COSC265 — Relational Database Systems

Neville Churcher

Department of Computer Science & Software Engineering
University of Canterbury

2021

Embedded Dependencies

Add *Days* — number of days a text is used by teacher for a given course.

$R = \{Course, Lecturer, Text, Days\}$ $F = \{Course, Lecturer, Text \to Days\}$

Course	**Lecturer**	**Text**	**Days**
Physics	Jones	Mechanics	7
Physics	Jones	Optics	5
Physics	Smith	Mechanics	8
Physics	Smith	Optics	4
Maths	Jones	Mechanics	3
Maths	Jones	Algebra	3
Maths	Jones	Calculus	6

- ☆ $Course \not\twoheadrightarrow Lecturer, Course \not\twoheadrightarrow Text$ in R
- ☆ But both must hold in the projection onto $R = \{Course, Lecturer, Text\}$

Definition (Embedded MVD)

Embedded MVD $X \twoheadrightarrow Y|Z$ holds in $r(R)$ if MVD $X \twoheadrightarrow Y$ holds in $\pi_{XYZ}(r)$ □

☞ $Course \twoheadrightarrow Lecturer|Text$ is an *embedded* MVD here.

Inclusion Dependencies

- Dependencies studied so far have been :

  Uni-relational: dealing with a single relation rather than inter-relation relationships

  Typed: no symbol appears in more than one column of its tableau representation (see later)

- Inclusion dependencies have neither of these properties
- An IND has the form

$$p[A_1 \ldots A_m] \subseteq q[B_1 \ldots B_m]$$

  where $p(P), q(Q)$ are relation identifiers (possibly the same) and A_i, B_i are attributes.

- If $\xi \in p(P), \zeta \in q(Q)$ then the above IND holds if

$$\forall \xi \, \exists \zeta : \xi[A_1 \ldots A_m] = \zeta[B_1 \ldots B_m]$$

Inclusion Dependencies (Continued)

- INDs tell us when values of an attribute must also be values of another (in same or *different relation*).
- A way to define FK/PK relationships

Example (Company Cars)

car_alloc	
Manager#	Car#
27	HK123
42	GP670

emp_alloc	
Emp#	Dept#
18	1
27	2
28	2
42	1

$$car\_alloc[Manager\#] \subseteq dept\_alloc[Emp\#]$$

- ☆ INDs useful for specifying when data should be duplicated
- ☆ Closely related to IS-A relationships used in EER, OO...

Inference Rules for INDs

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

Projection & Permutation: $\forall$ sequences $i_1 \ldots i_k$ of distinct integers in $1 \ldots m$

$R[A_1 \ldots A_m] \subseteq S[B_1 \ldots B_m] \models R[A_{i_1} \ldots A_{i_k}] \subseteq S[B_{i_1} \ldots B_{i_k}]$

Thus, if $\iota = R[A_1 A_2 A_3 A_4 A_5 A_6] \subseteq S[B_1 B_2 B_3 B_4 B_5 B_6]$

$$\begin{aligned} \iota &\models R[A_4] \subseteq S[B_4] \\ \iota &\models R[A_3 A_1 A_6] \subseteq S[B_3 B_1 B_6] \end{aligned}$$

Transitivity: $\{R[X] \subseteq S[Y], S[Y] \subseteq T[Z]\} \models R[X] \subseteq T[Z]$

- ☆ To date, no normal forms based on INDs have been developed.
- ☆ INDs with a single attribute on each side are *unary* (UINDs). In this special case the implication problem is solvable in polynomial time for dependency sets containing FDs and UINDs.

Generalized (Template) Dependencies

Bringing it all together

- ☆ Dependencies encountered so far have been of the form:

$$\frac{\text{“If you see this pattern”} \quad \textit{(hypothesis tuples)}}{\text{“you must also see this”} \textit{(template conclusion)}}$$

- ☆ The conclusion may be:
  - ★ another tuple—for *tuple-generating* dependencies like MVDs
  - ★ an expression—for *equality-generating* dependencies like FDs
- ☆ In tableau notation:

$$\frac{\begin{array}{l}\text{hypothesis tuple 1}\\ \text{hypothesis tuple} \ldots\\ \text{hypothesis tuple n}\end{array}}{\text{conclusion}}$$

FD & MVD Template Examples

NAIP & CLT revisited

$$\begin{array}{cccc} \multicolumn{4}{c}{NI \to P} \\ n & a_1 & i & p_1 \\ n & a_2 & i & p_2 \\ \hline \multicolumn{4}{c}{p_1 = p_2} \end{array}$$

☆ FD conclusion says values must be equal

$$\begin{array}{ccc} \multicolumn{3}{c}{C \twoheadrightarrow L} \\ c & l_1 & t_1 \\ c & l_2 & t_2 \\ \hline c & l_1 & t_2 \end{array}$$

☆ MVD conclusion says tuple must be present

☆ Conclusion tuple could have been chosen as (c, l_2, t_1) — why?

JD Templates

This looks familiar

- ☆ Revisit NAIP example
- ☆ Recall lossless-join test
- ☆ Conclusion says tuple must be present

$$\bowtie (NA, NIP)$$

$$\begin{array}{cccc}
n & a & i_1 & p_1 \\
n & a_2 & i & p \\
\hline
n & a & i & p
\end{array}$$

More Template Examples

In the following $R = ABCD$, $S = EFG$ are assumed.

FDs

$$\begin{array}{cccc} a_1 & b_1 & c_1 & d_1 \\ a_1 & b_1 & c_2 & d_2 \\ \hline \multicolumn{4}{c}{c_1 = c_2 \wedge d_1 = d_2} \end{array} \qquad AB \rightarrow CD$$

MVDs

$$\begin{array}{cccc} a_1 & b_1 & c_1 & d_1 \\ a_1 & b_2 & c_2 & d_2 \\ \hline a_1 & b_1 & c_2 & d_2 \end{array} \qquad A \twoheadrightarrow CD$$

INDs

$$\begin{array}{ccccccc} a_1 & b_1 & c_1 & d_1 & & & \\ \hline & & & & c_1 & d_1 & g \end{array} \qquad R[CD] \subseteq S[EF]$$

- ☆ In *typed* dependencies no symbol appears in more than one column. Thus FDs and MVDs are typed, while INDs are not.
- ☆ Tableaux symbols will be manipulated using techniques similar to that used to test whether decompositions had the lossless-join property.

Embedded Dependencies

- ☆ Symbols in the conclusion need not appear in the hypotheses
- ☆ Such symbols are called *unique*
- ☆ FDs are full, INDs are embedded

Definition (Embedded Dependency)

A generalized dependency is embedded if it has one or more unique symbols in its tableau representation. Otherwise it is full.

Example (Embedded MVD)

$R = \{Course, Lecturer, Text, Days\}$

$$\begin{array}{cccc} c & l_1 & t_1 & d_1 \\ c & l_2 & t_2 & d_2 \\ \hline c & l_1 & t_2 & d_3 \end{array} \qquad\qquad C \twoheadrightarrow L | T$$

d_3 is a unique symbol.

The symbols may be mapped to values such as 'Optics' and 'Prof. Smith'

Symbol Mappings

In order to determine conclusions from hypotheses a sequence of symbol mappings will be applied. These allow dependencies to be 'applied' by establishing mappings from the sets of symbols in their templates to the set of symbols in the current dependency.

Example (Symbol Mappings)

Consider the two sets of tuples $A = \{abc, ade, fbe\}$, $B = \{xyz, wyz\}$
The mapping $h(a) = x \dots$ maps all three tuples of A to the single tuple xyz of B while h' maps abc and ade to xyz and fbe to wyz

h			
	$a\ f$	$b\ d$	$c\ e$
w	x	y	z

h'			
f	a	$b\ d$	$c\ e$
w	x	y	z

□

The Chase

- ☆ Want to test $\mathcal{D} \models d$. Begin with hypotheses of d
- ☆ Apply $d_i \in \mathcal{D}$, using mappings to generate new tuples or equate symbols
- ☆ If the conclusion of d is obtained then we have a *proof* that $D \models d$
- ☆ If not, have constructed a *counter-example*
- ☆ Application of dependencies from $\mathcal{D}$ continues until no more changes can be made, or until one of the following occurs
  - ★ Can equate conclusion symbols for equality-generating dependencies
  - ★ Find a tuple that agrees with conclusion (apart from unique symbols) for tuple-generating dependencies

The Catch

If there are embedded dependencies then chase may not terminate (i.e. generating infinite counter-example). However, if it answers at all then it answers correctly.