These lecture notes include some material from Professors Guagliardo, Bertossi, Kolaitis, Libkin, Vardi, Barland, McMahan

Database Constraints

Lecture Handout

Dr Eugenia Ternovska

Simon Fraser University

Database Constraints

Updates to databases can be wrong in a variety of ways. For example,

- There may be typographical and transcription errors
- Data values may be outside of an allowed range
- An interrelationship between data values can be violated

These constraints can be written at the application level, however it is better to store these checks in the database and let the DBMS administer them. Duplication is avoided and the check is never "forgotten".

Support for integrity constraints is present in most DBMSs, but support for "checks", "assertions" and "triggers" is considerably weaker.

Integrity constraints

A **constraint** is a relationship among data elements that the DBMS is required to enforce.

Kinds of Constraints

- Keys, functional dependencies (FDs).
- Foreign-keys, inclusion dependencies (INDs) or referential dependencies.
- Value-based constraints. Constrain values of a particular attribute.
- Tuple-based constraints. Specify relationship among components.
- Assertions: any SQL boolean expression.

Instances that satisfy the constraints are called **legal**

Many kinds of constraints can be expressed in Relational Algebra.

Relational Algebra as a Constraint Language

There are two ways in which we can use expressions of relational algebra to express constraints.

- If R is an expression of relational algebra, then $R = \varnothing$ is a constraint that says "The value of R must be empty," or, equivalently, "There are no tuples in the result of R."
- If R and S are expressions of relational algebra, then $R \subseteq S$ is a constraint that says "Every tuple in the result of R must also be in the result of S." Of course the result of S may contain additional tuples not produced by R.

These ways of expressing constraints are equivalent in what they can express, but sometimes one or the other is clearer or more succinct.

The two ways of expressing constraints are equivalent:

$$R \subseteq S \quad \equiv \quad (R - S = \varnothing)$$
$$R = \varnothing \quad \equiv \quad R \subseteq \varnothing$$

Technically, $\varnothing$ is not an expression of relational algebra, but since there are expressions that evaluate to $\varnothing$, e.g. $R - R$, we may as well use it.

Equal-to-emptyset style of expressing constraints is most common in SQL, but sometimes it is easier to think in terms of set inclusion.

Constraints involving permitted values in a context

A common type of constraints

Often, quite straightforward, e.g.,
"integers only" or "strings of length 20".

A *domain constraint* for an attribute:

Example

Specify that the only legal values for the attribute BRANCH are Vancouver and Calgary.

$$\sigma_{\mathrm{branch} \neq '\mathrm{Vancouver}' \wedge \mathrm{branch} \neq '\mathrm{Calgary}'}(\mathrm{Account}) = \varnothing$$

Key constraints

A set of attributes forms a **key** for a relation if we do not allow two tuples in a relation instance to have the same values in all the attributes of the key.

$$\text{Account}(\underline{\text{AccNum}}, \underline{\text{CustID}}, \text{Balance})$$

Account

AccNum	CustID	Balance
123321	cust3	1330.00
243576	cust1	-120.00

Account

AccNum	CustID	Balance
123321	cust3	1330.00
243576	cust1	-120.00
243576	**cust1**	**654.00**

There should never be two accounts that have both the same AccNum and the same CustId.

Foreign-key constraints

Foreign-key constraints assert that a value appearing in an attribute or attributes of one relation must also appear as a value in attribute or attributes that are a key of another relation.

Example

Every value for attribute `custid` in Account must appear among the values of the **key** `custid` in Customer

$$\pi_{\text{CustID}}(\text{Account}) \subseteq \pi_{\text{CustID}}(\text{Customer})$$

Special cases of common types of dependencies: FDs, IDNs

Key constraints and foreign key constraints are special cases of more general constraints, respectively:

- Functional dependencies (FDs)
- Inclusion dependencies (INDs) (or Referential Integrity Constraints)

We will study FDs and INDs in more detail.

Functional dependencies (FDs)

Constraints of the form $X \to Y$, where X, Y are sets of attributes

Semantics

A relation R satisfies $X \to Y$ if for every two tuples $t_1, t_2 \in R$

$$\pi_X(t_1) = \pi_X(t_2) \quad \Longrightarrow \quad \pi_Y(t_1) = \pi_Y(t_2)$$

Intuition: The values for the X attributes **determine** the values for the Y attributes

Trivial FDs: $X \to Y$ where $Y \subseteq X$

Examples of FDs

Employee	Department	Manager
John	Finance	Smith
Mary	HR	Taylor
Susan	HR	Taylor
John	Sales	Smith

Which of the following FDs would the above relation satisfy?

- Department $\rightarrow$ Manager **Yes**
- Manager $\rightarrow$ Department **No**
- Employee $\rightarrow$ Department **No**
- Employee, Manager $\rightarrow$ Department **No**

Keys

Recall that a set of attributes forms a *key* for a relation if we do not allow two tuples in a relation instance to have the same values in all the attributes of the key.

Semantics

A set of attributes X is a key for relation R if for every $t_1, t_2 \in R$

$$\pi_X(t_1) = \pi_X(t_2) \implies t_1 = t_2$$

Special case of FD $X \rightarrow Y$
where Y is the **whole set of attributes** of a relation

Key constraints in relational algebra

$$\text{Account}(\underline{\text{AccNum}}, \text{CustID}, \text{Balance}, \text{Branch}) :$$

Account

AccNum	**CustID**	**Balance**	**Branch**
123321	cust3	1330.00	London
243576	cust1	-120.00	Paris

no two distinct tuples agree on the AccNum component.

Key constraints in relational algebra

Example: Express, algebraically, one of several implications of the key constraint: if two tuples agree on AccNum then they must also agree on the Balance.

Idea: if we construct all pairs of Account tuples (t_1, t_2), we must not find a pair that agree in the AccNum component and disagree in the Balance component.

$$\sigma_{\text{A1.AccNum}=\text{A2.AccNum} \wedge \text{A1.Balance} \neq \text{A2.Balance}}(A1 \times A2) = \varnothing$$

where $A1$ is shorthand for the renaming of the whole relation :

$$\rho_{A1(\text{AccNum},\text{CustID},\text{Balance},\text{Branch})}(\text{Account})$$

and $A2$ is a similar shorthand (i.e., a renaming of Account)

Inclusion dependencies (INDs)

Constraints of the form $R[X] \subseteq S[Y]$
where R, S are relations and X, Y are **sequences** of attributes

Semantics

R and S satisfy $R[X] \subseteq S[Y]$ if

for every $t_1 \in R$ there exists $t_2 \in S$ such that $\pi_X(t_1) = \pi_Y(t_2)$

Important: the projection must respect the attributes order

INDs are referential constraints: **link** the contents of one table
with the contents of another table

Foreign key: special case of IND $R[X] \subseteq S[Y]$ where Y is key for S

Examples of INDs

Employees

Name	Dep
John	Finance
Mary	HR
John	HR
Linda	Finance
Susan	Sales

Departments

Name	Mgr
Finance	John
HR	Mary
Sales	Linda

Which of the following INDs would the above relation satisfy?

- Employees[Dep] $\subseteq$ Departments[Name] **Yes**
- Employees[Name] $\subseteq$ Departments[Mgr] **No**
- Departments[Mgr] $\subseteq$ Employees[Name] **Yes**
- Departments[Mgr,Name] $\subseteq$ Employees[Name,Dep] **No**

Implication of constraints

A set Σ of constraints **implies** (or **entails**) a constraint ϕ if

every instance that satisfies Σ also satisfies ϕ

Notation: $\Sigma \models \phi$

Implication problem

Given Σ and ϕ, does Σ imply ϕ ?

Important because

- We never get the list of all constraints that hold in a database
- The given constraints may look fine, but imply some bad ones
- The given constraints may look bad, but imply only good ones

Axiomatization of constraints

Set of rules (**axioms**) to derive constraints

Sound	every derived constraint is implied
Complete	every implied constraint can be derived

Sound and complete axiomatization gives a procedure $\vdash$ such that

$$\Sigma \models \phi \quad \textbf{if and only if} \quad \Sigma \vdash \phi$$

Notation

Attributes are denoted by A, B, C, ...

If A and B are attributes, AB denotes the set $\{A, B\}$

Sets of attributes are denoted by X, Y, Z, ...

If X and Y are sets of attributes, XY denotes their union $X \cup Y$

If X is a set of attributes and A is an attribute,
XA denotes $X \cup \{A\}$

Armstrong's axioms

Sound and complete axiomatization for FDs

Essential axioms

Reflexivity: If $Y \subseteq X$, then $X \to Y$

Augmentation: If $X \to Y$, then $XZ \to YZ$ for any Z

Transitivity: If $X \to Y$ and $Y \to Z$, then $X \to Z$

Other axioms

Union: If $X \to Y$ and $X \to Z$, then $X \to YZ$

Decomposition: If $X \to YZ$, then $X \to Y$ and $X \to Z$

Closure of a set of FDs

Let F be a set of FDs
The closure F^+ of F is the set of all FDs **implied** by the FDs in F

Can be computed using Armstrong's axioms

Example
Given $F = \{A \to B, B \to C\}$

$$\begin{aligned} F^+ = F &\cup \{A \to C, AC \to BC, AB \to AC, AB \to BC\} \\ &\cup \{\text{all trivial FDs on } A, B, C\} \end{aligned}$$

Attribute closure

The closure $C_F(X)$ of a set X of attributes w.r.t. a set F of FDs
is the set of attributes we can derive from X using the FDs in F
(i.e., all the attributes A such that $F \vdash X \to A$)

Properties

- $X \subseteq C_F(X)$
- If $X \subseteq Y$, then $C_F(X) \subseteq C_F(Y)$
- $C_F\big(C_F(X)\big) = C_F(X)$

Solution to the implication problem:

$$F \models Y \to Z \quad \textbf{if and only if} \quad Z \subseteq C_F(Y)$$

Closure algorithm

Input: a set F of FDs, and a set X of attributes

Output: $C_F(X)$, the closure of X w.r.t. F

1. unused $:= F$
2. closure $:= X$
3. **while** ($(Y \to Z) \in$ unused and $Y \subseteq$ closure)

   closure $:=$ closure $\cup Z$

   unused $:=$ unused $- \{Y \to Z\}$
4. **return** closure

Example

Closure of A w.r.t. $\{AB \to C, A \to B, CD \to A\}$ (blackboard)

Keys, candidate keys, and prime attributes

Let R be a relation with set of attributes U and FDs F

$X \subseteq U$ is a key for R if $F \models X \to U$

Equivalently, X is a key if $C_F(X) = U$ (why?)

Candidate keys

Keys X such that, for each $Y \subset X$, Y is not a key

Intuitively, keys with a minimal set of attributes

Prime attribute: an attribute of a candidate key

Attribute closure and candidate keys

Given a set F of FDs on attributes U,
how do we compute all candidate keys?

1. ck $:= \varnothing$
2. $G :=$ DAG of the powerset 2^U of U
   - Nodes are elements of 2^U (sets of attributes)
   - There is an edge from X to Y if $Y \subset X$
3. Repeat until G is empty:

   Find a node X without children
   if $C_F(X) = U$:
       ck $:=$ ck $\cup \{X\}$
       Delete X and all its ancestors from G
   else:
       Delete X from G

Implication of INDs

Given a set of INDs, what other INDs can we infer from it?

Axiomatization

Reflexivity: $R[X] \subseteq R[X]$

Transitivity: If $R[X] \subseteq S[Y]$ and $S[Y] \subseteq T[Z]$, then $R[X] \subseteq T[Z]$

Projection: If $R[X,Y] \subseteq S[W,Z]$ with $|X| = |W|$,
then $R[X] \subseteq S[W]$

Permutation: If $R[A_1, \dots, A_n] \subseteq S[B_1, \dots, B_n]$,
then $R[A_{i_1}, \dots, A_{i_n}] \subseteq S[B_{i_1}, \dots, B_{1_n}]$,
where $i_1, \dots, i_n$ is a permutation of $1, \dots, n$

Sound and complete derivation procedure for INDs

FDs and INDs together

Given a set F of FDs and an FD f, we can decide whether $F \models f$

Given a set G of INDs and an IND g, we can decide whether $G \models g$

What about $F \cup G \models f$ or $F \cup G \models g$?

This problem is **undecidable**: no algorithm can solve it

What if we consider only keys and foreign keys?

The implication problem is still **undecidable**

Unary inclusion dependencies (UINDs)

INDs of the form $R[A] \subseteq S[B]$ where A, B are attributes

The implication problem for FDs and UINDs is decidable in PTIME

Acknowledgements

[1] Database Systems: The Complete Book, 2nd EditionHector Garcia-Molina, Jeffrey D. Ullman, Jennifer WidomPrentice Hall, 2009

[2] Database System Concepts, Seventh EditionAvi Silberschatz, Henry F. Korth, S. SudarshanMcGraw-Hill, March 2019www.db-book.com

Additional references and resources used in preparation of this course are listed on the course webpage or mentioned in slides.