Databases

Ken Moody

Computer Laboratory
University of Cambridge, UK

Lecture notes by Timothy G. Griffin

Lent 2012

Lecture 01 : What is a DBMS?

- DB vs. IR
- Relational Databases
- ACID properties
- Two fundamental trade-offs
- OLTP vs. OLAP
- Course outline

Example Database Management Systems (DBMSs)

A few database examples

- Banking : supporting customer accounts, deposits and withdrawals
- University : students, past and present, marks, academic status
- Business : products, sales, suppliers
- Real Estate : properties, leases, owners, renters
- Aviation : flights, seat reservations, passenger info, prices, payments
- Aviation : Aircraft, maintenance history, parts suppliers, parts orders

Some observations about these DBMSs ...

- They contains highly structured data that has been engineered to model some restricted aspect of the real world
- They support the activity of an organization in an essential way
- They support concurrent access, both read and write
- They often outlive their designers
- Users need to know very little about the DBMS technology used
- Well designed database systems are nearly transparent, just part of our infrastructure

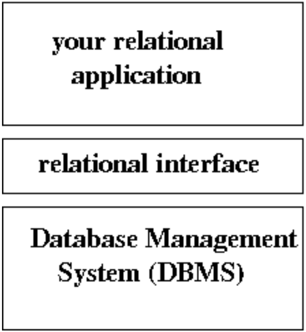
Databases vs Information Retrieval

Always ask What problem am I solving?

DBMS	IR system
exact query results	fuzzy query results
optimized for concurrent updates	optimized for concurrent reads
data models a narrow domain	domain often open-ended
generates documents (reports)	search existing documents
increase control over information	reduce information overload

And of course there are many systems that combine elements of DB and IR.

Still the dominant approach : Relational DBMSs



- The problem : in 1970 you could not write a database application without knowing a great deal about the low-level physical implementation of the data.
- Codd's radical idea [C1970]: give users a model of data and a language for manipulating that data which is completely independent of the details of its physical representation/implementation.
- This decouples development of Database Management Systems (DBMSs) from the development of database applications (at least in an idealized world).

What "services" do applications expect from a DBMS?

Transactions — ACID properties (Concurrent Systems course)

Atomicity Either all actions are carried out, or none are

- logs needed to undo operations, if needed

Consistency If each transaction is consistent, and the database is initially consistent, then it is left consistent

- **Applications designers must exploit the DBMS's capabilities.**

Isolation Transactions are isolated, or protected, from the effects of other scheduled transactions

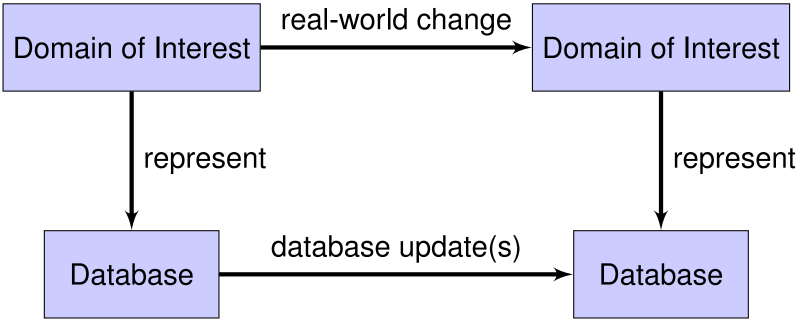
- Serializability, 2-phase commit protocol

Durability If a transactions completes successfully, then its effects persist

- Logging and crash recovery

These concepts should be familiar from Concurrent Systems and Applications.

What constitutes a good DBMS application design?



At the very least, this diagram should commute!

- Does your database design support all required changes?
- Can an update corrupt the database?

Relational Database Design

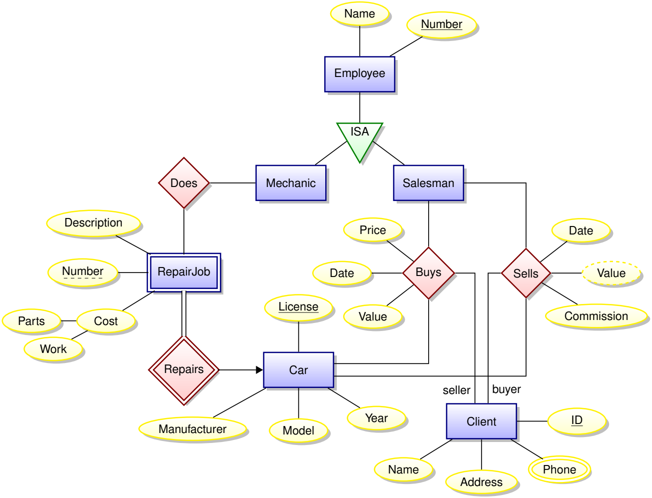
Our tools

Entity-Relationship (ER) modeling	high-level, diagram-based design
Relational modeling	formal model normal forms based on Functional Dependencies (FDs)
SQL implementation	Where the rubber meets the road

The ER and FD approaches are complementary

- ER facilitates design by allowing communication with *domain experts* who may know little about database technology.
- FD allows us formally explore general design trade-offs. Such as — **A Fundamental Trade-off in Database Design:** the more we reduce data redundancy, the harder it is to enforce some types of data integrity. (An example of this is made precise when we look at 3NF vs. BCNF.)

ER Demo Diagram (Notation follows SKS book)[1]



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

A Fundamental Trade-off in Database Implementation — Query response vs. update throughput

Redundancy is a Bad Thing.

- One of the main goals of ER and FD modeling is to reduce data redundancy. We seek *normalized* designs.
- A normalized database can support high update throughput and greatly facilitates the task of ensuring semantic consistency and data integrity.
- Update throughput is increased because in a normalized database a typical transaction need only lock a few data items — perhaps just one field of one row in a very large table.

Redundancy is a Good Thing.

- A de-normalized database can greatly improve the response time of read-only queries.
- Selective and controlled de-normalization is often required in

OLAP vs. OLTP

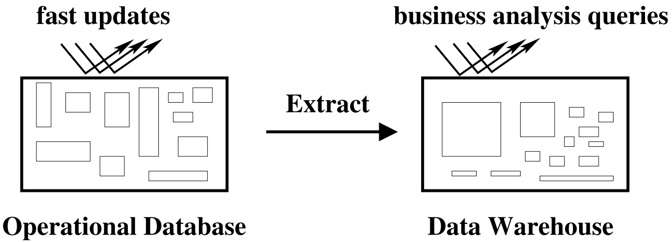
OLTP Online Transaction Processing

OLAP Online Analytical Processing

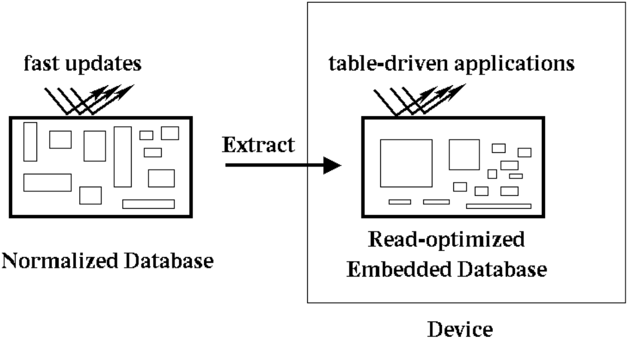
- Commonly associated with terms like Decision Support, Data Warehousing, etc.

	OLAP	OLTP
Supports	analysis	day-to-day operations
Data is	historical	current
Transactions mostly	reads	updates
optimized for	query processing	updates
Normal Forms	not important	important

Example : Data Warehouse (Decision support)

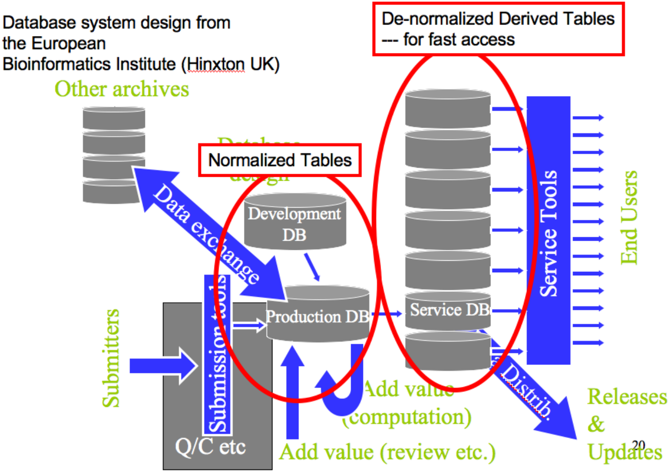


Example : Embedded databases



FIDO = Fetch Intensive Data Organization

Example : Hinxton Bio-informatics



NoSQL Movement

Technologies

- Key-value store
- Directed Graph Databases
- Main memory stores
- Distributed hash tables

Applications

- Facebook
- Google
- iMDB
- ...

Always remember to ask : What problem am I solving?

Term Outline

Recommended Reading

Textbooks

SKS **Silberschatz, A., Korth, H.F. and Sudarshan, S. (2002). Database system concepts. McGraw-Hill (4th edition).**

(Adjust accordingly for other editions)
Chapters 1 (DBMSs)
2 (Entity-Relationship Model)
3 (Relational Model)
4.1 – 4.7 (basic SQL)
6.1 – 6.4 (integrity constraints)
7 (functional dependencies and normal forms)
22 (OLAP)

UW Ullman, J. and Widom, J. (1997). A first course in database systems. Prentice Hall.

CJD Date, C.J. (2004). An introduction to database systems. Addison-Wesley (8th ed.).

Reading for the fun of it ...

Research Papers (Google for them)

C1970 E.F. Codd, (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM.

F1977 Ronald Fagin (1977) Multivalued dependencies and a new normal form for relational databases. TODS 2 (3).

L2003 L. Libkin. Expressive power of SQL. TCS, 296 (2003).

C+1996 L. Colby et al. Algorithms for deferred view maintenance. SIGMOD 199.

G+1997 J. Gray et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals (1997) Data Mining and Knowledge Discovery.

H2001 A. Halevy. Answering queries using views: A survey. VLDB Journal. December 2001.

Lecture 02 : The relational data model

- Mathematical relations and relational schema
- Using SQL to implement a relational schema
- Keys
- Database query languages
- The Relational Algebra
- The Relational Calculi (tuple and domain)
- a bit of SQL

Let's start with mathematical relations

Suppose that S_1 and S_2 are sets. The Cartesian product, $S_1 \times S_2$, is the set

$$S_1 \times S_2 = \{(s_1,\ s_2) \mid s_1 \in S_1,\ s_2 \in S_2\}$$

A (binary) relation over $S_1 \times S_2$ is any set r with

$$r \subseteq S_1 \times S_2.$$

In a similar way, if we have n sets,

$$S_1,\ S_2,\ \ldots, S_n,$$

then an *n*-ary relation r is a set

$$r \subseteq S_1 \times S_2 \times \cdots \times S_n = \{(s_1,\ s_2,\ \ldots, s_n) \mid s_i \in S_i\}$$

Relational Schema

Let **X** be a set of k attribute names.

- We will often ignore domains (types) and say that $R(\mathbf{X})$ denotes a relational schema.
- When we write $R(\mathbf{Z},\ \mathbf{Y})$ we mean $R(\mathbf{Z} \cup \mathbf{Y})$ and $\mathbf{Z} \cap \mathbf{Y} = \phi$.
- $u.[\mathbf{X}] = v.[\mathbf{X}]$ abbreviates $u.A_1 = v.A_1 \wedge \cdots \wedge u.A_k = v.A_k$.
- $\vec{\mathbf{X}}$ represents some (unspecified) ordering of the attribute names, $A_1,\ A_2,\ \ldots,\ A_k$

Mathematical vs. database relations

Suppose we have an n-tuple $t \in S_1 \times S_2 \times \cdots \times S_n$. Extracting the i-th component of t, say as $\pi_i(t)$, feels a bit low-level.

- Solution: (1) Associate a name, A_i (called an attribute name) with each domain S_i. (2) Instead of tuples, use records — sets of pairs each associating an attribute name A_i with a value in domain S_i.

A database relation R over the schema $A_1 : S_1 \times A_2 : S_2 \times \cdots \times A_n : S_n$ is a finite set

$$R \subseteq \{\{(A_1,\ s_1),\ (A_2,\ s_2),\ \ldots,(A_n,\ s_n)\} \mid s_i \in S_i\}$$

Example

A relational schema

Students(**name**: string, **sid**: string, **age** : integer)

A relational instance of this schema

$$\begin{array}{lcl} \textbf{Students} & = & \{ \\ & & \{(\textbf{name}, \text{ Fatima}), (\textbf{sid}, \text{ fm21}), (\textbf{age}, \text{ 20})\}, \\ & & \{(\textbf{name}, \text{ Eva}), (\textbf{sid}, \text{ ev77}), (\textbf{age}, \text{ 18})\}, \\ & & \{(\textbf{name}, \text{ James}), (\textbf{sid}, \text{ jj25}), (\textbf{age}, \text{ 19})\} \\ & & \} \end{array}$$

A tabular presentation

name	**sid**	**age**
Fatima	fm21	20
Eva	ev77	18
James	jj25	19

Key Concepts

Relational Key

Suppose $R(\mathbf{X})$ is a relational schema with $\mathbf{Z} \subseteq \mathbf{X}$. If for any records u and v in any instance of R we have

$$u.[\mathbf{Z}] = v.[\mathbf{Z}] \Longrightarrow u.[\mathbf{X}] = v.[\mathbf{X}],$$

then $\mathbf{Z}$ is a superkey for R. If no proper subset of $\mathbf{Z}$ is a superkey, then $\mathbf{Z}$ is a key for R. We write $R(\underline{\mathbf{Z}},\ \mathbf{Y})$ to indicate that $\mathbf{Z}$ is a key for $R(\mathbf{Z} \cup \mathbf{Y})$.

Note that this is a semantic assertion, and that a relation can have multiple keys.

Creating Tables in SQL

```
create table Students
       (sid varchar(10),
        name varchar(50),
        age int);

-- insert record with attribute names
insert into Students set
       name = 'Fatima', age = 20, sid = 'fm21';

-- or insert records with values in same order
-- as in create table
insert into Students values
       ('jj25' , 'James' ,  19),
       ('ev77' , 'Eva' ,  18);
```

Listing a Table in SQL

```
-- list by attribute order of create table
mysql> select * from Students;
+-------+---------+-------+
| sid   | name    | age   |
+-------+---------+-------+
| ev77  | Eva     |    18 |
| fm21  | Fatima  |    20 |
| jj25  | James   |    19 |
+-------+---------+-------+
3 rows in set (0.00 sec)
```

Listing a Table in SQL

```
-- list by specified attribute order
mysql> select name, age, sid from Students;
+--------+------+------+
| name   | age  | sid  |
+--------+------+------+
| Eva    |   18 | ev77 |
| Fatima |   20 | fm21 |
| James  |   19 | jj25 |
+--------+------+------+
3 rows in set (0.00 sec)
```

Keys in SQL

A key is a set of attributes that will uniquely identify any record (row) in a table.

```
-- with this create table
create table Students
       (sid varchar(10),
        name varchar(50),
        age int,
        primary key (sid));

-- if we try to insert this (fourth) student ...
mysql> insert into Students set
       name = 'Flavia', age = 23, sid = 'fm21';

 ERROR 1062 (23000): Duplicate
       entry 'fm21' for key 'PRIMARY'
```

What is a (relational) database query language?

Input : a collection of relation instances

Output : a single relation instance

$$R_1,\ R_2,\ \cdots,\ R_k \quad \Longrightarrow \quad Q(R_1,\ R_2,\ \cdots,\ R_k)$$

How can we express Q?

In order to meet Codd's goals we want a query language that is high-level and independent of physical data representation.

There are many possibilities ...

The Relational Algebra (RA)

$$
\begin{array}{rrrr}
Q & ::= & R & \text{base relation} \\
 & \mid & \sigma_p(Q) & \text{selection} \\
 & \mid & \pi_{\mathbf{X}}(Q) & \text{projection} \\
 & \mid & Q \times Q & \text{product} \\
 & \mid & Q - Q & \text{difference} \\
 & \mid & Q \cup Q & \text{union} \\
 & \mid & Q \cap Q & \text{intersection} \\
 & \mid & \rho_M(Q) & \text{renaming}
\end{array}
$$

- p is a simple boolean predicate over attributes values.
- $\mathbf{X} = \{A_1,\ A_2,\ \ldots,\ A_k\}$ is a set of attributes.
- $M = \{A_1 \mapsto B_1,\ \ A_2 \mapsto B_2,\ \ldots,\ A_k \mapsto B_k\}$ is a renaming map.

Relational Calculi

The Tuple Relational Calculus (TRC)

$$Q = \{t \mid P(t)\}$$

The Domain Relational Calculus (DRC)

$$Q = \{(A_1 = v_1,\ A_2 = v_2, \ldots, A_k = v_k) \mid P(v_1,\ v_2, \cdots,\ v_k)\}$$

The SQL standard

- Origins at IBM in early 1970's.
- SQL has grown and grown through many rounds of standardization :
  - ANSI: SQL-86
  - ANSI and ISO : SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008
- SQL is made up of many sub-languages :
  - Query Language
  - Data Definition Language
  - System Administration Language
  - ...

Selection

R

A	B	C	D
20	10	0	55
11	10	0	7
4	99	17	2
77	25	4	0

$\Longrightarrow$

$Q(R)$

A	B	C	D
20	10	0	55
77	25	4	0

RA $Q = \sigma_{A>12}(R)$

TRC $Q = \{t \mid t \in R \wedge t.A > 12\}$

DRC $Q = \{\{(A,\ a),\ (B,\ b),\ (C,\ c), (D,\ d)\} \mid \{(A,\ a),\ (B,\ b),\ (C,\ c), (D,\ d)\} \in R \wedge a > 12\}$

SQL `select * from R where R.A > 12`

Projection

R

A	B	C	D
20	10	0	55
11	10	0	7
4	99	17	2
77	25	4	0

$\Longrightarrow$

$Q(R)$

B	C
10	0
99	17
25	4

RA $Q = \pi_{B,C}(R)$

TRC $Q = \{t \mid \exists u \in R \wedge t.[B, C] = u.[B, C]\}$

DRC $Q = \{\{(B,\ b),\ (C, c)\} \mid$
$\exists\{(A,\ a),\ (B,\ b),\ (C, c),\ (D,\ d)\} \in R\}$

SQL `select distinct B, C from R`

Why the `distinct` in the SQL?

The SQL query

```
select B, C from R
```

will produce a bag (multiset)!

R

A	B	C	D
20	10	0	55
11	10	0	7
4	99	17	2
77	25	4	0

$\Longrightarrow$

$Q(R)$

B	C	
10	0	$\star\star\star$
10	0	$\star\star\star$
99	17	
25	4	

SQL is actually based on multisets, not sets. We will look into this more in Lecture 11.

Lecture 03 : Entity-Relationship (E/R) modelling

Outline

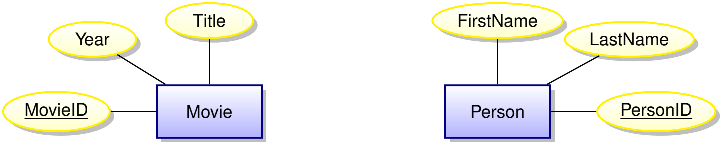
- Entities
- Relationships
- Their relational implementations
- n-ary relationships
- Generalization
- On the importance of SCOPE

Some real-world data ...

... from the Internet Movie Database (IMDb).

Title	**Year**	**Actor**
Austin Powers: International Man of Mystery	1997	Mike Myers
Austin Powers: The Spy Who Shagged Me	1999	Mike Myers
Dude, Where's My Car?	2000	Bill Chott
Dude, Where's My Car?	2000	Marc Lynn

Entities diagrams and Relational Schema



These diagrams represent relational schema

Movie(<u>*MovieID*</u>, *Title*, *Year*)

Person(<u>*PersonID*</u>, *FirstName*, *LastName*)

Yes, this ignores types ...

Entity sets (relational instances)

Movie

MovieID	Title	Year
55871	Austin Powers: International Man of Mystery	1997
55873	Austin Powers: The Spy Who Shagged Me	1999
171771	Dude, Where's My Car?	2000

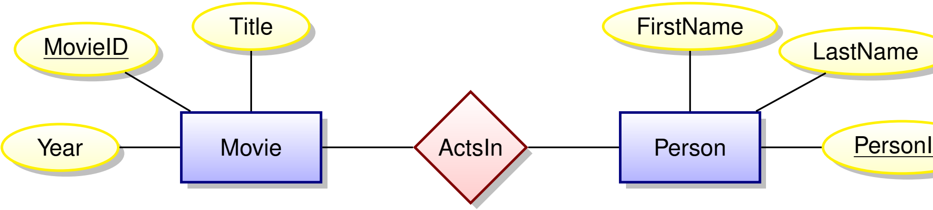
(Tim used line number from IMDb raw file movies.list as MovieID.)

Person

PersonID	FirstName	LastName
6902836	Mike	Myers
1757556	Bill	Chott
5882058	Marc	Lynn

(Tim used line number from IMDb raw file actors.list as PersonID)

Relationships



Foreign Keys and Referential Integrity

Foreign Key

Suppose we have $R(\underline{\mathbf{Z}},\ \mathbf{Y})$. Furthermore, let $S(\mathbf{W})$ be a relational schema with $\mathbf{Z} \subseteq \mathbf{W}$. We say that $\mathbf{Z}$ represents a Foreign Key in S for R if for any instance we have $\pi_{\mathbf{Z}}(S) \subseteq \pi_{\mathbf{Z}}(R)$. This is a semantic assertion.

Referential integrity

A database is said to have referential integrity when all foreign key constraints are satisfied.

A relational representation

A relational schema

$$ActsIn(\underline{MovieID}, \underline{PersonID})$$

With referential integrity constraints

$$\pi_{MovieID}(ActsIn) \subseteq \pi_{MovieID}(Movie)$$

$$\pi_{PersonID}(ActsIn) \subseteq \pi_{PersonID}(Person)$$

ActsIn

PersonID	**MovieID**
6902836	55871
6902836	55873
1757556	171771
5882058	171771

Foreign Keys in SQL

```
create table ActsIn
(  MovieID int not NULL,
   PersonID int not NULL,
   primary key (MovieID, PersonID),
   constraint actsin_movie
         foreign key (MovieID)
         references Movie(MovieID),
   constraint actsin_person
         foreign key (PersonID)
         references Person(PersonID))
```

Relational representation of relationships, in general?

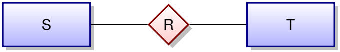
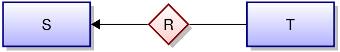
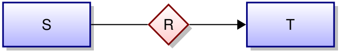
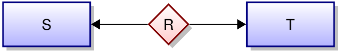
That depends ...

Mapping Cardinalities for binary relations, $R \subseteq S \times T$

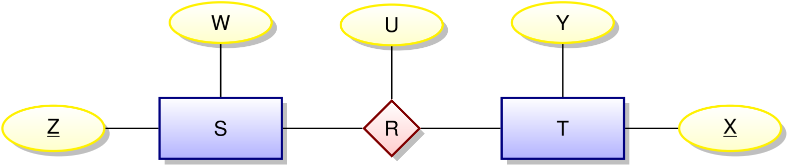
Relation R is	meaning
many to many	no constraints
one to many	$\forall t \in T, s_1, s_2 \in S.(R(s_1, t) \wedge R(s_2, t)) \implies s_1 = s_2$
many to one	$\forall s \in S, t_1, t_2 \in T.(R(s, t_1) \wedge R(s, t_2)) \implies t_1 = t_2$
one to one	one to many and many to one

Note that the database terminology differs slightly from standard mathematical terminology.

Diagrams for Mapping Cardinalities

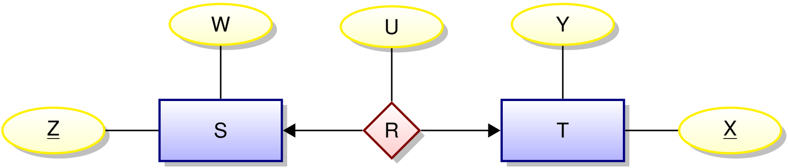
ER diagram	Relation R is
	many to many ($M : N$)
	one to many ($1 : M$)
	many to one ($M : 1$)
	one to one ($1 : 1$)

Relationships to Relational Schema



Relation R is	Schema
many to many ($M : N$)	$R(\underline{X}, \underline{Z}, U)$
one to many ($1 : M$)	$R(X, \underline{Z}, U)$
many to one ($M : 1$)	$R(\underline{X}, Z, U)$
one to one ($1 : 1$)	$R(\underline{X}, Z, U)$ and/or $R(X, \underline{Z}, U)$ (alternate keys

“one to one” does not mean a "1-to-1 correspondence”



This database instance is OK

S

Z	W
z_1	w_1
z_2	w_2
z_3	w_3

R

Z	X	U
z_1	x_2	u_1

T

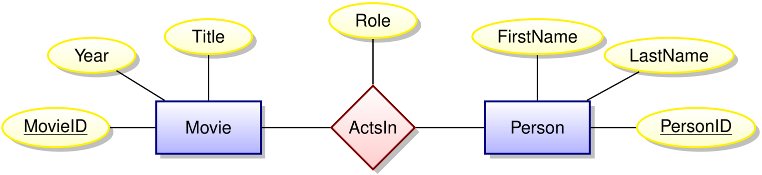
X	Y
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4

Some more real-world data ... (a slight change of SCOPE)

Title	Year	Actor	Role
Austin Powers: International Man of Mystery	1997	Mike Myers	Austin Powers
Austin Powers: International Man of Mystery	1997	Mike Myers	Dr. Evil
Austin Powers: The Spy Who Shagged Me	1999	Mike Myers	Austin Powers
Austin Powers: The Spy Who Shagged Me	1999	Mike Myers	Dr. Evil
Austin Powers: The Spy Who Shagged Me	1999	Mike Myers	Fat Bastard
Dude, Where's My Car?	2000	Bill Chott	Big Cult Guard 1
Dude, Where's My Car?	2000	Marc Lynn	Cop with Whips

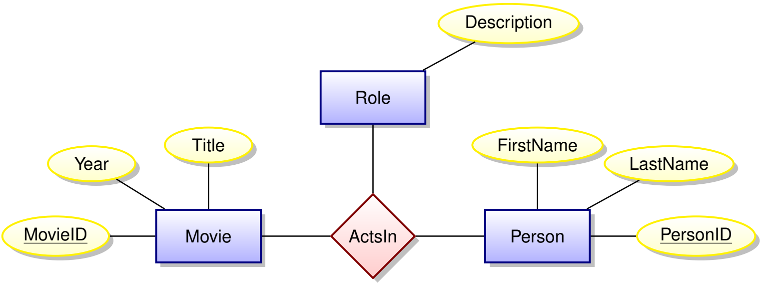
How will this change our model?

Will **ActsIn** remain a binary Relationship?



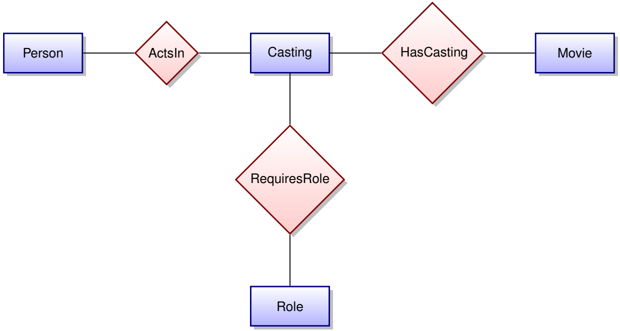
No! An actor can have many roles in the same movie!

Could **ActsIn** be modeled as a Ternary Relationship?



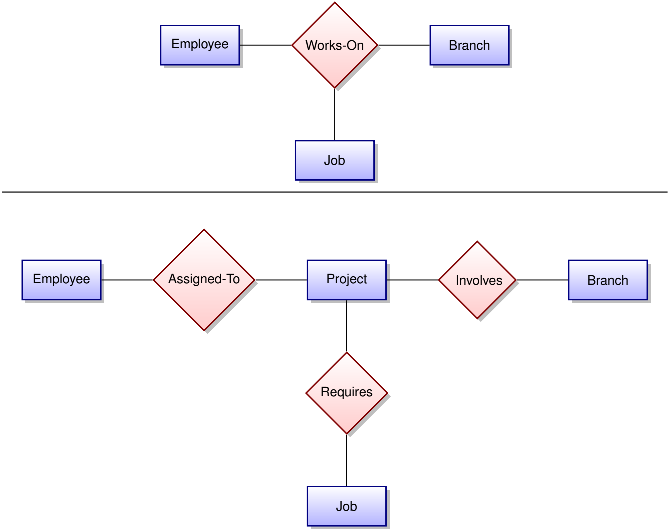
Yes, this works!

Can a ternary relationship be modeled with multiple binary relationships?

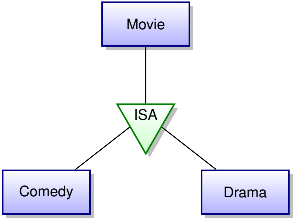


The **Casting** entity seems artificial. What attributes would it have?

Sometimes ternary to multiple binary makes more sense ...



Generalization



Questions

- Is every movie either comedy or a drama?
- Can a movie be a comedy and a drama?

But perhaps this isn't a good model ...

- What attributes would distinguish Drama and Comedy entities?
- What abound **Science Fiction**?
- Perhaps **Genre** would make a nice entity, which could have a relationship with **Movie**.

Question: What is the right model?

Answer: The question doesn't make sense!

- There is no "right" model ...
- It depends on the intended use of the database.
- What activity will the DBMS support?
- What data is needed to support that activity?

The issue of SCOPE is missing from most textbooks

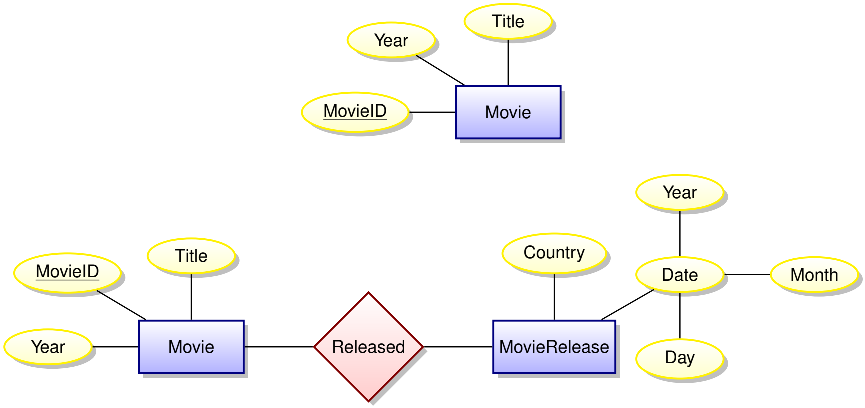
- **Suppose** that all databases begin life with beautifully designed schemas.
- **Observe** that many operational databases are in a sorry state.
- **Conclude** that the scope and goals of a database continually change, and that schema evolution is a difficult problem to solve, in practice.

Another change of SCOPE ...

Movies with detailed release dates

Title	Country	Day	Month	Year
Austin Powers: International Man of Mystery	USA	02	05	1997
Austin Powers: International Man of Mystery	Iceland	24	10	1997
Austin Powers: International Man of Mystery	UK	05	09	1997
Austin Powers: International Man of Mystery	Brazil	13	02	1998
Austin Powers: The Spy Who Shagged Me	USA	08	06	1999
Austin Powers: The Spy Who Shagged Me	Iceland	02	07	1999
Austin Powers: The Spy Who Shagged Me	UK	30	07	1999
Austin Powers: The Spy Who Shagged Me	Brazil	08	10	1999
Dude, Where's My Car?	USA	10	12	2000
Dude, Where's My Car?	Iceland	9	02	2001
Dude, Where's My Car?	UK	9	02	2001
Dude, Where's My Car?	Brazil	9	03	2001
Dude, Where's My Car?	Russia	18	09	2001

... and an attribute becomes an entity with a connecting relation.



Lecture 04 : Relational algebra and relational calculus

Outline

- Constructing new tuples!
- Joins
- Limitations of Relational Algebra

Renaming

R			
A	B	C	D
20	10	0	55
11	10	0	7
4	99	17	2
77	25	4	0

$\Longrightarrow$

$Q(R)$			
A	E	C	F
20	10	0	55
11	10	0	7
4	99	17	2
77	25	4	0

RA $Q = \rho_{\{B \mapsto E,\ D \mapsto F\}}(R)$

TRC $Q = \{t \mid \exists u \in R \wedge t.A = u.A \wedge t.E = u.E \wedge t.C = u.C \wedge t.F = u.D\}$

DRC $Q = \{\{(A,\ a),\ (E,\ b),\ (C,\ c), (F,\ d)\} \mid \exists\{(A,\ a),\ (B,\ b),\ (C,\ c), (D,\ d)\} \in R\}$

SQL `select A, B as E, C, D as F from R`

Union

R

A	B
20	10
11	10
4	99

S

A	B
20	10
77	1000

$\Longrightarrow$

$Q(R,\ S)$

A	B
20	10
11	10
4	99
77	1000

RA $Q = R \cup S$

TRC $Q = \{t \mid t \in R \vee t \in S\}$

DRC $Q = \{\{(A,\ a),\ (B,\ b)\} \mid \{(A,\ a),\ (B,\ b)\} \in R \vee \{(A,\ a), (B,\ b)\} \in S\}$

SQL `(select * from R) union (select * from S)`

Intersection

R

A	B
20	10
11	10
4	99

S

A	B
20	10
77	1000

$\Longrightarrow$

$Q(R)$

A	B
20	10

RA $Q = R \cap S$

TRC $Q = \{t \mid t \in R \wedge t \in S\}$

DRC $Q = \{\{(A,\ a),\ (B,\ b)\} \mid \{(A,\ a),\ (B,\ b)\} \in R \wedge \{(A,\ a), (B,\ b)\} \in S\}$

SQL

```
(select * from R) intersect (select * from S)
```

Difference

R

A	B
20	10
11	10
4	99

S

A	B
20	10
77	1000

$\Longrightarrow$

$Q(R)$

A	B
11	10
4	99

RA $Q = R - S$

TRC $Q = \{t \mid t \in R \wedge t \notin S\}$

DRC $Q = \{\{(A,\ a),\ (B,\ b)\} \mid \{(A,\ a),\ (B,\ b)\} \in R \wedge \{(A,\ a), (B,\ b)\} \notin S\}$

SQL `(select * from R) except (select * from S)`

Wait, are we missing something?

Suppose we want to add information about college membership to our Student database. We could add an additional attribute for the college.

```
StudentsWithCollege :
+--------+-------+------+---------+
| name   | age   | sid  | college|
+--------+-------+------+---------+
| Eva    |    18 | ev77 | King's  |
| Fatima |    20 | fm21 | Clare   |
| James  |    19 | jj25 | Clare   |
+--------+-------+------+---------+
```

Put logically independent data in distinct tables?

```
Students : +--------+------+------+-----+
           | name   | age  | sid  | cid |
           +--------+------+------+-----+
           | Eva    |   18 | ev77 | k   |
           | Fatima |   20 | fm21 | cl  |
           | James  |   19 | jj25 | cl  |
           +--------+------+------+-----+

Colleges : +-----+---------------+
           | cid | college_name  |
           +-----+---------------+
           | k   | King's        |
           | cl  | Clare         |
           | sid | Sidney Sussex |
           | q   | Queens'       |
             ...       .....
```

But how do we put them back together again?

Product

R

A	B
20	10
11	10
4	99

S

C	D
14	99
77	100

$\Longrightarrow$

$Q(R,\ S)$

A	B	C	D
20	10	14	99
20	10	77	100
11	10	14	99
11	10	77	100
4	99	14	99
4	99	77	100

Note the automatic flattening

RA $Q = R \times S$

TRC $Q = \{t \mid \exists u \in R, v \in S,\ t.[A, B] = u.[A, B] \wedge t.[C, D] = v.[C, D]\}$

DRC $Q = \{\{(A,\ a),\ (B,\ b),\ (C,\ c), (D,\ d)\} \mid \{(A,\ a),\ (B,\ b)\} \in R \wedge \{(C,\ c), (D,\ d)\} \in S\}$

SQL `select A, B, C, D from R, S`

Product is special!

R

A	B
20	10
4	99

$\Longrightarrow$

$R \times \rho_{A \mapsto C,\ B \mapsto D}(R)$

A	B	C	D
20	10	20	10
20	10	4	99
4	99	20	10
4	99	4	99

- $\times$ is the only operation in the Relational Algebra that created new records (ignoring renaming),
- But $\times$ usually creates too many records!
- Joins are the typical way of using products in a constrained manner.

Natural Join

Natural Join

Given $R(\mathbf{X},\ \mathbf{Y})$ and $S(\mathbf{Y},\ \mathbf{Z})$, we define the natural join, denoted $R \bowtie S$, as a relation over attributes $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ defined as

$$R \bowtie S \equiv \{t \mid \exists u \in R,\ v \in S,\ u.[\mathbf{Y}] = v.[\mathbf{Y}] \wedge t = u.[\mathbf{X}] \cup u.[\mathbf{Y}] \cup v.[\mathbf{Z}]\}$$

In the Relational Algebra:

$$R \bowtie S = \pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}}(\sigma_{\mathbf{Y}=\mathbf{Y}'}(R \times \rho_{\vec{\mathbf{Y}} \mapsto \vec{\mathbf{Y}'}}(S)))$$

Join example

Students

name	sid	age	cid
Fatima	fm21	20	cl
Eva	ev77	18	k
James	jj25	19	cl

Colleges

cid	cname
k	King's
cl	Clare
q	Queens'
⋮	⋮

$\pi_{\mathbf{name},\mathbf{cname}}(\text{Students} \bowtie \text{Colleges})$

$\Longrightarrow$

name	cname
Fatima	Clare
Eva	King's
James	Clare

The same in SQL

```
select name, cname
from Students, Colleges
where Students.cid = Colleges.cid
```

```
+--------+--------+
| name   | cname  |
+--------+--------+
| Eva    | King's |
| Fatima | Clare  |
| James  | Clare  |
+--------+--------+
```

Division

Given $R(\mathbf{X},\ \mathbf{Y})$ and $S(\mathbf{Y})$, the division of R by S, denoted $R \div S$, is the relation over attributes $\mathbf{X}$ defined as (in the TRC)

$$R \div S \equiv \{x \mid \forall s \in S,\ x \cup s \in R\}.$$

name	award
Fatima	writing
Fatima	music
Eva	music
Eva	writing
Eva	dance
James	dance

$\div$

award
music
writing
dance

$=$

name
Eva

Division in the Relational Algebra?

Clearly, $R \div S \subseteq \pi_{\mathbf{X}}(R)$. So $R \div S = \pi_{\mathbf{X}}(R) - C$, where C represents counter examples to the division condition. That is, in the TRC,

$$C = \{x \mid \exists s \in S,\ x \cup s \notin R\}.$$

- $U = \pi_{\mathbf{X}}(R) \times S$ represents all possible $x \cup s$ for $x \in \mathbf{X}(R)$ and $s \in S$,
- so $T = U - R$ represents all those $x \cup s$ that are not in R,
- so $C = \pi_{\mathbf{X}}(T)$ represents those records x that are counter examples.

Division in RA

$$R \div S \equiv \pi_{\mathbf{X}}(R) - \pi_{\mathbf{X}}((\pi_{\mathbf{X}}(R) \times S) - R)$$

Query Safety

A query like $Q = \{t \mid t \in R \wedge t \notin S\}$ raises some interesting questions. Should we allow the following query?

$$Q = \{t \mid t \notin S\}$$

We want our relations to be finite!

Safety

A (TRC) query

$$Q = \{t \mid P(t)\}$$

is safe if it is always finite for any database instance.

- Problem : query safety is not decidable!
- Solution : define a restricted syntax that guarantees safety.

Safe queries can be represented in the Relational Algebra.

Limitations of simple relational query languages

- The expressive power of RA, TRC, and DRC are essentially the same.
  - None can express the transitive closure of a relation.
- We could extend RA to more powerful languages (like Datalog).
- SQL has been extended with many features beyond the Relational Algebra.
  - stored procedures
  - recursive queries
  - ability to embed SQL in standard procedural languages

Lecture 05 : SQL and integrity constraints

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

$\stackrel{\texttt{group by}}{\Longrightarrow}$

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

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

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)
```

Lectures 06 : Case Study - Cancer registry for the NHS

ECRIC is a cancer registry, recording details about all tumours in people in the East of England. This data is particularly sensitive, and its use is strictly controlled. The lecture focusses on the challenges of scaling up the registration system to cover all cancer patients in England, while still maintaining the long term accuracy and continuity of the data set.

Lecture 07 : Schema refinement I

Outline

- ER is for top-down and informal (but rigorous) design
- FDs are used for bottom-up and formal design and analysis
- update anomalies
- Reasoning about Functional Dependencies
- Heath's rule

Update anomalies

Big Table

sid	name	college	course	part	term_name
yy88	Yoni	New Hall	Algorithms I	IA	Easter
uu99	Uri	King's	Algorithms I	IA	Easter
bb44	Bin	New Hall	Databases	IB	Lent
bb44	Bin	New Hall	Algorithms II	IB	Michaelmas
zz70	Zip	Trinity	Databases	IB	Lent
zz70	Zip	Trinity	Algorithms II	IB	Michaelmas

- How can we tell if an insert record is consistent with current records?
- Can we record data about a course before students enroll?
- Will we wipe out information about a college when last student associated with the college is deleted?

Redundancy implies more locking ...

... at least for correct transactions!

Big Table

sid	name	college	course	part	term_name
yy88	Yoni	New Hall	Algorithms I	IA	Easter
uu99	Uri	King's	Algorithms I	IA	Easter
bb44	Bin	New Hall	Databases	IB	Lent
bb44	Bin	New Hall	Algorithms II	IB	Michaelmas
zz70	Zip	Trinity	Databases	IB	Lent
zz70	Zip	Trinity	Algorithms II	IB	Michaelmas

- Change New Hall to Murray Edwards College
  - Conceptually simple update
  - May require locking entire table.

Redundancy is the root of (almost) all database evils

- It may not be obvious, but redundancy is also the cause of update anomalies.
- By redundancy we do not mean that some values occur many times in the database!
  - A foreign key value may be have millions of copies!
- But then, what do we mean?

Functional Dependency

Functional Dependency (FD)

Let $R(\mathbf{X})$ be a relational schema and $\mathbf{Y} \subseteq \mathbf{X}$, $\mathbf{Z} \subseteq \mathbf{X}$ be two attribute sets. We say $\mathbf{Y}$ functionally determines $\mathbf{Z}$, written $\mathbf{Y} \rightarrow \mathbf{Z}$, if for any two tuples u and v in an instance of $R(\mathbf{X})$ we have

$$u.\mathbf{Y} = v.\mathbf{Y} \rightarrow u.\mathbf{Z} = v.\mathbf{Z}.$$

We call $\mathbf{Y} \rightarrow \mathbf{Z}$ a functional dependency.

A functional dependency is a <u>semantic</u> assertion. It represents a rule that should always hold in any instance of schema $R(\mathbf{X})$.

Example FDs

Big Table

sid	name	college	course	part	term_name
yy88	Yoni	New Hall	Algorithms I	IA	Easter
uu99	Uri	King's	Algorithms I	IA	Easter
bb44	Bin	New Hall	Databases	IB	Lent
bb44	Bin	New Hall	Algorithms II	IB	Michaelmas
zz70	Zip	Trinity	Databases	IB	Lent
zz70	Zip	Trinity	Algorithms II	IB	Michaelmas

- **sid** $\rightarrow$ **name**
- **sid** $\rightarrow$ **college**
- **course** $\rightarrow$ **part**
- **course** $\rightarrow$ **term_name**

Keys, revisited

Candidate Key

Let $R(\mathbf{X})$ be a relational schema and $\mathbf{Y} \subseteq \mathbf{X}$. $\mathbf{Y}$ is a candidate key if

1. The FD $\mathbf{Y} \rightarrow \mathbf{X}$ holds, and
2. for no proper subset $\mathbf{Z} \subset \mathbf{Y}$ does $\mathbf{Z} \rightarrow \mathbf{X}$ hold.

Prime and Non-prime attributes

An attribute A is prime for $R(\mathbf{X})$ if it is a member of some candidate key for R. Otherwise, A is non-prime.

Database redundancy roughly means the existence of non-key functional dependencies!

Semantic Closure

Notation

$$F \models \mathbf{Y} \rightarrow \mathbf{Z}$$

means that any database instance that that satisfies every FD of F, must also satisfy $\mathbf{Y} \rightarrow \mathbf{Z}$.

The semantic closure of F, denoted F^+, is defined to be

$$F^+ = \{\mathbf{Y} \rightarrow \mathbf{Z} \mid \mathbf{Y} \cup \mathbf{Z} \subseteq \text{atts}(F) and \wedge F \models \mathbf{Y} \rightarrow \mathbf{Z}\}.$$

The membership problem is to determine if $\mathbf{Y} \rightarrow \mathbf{Z} \in F^+$.

Reasoning about Functional Dependencies

We write $F \vdash \mathbf{Y} \rightarrow \mathbf{Z}$ when $\mathbf{Y} \rightarrow \mathbf{Z}$ can be derived from F via the following rules.

Armstrong's Axioms

Reflexivity If $\mathbf{Z} \subseteq \mathbf{Y}$, then $F \vdash \mathbf{Y} \rightarrow \mathbf{Z}$.

Augmentation If $F \vdash \mathbf{Y} \rightarrow \mathbf{Z}$ then $F \vdash \mathbf{Y}, \mathbf{W} \rightarrow \mathbf{Z}, \mathbf{W}$.

Transitivity If $F \vdash \mathbf{Y} \rightarrow \mathbf{Z}$ and $F \models \mathbf{Z} \rightarrow \mathbf{W}$, then $F \vdash \mathbf{Y} \rightarrow \mathbf{W}$.

Logical Closure (of a set of attributes)

Notation

$$\text{closure}(F,\ \mathbf{X}) = \{A \mid F \vdash \mathbf{X} \rightarrow A\}$$

Claim 1

If $\mathbf{Y} \rightarrow \mathbf{W} \in F$ and $\mathbf{Y} \subseteq \text{closure}(F,\ \mathbf{X})$, then $\mathbf{W} \subseteq \text{closure}(F,\ \mathbf{X})$.

Claim 2

$\mathbf{Y} \rightarrow \mathbf{W} \in F^+$ if and only if $\mathbf{W} \subseteq \text{closure}(F,\ \mathbf{Y})$.

Soundness and Completeness

Soundness

$$F \vdash f \implies f \in F^+$$

Completeness

$$f \in F^+ \implies F \vdash f$$

Proof of Completeness (soundness left as an exercise)

Show $\neg(F \vdash f) \implies \neg(F \models f)$:

- Suppose $\neg(F \vdash \mathbf{Y} \rightarrow \mathbf{Z})$ for $R(\mathbf{X})$.
- Let $\mathbf{Y}^+ = \text{closure}(F, \ \mathbf{Y})$.
- $\exists B \in \mathbf{Z}$, with $B \notin \mathbf{Y}^+$.
- Construct an instance of R with just two records, u and v, that agree on $\mathbf{Y}^+$ but not on $\mathbf{X} - \mathbf{Y}^+$.
- By construction, this instance does not satisfy $\mathbf{Y} \rightarrow \mathbf{Z}$.
- But it does satisfy F! Why?
    - let $\mathbf{S} \rightarrow \mathbf{T}$ be any FD in F, with $u.[\mathbf{S}] = v.[\mathbf{S}]$.
    - So $\mathbf{S} \subseteq \mathbf{Y}+$. and so $\mathbf{T} \subseteq \mathbf{Y}+$ by claim 1,
    - and so $u.[T] = v.[T]$

Closure

By soundness and completeness

$$\text{closure}(F,\ \mathbf{X}) = \{A \mid F \vdash \mathbf{X} \to A\} = \{A \mid \mathbf{X} \to A \in F^+\}$$

Claim 2 (from previous lecture)

$\mathbf{Y} \to \mathbf{W} \in F^+$ if and only if $\mathbf{W} \subseteq \text{closure}(F,\ \mathbf{Y})$.

If we had an algorithm for closure$(F,\ \mathbf{X})$, then we would have a (brute force!) algorithm for enumerating F^+:

F^+

- for every subset $\mathbf{Y} \subseteq \text{atts}(F)$
  - for every subset $\mathbf{Z} \subseteq \text{closure}(F,\ \mathbf{Y})$,
    - output $\mathbf{Y} \to \mathbf{Z}$

Attribute Closure Algorithm

- Input : a set of FDs F and a set of attributes $\mathbf{X}$.
- Output : $\mathbf{Y} = \text{closure}(F, \mathbf{X})$

1. $\mathbf{Y} := \mathbf{X}$
2. while there is some $\mathbf{S} \rightarrow \mathbf{T} \in F$ with $\mathbf{S} \subseteq \mathbf{Y}$ and $\mathbf{T} \not\subseteq \mathbf{Y}$, then $\mathbf{Y} := \mathbf{Y} \cup \mathbf{T}$.

An Example (UW1997, Exercise 3.6.1)

$R(A, B, C, D)$ with F made up of the FDs

$$A, B \rightarrow C$$
$$C \rightarrow D$$
$$D \rightarrow A$$

What is F^+?

Brute force!

Let's just consider all possible nonempty sets **X** — there are only 15...

Example (cont.)

$$F = \{A, B \to C, \quad C \to D, \quad D \to A\}$$

For the single attributes we have

- $\{A\}^+ = \{A\}$,
- $\{B\}^+ = \{B\}$,
- $\{C\}^+ = \{A,\ C,\ D\}$,
  - $\{C\} \stackrel{C \to D}{\Longrightarrow} \{C,\ D\} \stackrel{D \to A}{\Longrightarrow} \{A,\ C,\ D\}$
- $\{D\}^+ = \{A,\ D\}$
  - $\{D\} \stackrel{D \to A}{\Longrightarrow} \{A,\ D\}$

The only new dependency we get with a single attribute on the left is $C \to A$.

Example (cont.)

$$F = \{A, B \rightarrow C, \;\; C \rightarrow D, \;\; D \rightarrow A\}$$

Now consider pairs of attributes.

- $\{A,\ B\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $A, B \rightarrow D$ is a new dependency
- $\{A,\ C\}^+ = \{A,\ C,\ D\}$,
  - so $A, C \rightarrow D$ is a new dependency
- $\{A,\ D\}^+ = \{A,\ D\}$,
  - so nothing new.
- $\{B,\ C\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $B, C \rightarrow A, D$ is a new dependency
- $\{B,\ D\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $B, D \rightarrow A, C$ is a new dependency
- $\{C,\ D\}^+ = \{A,\ C,\ D\}$,
  - so $C, D \rightarrow A$ is a new dependency

Example (cont.)

$$F = \{A, B \to C, \;\; C \to D, \;\; D \to A\}$$

For the triples of attributes:

- $\{A,\ C,\ D\}^+ = \{A,\ C,\ D\}$,
- $\{A,\ B,\ D\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $A, B, D \to C$ is a new dependency
- $\{A,\ B,\ C\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $A, B, C \to D$ is a new dependency
- $\{B,\ C,\ D\}^+ = \{A,\ B,\ C,\ D\}$,
  - so $B, C, D \to A$ is a new dependency

And since $\{A,\ B,\ C,\ D\}+ = \{A,\ B,\ C,\ D\}$, we get no new dependencies with four attributes.

Example (cont.)

We generated 11 new FDs:

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
C & \rightarrow & A & A, B & \rightarrow & D \\
A, C & \rightarrow & D & B, C & \rightarrow & A \\
B, C & \rightarrow & D & B, D & \rightarrow & A \\
B, D & \rightarrow & C & C, D & \rightarrow & A \\
A, B, C & \rightarrow & D & A, B, D & \rightarrow & C \\
B, C, D & \rightarrow & A & & &
\end{array}
$$

Can you see the Key?

$\{A, B\}$, $\{B, C\}$, and $\{B, D\}$ are keys.

Note: this schema is already in 3NF! Why?

Consequences of Armstrong's Axioms

Union If $F \models \mathbf{Y} \to \mathbf{Z}$ and $F \models \mathbf{Y} \to \mathbf{W}$, then $F \models \mathbf{Y} \to \mathbf{W}, \mathbf{Z}$.

Pseudo-transitivity If $F \models \mathbf{Y} \to \mathbf{Z}$ and $F \models \mathbf{U}, \mathbf{Z} \to \mathbf{W}$, then $F \models \mathbf{Y}, \mathbf{U} \to \mathbf{W}$.

Decomposition If $F \models \mathbf{Y} \to \mathbf{Z}$ and $\mathbf{W} \subseteq \mathbf{Z}$, then $F \models \mathbf{Y} \to \mathbf{W}$.

Exercise : Prove these using Armstrong's axioms!

Proof of the Union Rule

Suppose we have

$$F \models \mathbf{Y} \rightarrow \mathbf{Z},$$
$$F \models \mathbf{Y} \rightarrow \mathbf{W}.$$

By augmentation we have

$$F \models \mathbf{Y}, \mathbf{Y} \rightarrow \mathbf{Y}, \mathbf{Z},$$

that is,

$$F \models \mathbf{Y} \rightarrow \mathbf{Y}, \mathbf{Z}.$$

Also using augmentation we obtain

$$F \models \mathbf{Y}, \mathbf{Z} \rightarrow \mathbf{W}, \mathbf{Z}.$$

Therefore, by transitivity we obtain

$$F \models \mathbf{Y} \rightarrow \mathbf{W}, \mathbf{Z}.$$

Example application of functional reasoning.

Heath's Rule

Suppose $R(A, B, C)$ is a relational schema with functional dependency $A \rightarrow B$, then

$$R = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R).$$

Proof of Heath's Rule

We first show that $R \subseteq \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

- If $u = (a,\ b,\ c) \in R$, then $u_1 = (a,\ b) \in \pi_{A,B}(R)$ and $u_2 = (a,\ c) \in \pi_{A,C}(R)$.
- Since $\{(a,\ b)\} \bowtie_A \{(a,\ c)\} = \{(a,\ b,\ c)\}$ we know $u \in \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

In the other direction we must show $R' = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R) \subseteq R$.

- If $u = (a,\ b,\ c) \in R'$, then there must exist tuples $u_1 = (a,\ b) \in \pi_{A,B}(R)$ and $u_2 = (a,\ c) \in \pi_{A,C}(R)$.
- This means that there must exist a $u' = (a,\ b',\ c) \in R$ such that $u_2 = \pi_{A,C}(\{(a,\ b',\ c)\})$.
- However, the functional dependency tells us that $b = b'$, so $u = (a,\ b,\ c) \in R$.

Closure Example

$R(A,\ B, C, D, E, F)$ with

$$\begin{array}{l} A, B \rightarrow C \\ B, C \rightarrow D \\ D \rightarrow E \\ C, F \rightarrow B \end{array}$$

What is the closure of $\{A,\ B\}$?

$$\begin{array}{rcl} \{A,\ B\} & \overset{A,B \rightarrow C}{\Longrightarrow} & \{A,\ B,\ C\} \\ & \overset{B,C \rightarrow D}{\Longrightarrow} & \{A,\ B,\ C,\ D\} \\ & \overset{D \rightarrow E}{\Longrightarrow} & \{A,\ B,\ C,\ D,\ E\} \end{array}$$

So $\{A,\ B\}^+ = \{A,\ B,\ C,\ D,\ E\}$ and $A, B \rightarrow C, D, E$.

Lecture 08 : Normal Forms

Outline

- First Normal Form (1NF)
- Second Normal Form (2NF)
- 3NF and BCNF
- Multi-valued dependencies (MVDs)
- Fourth Normal Form

The Plan

Given a relational schema $R(\mathbf{X})$ with FDs F :

- Reason about FDs
  - Is F missing FDs that are logically implied by those in F?
- Decompose each $R(\mathbf{X})$ into smaller $R_1(\mathbf{X}_1)$, $R_2(\mathbf{X}_2)$, $\cdots R_k(\mathbf{X}_k)$, where each $R_i(\mathbf{X}_i)$ is in the desired Normal Form.

Are some decompositions better than others?

Desired properties of any decomposition

Lossless-join decomposition

A decomposition of schema $R(\mathbf{X})$ to $S(\mathbf{Y} \cup \mathbf{Z})$ and $T(\mathbf{Y} \cup (\mathbf{X} - \mathbf{Z}))$ is a lossless-join decomposition if for every database instances we have $R = S \bowtie T$.

Dependency preserving decomposition

A decomposition of schema $R(\mathbf{X})$ to $S(\mathbf{Y} \cup \mathbf{Z})$ and $T(\mathbf{Y} \cup (\mathbf{X} - \mathbf{Z}))$ is dependency preserving, if enforcing FDs on S and T individually has the same effect as enforcing all FDs on $S \bowtie T$.

We will see that it is not always possible to achieve both of these goals.

First Normal Form (1NF)

We will assume every schema is in 1NF.

1NF

A schema $R(A_1 : S_1,\ A_2 : S_2,\ \cdots,\ A_n : S_n)$ is in First Normal Form (1NF) if the domains S_1 are elementary — their values are atomic.

name
Timothy George Griffin

$\Longrightarrow$

first_name	**middle_name**	**last_name**
Timothy	George	Griffin

Second Normal Form (2NF)

Second Normal Form (2NF)

A relational schema R is in 2NF if for every functional dependency $\mathbf{X} \rightarrow A$ either

- $A \in \mathbf{X}$, or
- $\mathbf{X}$ is a superkey for R, or
- A is a member of some key, or
- $\mathbf{X}$ is not a proper subset of any key.

3NF and BCNF

Third Normal Form (3NF)

A relational schema R is in 3NF if for every functional dependency $\mathbf{X} \rightarrow A$ either

- $A \in \mathbf{X}$, or
- $\mathbf{X}$ is a superkey for R, or
- A is a member of some key.

Boyce-Codd Normal Form (BCNF)

A relational schema R is in BCNF if for every functional dependency $\mathbf{X} \rightarrow A$ either

- $A \in \mathbf{X}$, or
- $\mathbf{X}$ is a superkey for R.

Is something missing?

Another look at Heath's Rule

Given $R(\mathbf{Z}, \mathbf{W}, \mathbf{Y})$ with FDs F

If $\mathbf{Z} \rightarrow \mathbf{W} \in F^+$, the

$$R = \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R)$$

What about an implication in the other direction? That is, suppose we have

$$R = \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R).$$

Q Can we conclude anything about FDs on R? In particular, is it true that $\mathbf{Z} \rightarrow \mathbf{W}$ holds?

A No!

We just need one counter example ...

$$R = \pi_{A,B}(R) \bowtie \pi_{A,C}(R)$$

A	B	C
a	b_1	c_1
a	b_2	c_2
a	b_1	c_2
a	b_2	c_1

A	B
a	b_1
a	b_2

A	C
a	c_1
a	c_2

Clearly $A \rightarrow B$ is not an FD of R.

A concrete example

course_name	lecturer	text
Databases	Tim	Ullman and Widom
Databases	Fatima	Date
Databases	Tim	Date
Databases	Fatima	Ullman and Widom

Assuming that texts and lecturers are assigned to courses independently, then a better representation would in two tables:

course_name	lecturer
Databases	Tim
Databases	Fatima

course_name	text
Databases	Ullman and Widom
Databases	Date

Time for a definition! MVDs

Multivalued Dependencies (MVDs)

Let $R(\mathbf{Z}, \mathbf{W}, \mathbf{Y})$ be a relational schema. A multivalued dependency, denoted $\mathbf{Z} \twoheadrightarrow \mathbf{W}$, holds if whenever t and u are two records that agree on the attributes of $\mathbf{Z}$, then there must be some tuple v such that

1. v agrees with both t and u on the attributes of $\mathbf{Z}$,
2. v agrees with t on the attributes of $\mathbf{W}$,
3. v agrees with u on the attributes of $\mathbf{Y}$.

A few observations

Note 1

Every functional dependency is multivalued dependency,

$$(\mathbf{Z} \rightarrow \mathbf{W}) \implies (\mathbf{Z} \twoheadrightarrow \mathbf{W}).$$

To see this, just let $v = u$ in the above definition.

Note 2

Let $R(\mathbf{Z},\ \mathbf{W},\ \mathbf{Y})$ be a relational schema, then

$$(\mathbf{Z} \twoheadrightarrow \mathbf{W}) \iff (\mathbf{Z} \twoheadrightarrow \mathbf{Y}),$$

by symmetry of the definition.

MVDs and lossless-join decompositions

Fun Fun Fact

Let $R(\mathbf{Z}, \ \mathbf{W}, \ \mathbf{Y})$ be a relational schema. The decomposition $R_1(\mathbf{Z}, \ \mathbf{W})$, $R_2(\mathbf{Z}, \ \mathbf{Y})$ is a lossless-join decomposition of R if and only if the MVD $\mathbf{Z} \twoheadrightarrow \mathbf{W}$ holds.

Proof of Fun Fun Fact

Proof of $(\mathbf{Z} \twoheadrightarrow \mathbf{W}) \implies R = \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R)$

- Suppose $\mathbf{Z} \twoheadrightarrow \mathbf{W}$.
- We know (from proof of Heath's rule) that $R \subseteq \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R)$. So we only need to show $\pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R) \subseteq R$.
- Suppose $r \in \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R)$.
- So there must be a $t \in R$ and $u \in R$ with $\{r\} = \pi_{\mathbf{Z},\mathbf{W}}(\{t\}) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(\{u\})$.
- In other words, there must be a $t \in R$ and $u \in R$ with $t.\mathbf{Z} = u.\mathbf{Z}$.
- So the MVD tells us that then there must be some tuple $v \in R$ such that
  1. v agrees with both t and u on the attributes of $\mathbf{Z}$,
  2. v agrees with t on the attributes of $\mathbf{W}$,
  3. v agrees with u on the attributes of $\mathbf{Y}$.
- This v must be the same as r, so $r \in R$.

Proof of Fun Fun Fact (cont.)

Proof of $R = \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R) \implies (\mathbf{Z} \twoheadrightarrow \mathbf{W})$

- Suppose $R = \pi_{\mathbf{Z},\mathbf{W}}(R) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(R)$.
- Let t and u be any records in R with $t.\mathbf{Z} = u.\mathbf{Z}$.
- Let v be defined by $\{v\} = \pi_{\mathbf{Z},\mathbf{W}}(\{t\}) \bowtie \pi_{\mathbf{Z},\mathbf{Y}}(\{u\})$ (and we know $v \in R$ by the assumption).
- Note that by construction we have
  1. $v.\mathbf{Z} = t.\mathbf{Z} = u.\mathbf{Z}$,
  2. $v.\mathbf{W} = t.\mathbf{W}$,
  3. $v.\mathbf{Y} = u.\mathbf{Y}$.
- Therefore, $\mathbf{Z} \twoheadrightarrow \mathbf{W}$ holds.

Fourth Normal Form

Trivial MVD

The MVD $\mathbf{Z} \twoheadrightarrow \mathbf{W}$ is trivial for relational schema $R(\mathbf{Z},\ \mathbf{W},\ \mathbf{Y})$ if

1. $\mathbf{Z} \cap \mathbf{W} \neq \{\}$, or
2. $\mathbf{Y} = \{\}$.

4NF

A relational schema $R(\mathbf{Z},\ \mathbf{W},\ \mathbf{Y})$ is in 4NF if for every MVD $\mathbf{Z} \twoheadrightarrow \mathbf{W}$ either

- $\mathbf{Z} \twoheadrightarrow \mathbf{W}$ is a trivial MVD, or
- $\mathbf{Z}$ is a superkey for R.

Note : 4NF $\subset$ BCNF $\subset$ 3NF $\subset$ 2NF

Summary

We always want the lossless-join property. What are our options?

	3NF	BCNF	4NF
Preserves FDs	Yes	Maybe	Maybe
Preserves MVDs	Maybe	Maybe	Maybe
Eliminates FD-redundancy	Maybe	Yes	Yes
Eliminates MVD-redundancy	No	No	Yes

Inclusions

Clearly BCNF $\subseteq$ 3NF $\subseteq 2NF$. These are proper inclusions:

In 2NF, but not 3NF

$R(A,\ B,\ C)$, with $F = \{A \rightarrow B,\ B \rightarrow C\}$.

In 3NF, but not BCNF

$R(A,\ B,\ C)$, with $F = \{A, B \rightarrow C,\ C \rightarrow B\}$.

- This is in 3NF since AB and AC are keys, so there are no non-prime attributes
- But not in BCNF since C is not a key and we have $C \rightarrow B$.

Schema refinement III and advanced design

Outline

- General Decomposition Method (GDM)
- The lossless-join condition is guaranteed by GDM
- The GDM does not always preserve dependencies!
- FDs vs ER models?
- Weak entities
- Using FDs and MVDs to refine ER models
- Another look at ternary relationships

General Decomposition Method (GDM)

GDM

1. Understand your FDs F (compute F^+),
2. find $R(\mathbf{X}) = R(\mathbf{Z},\ \mathbf{W},\ \mathbf{Y})$ (sets **Z**, **W** and **Y** are disjoint) with FD $\mathbf{Z} \rightarrow \mathbf{W} \in F^+$ violating a condition of desired NF,
3. split R into two tables $R_1(\mathbf{Z},\ \mathbf{W})$ and $R_2(\mathbf{Z},\ \mathbf{Y})$
4. wash, rinse, repeat

Reminder

For $\mathbf{Z} \rightarrow \mathbf{W}$, if we assume $\mathbf{Z} \cap \mathbf{W} = \{\}$, then the conditions are

1. **Z** is a superkey for R (2NF, 3NF, BCNF)
2. **W** is a subset of some key (2NF, 3NF)
3. **Z** is not a proper subset of any key (2NF)

The lossless-join condition is guaranteed by GDM

- This method will produce a lossless-join decomposition because of (repeated applications of) Heath's Rule!
- That is, each time we replace an S by S_1 and S_2, we will always be able to recover S as $S_1 \bowtie S_2$.
- Note that in GDM step 3, the FD $\mathbf{Z} \rightarrow \mathbf{W}$ may represent a key constraint for R_1.

But does the method always terminate? Please think about this ....

General Decomposition Method Revisited

GDM++

1. Understand your FDs and MVDs F (compute F^+),
2. find $R(\mathbf{X}) = R(\mathbf{Z},\ \mathbf{W},\ \mathbf{Y})$ (sets $\mathbf{Z}$, $\mathbf{W}$ and $\mathbf{Y}$ are disjoint) with either FD $\mathbf{Z} \rightarrow \mathbf{W} \in F^+$ or MVD $\mathbf{Z} \twoheadrightarrow \mathbf{W} \in F^+$ violating a condition of desired NF,
3. split R into two tables $R_1(\mathbf{Z},\ \mathbf{W})$ and $R_2(\mathbf{Z},\ \mathbf{Y})$
4. wash, rinse, repeat

Return to Example — Decompose to BCNF

$R(A, \ B, C, D)$

$$F = \{A, B \to C, \quad C \to D, \quad D \to A\}$$

Which FDs in F^+ violate BCNF?

$$\begin{array}{rcl} C & \to & A \\ C & \to & D \\ D & \to & A \\ A, C & \to & D \\ C, D & \to & A \end{array}$$

Return to Example — Decompose to BCNF

Decompose $R(A, B, C, D)$ to BCNF

Use $C \rightarrow D$ to obtain

- $R_1(C, D)$. This is in BCNF. Done.
- $R_2(A, B, C)$ This is not in BCNF. Why? A, B and B, C are the only keys, and $C \rightarrow A$ is a FD for R_1. So use $C \rightarrow A$ to obtain
  - $R_{2.1}(A, C)$. This is in BCNF. Done.
  - $R_{2.2}(B, C)$. This is in BCNF. Done.

Exercise : Try starting with any of the other BCNF violations and see where you end up.

The GDM does not always preserve dependencies!

$R(A,\ B,\ C,\ D,\ E)$

$$\begin{array}{rcl} A, B & \rightarrow & C \\ D, E & \rightarrow & C \\ B & \rightarrow & D \end{array}$$

- $\{A,\ B\}^+ = \{A,\ B,\ C,\ D\}$,
- so $A, B \rightarrow C, D$,
- and $\{A,\ B,\ E\}$ is a key.

- $\{B,\ E\}^+ = \{B,\ C,\ D,\ E\}$,
- so $B, E \rightarrow C, D$,
- and $\{A,\ B,\ E\}$ is a key (again)

Let's try for a BCNF decomposition ...

Decomposition 1

Decompose $R(A,\ B,\ C,\ D,\ E)$ using $A, B \rightarrow C, D$:

- $R_1(A,\ B,\ C,\ D)$. Decompose this using $B \rightarrow D$:
  - $R_{1.1}(B,\ D)$. Done.
  - $R_{1.2}(A,\ B,\ C)$. Done.
- $R_2(A,\ B,\ E)$. Done.

But in this decomposition, how will we enforce this dependency?

$$D, E \rightarrow C$$

Decomposition 2

Decompose $R(A, B, C, D, E)$ using $B, E \rightarrow C, D$:

- $R_3(B, C, D, E)$. Decompose this using $D, E \rightarrow C$
  - $R_{3.1}(C, D, E)$. Done.
  - $R_{3.2}(B, D, E)$. Decompose this using $B \rightarrow D$:
    - $R_{3.2.1}(B, D)$. Done.
    - $R_{3.2.2}(B, E)$. Done.
- $R_4(A, B, E)$. Done.

But in this decomposition, how will we enforce this dependency?

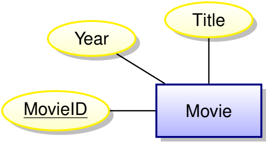
$$A, B \rightarrow C$$

Summary

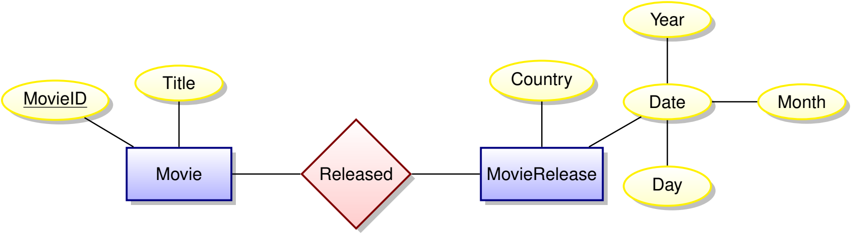
- It is always possible to obtain BCNF that has the lossless-join property (using GDM)
  - But the result may not preserve all dependencies.
- It is always possible to obtain 3NF that preserves dependencies and has the lossless-join property.
  - Using methods based on "minimal covers" (for example, see EN2000).

Recall : a small change of scope ...

... changed this entity

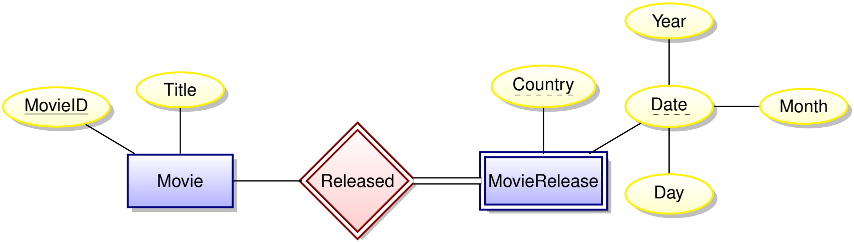


into two entities and a relationship :



But is there something odd about the MovieRelease entity?

MovieRelease represents a Weak entity set



Definition

- Weak entity sets do not 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?

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

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

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

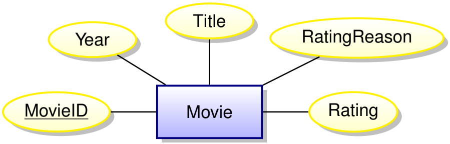
Title	Year	Rating
Austin Powers: International Man of Mystery	1997	15
Austin Powers: The Spy Who Shagged Me	1999	12
Dude, Where's My Car?	2000	15

Scope = Earth

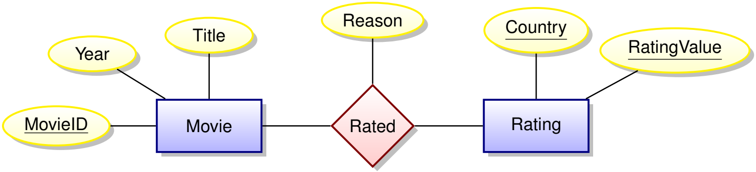
Title	Year	Country	Rating
Austin Powers: International Man of Mystery	1997	UK	15
Austin Powers: International Man of Mystery	1997	Malaysia	18SX
Austin Powers: International Man of Mystery	1997	Portugal	M/12
Austin Powers: International Man of Mystery	1997	USA	PG-13
Austin Powers: The Spy Who Shagged 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:



Note that relation Rated has an attribute!

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.

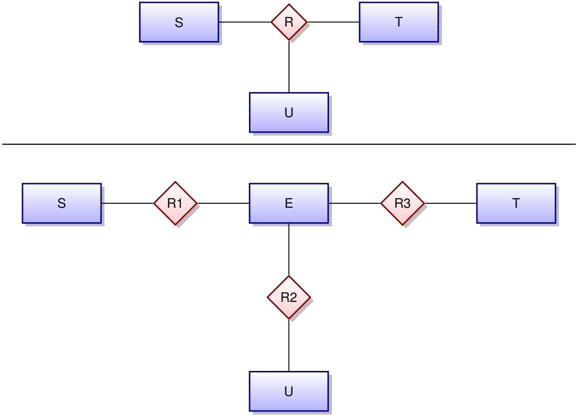
This is a mildly amusing illustration of a real and pervasive problem — deriving a functional dependency after the examination of a limited set of data (or after talking to only a few domain experts).

Oh, but the real world is such a bother!

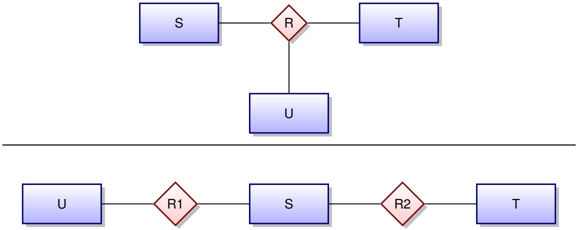
from IMDb raw data file certificates.list

```
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?

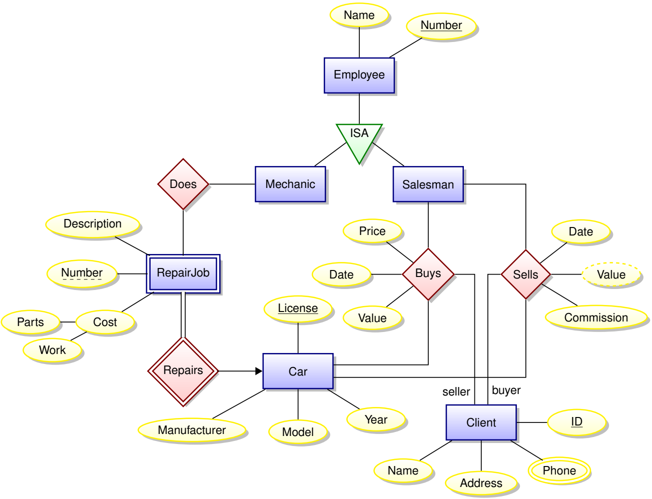


Ternary or multiple binary relationships?



Look again at ER Demo Diagram[2]

How might this be refined using FDs or MVDs?



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

Lecture 10 : On-line Analytical Processing (OLAP)

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]

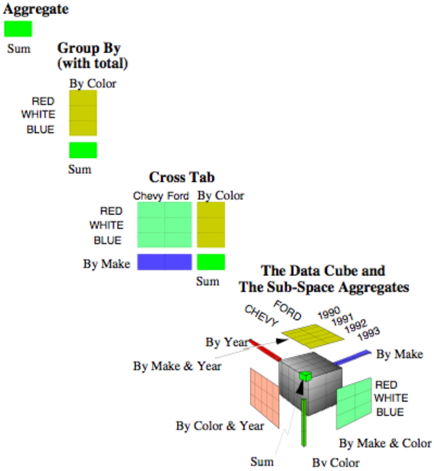
Data Mining and Knowledge Discovery 1, 29–53 (1997)

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

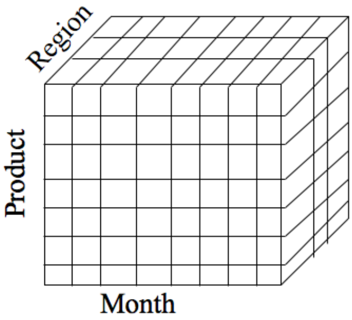
JIM GRAY Gray@Microsoft.com
SURAJIT CHAUDHURI SurajitC@Microsoft.com
ADAM BOSWORTH AdamB@Microsoft.com
ANDREW LAYMAN AndrewL@Microsoft.com
DON REICHART DonRei@Microsoft.com
MURALI VENKATRAO MuraliV@Microsoft.com
Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way,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



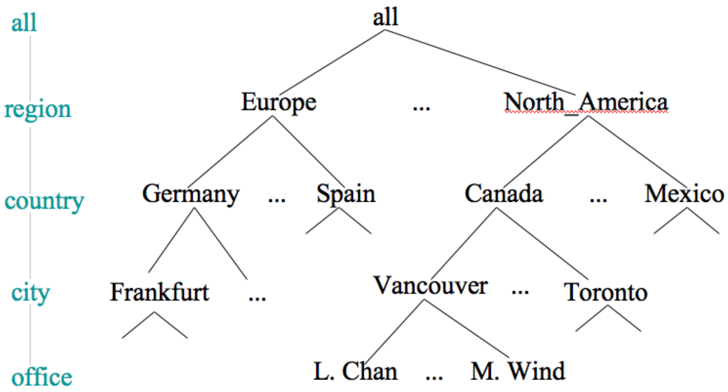
The Data Cube



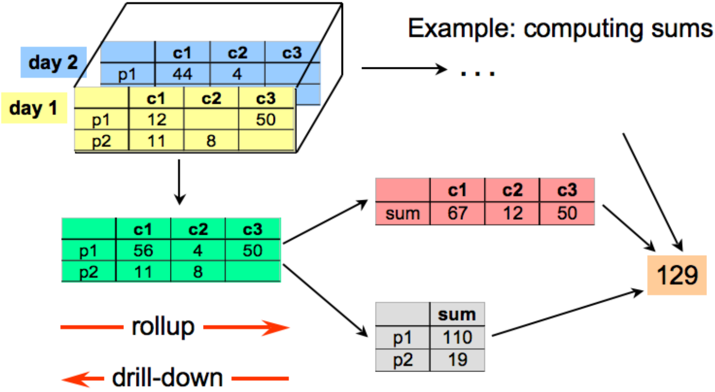
Dimensions:
Product,
Location,
Time

- 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

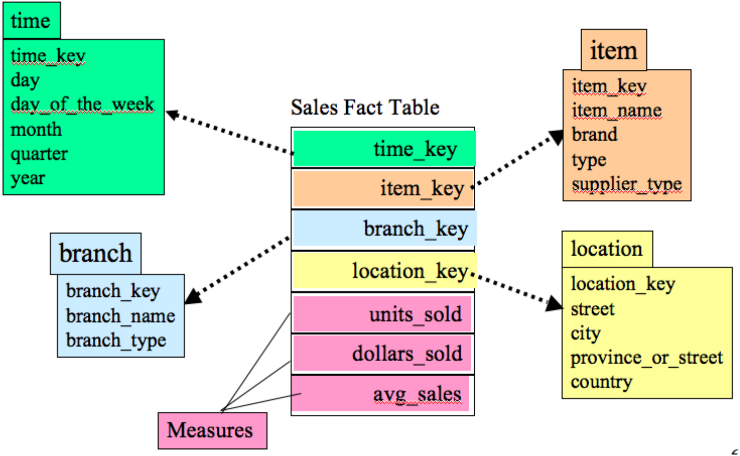
Hierarchy for **Location** Dimension



Cube Operations



The Star Schema as a design tool



Lectures 11 : Case Study - Cancer registry for the NHS, Part II

The extension of ECRIC to cover all of England requires schema reconciliation, a problem that remains unresolved since it was first encountered in the 1980s. Jem Rashbass has a long track record in NHS IT, and is now CEO of ECRIC. Jem will explain what the NHS needs and why - some of the existing challenges and future opportunities. The session will close with an open forum in which the DBA of the now national level Cancer Registry DBMS will join Jem.

Lecture 12 : XML as a data exchange format

Outline

- HTML vs. XML
- Using XML to solve the data exchange problem
- Domain-specific XML schema
- Native XML databases

HTML vs XML

HTML

HTML	=	Content + (fixed) Schema + (fixed) presentation

Untangle these and generalize to

XML

XML	=	Content
XSL	=	defines presentations
DTD or XSchema	=	defines schema

HTML	:	Hypertext Markup Language
XML	:	eXtensible Markup Language
XSL	:	Extensible Stylesheet Language (similar to CSS)
CSS	:	Cascading Style Sheets
DTD	:	Document Type Definition

XML data is "semi-structured" UniCode text

```
<TAGNAME VAL1="some value" VAL2="some value">
Body of text, and possibly nested tags.
</TAGNAME>
```

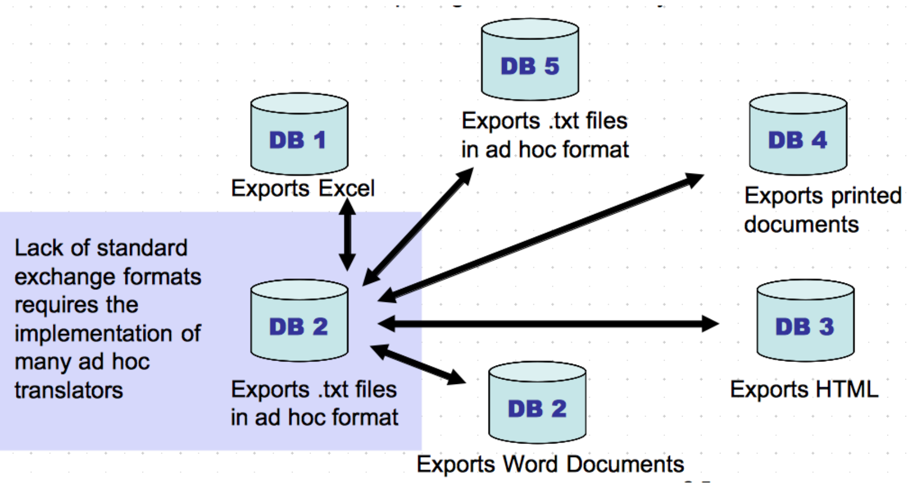
An XML schema defines

- tag names
- which associated values are optional or required
- types of associated values
- type of the associated body

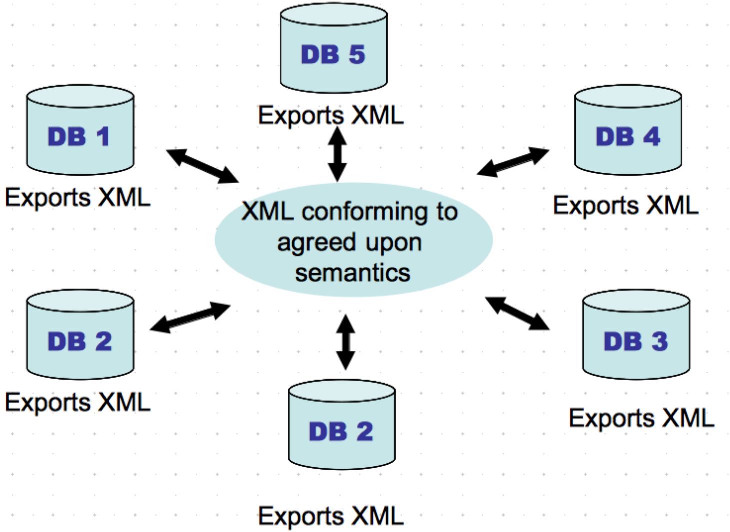
What would Churchill say?

XML is the worst form of data representation, except for all those other forms that have been tried from time to time.

The data exchange problem



XML as a data exchange standard



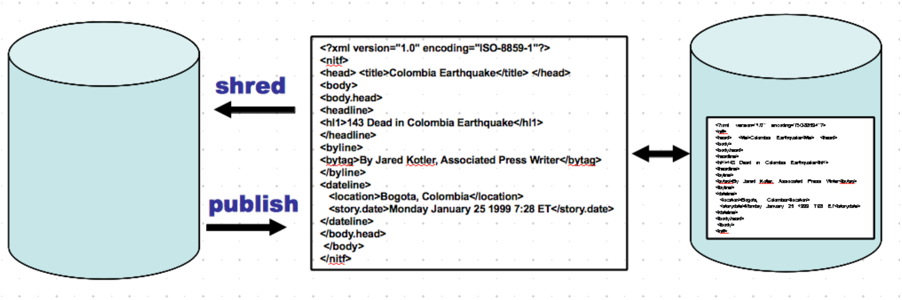
Domain-specific schema can become standards.

There are now thousands of domain-specific schema

WML: Wireless markup language (WAP)
OFX: Open financial exchange
CML: Chemical markup language
AML: Astronomical markup language
MathML: Mathematics markup language
SMIL: Synchronized multimedia integration language
ThML: Theological markup language
.....

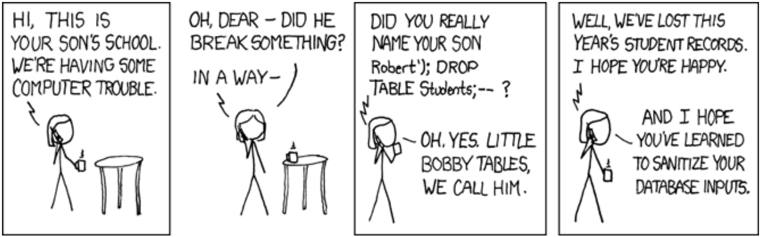
The public XML schema is in some many ways "dual" to the many private SQL schemas involved in data exchange.

Two basic kinds of XML databases (hybrids possible)



XML-enabled databases	Native XML database
Relational (XML for exchange)	direct storage of XML data
"Data-centric"	"Document-centric"
SQL	XPath and XQuery
http://www.mysql.com/	http://basex.org
	http://exist.sourceforge.net

The End



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