Databases 2011
Lectures 08 — 12

Timothy G. Griffin

Computer Laboratory
University of Cambridge, UK

Databases, Easter 2011

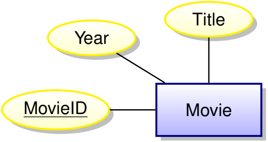
Lectures 08 and 09 : Top-Down vs Bottom-up Modeling

Outline

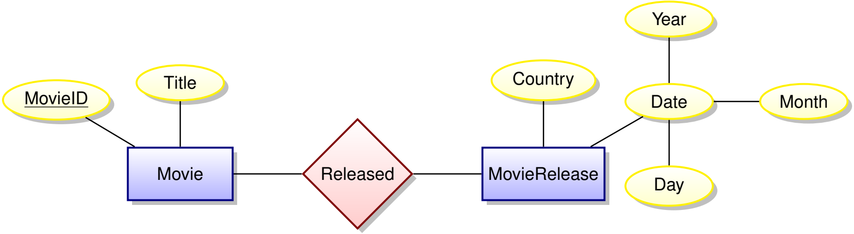
- Weak entities
- Using FDs and MVDs to refine ER models
- Another look at ternary relationships

Recall : a small change of scope ...

... changed this entity

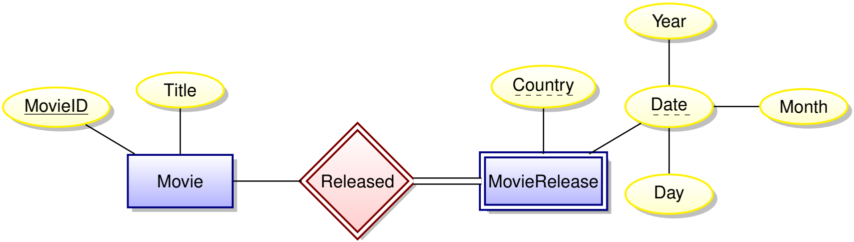


into two entities and a relationship :



But is there something odd about the MovieRelease entity?

MovieRelease represents a Weak entity set



Definition

- Weak entity sets do no have a primary key.
- The existence of a weak entity depends on an identifying entity set through an identifying relationship.
- The primary key of the identifying entity together with the weak entities discriminators (dashed underline in diagram) identify each weak entity element.

Can FDs help us think about implementation?

$$R(I, T, D, C)$$
$$I \rightarrow T$$

$$\begin{array}{rcl} I & = & \text{MovieID} \\ T & = & \text{Title} \\ D & = & \text{Date} \\ C & = & \text{Country} \end{array}$$

Turn the decomposition crank to obtain

$$R_1(I, T) \quad R_2(I, D, C)$$
$$\pi_I(R_2) \subseteq \pi_I(R_1)$$

Movie Ratings example

Scope = UK

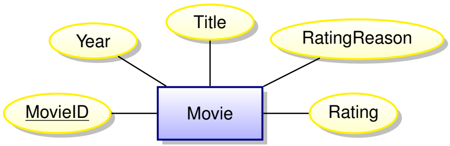
Title	Year	Rating
Austin Powers: International Man of Mystery	1997	15
Austin Powers: The Spy Who Shagged Me	1999	12
Dude, Where's My Car?	2000	15

Scope = Earth

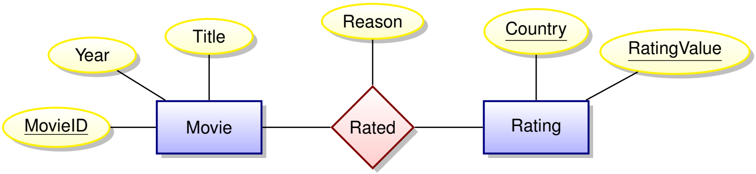
Title	Year	Country	Rating
Austin Powers: International Man of Mystery	1997	UK	15
Austin Powers: International Man of Mystery	1997	Malaysia	18SX
Austin Powers: International Man of Mystery	1997	Portugal	M/12
Austin Powers: International Man of Mystery	1997	USA	PG-13
Austin Powers: The Spy Who Shagged Me	1999	UK	12
Austin Powers: The Spy Who Shagged Me	1999	Portugal	M/12
Austin Powers: The Spy Who Shagged Me	1999	USA	PG-13
Dude, Where's My Car?	2000	UK	15
Dude, Where's My Car?	2000	USA	PG-13
Dude, Where's My Car?	2000	Malaysia	18PL

Example of attribute migrating to strong entity set

From single-country scope,



to multi-country scope:



Beware of FFDs = Faux Functional Dependencies

(US ratings)

Title	Year	Rating	RatingReason
Stoned	2005	R	drug use
Wasted	2006	R	drug use
High Life	2009	R	drug use
Poppies: Odyssey of an opium eater	2009	R	drug use

But

$$\textbf{Title} \rightarrow \{\textbf{Rating}, \ \textbf{RatingReason}\}$$

is not a functional dependency.

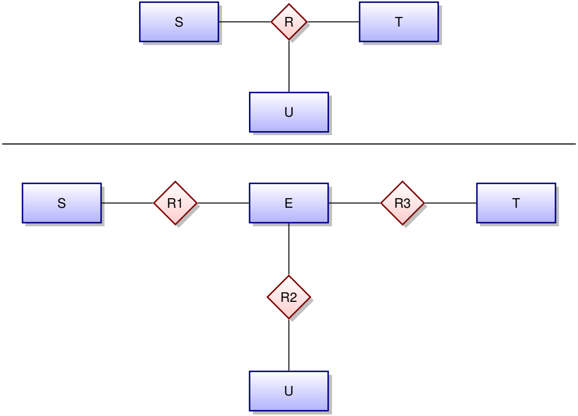
This is a mildly amusing illustration of a real and pervasive problem — deriving a functional dependency after the examination a limited set of data (or after talking to only a few domain experts).

Oh, but the real world is such a bother!

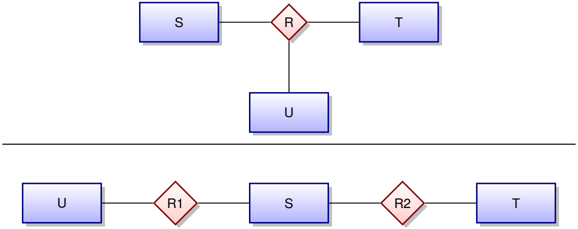
from IMDb raw data file certificates.list

```
2 Fast 2 Furious (2003) Switzerland:14 (canton of Vaud)
2 Fast 2 Furious (2003) Switzerland:16 (canton of Zurich)
28 Days (2000) Canada:13+ (Quebec)
28 Days (2000) Canada:14 (Nova Scotia)
28 Days (2000) Canada:14A (Alberta)
28 Days (2000) Canada:AA (Ontario)
28 Days (2000) Canada:PA (Manitoba)
28 Days (2000) Canada:PG (British Columbia)
```

Ternary or multiple binary relationships?

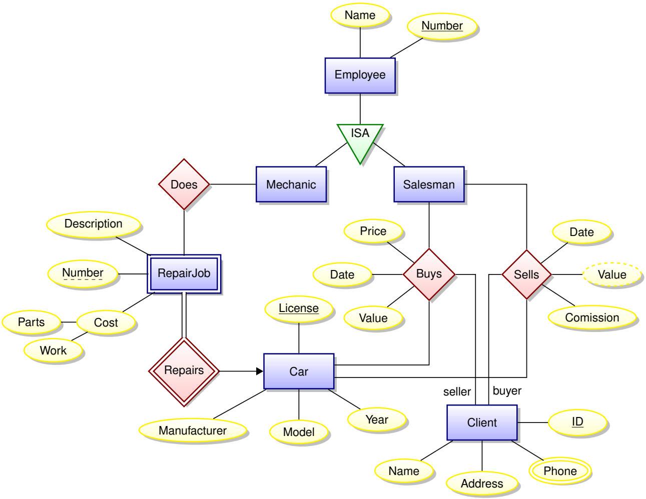


Ternary or multiple binary relationships?



Look again at ER Demo Diagram[1]

How might this be refined using FDs or MVDs?



[1] By Pável Calado, http://www.texample.net/tikz/examples/entity-relationship-diagram

Lecture 10 : Missing data and derived data in SQL

Outline

- NULL in SQL
- three-valued logic
- Multisets and aggregation in SQL
- Views
- General integrity constraints

What is `NULL` in SQL?

What if you don't know Kim's age?

```
mysql> select * from students;
   +------+--------+------+
   | sid  | name   | age  |
   +------+--------+------+
   | ev77 | Eva    |   18 |
   | fm21 | Fatima |   20 |
   | jj25 | James  |   19 |
   | ks87 | Kim    | NULL |
   +------+--------+------+
```

What is NULL?

- NULL is a place-holder, not a value!
- NULL is not a member of any domain (type),
- For records with NULL for **age**, an expression like `age > 20` must unknown!
- This means we need (at least) three-valued logic.

Let $\perp$ represent **We don't know!**

$\wedge$	**T**	**F**	$\perp$
T	**T**	**F**	$\perp$
F	**F**	**F**	**F**
$\perp$	$\perp$	**F**	$\perp$

$\vee$	**T**	**F**	$\perp$
T	**T**	**T**	**T**
F	**T**	**F**	$\perp$
$\perp$	**T**	$\perp$	$\perp$

v	$\neg v$
T	**F**
F	**T**
$\perp$	$\perp$

`NULL` can lead to unexpected results

```
mysql> select * from students;
+------+--------+------+
| sid  | name   | age  |
+------+--------+------+
| ev77 | Eva    |   18 |
| fm21 | Fatima |   20 |
| jj25 | James  |   19 |
| ks87 | Kim    | NULL |
+------+--------+------+

mysql> select * from students where age <> 19;
+------+--------+------+
| sid  | name   | age  |
+------+--------+------+
| ev77 | Eva    |   18 |
| fm21 | Fatima |   20 |
+------+--------+------+
```

The ambiguity of `NULL`

Possible interpretations of `NULL`

- There is a value, but we don't know what it is.
- No value is applicable.
- The value is known, but you are not allowed to see it.
- ...

A great deal of semantic muddle is created by conflating all of these interpretations into one non-value.

On the other hand, introducing distinct NULLs for each possible interpretation leads to very complex logics ...

Not everyone approves of `NULL`

C. J. Date [D2004], Chapter 19

"Before we go any further, we should make it very clear that in our opinion (and in that of many other writers too, we hasten to add), NULLs and 3VL are and always were a serious mistake and have no place in the relational model."

age is not a good attribute ...

The **age** column is guaranteed to go out of date! Let's record dates of birth instead!

```
create table Students
     (  sid varchar(10) not NULL,
        name varchar(50) not NULL,
        birth_date  date,
        cid varchar(3) not NULL,
        primary key (sid),
        constraint student_college foreign key (cid)
        references Colleges(cid)   )
```

age is not a good attribute ...

```
mysql> select * from Students;
+------+----------+------------+-----+
| sid  | name     | birth_date | cid |
+------+----------+------------+-----+
| ev77 | Eva      | 1990-01-26 | k   |
| fm21 | Fatima   | 1988-07-20 | cl  |
| jj25 | James    | 1989-03-14 | cl  |
+------+----------+------------+-----+
```

Use a view to recover original table

(Note : the age calculation here is not correct!)

```
create view StudentsWithAge as
  select sid, name,
   (year(current_date()) - year(birth_date)) as age,
   cid
  from Students;
```

```
mysql> select * from StudentsWithAge;
+------+----------+------+-----+
| sid  | name     | age  | cid |
+------+----------+------+-----+
| ev77 | Eva      |   19 | k   |
| fm21 | Fatima   |   21 | cl  |
| jj25 | James    |   20 | cl  |
+------+----------+------+-----+
```

Views are simply identifiers that represent a query. The view's name can be used as if it were a stored table.

But that calculation is not correct ...

Clearly the calculation of age does not take into account the day and month of year.

From 2010 Database Contest (winner : Sebastian Probst Eide)

```
SELECT year(CURRENT_DATE()) - year(birth_date) -
  CASE WHEN month(CURRENT_DATE()) < month(birth_date)
  THEN 1
  ELSE
      CASE WHEN month(CURRENT_DATE()) = month(birth_date)
      THEN
          CASE WHEN day(CURRENT_DATE()) < day(birth_date)
          THEN 1
          ELSE 0
          END
      ELSE 0
      END
  END
AS age FROM Students
```

An Example ...

```
mysql> select * from marks;
   +-------+------------+------+
   | sid   | course     | mark |
   +-------+------------+------+
   | ev77  | databases  |   92 |
   | ev77  | spelling   |   99 |
   | tgg22 | spelling   |    3 |
   | tgg22 | databases  |  100 |
   | fm21  | databases  |   92 |
   | fm21  | spelling   |  100 |
   | jj25  | databases  |   88 |
   | jj25  | spelling   |   92 |
   +-------+------------+------+
```

... of duplicates

```
mysql> select mark from marks;
+------+
| mark |
+------+
|   92 |
|   99 |
|    3 |
|  100 |
|   92 |
|  100 |
|   88 |
|   92 |
+------+
```

Why Multisets?

Duplicates are important for aggregate functions.

```
mysql> select min(mark),
              max(mark),
              sum(mark),
              avg(mark)
       from marks;
+-----------+-----------+-----------+-----------+
| min(mark) | max(mark) | sum(mark) | avg(mark) |
+-----------+-----------+-----------+-----------+
|         3 |       100 |       666 |   83.2500 |
+-----------+-----------+-----------+-----------+
```

The `group by` clause

```
mysql> select course,
              min(mark),
              max(mark),
              avg(mark)
       from marks
       group by course;
+-----------+-----------+-----------+-----------+
| course    | min(mark) | max(mark) | avg(mark) |
+-----------+-----------+-----------+-----------+
| databases |        88 |       100 |   93.0000 |
| spelling  |         3 |       100 |   73.5000 |
+-----------+-----------+-----------+-----------+
```

Visualizing group by

sid	course	mark
ev77	databases	92
ev77	spelling	99
tgg22	spelling	3
tgg22	databases	100
fm21	databases	92
fm21	spelling	100
jj25	databases	88
jj25	spelling	92

$\xrightarrow{\texttt{group by}}$

course	mark
spelling	99
spelling	3
spelling	100
spelling	92

course	mark
databases	92
databases	100
databases	92
databases	88

Visualizing group by

course	mark
spelling	99
spelling	3
spelling	100
spelling	92

course	mark
databases	92
databases	100
databases	92
databases	88

$\xrightarrow{\text{min}(\textbf{mark})}$

course	min(**mark**)
spelling	3
databases	88

The `having` clause

How can we select on the aggregated columns?

```
mysql> select course,
              min(mark),
              max(mark),
              avg(mark)
       from marks
       group by course
       having min(mark) > 60;
+-----------+-----------+-----------+-----------+
| course    | min(mark) | max(mark) | avg(mark) |
+-----------+-----------+-----------+-----------+
| databases |        88 |       100 |   93.0000 |
+-----------+-----------+-----------+-----------+
```

Use renaming to make things nicer ...

```
mysql> select course,
              min(mark) as minimum,
              max(mark) as maximum,
              avg(mark) as average
       from marks
       group by course
       having minimum > 60;
+-----------+---------+---------+---------+
| course    | minimum | maximum | average |
+-----------+---------+---------+---------+
| databases |      88 |     100 | 93.0000 |
+-----------+---------+---------+---------+
```

Materialized Views

- Suppose Q is a very expensive, and very frequent query.
- Why not de-normalize some data to speed up the evaluation of Q?
  - This might be a reasonable thing to do, or ...
  - ... it might be the first step to destroying the integrity of your data design.
- Why not store the value of Q in a table?
  - This is called a materialized view.
  - But now there is a problem: How often should this view be refreshed?

General integrity constraints

- Suppose that C is some constraint we would like to enforce on our database.
- Let $Q_{\neg C}$ be a query that captures all violations of C.
- Enforce (somehow) that the assertion that is always $Q_{\neg C}$ empty.

Example

- $C = \mathbf{Z} \rightarrow \mathbf{W}$, and FD that was not preserved for relation $R(\mathbf{X})$,
- Let Q_R be a join that reconstructs R,
- Let Q'_R be this query with $\mathbf{X} \mapsto \mathbf{X}'$ and
- $Q_{\neg C} = \sigma_{\mathbf{W} \neq \mathbf{W}'}(\sigma_{\mathbf{Z} = \mathbf{Z}'}(Q_R \times Q'_R))$

Assertions in SQL

```
create view C_violations as ....

create assertion check_C
       check not (exists C_violations)
```

Lecture 11 and 12 : Relational Limitations and Alternatives

Outline

- Limits of SQL aggregation
- OLAP : Online Analytic Processing
- Data cubes
- Star schema

Limits of SQL aggregation

sale	prodId	storeId	amt
	p1	c1	12
	p2	c1	11
	p1	c3	50
	p2	c2	8

⟷

	c1	c2	c3
p1	12		50
p2	11	8	

- Flat tables are great for processing, but hard for people to read and understand.
- Pivot tables and cross tabulations (spreadsheet terminology) are very useful for presenting data in ways that people can understand.
- SQL does not handle pivot tables and cross tabulations well.

OLAP vs. OLTP

- OLTP : Online Transaction Processing (traditional databases)
  - Data is normalized for the sake of updates.
- OLAP : Online Analytic Processing
  - These are (almost) read-only databases.
  - Data is de-normalized for the sake of queries!
  - Multi-dimensional data cube emerging as common data model.
    - This can be seen as a generalization of SQL's group by

OLAP Databases : Data Models and Design

The big question

Is the relational model and its associated query language (SQL) well suited for OLAP databases?

- Aggregation (sums, averages, totals, ...) are very common in OLAP queries
  - Problem : SQL aggregation quickly runs out of steam.
  - Solution : Data Cube and associated operations (spreadsheets on steroids)
- Relational design is obsessed with normalization
  - Problem : Need to organize data well since all analysis queries cannot be anticipated in advance.
  - Solution : Multi-dimensional fact tables, with hierarchy in dimensions, star-schema design.

A very influential paper [G+1997]

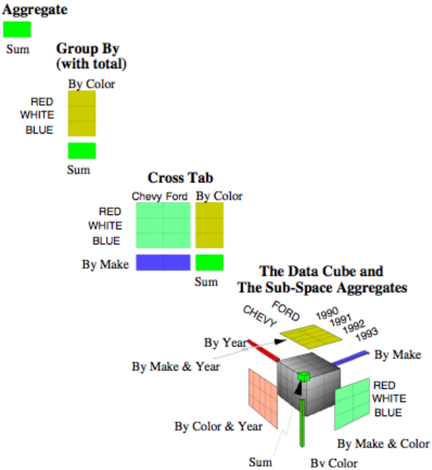
Data Mining and Knowledge Discovery 1, 29–53 (1997)

Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals*

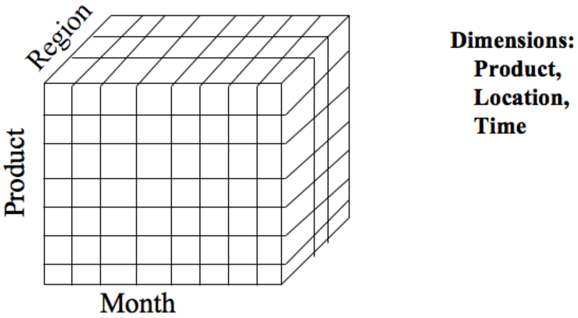
JIM GRAY Gray@Microsoft.com
SURAJIT CHAUDHURI SurajitC@Microsoft.com
ADAM BOSWORTH AdamB@Microsoft.com
ANDREW LAYMAN AndrewL@Microsoft.com
DON REICHART DonRei@Microsoft.com
MURALI VENKATRAO MuraliV@Microsoft.com
Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

FRANK PELLOW Pellow@vnet.IBM.com
HAMID PIRAHESH Pirahesh@Almaden.IBM.com
IBM Research, 500 Harry Road, San Jose, CA 95120

From aggregates to data cubes

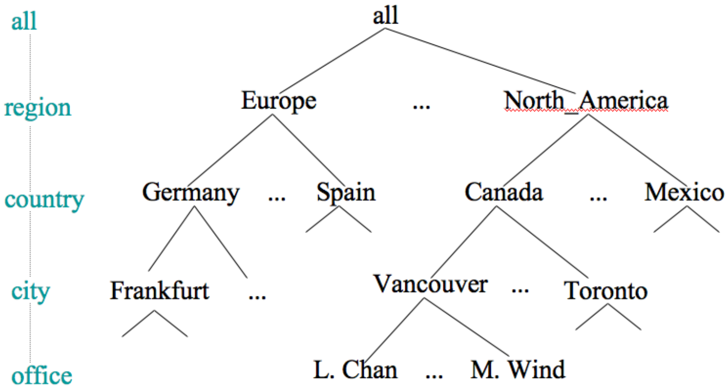


The Data Cube

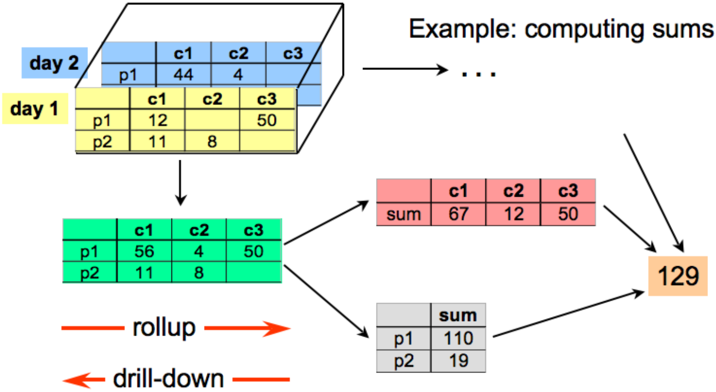


- Data modeled as an *n*-dimensional (hyper-) cube
- Each dimension is associated with a hierarchy
- Each "point" records facts
- Aggregation and cross-tabulation possible along all dimensions

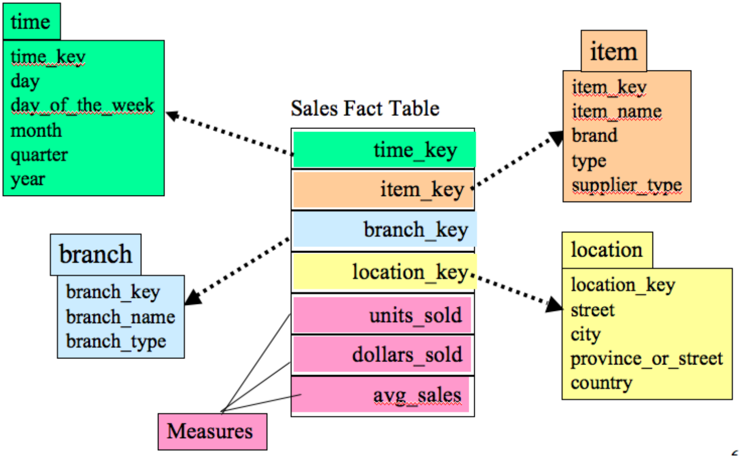
Hierarchy for **Location** Dimension



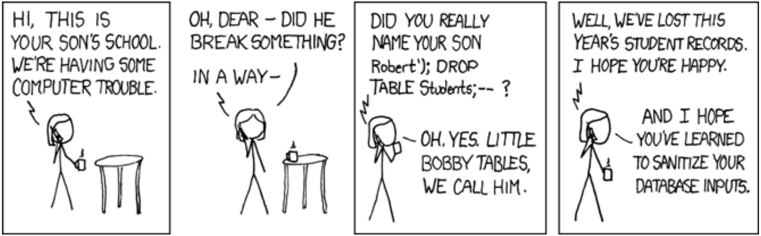
Cube Operations



The Star Schema as a design tool



The End



(`http://xkcd.com/327`)