More on the Relational Data Model
Enforcing Integrity

CS1F

IM Lecture 6

Craig Macdonald

Overview

- More on the Relational Model
- Integrity

- Next lectures
  - Sets and Relations
  - Relational Algebra

Reminder: ER -> Schema

- Strong entities
  - build a table with columns for each attribute
- Weak entities
  - build a table with columns for each attribute
  - Add the PK of the owner entity as FK; composite PK
- Relationships
  - 1-1– FK any side
  - 1-N – FK N side
  - N-M – new table containing FKs; composite PK

THE RELATIONAL MODEL

4

The Relational Model

5

- Recall: In the relational model of a database, all data is represented in terms of tuples, grouped into relations.
- A database organised in terms of the relational model is a relational database.

A relation

6

- A relation may be thought of as a 2D table

STUDENT

name	Student ID	exam1	exam2
Gedge	891023	12	58
Kerr	892361	66	90
Fraser	880123	50	65

- A relation has
  - a **name - STUDENT**
  - In the header: an unchanging set of **columns** which are *named* and *typed (domain)* – i.e. the scheme
    - Student(name, StudentID, exam1, exam2)
  - In the body: a time varying set of **tuples**, which are the current set of **records** for the relation

Degree and Cardinality of a Relation

7

STUDENT

name	matric	exam1	exam2
Gedge	891023	12	58
Kerr	892361	66	90
Fraser	880123	50	65

- The relation student has:
  - Degree of 4 (number of attributes/columns)
  - Cardinality of 3 (number of rows/tuples)

GOTCHA: Do not confuse this with the cardinality of a relationship type in an E/R diagram

Characteristics of the Relational Model (1)

8

- **A relation is a set of tuples**
  - e.g. {<Fraser,880123,50,65>, ….}
- **There are no duplicate tuples**
  - because the tuples form a set
  - this must be checked when
    - a new tuple is added
    - a value is modified
    - a new relation is created as a restriction of an old one
- This implies that **a primary key always exists**
  - Worst case = a key composed of all of the attributes

Characteristics of the Relational Model (2)

9

- **The tuples are unordered**
  - a property of sets
  - no meaning is imposed on the relation (we could use any order)
  - a table is only one possible representation of a relation
  - physical storage of a relation must however have an order

Characteristics of the Relational Model (3)

10

- **Attributes are also an unordered set**
  - nothing can be inferred from the way we write the set of names or draw a table representation
  - no notion of getting the first attribute, next attribute, etc.
    - no first tuple, next tuple for that matter

Characteristics of the Relational Model (4)

11

- **All values are atomic**
  - no attribute can have a value which is either a set or is decomposable
  - So NO MULTI-VALUED attributes

S#	PQ
S1	{(P1,200),(P2,300)}

must become

S#	P	Q
S1	P1	200
S1	P2	300

- This simplifies the model
  - called the **First Normal Form Assumption**
  - normalisation covered in later years

Characteristics of the Relational Model (5)

12

- Relations represent
  - an entity type and its attributes
  - a relationship
  - a set of values
- Unknown values must be represented
  - replaced by **NULL**

NULL Values

13

- When adding a new tuple, we might not have all the data we need

name	matric	exam1	exam2
Gedge	891023	12	58
Kerr	892361	66	90
Fraser	880123	50	65
Smith	882854	89	NULL

- The tuple for STUDENT Smith has a NULL value for Exam2
- This is okay if the model allows….but what about if we wanted a NULL value for matric number?
  - Primary keys cannot be allowed to take NULL values

INTEGRITY

14

Constraints on relational databases

15

- **Inherent integrity constraints:**
  - must hold for all relational databases
  - typically enforced by DBMS
- **Enterprise constraints:**
  - specific to a particular application

Integrity constraints

16

- Primary key values must be unique
- Primary key values cannot be NULL
- Foreign key values:
  - must exist in the primary key of the referenced relations – we call this **referential integrity**
  - may be NULL (if it is not a mandatory participation)
- NB: In MySQL, foreign keys are constraints that need a unique name.
  - I suggest a notation including both tables: e.g. “fk_Star_Studio”

Enterprise constraints

17

- Application dependent
- Examples:
  - specified non-key attributes must not be NULL
    e.g.: all students must have a name, even if the primary key is the student number
  - values of one attribute must be less than values in another attribute
    e.g.: age of parent must be greater than age of child
- SQL standards compliant DBMS platforms allow user-defined constraints to be checked when data is inserted/updated

Referential Integrity

18

- Concerns the use of Foreign Keys
- Guarantees that relationships between tuples are coherent
  - Every non NULL value in a Foreign Key must also exist in the relation for which it is the Primary Key

(Recap) Foreign Keys: Definition

19

- A **foreign key** is an attribute (or set of attributes) that *exist in more than one table* and which is the *primary key* for *one* of those tables.
- The foreign key is used to *cross-reference* between tables
- A foreign key is a *referential* ***constraint*** between two tables, i.e. a value in a foreign key MUST exist in the referenced primary key
- A table may have multiple foreign keys and each foreign key may reference a different table
- Foreign keys need *not* have the same attribute name across tables
  - Could have be Release(MovieTitle,Year, Length)
  - Names should differ to help readability!

Enforcing Referential Integrity

20

- Rules for each foreign key must be enforced independently by the database system
- Improper foreign key/primary key relationships or not enforcing those relationships are often the source of many database and data modeling problems

Enforcing Referential Integrity

21

- What happens if we want to update or delete a row of a table?

(A very likely database operation)

Referential Integrity: 3 Strategies

22

- **Restrict** – ban any alterations to a primary key if there are foreign key references to it
- **Cascade** - cascade the effect to all relations in all tables that refer to it
- **Set to NULL** – allow update in original table, set all corresponsing FK values to null

Referential Integrity - Examples

23

- Schema:
  - Staff(Payroll# : INT, name: VARCHAR(10))
  - Students(matric# : INT, name : VARCHAR(10), adviser : INT)
- A foreign key constraint exists, where Students.adviser refers to Staff.payroll#

STAFF

payroll#	name
8000	Johnson
8450	Gray
8556	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	8556
789	Blair	8450

What is the domain of the the adviser attribute?

Adviser : { 8000, 8450, 8856, NULL }

Referential Integrity - Examples

24

- What to do if the referent of a foreign key is changed (edited or deleted)?

STAFF

payroll#	name
8000	Johnson
8450	Gray
8556	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	8556
789	Blair	8450

EXAMPLES:
1. What if Lennon's payroll number changes to 7990?
2. What if Gray's record is deleted?

Referential Integrity – example 1

25

- What if Lennon's payroll number changes to 7990?

STAFF

payroll#	name
8000	Johnson
~~8450~~	Gray
8556	**Lennon**

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	**Smith**	**8556**
789	Blair	8450

Referential Integrity – example 1

26

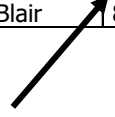
- What if Lennon's payroll number changes to 7990?

STAFF

payroll#	name
8000	Johnson
~~8450~~	Gray
7990	**Lennon**

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	**Smith**	**8556**
789	Blair	8450



referential integrity violated

Referential Integrity – example 1

27

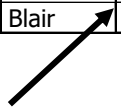
- What if Lennon's payroll number changes to 7990?

STAFF

payroll#	name
8000	Johnson
8450	Gray
7990	**Lennon**

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	**Smith**	**7990**
789	Blair	8450



solution: cascade

Referential Integrity – example 2

28

- What if Gray's record is deleted?

STAFF

payroll#	name
8000	Johnson
8450	**Gray**
7990	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	7990
789	Blair	8450

Referential Integrity – example 2

29

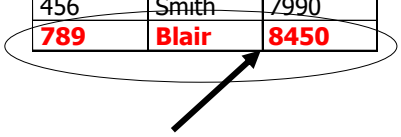
- What if Gray's record is deleted?

STAFF

payroll#	name
8000	Johnson
8450	**Gray**
7990	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	7990
789	**Blair**	**8450**



referential integrity violated

Referential Integrity – example 2

30

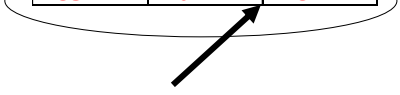
- What if Gray's record is deleted?

STAFF

payroll#	name
8000	Johnson
8450	**Gray**
7990	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	7990
789	**Blair**	**NULL**



Solution: Set to NULL

Referential Integrity

31

- What if Gray's record is delet

Error Number: 1451

Cannot delete or update a parent row: a foreign key constraint fails

STAFF

payroll#	name
8000	Johnson
8450	**Gray**
7990	Lennon

STUDENTS

matric#	name	adviser
123	Jones	NULL
456	Smith	7990
789	**Blair**	**NULL**

Solution: Restrict – raise an error, abort the transaction

Integrity in MySQL Workbench

32

- Integrity – controlled by a dialogue box
  - See Foreign Keys Tab of Table Editor

Steps for Creating a Foreign Key

33

Our Schema: Movie(Title, Type, Studio, SequelOf)
Release(MovieTitle, Year, Length)
where Release Title is a FK reference to Movie Title

1. Check that Movie's Title is a Primary Key
   - And note its datatype e.g. VARCHAR(50)

Steps for Creating a Foreign Key

34

Our Schema: Movie(Title, Type, Studio, SequelOf)
Release(MovieTitle, Year, Length)
where Release Title is a FK reference to Movie Title

2. Create all attributes in Release, including those attributes that will be Foreign Keys
   - Make sure that FKs attributes have correct datatype – it MUST be the SAME as the PK attribute you intend do refer to
   - Press Apply BEFORE creating the FK constraints

Steps for Creating a Foreign Key

35

Our Schema: `Movie(Title, Type, Studio,`
`Release(MovieTitle, Year, L`
where Release Title is a FK reference to Movie

This name must be globally unique in your DB: "Title" will not work

3. Add a Foreign Key *constraint* for Release
   - Name the constraint, e.g. "fk_release_movie"
   - Select Movie as the referenced table
   - Select MovieTitle as the (foreign key) column of this table, and Title as the referenced (primary key) column
- 4. Choose a referencing strategy (e.g. CASCADE)

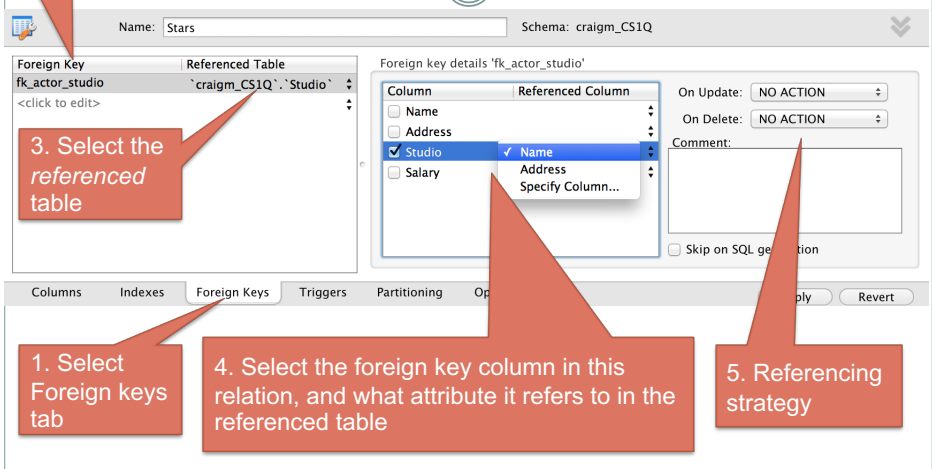
Names…

(i) the foreign key attribute *name*
(ii) the referenced table & its PK attribute *name(s)*
(iii) a *name* for the constraint itself

rkbench Table Editor: Foreign Keys

36

2. Make a name for this constraint



3. Select the *referenced* table

1. Select Foreign keys tab

4. Select the foreign key column in this relation, and what attribute it refers to in the referenced table

5. Referencing strategy

Then click 'Apply' to save the table to the DBMS

FK Checklist

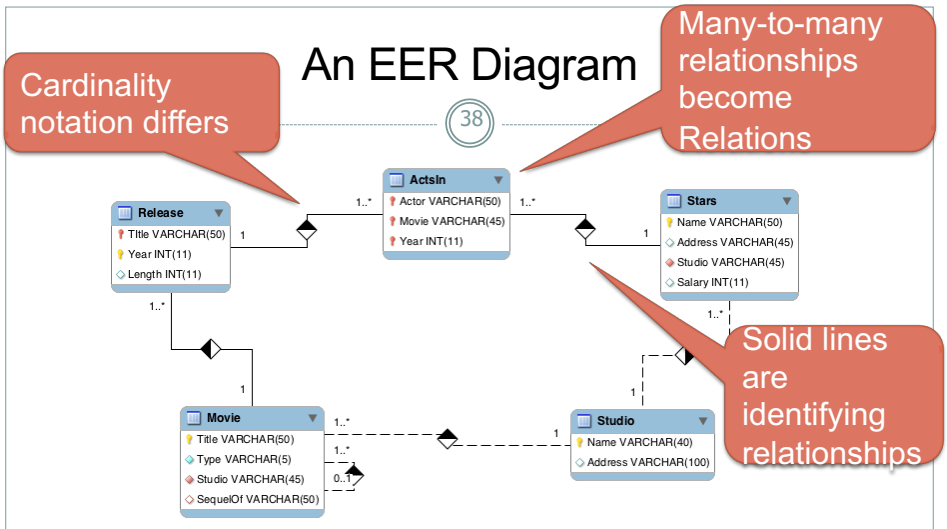
37

- Foreign Keys are difficult! If things go wrong:

1. Do not panic
2. *Read* the SQL error message, it will give a hint
3. Check…
   - … that you are referencing a PK
   - … that the datatypes of the PK and FK attributes match
   - … If you have a composite PK, you must also have a composite FK
4. *Google* the error message
5. If all else fails, close the Workbench tabs and try again

An EER Diagram

38



- Use the 'reverse engineer schema' functionality to view the relationships
  - NB: Looks slightly different from the ER diagram?

Practical Tips

39

- Create the tables in the same order as the **Translating E-R Diagram to Relational Schema** rules
  - Create Strong entities first
  - Weak entities next
  - When creating a foreign key constraint, the referenced table & column must already exist!
- Which order to create these tables?
  - Movie(Title, Type, Studio, SequelOf)
  - Release(Title, Year, Length)
  - Stars(Name, Address, Studio, Salary)
  - Studio(Name, Address)

  - 3
  - 4
  - 2
  - 1

What to do now

- This week's lab is on converting your ER diagram into tables in MySQL
  - Follow the labsheet instructions to connect to the MySQL database server
  - Use the step wise guide included in this lecture to design your relational schema, then create your tables
- Once you have created the tables you will have to create the relationships between the tables
  - This will be easier if you spend time getting the tables correct
  - Get your tutor to keep checking your tables are ok

Some Tips!

41

- Follow the stepwise guide – it works!

- Write a schema first – then go to access to build the tables
- Add the entities OWN attributes – then decide what FKs to add
- Use “Reverse Engineer Schema” to view the result
- Be careful to select good data types – they must **match** when you go to connect PKs and FKs
- When connecting relationships – think what is the referrent PK and what is referring FK