to continue firing σ_1, σ_2, and σ_3 in such a way as to obtain a chase that does not terminate, perpetually introducing new nulls.

If the standard chase (or any other chase listed below) terminates, it yields a strongly-universal model of Σ and I and it is straightforward to verify that all such models are homomorphically equivalent. Therefore the result of the standard chase is unique up to homomorphic equivalence. However, the choice of what constraint to fire and on what tuple may affect whether the chase terminates or not.

There are several variations of the chase, which we shall call here the *standard* chase, the *parallel* chase, and the *core* chase. The standard chase was described above. In the *parallel chase*, at every chase step σ is fired on $A_n, \bar{a}$ for all pairs $(\sigma, \bar{a})$ such that σ fails on $A, \bar{a}$.

We write I^Σ for the result of the chase on Σ and I, if the chase terminates. In that case, we say that I^Σ is defined. In general, it holds that if $A \to B$, then $A^\Sigma \to B^\Sigma$, whenever the latter are defined.

It was shown in [7] that the standard chase is incomplete, in the following sense: it may be that Σ and I have a strongly-universal model, yet the standard chase does not terminate. The parallel chase is also incomplete in this sense. In contrast, the *core chase* introduced in [7] is complete: if a strongly universal-model exists, the core chase terminates and yields such a model. A chase step of the core chase consists of one chase step of the parallel chase, followed by computing the core of the resulting instance.

Any of the above mentioned variations of the chase can be applied to sets of constraints which consist of

1. tgds only,
2. tgds and egds,
3. tgds and egds with disjunctions
4. tgds and egds with disjunctions and negation, which are equivalent to general $\forall\exists$ sentences.

The chase with tgds and egds has been described above. The chase has been extended to handle disjunction and negation. In this case, it gives not a single model, but a set S of models which is strongly universal, in the sense that for any model J (finite or infinite) of Σ and I, there is a model $A \in S$ such that $A \to J$. Such a set arises from a single initial model by branching due to disjunction. For example, consider the set Σ with the single disjunctive tgd

$$\sigma : \qquad R(x) \to S(x) \vee T(x)$$

and the instance I containing the single fact $R(1)$. Clearly every model of Σ and I, must contain either $S(1)$ or $T(1)$. It is easy to verify that the set $S = \{I_1, I_2\}$ where $I_1 = \{R(1), S(1)\}$ and $I_2 = \{R(1), T(1)\}$ is strongly universal for I and Σ, but no proper subset of S is. The disjunctive chase with Σ on I consists of a single step, which produces not a single model, but the set S of models. Intuitively, whenever a disjunctive tgd fires on a set W of models, it produces, for every instance $A \in W$, one instance for every disjunct in its conclusion. For details, to see how negation is handled, and to see how universality for functions other than homomorphisms is achieved, see [7, 6].

It was shown in [7] that it is undecidable whether the standard, parallel, or core chase with a set of tgds terminates. A widely-applicable, efficiently-checkable condition on a set Σ of tgds, which is sufficient to guarantee that the chase with Σ on any instance I terminates, was introduced in [9, 11]. A set of tgds satisfying this condition is called *weakly acyclic* in [11] and is said to have *stratified witnesses* in [9]. A more widely-applicable condition, also sufficient for chase termination, was introduced in [7], where a set of tgds satisfying this condition is called *stratified*.

KEY APPLICATIONS*

The chase has been used in many applications, including

- •checking containment of queries under constraints (which in turn is used in such query rewriting tasks as minimization, rewriting using views, and semantic optimization),
- •rewriting queries using views,
- •checking implication of constraints,
- •computing solutions to data exchange problems, and
- •computing certain answers in data integration settings.

To check whether a query P is contained in a query Q under constraints Σ, written $P \sqsubseteq_\Sigma Q$, it is sufficient to (1) treat P as if it was an instance in which the free variables are constants and the bound variables are nulls (this is known as the "frozen instance" or "canonical database" [1] corresponding to P) (2) chase it with Σ, and if this

chase terminates to yield P^Σ (3) check whether the result of this chase is contained in Q, written $P^\Sigma \sqsubseteq Q$. In symbols, if the chase described above terminates, then

$$P \sqsubseteq_\Sigma Q \text{ iff } P^\Sigma \sqsubseteq Q.$$

That is, the chase reduces the problem of query containment under constraints to one of query containment without constraints.

To check whether a set Σ of tgds implies a tgd σ of the form

$$\forall \bar{x}, \bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\beta(\bar{x}, \bar{z}))$$

which is logically equivalent to

$$\forall \bar{x}(\underbrace{\exists \bar{y}\alpha(\bar{x}, \bar{y})}_{Q_\alpha(\bar{x})} \rightarrow \underbrace{\exists \bar{z}\beta(\bar{x}, \bar{z})}_{Q_\beta(\bar{x})}),$$

it suffices to check whether the query Q_α in the premise of σ is contained under the constraints Σ in the query Q_β in the conclusion of σ. That is, if Q_α^Σ is defined, then

$$\Sigma \models \sigma \text{ iff } Q_\alpha \sqsubseteq_\Sigma Q_\beta \text{ iff } Q_\alpha^\Sigma \sqsubseteq Q_\beta.$$

The chase was also employed to find equivalent rewritings of conjunctive queries using conjunctive query views, in the presence of constraints. Given a set $\mathcal{V}$ of conjunctive query views and a conjunctive query Q, one can construct, using the chase, a query R expressed in terms of $\mathcal{V}$, such that Q has some equivalent rewriting using $\mathcal{V}$ if and only if R is itself such a rewriting. Moreover, every minimal rewriting of Q is guaranteed to be a sub-query of R. The algorithm for constructing R and exploring all its sub-queries is called the *Chase&Backchase (CB)* [8], and it is sound and complete for finding all minimal rewritings under a set Σ of embedded dependencies, provided the chase with Σ terminates [9]. The CB algorithm constructs R by simply (i) constructing a set $\Sigma_\mathcal{V}$ of tgds extracted from the view definitions, and (ii) chasing Q with $\Sigma \cup \Sigma_\mathcal{V}$ and restricting the resulting query to only the atoms using views in $\mathcal{V}$.

In [11] is was shown that the certain answers to a union Q of conjunctive queries on a ground instance I under a set Σ of source-to-target tgds and target tgds and egds can be obtained by computing $Q(U)$—where U is a *universal solution* for I under Σ—then discarding any tuples with nulls. Universal solutions, which are the preferred solutions to materialize in data exchange, are closely related to strongly-universal models [7] and it was shown in [11] that they can be obtained using the chase.

CROSS REFERENCE*

Data exchange – Data integration – Database dependencies – Equality-generating dependencies – Query containment – Query optimization – Query rewriting – Query rewriting using views – Tuple-generating dependencies

RECOMMENDED READING

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, 1979.

[3] A. V. Aho, Y. Sagiv, and J. D. Ullman. Efficient optimization of a class of relational expressions. *ACM Trans. on Database Systems (TODS)*, 4(4):435–454, 1979.

[4] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalence of relational expressions. *SIAM J. on Computing*, 8(2):218–246, 1979.

[5] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[6] A. Deutsch, B. Ludaescher, and A. Nash. Rewriting queries using views with access patterns under integrity constraints. In *ICDT*, 2005.
[7] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, 2008.
[8] A. Deutsch, L. Popa, and V. Tannen. Physical data independence, constraints, and optimization with universal plans. In *VLDB*, pages 459–470, 1999.
[9] A. Deutsch and V. Tannen. XML queries and constraints, containment and reformulation. *Theor. Comput. Sci. (TCS)*, 336(1):57–87, 2005. Preliminary version in ICDT 2003.
[10] R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.
[11] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theor. Comput. Sci. (TCS)*, 336(1):89–124, 2005. Preliminary version in ICDT 2003.
[12] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM Trans. Database Syst. (TODS)*, 31(4):1454–1498, 2006. Preliminary version in PODS 2005.
[13] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *ICDT*, pages 332–347, 1999.
[14] Maier, Sagiv, and Yannakakis. On the complexity of testing implication of functional and join dependencies. *J. ACM*, 1981.
[15] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.
[16] L. Popa and V. Tannen. An equational chase for path-conjunctive queries, constraints, and views. In *ICDT*, pages 39–57, 1999.
[17] M. Vardi. Inferring multivalued dependencies from functional and join dependencies. *Acta Informatica*, 1983.