Normalisation

Informal guidelines

- Semantics of the attributes
  - *easy to explain relation*
  - *doesn't mix concepts*
- Reducing the redundant values in tuples
- Choosing attribute domains that are atomic
- Reducing the null values in tuples
- Disallowing spurious tuples

Functional Dependency

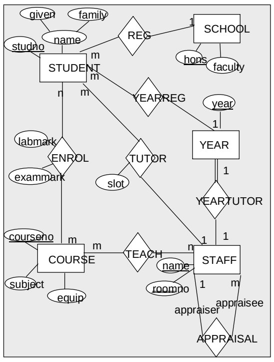
- an attribute A is functionally dependent on a set of attributes X if and only if
  - value of A is determined solely by the values of X
  - values of X uniquely determine a value of A

$X \rightarrow A$

child → mother
mother ↛ child

The value of child implies the value of mother
Value of mother does NOT imply value of child
Child is the determinant
Mother is the dependent/determined

Our case study example



STUDENT(<u>studno</u>,givenname,familyname, hons,tutor,slot,year)
studno → studno, givenname, familyname, hons tutor, slot, year

ENROL(<u>studno,courseno</u>,labmark,exammark)
studno, courseno → labmark, exammark

COURSE(<u>courseno</u>,subject,equip)
courseno → courseno, subject, equip

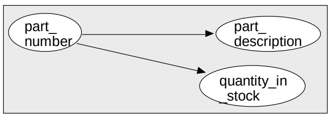
STAFF(<u>lecturer</u>,roomno,appraiser)
lecturer → lecturer, roomno, appraiser
roomno → lecturer, appraiser, roomno

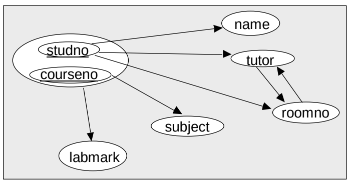
YEAR(<u>year</u>,yeartutor)
year → year, yeartutor
yeartutor → year, yeartutor

SCHOOL(<u>hons</u>,faculty)
hons → hons, faculty

TEACH(<u>courseno,lecturer</u>)
courseno, lecturer → courseno, lecturer

More Examples of Functional Dependency





Use functional dependencies to …
check that a relation is legal or good. e.g keys

- K is a superkey of relation R if K → R

i.e.
whenever t1[k] = t2[k]
then t1[R]= t2[R]

K functionally determines all attributes in a tuple in R

STUDENT (<u>studno</u>,name,hons,tutor,slot,year)

studno → studno, name, hons, tutor, slot, year

Use functional dependencies to …

check that a relation is legal or good. e.g. remove redundancy

- Partial Dependency
  - studno, courseno → subject
    - (studno, courseno, subject)
- Transitive Dependency
  - studno → yeartutor
    - studno → year
      - year → yeartutor   so,
        - studno → yeartutor
    - (studno, yeartutor)
- Base functional dependencies F
- Set of logically implied functional dependencies
  CLOSURE F+

Normalisation

Given a relation R with a set of functional dependencies F, and a key K

We must identify independent attributes

1. the key identifies all the attributes but…
2. ... if an attribute only depends on part of the key, then it is independent of the rest of it.

   Attribute is partially dependent on the key
3. ... if an attribute only depends on the key transitively, then it really depends directly on another attribute and is independent of the key.

   Attribute is transitively dependent on the key

Boyce-Codd Normal Form

A relation scheme R is in BCNF if, for all functional dependencies that hold on R of the form X → Y where R ⊇ X and R ⊇ Y at least one of the following holds

- X → Y is trivial
- X is a *candidate key* for the scheme R
  i.e. X → R

Every attribute must depend on the key, the whole key and nothing but the key

- Other Normal Forms: 1NF, 2NF and 3NF ... uses primary key only
- BCNF... generalised for candidate keys

Use functional dependencies to …

check constraints on the set of legal relations

studno	name	tutor	roomno	course no	labmark	subject
s1	jones	bush	2.26	cs250	65	prog
s1	jones	bush	2.26	cs260	80	graphics
s1	jones	wibby	2.26	cs270	47	elecs
s2	brown	kahn	IT206	cs250	67	prog
s2	brown	kahn	IT206	cs270	65	elecs
s3	smith	goble	2.82	cs270	49	comms
s4	blogg	goble	2.82	cs280	50	design
s5	jones	zobel	2.34	cs250	0	prog
s6	peters	kahn	A17	cs250	2	prog
null	*null*	capon	A14	*null*	*null*	*null*
null	*null*	*null*	*null*	cs290	*null*	specs
s7	patel	*null*	*null*	*null*	*null*	*null*

F

studno → name, tutor

tutor → roomno

roomno → tutor

courseno → subject

studno, courseno → labmark

F+

studno, courseno → name   *partial*

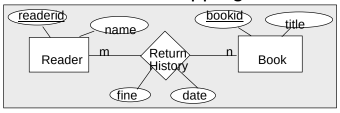
studno → roomno   *transitive*

Consequences of redundancy

- Wasted space
- Potential performance cost
- Potential inconsistency
- Inability to represent data

Use functional dependencies to …

check the EER model mapping correctness



ReturnHistory(readerid, bookid, date, fine)

ReturnHistory			
readerid	bookid	date	fine
123	Macbeth	10/4	0
123	Macbeth	5/5	1.5
123	Macbeth	7/6	0
123	Hamlet	16/9	0
456	Macbeth	16/9	2
.	.	.	.

readerid → readerid

readerid → name

bookid → bookid

bookid → title

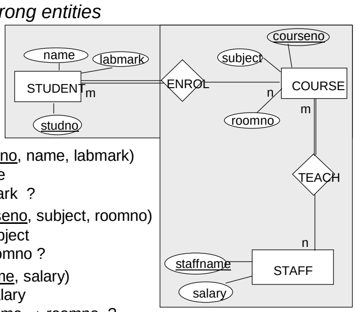
readerid, bookid → date ?

readerid, bookid → fine ?

Many:many relationships that could be weak entity types because they have hidden partial keys.

Using Functional Dependencies to ... check EER mappings

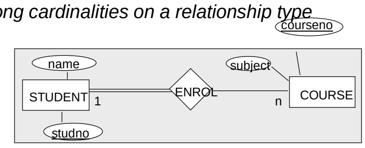
Attributes on wrong entities



- STUDENT(<u>studno</u>, name, labmark)
  studno → name
  studno → labmark ?
- COURSE(<u>courseno</u>, subject, roomno)
  courseno → subject
  courseno → roomno ?
- STAFF(<u>staffname</u>, salary)
  staffname → salary
  where is staffname → roomno ?

Using Functional Dependencies to ... check EER mappings

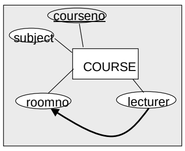
Wrong cardinalities on a relationship type



- STUDENT(<u>studno</u>, name)
  studno → name
- COURSE(<u>courseno</u>, subject, studno)
  courseno → subject
  courseno → studno ?

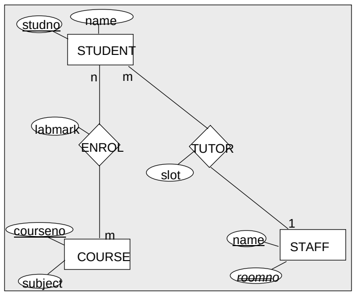
Using Functional Dependencies to ... check EER mappings

Missing 1:many relationship type and entity type or missing multi-valued attribute



- COURSE (<u>courseno</u>, subject, lecturer,roomno)
  courseno → subject
  courseno → lecturer ?
  courseno → roomno
  lecturer → roomno

Functional Dependencies are hidden in EER Model



Using the EER Model and Functional Dependencies

1. Draw EER model
2. Map EER schema to relational schema
3. For every relation
   - List the functional dependencies
   - what does determine every attribute?
   - Check that every relation is in BCNF
     - does the key really solely uniquely identify each attribute?
     - if its not in BCNF then why?
     - Fix the problem
       - normalise and/or
       - trace back to EER model
4. Are there any functional dependencies missing?
5. Optimise the relational schema

Database design

- Extended Entity Relationship
  - Top Down
  - Conceptual/Abstract View
- Functional Dependencies
  - Bottom Up
  - Implementation View
  - The Determinancy Approach
  - Synthesise relations

1. List all attributes
2. Consider the relationships between them

- those which determine the values of others are entities
- those whose values are determined by other items are attributes.

Use functional dependencies to…Synthesise relations

studno → studno
studno → familyname
studno → givenname
studno → hons
studno → tutor
studno → slot
studno → year

STUDENT (studno,givenname,familyname,hons,tutor,slot,year)

studno, courseno → labmark
studno, courseno → exammark

ENROL(studno,courseno,labmark,exammark)

courseno → courseno
courseno → subject
courseno → equip

COURSE(courseno,subject,equip)

lecturer → lecturer
lecturer → roomno
lecturer → appraiser
roomno → lecturer
roomno → roomno
roomno → appraiser

STAFF(lecturer,*roomno*,appraiser)

hons → faculty
hons → hons

year → year
year → yeartutor
yeartutor → year
yeartutor → yeartutor

YEAR(year,*yeartutor*)

SCHOOL(hons,faculty)

er….

TEACH(courseno,lecturer)

courseno, lecturer → courseno, lecturer

TEACH(courseno,lecturer, num_of_lectures)

courseno, lecturer → num_of_lectures

Complementary Approaches

- Disadvantages of EER Top Down
  1. Not all entity types are represented by nouns or noun-phrases
     - association entity types
  2. Not all nouns and noun-phrases correspond to entities
     - single attribute entities
- Disadvantages of determinancy bottom-up
  1. Long-winded
  2. Hides overall picture of data model

The Steps of Normalisation

- Take one dependency at a time
- Treat each relation separately and independently
- Iterative process

Use functional dependencies to…

NORMALISE relations

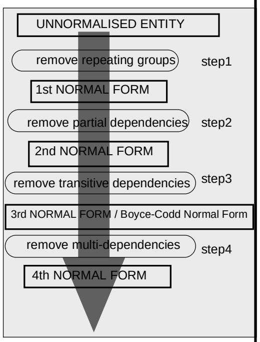
- Systematically create legal relations
- Derive relations which avoid anomalies in
  - Insertion
  - Deletion
  - Modification
  - Accessing
- Ensure single valued-ness of facts represented in attributes in keyed relations
- Ensure the removal of redundancy in a relation

Normalisation

- Given
  - a universal relation that is unnormalised
  - a set of functional dependencies on the attributes in the relation
  - produce a set of relations where each relation is normalised for the functional dependencies on the attributes in the relation
  - Three approaches:
  - 1. Relational synthesis
  - 2. Step-wise normalisation
  - 3. Using BCNF decomposition

The Process of Normalisation

- Usually four steps giving rise to
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
- At each step we consider relationships between the functional dependencies of a relation's attributes
- Normalisation is a:
  - framework
  - series of tests



First Normal Form

- Attributes form Repeating Groups
- When a group of attributes has multiple values then we say there is a repeating group of attributes in the relation
- An relation is in 1NF if there are no repeating groups of attribute types
- Any un-normalised relation is transformed to 1NF
  - Remove all repeating attribute groups
  - Repeating attribute groups become new relations in their own right
  - The key of the original relation must be an attribute (but not necessarily a key) of the derived relation.

First Normal Form : Repeating Groups

STUDENT_DETAILS

<u>stud no</u>	name	tutor	roomno	<u>course no</u>	lab mark	subject
s1	jones	bush	2.26	cs250	65	prog
				cs260	80	graphics
				cs270	47	elecs
s2	brown	kahn	IT206	cs250	67	prog
				cs270	65	elecs

STUDENT_DETAILS
(<u>studno</u>, name, tutor, roomno, {<u>courseno</u>, labmark, subject})
studno → name, tutor
tutor → roomno, roomno → tutor
courseno → subject
studno, courseno → labmark

STUDENT
(<u>studno</u>, name, tutor, roomno)
studno → name, tutor
tutor → roomno,
roomno → tutor

ENROL (<u>studno, courseno</u>, subject, labmark)
courseno → subject
studno, courseno → labmark

Benefits from First Normal Form

- Any 'hidden' relations (entities) are identified
- Process results in separation of different objects
- BUT anomalies may still exist

ENROL (<u>studno, courseno</u>, subject, labmark)

- subject appears on every enrolment occurrence.
- This may result in anomalies when updating or deleting tuples
- The problem in example is that subject is functionally dependent only on `courseno` which is only *part* of the key

Second Normal Form

- A relation is in 2NF if it is in 1NF and each non identifying attribute depends upon the *whole* key (identifier)
- Any relation in 1NF is transformed to 2NF
  - Identify functional dependencies
  - Re-write relations so that each non-identifying attribute is functionally dependent on the *whole* of the key
  - Decompose ENROL into two relations

ENROL (<u>studno, courseno</u>, subject, labmark)
courseno → subject
studno, courseno → labmark

ENROL' (<u>studno, courseno</u>, labmark)
studno, courseno → labmark

COURSE (<u>courseno</u>, subject)
courseno → subject

Second Normal Form

STUDENT(<u>studno</u>, name, tutor, roomno)

studno → name, tutor
tutor → roomno
roomno → tutor

ENROL' (<u>studno, courseno</u>, labmark)

studno, courseno → labmark

COURSE (<u>courseno</u>, subject)

courseno → subject

Third Normal Form

- An relation is in 3NF if it is in 2NF and all non-identifying attributes are independent
- Any relation in 2NF is transformed in 3NF
- Determine functional dependencies between non identifying attributes
- Decompose relation into new relations

STUDENT (<u>studno</u>, name, tutor, roomno)
studno → name, tutor
tutor → roomno
roomno → tutor

STUDENT (<u>studno</u>, name, tutor)
studno → name, tutor

TUTOR (<u>tutor</u>, roomno)
tutor → roomno
roomno → tutor

Student Relational Schema in 3NF

- STUDENT (<u>studno</u>, name, tutor)
  studno → name, tutor
- TUTOR (<u>tutor</u>, *roomno*)
  tutor → roomno
  roomno → tutor
- ENROL
  (<u>studno, courseno</u>, labmark)
  studno, courseno → labmark
- COURSE (<u>courseno</u>, subject)
  courseno → subject

Decomposition: Lossless or Non-additive Join

- R is a relational scheme, F is a set of functional dependencies on R. R1 and R2 form a decomposition of R.
- The decomposition of R is non-additive if at least one of the following functional dependencies are in F+
  $R1 \cap R2 \rightarrow R1$
  $R1 \cap R2 \rightarrow R2$
- The decomposition of R is non-additive if for every state r of R that satisfies F
  $\bowtie (\pi_{<R1>}(r), ..., \pi_{<Rm>}(r)) = r$
  where $\bowtie$ condition is the natural join

Decomposition: Lossless or Non-additive Join

ENROL (<u>studno, courseno</u>, subject, labmark)
courseno → subject
studno, courseno → labmark

ENROL' (<u>studno, courseno</u>, labmark)
studno, courseno → labmark

COURSE (<u>courseno</u>, subject)
courseno → subject

- ENROL' ∩ COURSE = courseno
- courseno → subject
- (<u>courseno</u>, subject) = COURSE

Lossless or Non-additive Join

STUDENT

<u>studno</u>	name	tutor	roomno
s1	jones	bush	2.26
s2	brown	kahn	IT206
s3	smith	goble	2.82
s4	bloggs	goble	2.82
s5	jones	zobel	2.34
s6	peters	kahn	IT206

studno → name
studno → tutor
tutor → roomno
roomno → tutor

STUDENT1

<u>studno</u>	name	tutor
s1	jones	bush
s2	brown	kahn
s3	smith	goble
s4	bloggs	goble
s5	jones	zobel
s6	peters	kahn

studno → name
studno → tutor

TUTOR

<u>tutor</u>	*roomno*
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34

tutor → roomno
roomno → tutor

- STUDENT1 ⋈(tutor = tutor)TUTORS = STUDENT

Spurious Tuples Lossless or Non-additive Join

TEACH

<u>courseno</u>	<u>lecturer</u>	num_lecture
cs250	lindsey	20
cs250	capon	4
cs260	kahn	15
cs260	bush	9
cs270	zobel	12
cs270	woods	12
cs280	capon	24

TEACH'

<u>courseno</u>	<u>lecturer</u>
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

LECTURES

<u>lecturer</u>	num_lecture
lindsey	20
capon	4
kahn	15
bush	9
zobel	12
woods	12
capon	24

TEACH' ⋈ LECTURES

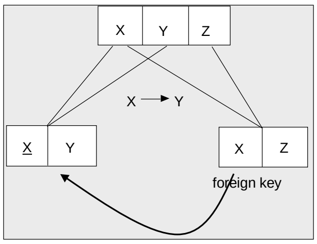
<u>courseno</u>	<u>lecturer</u>	num_lecture
cs250	lindsey	20
cs250	capon	4
cs250	capon	24
cs260	kahn	15
cs260	bush	9
cs270	zobel	12
cs270	woods	12
cs280	capon	4
cs280	capon	24

Decomposition Algorithm:
Decomposition D, relation R

- set D := { R } ;
- while there is a relation schema Q in D that is not in BCNF do
- begin
  - choose a relation schema Q in D that is not in BCNF;
  - find a functional dependency $X \rightarrow Y$ in Q that violates BCNF;
    - violation means that $(X)^+$ fails to find all of Q, so X can't be a key.
  - replace Q in D by two schemas
    - R1 $(Q - (Y)^+ \cup X)$
      - *leave copy of X in relation to be the foreign key for R2*

    and
    - R2 $(X \cup (Y)^+)$
      - *new relation for functional dependency and its closure, X will be the primary key*
- end;

Lossless or Non-additive Join



Decomposition: Dependency Preservation

- When an update is made to a database, should be able to check that update satisfies all functional dependencies.
- It is desirable to allow validation of relational database schemes that allow update validation without the computation of joins.
- independent manipulation of relations.

Dependency Preservation

- The union of dependencies that hold on the individual relations in decomposition D must be equivalent to F.
- Given F on R, $\pi_F(R_i)$ where $R_i \subseteq R$ is the set of dependencies X Y in F^+ such that the attributes in $X \cup Y$ are all contained in R_i
- Decomposition D = $\{R_1, R_2, ..., R_m\}$ of R is dependency preserving w.r.t. F if

$$(\pi_F(R_1)) \cup .... \cup \pi_F(R_m)))^+ = F^+$$

- Given the restriction of functional dependencies to a relation is the fds that involve attributes of that relation F_i for R_i

$$\bigcup_{i=1}^{n} Fi \neq F \text{ possible, } \quad \text{but...} \quad (\bigcup_{i=1}^{n} F_i)^+ = F^+$$

Dependency Preservation

- STUDENT (<u>studno</u>, name, tutor, roomno, appraiser)
  studno → name, tutor
  tutor → roomno, appraiser
  roomno → tutor, appraiser
- STUDENT1 (<u>studno</u>, name, tutor)
  studno → name, tutor
- TUTOR (<u>studno</u>, roomno, appraiser)
  studno → roomno, appraiser

  This is in Boyce-Codd Normal Form and is a lossless (nonadditive) join decomposition but we have lost....
- tutor → roomno, appraiser
  roomno → tutor, appraiser

Dependency Preservation

STUDENT

studno	name	tutor	roomno	appraiser
s1	jones	bush	2.26	capon
s2	brown	kahn	IT206	watson
s3	smith	goble	2.82	capon
s4	bloggs	goble	2.82	capon
s5	jones	zobel	2.34	watson
s6	peters	kahn	IT206	watson

studno → name
studno → tutor

tutor → roomno
tutor → appraiser
roomno → tutor
roomno → appraiser
studno → appraiser
studno → roomno

studno → appraiser
studno → roomno

STUDENT'

studno	name	tutor
s1	jones	bush
s2	brown	kahn
s3	smith	goble
s4	bloggs	goble
s5	jones	zobel
s6	peters	kahn

TUTOR

studno	roomno	appraiser
s1	2.26	capon
s2	IT206	watson
s3	2.82	capon
s4	2.82	capon
s5	2.34	watson
s6	IT206	watson

studno → name
studno → tutor

STUDENT' ⋈ TUTOR = STUDENT

Designing a relational schema

- Build a relational database
  - without redundancy
    - *normalisation*
  - without loss of information or gain of data
    - *lossless join decomposition*
  - without losing dependency integrity
    - *dependency preservation*

Multi-valued Dependencies and Fourth Normal Form

Multi-valued Dependencies

- a course has many lecturers
- a course has many texts
- lecturers and texts are independent
- a lecturer teaches many courses
- a text is used by many courses
- lecturer and text are independent sets
- for each courseno there is an associated set of lecturers
- for each courseno there is an associated set of texts
- the sets are independent.

courseno	lecturer	text
cs250	lindsey	Intro to SML
	capon	SML for beginners
		More SML
cs260	kahn	Raster Graphics
	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
	woods	Intro to Electronics
cs280	capon	Object Design

Multi-valued Dependencies

courseno →→ lecturer
courseno →→ text

This is in BCNF
key is
{courseno,lecturer,text}

courseno, lecturer,text
→ courseno, lecturer,text

- trivial dependencies

courseno	lecturer	text
cs250	lindsey	Intro to SML
cs250	lindsey	SML for beginners
cs250	lindsey	More SML
cs250	capon	Intro to ML
cs250	capon	ML for beginners
cs250	capon	More SML
cs260	kahn	Raster Graphics
cs260	kahn	Ray Tracing for Fun
cs260	bush	Raster Graphics
cs260	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
cs270	zobel	Intro to Electronics
cs270	woods	Chips with everything
cs270	woods	Intro to Electronics
cs280	capon	Object Design

Multi-valued Dependencies

Each TEXT is associated with all the LECTURERS that teach a COURSE

The attribute TEXT contains redundant values.

If TEXT were deleted from rows 1, 2 & 3 the values could be deduced from rows 4,5 & 6

courseno	lecturer	text
cs250	lindsey	Intro to SML
cs250	lindsey	SML for beginners
cs250	lindsey	More SML
cs250	capon	Intro to ML
cs250	capon	ML for beginners
cs250	capon	More SML
cs260	kahn	Raster Graphics
cs260	kahn	Ray Tracing for Fun
cs260	bush	Raster Graphics
cs260	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
cs270	zobel	Intro to Electronics
cs270	woods	Chips with everything
cs270	woods	Intro to Electronics
cs280	capon	Object Design

Multivalued Dependencies

courseno →→ lecturer
courseno →→ text

- if (c,l,t) and (c,l’,t’) appear then
- (c,l,t’) and (c,l’,t) appear also
- tuple (c,l,t) appears if c can be taught by l using text t
- for each course all possible combinations of lecturer and text appear

courseno	lecturer	text
cs250	lindsey	Intro to SML
cs250	lindsey	SML for beginners
cs250	lindsey	More SML
cs250	capon	Intro to ML
cs250	capon	ML for beginners
cs250	capon	More SML
cs260	kahn	Raster Graphics
cs260	kahn	Ray Tracing for Fun
cs260	bush	Raster Graphics
cs260	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
cs270	zobel	Intro to Electronics
cs270	woods	Chips with everything
cs270	woods	Intro to Electronics
cs280	capon	Object Design

Multi-Valued Dependencies

- Whenever X →→ Y holds in R
  so does X →→(R - (XY)).
- a MVD is trivial if Y ⊂ X or X ∪ Y = R.
  i.e. the two attributes form the whole relation
- non-trivial MV dependencies need at least 3 attributes.

Fourth Normal Form

- A relation R is in 4NF if it is in 3NF and there are no multi-valued dependencies between its attribute types
- A relation R is in 4NF iff whenever there exists a non-trivial multi-valued dependency in F^+ for R
  X →→ Y
- X is a superkey for R, i.e. all attributes are functionally dependent on X.
- Any relation in 3NF is transformed in 4NF
  - Detect any multi-valued dependencies
  - Decompose relation

Fourth Normal Form

courseno	lecturer	text
cs250	lindsey	Intro to SML
cs250	lindsey	SML for beginners
cs250	lindsey	More SML
cs250	capon	Intro to ML
cs250	capon	ML for beginners
cs250	capon	More SML
cs260	kahn	Raster Graphics
cs260	kahn	Ray Tracing for Fun
cs260	bush	Raster Graphics
cs260	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
cs270	zobel	Intro to Electronics
cs270	woods	Chips with everything
cs270	woods	Intro to Electronics
cs280	capon	Object Design

courseno →→ lecturer
courseno →→ text

courseno	lecturer
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

trivial dependencies only

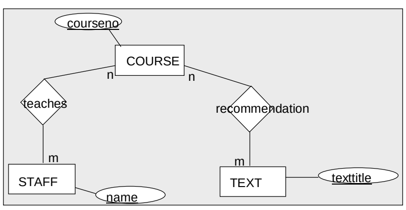
courseno	text
cs250	Intro to SML
cs250	SML for beginners
cs250	More SML
cs260	Raster Graphics
cs260	Ray Tracing for Fun
cs270	Chips with everything
cs270	Intro to Electronics
cs280	Object Design

Lossless join decomposition into 4NF

- Algorithm:
  Decomposition D, relation R
  1. set D := { R } ;
  2. while there is a relation schema Q in D that is not in 4NF do
     begin
     choose a relation schema Q in D that is not in 4NF;
     find a non-trivial MVD X →→ Y in Q that violates 4NF;
     replace Q in D by two schemas
     (Q -Y) and (X ∪ Y)
     end;

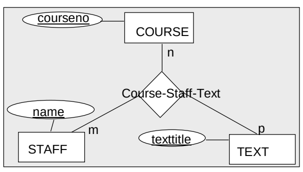
Fourth Normal Form EER modelling

- Leads to correctly normalised relational schema



Fourth Normal Form EER modelling

- Leads to relational schema that is not in 4NF



Conclusions

- Data Normalisation is a technique that ensures the basic properties of the relational model
  - no duplicate tuples
  - no nested relations
- Data normalisation is sometimes used as the only technique for database design—implementation view
- A more appropriate approach is to complement conceptual modelling with data normalisation

Lossless or Non-additive Join Algorithm

Decomposition D, relation R

1. set D := {R} ;
2. while there is a relation schema Q in D that is not in BCNF do
   begin
   choose a relation schema Q in D that is not in BCNF;
   find a functional dependency $X \rightarrow Y$ in Q that violates BCNF;
   replace Q in D by two schemas
   R1 (Q - Y)   leave copy of X in relation to be foreign key for R2
   and
   R2 ($X \cup Y$)   new relation for functional dependency and its closure,
   X will be the primary key

end;