The Relational Data Model

Tutorial

SWEN 304

Trimester 2, 2019

Lecturer: Dr Hui Ma

Engineering and Computer Science

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Outline

- Relation Schema and its instances
- Key constraints
  - Definitions
  - Procedure of identify keys
- Relational database schema and its instances
- referential integrity constraints
  - Definition
  - Algorithm
- Relational Database operations and constraints

Redefining Some Terms

- Relation schema $N\,(R, C\,)$
  - N is the name, R is the set of attributes, C is the set of constraints

Student({id: String, lname: String, fname: String, major: String}, {id})

- Tuple: the set of pairs $t = \{(A_1, a_1\,), \ldots, (A_n, a_n\,)\}$, where $A_i \in R$, $a_i \in dom\,(A_i)$, and $n = |R\,|$ is $Degree(r\,(N\,))$

t= {(id: 300111), (lname: Smith), (fname: Susan), (major: COMP)}

Redefining Some Terms

- Relation schema instance $r\ (N)$: Relation over R that satisfies all constraints from C
- A relation schema instance is a set of tuples

STUDENT			
id	**lname**	**fname**	**major**
300111	Smith	Susan	COMP
300121	Bond	James	MATH
300132	Smith	Susan	COMP

- *r(Student)*= {(300111, Smith, Susan, COMP), (300121, Bond, James, Math), (300132, Smith, Susan, COMP)}

Relation Schema Key and Primary Key

- Let $N(A_1,\ldots, A_n)$ be a relation schema and $X = \{A_k,\ldots, A_m\} \subseteq \{A_1,\ldots, A_n\}$, X is a relation schema key of N, if

$1^o\ (\forall r(N))(\forall u, v \in r(N))(u[X] = v[X] \Rightarrow u = v)$ (unique)

$2^o\ (\forall Y \subset X)(\neg 1^o)$ (minimal)

$3^o\ (\forall r(N))(\forall t \in r(N))(\forall A \in X)(t[A] \neq \omega)$ (not null)

1. For all relations $r(N)$ of relation schema N, for all pairs of tuples u, v in $r(N)$, if they agree on the values over X, they are the same tuple
2. For all proper subsets Y of X, the uniqueness property does not hold
3. For all relations $r(N)$, for all tuples t in $r(N)$, for all attributes A in X, the restriction of a tuple t over A is not null

Example keys

- Example
  - CAR ({LicPlateNo, EnigineNo, Make, Model, Year}, {LicPlateNo, EnigineNo })
    - Primary key: K_p = {LicPlateNo}
  - CAR (<u>LicPlateNo</u>, EnigineNo, Make, Model, Year)
    - the primary key is underlined
  - LicPlateNo and EngineNo satisfy *unique, minimal* and *not null* properties

Key Constraints

- You are given a relation schema $N(R, C)$ and an instance $r(N)$
- If C does not contain any key, inferring keys from instances is very hard if possible at all, since there are so many of them
- If any attribute might have null values, $Null(N, A) = Y$, we can conclude that A cannot be a part of any key
- Also, from instances we may infer the subsets of attributes that are not unique and cannot be a schema key

Find Key Constraints not Violated in r(N)

a) Suppose $Null(N, A) = N$ for all attributes except F in N_2

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_2	c_1	d_2	e_1	f_2
a_2	b_1	c_2	d_1	e_2	f_3
a_1	b_3	c_3	d_1	e_1	ω
a_3	b_1	c_1	d_3	e_2	f_4

$SatKey(N_2)(r\ (N_2)) =$ {AB, CD, BCE, BDE }

1. **Produce the power set of the set of attributes that cannot be null,**
   2^5= 32 subsets

P = {{}, {A}, {B}, {C}, {D}, {E} {A, B}, {A, C}, {A, D}, {A, E}, {B, C}, {B, D}, {B, E},{C, D}, {C, E}, {D, E}, {A, B, C}, {A, B, D}, {A, B, E}, {A, C, D}, {A, C, E}, {A, D, E}, {B, C, D}, {B, C, E}, {C, D, E}, {B,D,E}, {A, B, C, D}, {A, B, C, E}, {A, C, D, E}, {A, B, C, D, E}}

2. **Check subsets for key definition satisfaction, starting from the subsets with lower cardinality.**
   - **{A, B} and {C, D} are the keys, but {A, C }, {A, D }, {A, E }, {B, C }, {B, D }, {B, E } and {C, E} cannot not be keys, so it remains to check {A, C, E}, {A, D, E}, and {B, C, E}.**
   - **{A, C, E} is not a key, neither is {A, D, E}, but {B, C, E} is a key.**

Find Key Constraints not Violated in r(N)

a) SatKey(N_2)(r (N_2)) = {AB, CD, BCE, BDE }

The subsets of attributes that satisfy the key constraints of relation schema N_2 over relation $r(N_2)$

b) Suppose now $Null(N_2, D) = Y$ and $Null(N_2, F) = Y$,
Possible schema keys
SatKey(N_2)(r (N_2)) = {AB, BCE }

Relational Database Schema

- Relational database schema $N(S, IC)$
  - N is the name,
  - $S = \{N_1(R_1, C_1),\ldots, N_k(R_k, C_k)\}$ is a set of relation schemas, and
  - IC is a set of interrelation constraints
- A database schema *DBS* as a complex data type defines a finite, but very large number of different database instances
- An instance of the relational database schema $N(S, IC)$ is $db = \{r(N_1),\ldots, r(N_k)\}$ such that:
  - Each $r(N)$ is an instance of a relation schema $N(R, C)$ in S, and
  - *db* satisfies all constraints in IC

Referential Integrity

- Given $N_1(R_1, C_1)$ and $N_2(R_2, C_2)$ with X the primary key of N_1 and $Y = \{B_1, \ldots, B_m\} \subseteq R_2$, Y is a foreign key in N_2 with regard to X in N_1
- Relations $r(N_1)$ and $r(N_2)$ satisfy the referential integrity constraint $N_2[Y] \subseteq N_1[X]$ if:

$$(\forall u \in r(N_2))(\exists v \in r(N_1))(u[Y] = v[X] \vee (\exists i \in \{1, \ldots, m\})(u[B_i] = \omega))$$

  - N_2: the referencing relation schema and
  - N_1: the referenced relation schema
- Either tuples u and v are equal on X and Y values, or there exists at least one attribute in Y whose value in the tuple u is null

Incorrect Referential Integrity Constraints

S = {BOOK(<u>ISBN</u>, Title), LIBRARY(<u>LibId</u>,LibN),
BOOK_COPIES(<u>ISBN, LibId</u>, CopNum),
BOOK_LOANS (<u>ISBN, LibId, CardNo</u>, Date),
BORROWER(<u>CardNo</u>, Name)}

IC = {BOOK_COPIES[ISBN] $\subseteq$ BOOK[ISBN],
BOOK_COPIES[LibId] $\subseteq$ LIBRARY[LibId],
BOOK_LOANS[CardNo] $\subseteq$ BORROWER[CardNo],
BOOK_LOANS [ISBN] $\subseteq$ BOOK [ISBN],
BOOK_LOANS [LibId] $\subseteq$ LIBRARY[LibId],
BOOK_LOANS [ISBN] $\subseteq$ BOOK_COPIES [ISBN],
BOOK_LOANS [LibId] $\subseteq$ BOOK_COPIES [LibId]
}

Are the constraints correct?

A Consequence of Incorrect Instance

BOOK

ISBN	Title
1010	DB Sys
9999	Comp

LIBRARY

LibId	LibN
1	Vic
9	Massey

BORROWER

CardNo	Name
10	Susan
20	James

BOOK_COPIES

ISBN	LibId	NoOfCop
1010	1	10
9999	1	15
9999	9	5

BOOK_LOANS

ISBN	LibId	CardNo	Date
1010	1	10	01.03.01
9999	1	10	15.07.00
1010	9	20	01.03.01

IC = {BOOK_COPIES[ISBN] $\subseteq$ BOOK [ISBN],
BOOK_COPIES [LibId] $\subseteq$ LIBRARY [LibId],
BOOK_LOANS [CardNo] $\subseteq$ BORROWER [CardNo],
BOOK_LOANS [ISBN] $\subseteq$ BOOK [ISBN],
BOOK_LOANS [LibId] $\subseteq$ LIBRARY [LibId],
BOOK_LOANS [ISBN] $\subseteq$ BOOK_COPIES [ISBN],
BOOK_LOANS [LibId] $\subseteq$ BOOK_COPIES [LibId]

A Consequence of Incorrect RI

BOOK

ISBN	Title
1010	DB Sys
9999	Comp

LIBRARY

LibId	LibN
1	Vic
9	Massey

BORROWER

CardNo	Name
10	Susan
20	James

BOOK_COPIES

ISBN	LibId	NoOfCop
1010	1	10
9999	1	15
9999	9	5

BOOK_LOANS

ISBN	LibId	CardNo	Date
1010	1	10	01.03.01
9999	1	10	15.07.00
1010	9	20	01.03.01

Massey library doesn't posses the book DB Sys

Wrong tuple

Incorrect Referential Integrity Constraints

S = {BOOK($\underline{\text{ISBN}}$, Title), LIBRARY($\underline{\text{LibId}}$,LibN),
BOOK_COPIES($\underline{\text{ISBN, LibId}}$, CopNum),
BOOK_LOANS ($\underline{\text{ISBN, LibId, CardNo}}$, Date),
BORROWER($\underline{\text{CardNo}}$, Name)}

IC = {BOOK_COPIES[ISBN] $\subseteq$ BOOK[ISBN],
BOOK_COPIES[LibId] $\subseteq$ LIBRARY[LibId],
BOOK_LOANS[CardNo] $\subseteq$ BORROWER[CardNo],
BOOK_LOANS [ISBN] $\subseteq$ BOOK [ISBN],
BOOK_LOANS [LibId] $\subseteq$ LIBRARY [LibId],
BOOK_LOANS [ISBN] $\subseteq$ BOOK_COPIES [ISBN],
BOOK_LOANS [LibId] $\subseteq$ BOOK_COPIES [LibId],
BOOK_LOANS[(ISBN, LibId)] $\subseteq$ BOOK_COPIES[(ISBN, LibId)]
}

redundant

wrong

missing

Inferring Referential Integrities

- Convince yourself (by thinking) that the following implication is true

$$(BOOK\_COPIES\ [ISBN\ ] \subseteq BOOK\ [ISBN\ ] \wedge$$
$$BOOK\_COPIES\ [LibId\ ] \subseteq LIBRARY\ [LibId\ ] \wedge$$
$$BOOK\_LOANS\ [(ISBN,\ LibId\ )] \subseteq BOOK\_COPIES\ [(ISBN,\ LibId\ )]\ )$$
$$\models$$
$$(BOOK\_LOANS\ [ISBN\ ] \subseteq BOOK\ [ISBN\ ] \wedge$$
$$BOOK\_LOANS\ [LibId\ ] \subseteq LIBRARY\ [\mathit{LibId}\ ]\ )$$

- But also note that:

$$\neg((BOOK\_COPIES\ [ISBN\ ] \subseteq BOOK\ [ISBN\ ] \wedge$$
$$BOOK\_COPIES\ [LibId\ ] \subseteq LIBRARY\ [LibId\ ] \wedge$$
$$BOOK\_LOANS\ [ISBN\ ] \subseteq BOOK\ [ISBN\ ] \wedge$$
$$BOOK\_LOANS[LibId\ ] \subseteq LIBRARY\ [LibId\ ]\ ) \models$$
$$(BOOK\_LOANS\ [(ISBN,\ LibId\ )] \subseteq BOOK\_COPIES\ [(ISBN,\ LibId\ )]\ ))$$

Correct Referential Integrity Constraints

S = {BOOK (<u>ISBN</u>, Title), LIBRARY (<u>LibId</u>,LibN),
BOOK_COPIES (<u>ISBN, LibId</u>, NoOfCop),
BOOK_LOANS(<u>ISBN, LibId, CardNo</u>, Date),
BORROWER (<u>CardNo</u>, Name) }

IC = {BOOK_COPIES [ISBN] $\subseteq$ BOOK [ISBN],
BOOK_COPIES [LibId] $\subseteq$ LIBRARY [LibId],
BOOK_LOANS [(ISBN, LibId)] $\subseteq$
BOOK_COPIES [(ISBN, LibId)],
BOOK_LOANS [CardNo] $\subseteq$ BORROWER [CardNo] }

Extending Library Schema

- Suppose we want to keep track about customers requesting books that do not exist in a library
- We extend the Library schema by the relation schema

  REQ_BOOK({CardNo, ISBN, LibId, ReqDate},
  {CardNo + ISBN + LibId })
- … and add the referential integrity constraints:

  REQ_BOOK [(ISBN, LibId)] $\subseteq$ BOOK_COPIES [(ISBN, LibId)]

  REQ_BOOK [CardNo] $\subseteq$ BORROWER [CardNo]

- Is the referential integrity

  REQ_BOOK [(ISBN, LibId)] $\subseteq$ BOOK_COPIES [(ISBN, LibId)]

  correct?

Relational Database Operations

- Database Management System must implement **update** operations:
  - insert,
  - delete, and
  - modify

- Database Management System must implement **retrieval** operations:
  - query language
  - Need a well defined language

DB Updates and Constraints

- No update operation should leave a database in an inconsistent state (with violated constraints)
- A DBMS must take the actions necessary to prevent a constraint violation:
  - **reject**: do not allow the operation
  - **cascade**: propagate the operation by making necessary consequential changes
  - **set null**, or **set default**: reset other values to maintain consistency

Inserts and Constraint Violations

- Inserting a new tuple could violate
  - Attribute/domain constraints
    (a value is not of the right type or within the required range)
  - Uniqueness constraints
    (the values of the key attributes duplicate another tuple)
  - Not Null constraints
    (an attribute has the value null when it shouldn’t)
  - Referential Integrity constraints
    (the values of the attributes of a foreign key do not match any tuple in the other relation)
- Response:
  - **Reject** the operation – there is no change that the DBMS system could safely make to resolve the inconsistency

Deletes and Constraint Violations

- Deleting a tuple can only violate a referential integrity constraint:
  - If a tuple t is referred to by foreign keys in some tuples t_1, t_2, … t_n in other relations, then deleting t will make t_1, t_2, … t_n inconsistent.
  - Example:
    - Delete a student record from the database, and all their grade records will refer to nothing
- There are several options:
  - **Reject** the deletion
  - **Set null / set default**: insert null or a default value in the *foreign key* attributes of tuples in other relation(s) that refer to t (can't do set null if foreign key attributes are NOT NULL)
  - **Cascade**: delete tuples in other relation(s) that refer to t (appropriate only if the other tuples "existentially depend" on t)

Modify and Constraint Violations

- Modifying/updating the values of attributes in a tuple may violate constraints
  - Attribute/domain constraints
    Response: **reject** (like insert)
  - Key constraints (if attribute is part of a key)
    Response: treat as a *delete followed by an insert*
  - Referential integrity constraints (if attribute is part of a foreign key).
    Response: **reject** (like insert), or **cascade**, or **set null,** or **set default** (like delete)

DB Updates and Constraints

Update operation	Domain / Attribute constraint	Key / Entity integrity constraint,	Referential integrity
insert	reject	reject	reject
delete	no violation	no violation	reject, cascade, set null, set default
modify	reject	reject	reject, cascade, set null, set default

A Question for You

- Consider the following database instance

TEXTBOOK			
Title	ISBN	Pcod	Pnum
COD	1111	COMP	203
FDBS	2222	COMP	ω

COURSE		
Pcode	Pnum	Pname
COMP	203	CO
COMP	302	DBS

- Should a DBMS reject the following update operation: (Y/N)?

UPDATE TEXTBOOK SET PNum = 302 WHERE ISBN = 2222;

N

- Should a DBMS reject the following update operation: (Y/N)?

UPDATE TEXTBOOK SET PNum = 302 WHERE ISBN = 1111;

N

A Question for You

- Consider the following database instance

TEXTBOOK			
Title	**ISBN**	**Pcod**	**Pnum**
COD	1111	COMP	203
FDBS	2222	COMP	ω

COURSE		
Pcode	**Pnum**	**Pname**
COMP	203	CO
COMP	302	DBS

- Should a DBMS reject the following update operation: (Y/N)?

`UPDATE TEXTBOOK SET PNum = 403 WHERE ISBN = 2222;`

Y

- Should a DBMS reject the following update operation: (Y/N)?

`UPDATE COURSE SET PNum = 102 WHERE Pname = 'CO';`

Y/N

A Question for You

- Consider the following database instance

TEXTBOOK			
Title	**<u>ISBN</u>**	**Pcod**	**Pnum**
COD	1111	COMP	203
FDBS	2222	COMP	ω

COURSE		
<u>Pcode</u>	**<u>Pnum</u>**	**Pname**
COMP	203	CO
COMP	302	DBS

- Should a DBMS reject the following update operation: (Y/N)?

```
UPDATE TEXTBOOK SET Pcode = 'SWEN' WHERE ISBN = 2222;
```

N

Renaming Attributes with Different Roles (H)

- The referential integrity

  REQ_BOOK [(ISBN, LibId)] $\subseteq$ BOOK_COPIES [(ISBN, LibId)]

  is incorrect:

  - The attributes REQ_BOOK.ISBN and BOOK_COPIES.ISBN have different meanings
  - For a given LibId value, REQ_BOOK.ISBN and BOOK_COPIES.ISBN have disjoint sets of values
    - REQ_BOOK.ISBN are ISBNs of books not yet in the library
    - BOOK_COPIES.ISBN are ISBNs of books already in the library

- Instead we use the referential integrity constraints

  REQ_BOOK [ISBN] $\subseteq$ BOOK [ISBN]

  REQ_BOOK [LibId] $\subseteq$ LIBRARY [LibId]

to ensure that new books to be purchased are first recorded in the BOOK table (for bookkeeping) and are requested for existing libraries only

Improving Extended Library Schema (H)

- After the correction we have the relation schema
  REQ_BOOK({CardNo, ISBN, LibId, ReqDate},
  {CardNo + ISBN + LibId})

- ... and the referential integrity constraints:
  $REQ\_BOOK[ISBN] \subseteq BOOK[ISBN]$
  $REQ\_BOOK[LibId] \subseteq Library[LibId]$
  $REQ\_BOOK[CardNo] \subseteq BORROWER[CardNo]$