Constraints

Hugh Darwen

hughdarwen@gmail.com
web.onetel.com/~hughdarwen/M359/

A lecture derived from HD's course at Warwick University.
Terms and concepts not used in M359 are shown in this colour.

Constraints

Constraints express the integrity rules for a database.

Enforcement of constraints by the DBMS ensures that the database is at all times in a *consistent* state.

A constraint is a *truth-valued* expression, such as a *comparison*, declared as part of the *logical schema* of the database.

The comparands of a constraint are typically relation expressions or invocations of aggregate operators.

But the commonest kinds of constraint are expressed using special shorthands, like **primary key, alternate key, foreign key, is empty**.

KEY Constraints

The **Tutorial D** constraint shown below is a "uniqueness" constraint, meaning that no two distinct tuples can match on both StudentId and CourseId.

{ StudentId, CourseId } is a *superkey* of EXAM_MARK (also a key, see next slide)

EXAM_MARK

StudentId	CourseId	Mark
S1	C1	85
S1	C2	49
S2	C1	49
S3	C3	66
S4	C1	93

((EXAM_MARK GROUP { Mark } AS Marks
WHERE COUNT (Marks) > 1) { }) = RELATION { } { }

(M359 doesn't have a counterpart of this. Please ignore!)

When a Superkey Is a Key

If no proper subset of superkey *K* is a superkey, then *K* is a *key*.

So { StudentId, CourseId } is in fact a key of EXAM_MARK, and is in fact the only key of EXAM_MARK.

In general a relvar can have several keys, in which case it is sometimes useful to nominate one of them as being the *primary key*. For that reason, keys are sometimes referred to as *candidate keys*. When a primary key is nominated, any other keys are called *alternate keys*.

The Key Shorthands

Traditionally, a KEY constraint is declared as part of the definition of the relvar to which it pertains, thus:

relation EXAM_MARK
StudentId: CHAR,
CourseId: CHAR,
Mark INTEGER }
primary key (StudentId, CourseId) ;

alternate key (…) for any additional keys

Multiple Keys

The table shown depicts part of the relation PLUS, representing the predicate $a + b = c$:

a	b	c
1	2	3
2	3	5
2	1	3

Not a variable, of course, but we can still observe that {a, b}, {a, c} and {b, c} are all keys. We might even nominate {a, b} to be the primary key (for psychological reasons only).

Degenerate Cases of Keys

The entire heading can be a key. In that case it is the only key (why?).

The empty set can be a key. In that case it is the only key (why?). What special property is implied by such a key?

“Foreign Key” Constraints

IS_CALLED

StudentId	Name
S1	Anne
S2	Boris
S3	Cindy
S4	Devinder
S5	Boris

primary key (StudentId)

IS_ENROLLED_ON

StudentId	CourseId
S1	C1
S1	C2
S2	C1
S3	C3
S4	C1

Every StudentId value here must also appear in
project IS_CALLED **over** StudentId

Inclusion Dependency

foreign key (StudentId) **referencing** IS_CALLED
included in declaration of IS_ENROLLED_ON is shorthand for:

project IS_CALLED **over** StudentId $\supseteq$
project IS_ENROLLED_ON **over** StudentId

$\supseteq$ means "is a superset of", or "includes" (not used in M359)

Such constraints in general are sometimes called *inclusion dependencies*. An inclusion dependency is a foreign key if the heading common to the two comparands is a key of the referenced relvar.

is empty example

EXAM_MARK

StudentId	CourseId	Mark
S1	C1	85
S1	C2	49
S2	C1	49
S3	C3	66
S4	C1	93

This might be subject to the constraint:
$0 \leq \text{Mark} \leq 100$

(**select** EXAM_MARK **where**
Mark < 0 **or** Mark > 100)
is empty

In M359 could be written as **constraint** Mark >= 0 **and** Mark <= 100 in declaration of EXAM_MARK.

Declaration of Inclusion Dependency

(project *r1* **over … difference project** *r2* **over …) is empty**

E.g., to express that foreign key in IS_ENROLLED_ON:

(**project** IS_ENROLLED_ON **over** StudentId
difference
project IS_CALLED **over** StudentId **) is empty**

But now the operands can be arbitrary relation expressions, without the restrictions of **foreign key**.

“Exclusion Dependency”?

(*r1* **join** *r2*) **is empty**

E.g., to enforce disjointness of part-time and full-time employees:

(PART_TIMER **join** FULL_TIMER) **is empty**