Database System Concepts

Chapter 7: Relational Database Design

Departamento de Engenharia Informática
Instituto Superior Técnico

1st Semester
2009/2010

Outline

Outline

What is a "Good" Schema?

Suppose that, instead of *borrower* and *loan* we have a single relation:

$$loan(customer\_name, loan\_number, amount)$$

What is a "Good" Schema?

Suppose that, instead of *borrower* and *loan* we have a single relation:

$$loan(customer\_name, loan\_number, amount)$$

Result would be possible repetition of information

customer_id	*loan_number*	*amount*
...	...	...
Jones	L-100	10 000
Smith	L-100	10 000
Lindsay	L-100	10 000
...	...	...

Data Redundancy

- Data redundancy has implications in the coherence and completeness of the data
- The following problems (or *anomalies*) are well known:
  - Insertion: occurs when independent facts cannot be inserted as independent data in the database
  - Deletion: occurs when, by deleting an item, the deletion of other independent items also occurs
  - Update: occurs when updating an item implies the alteration of other independent items

An Example

- Take the following relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

- Anomalies occur when:

An Example

- Take the following relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

- Anomalies occur when:
  - Insertion: we can only store the price of an item if there are orders for the item

An Example

- Take the following relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

- Anomalies occur when:
  - Insertion: we can only store the price of an item if there are orders for the item
  - Deletion: if we delete an order, we loose the price information of the items

An Example

- Take the following relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

- Anomalies occur when:
  - Insertion: we can only store the price of an item if there are orders for the item
  - Deletion: if we delete an order, we loose the price information of the items
  - Update: if we change the price of an item, we need to update all the orders containing the item

A Theory for "Good" Relational Design

We need a formal method to:

- Decide whether a particular relation R is in "good" form.
- In the case that a relation R is not in *good* form, decompose it into a set of relations $\{R_1, R_2, \ldots, R_n\}$ such that
  - each relation is in good form
  - the decomposition has no loss of information
- Our theory is based on functional dependencies

Outline

An Example Revisited

- The problem with the relation

$$order(order\_id, item\_id, quantity, unit\_price)$$

as to do with dependency relations between its attributes

  - Through the *order_id* we know the *item_id*, *quantity* and *unit_price*
  - But we need only the *item_id* to know the *unit_price*

- We say that:
  - the *order_id* determines the *item_id*, *quantity* and *unit_price*
  - the *item_id* determines the *unit_price*

Functional Dependencies

- Let R be a relation schema where

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The functional dependency

$$\alpha \rightarrow \beta$$

holds on R if and only if, for any legal relations $r(R)$, for any tuples t_1 and t_2 of r

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t1[\beta] = t2[\beta]$$

- In other words

α determines β if, for every value of α there is one and only one value of β

Functional Dependencies (cont.)

Example

A	B
1	4
1	5
3	7

- Does $A \rightarrow B$ hold?
- Does $B \rightarrow A$ hold?

Functional Dependencies (cont.)

Example

A	B
1	4
1	5
3	7

- Does $A \rightarrow B$ hold?
- Does $B \rightarrow A$ hold?

- Functional dependencies are determined from the semantics of the context we are trying to model
  - They can not be determined from a given set of tuples
  - From a set of tuple we can only verify that a given dependency does not hold

Functional Dependencies and Keys

- A functional dependency is a generalization of the notion of a key
  - We say that K is a superkey of $r(R)$ if $K \rightarrow R$
  - K is a candidate key for $r(R)$ if and only if
    - $K \rightarrow R$ and
    - for no $\alpha \subset K$, $\alpha \rightarrow R$
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys.

Example

Consider the relation $loan(customer\_id, loan\_number, amount)$. We expect this functional dependency to hold:

$$loan\_number \rightarrow amount$$

but would not expect the following to hold:

$$amount \rightarrow customer\_name$$

Properties of Functional Dependencies

- Given a set F set of functional dependencies, there are certain other functional dependencies that are logically implied by F
  - For example: If $A \rightarrow B$ and $B \rightarrow C$, we can infer that $A \rightarrow C$
- The set of all functional dependencies logically implied by F is the closure of F
  - We denote the closure of F by F^+
- We can find all of F^+ by applying

Armstrong's Axioms

- Reflexivity: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
- Augmentation: if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$
- Transitivity: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

Procedure for Computing F^+

$F^+ = F$

repeat

for each functional dependency f in F^+

apply reflexivity and augmentation rules on f

add the resulting functional dependencies to F^+

for each pair of functional dependencies f_1 and f_2 in F^+

if f_1 and f_2 can be combined using transitivity

then add the resulting functional dependency to F^+

until F^+ does not change any further

An Example

- Consider the relation $R(A, B, C, G, H, I)$ and the set

$$F = \{ A \rightarrow B \\ A \rightarrow C \\ CG \rightarrow H \\ CG \rightarrow I \\ B \rightarrow H \}$$

- Some members of F^+ are
  - $A \rightarrow H$ by transitivity from $A \rightarrow B$ and $B \rightarrow H$
  - $AG \rightarrow I$ by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$ and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$ by augmenting $CG \rightarrow I$ to infer $CG \rightarrow CGI$, and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

Derived Rules

- We can further simplify manual computation of F^+ by using the following additional rules:
  - Self-reflexivity: $\alpha \rightarrow \alpha$ holds
  - Union: If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds
  - Decomposition: If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds
  - Pseudotransitivity: If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds
- The above rules can be inferred from Armstrong's axioms

Closure of Attribute Sets

Definition

Given a set of attributes α we define the closure of α under F (denoted by α^+) as the set of attributes that are functionally determined by α under F

- Algorithm to compute α^+:

  result $= \alpha$;
  while (changes to *result*) **do**
      for each $\beta \rightarrow \gamma$ in F **do**
      begin
          if $\beta \subseteq$ *result* **then** *result* $=$ *result* $\cup \gamma$
      end

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

1. $result = AG$

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

1. $result = AG$
2. $result = ABCG$ ($A \rightarrow C$ and $A \rightarrow B$)

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

1. $result = AG$
2. $result = ABCG$ ($A \rightarrow C$ and $A \rightarrow B$)
3. $result = ABCGH$ ($CG \rightarrow H$ and $CG \subseteq AGBC$)

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

1. $result = AG$
2. $result = ABCG$ ($A \rightarrow C$ and $A \rightarrow B$)
3. $result = ABCGH$ ($CG \rightarrow H$ and $CG \subseteq AGBC$)
4. $result = ABCGHI$ ($CG \rightarrow I$ and $CG \subseteq AGBCH$)

An Example

Consider the relation $R(A, B, C, G, H, I)$ and the set $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$:

1. $result = AG$
2. $result = ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
3. $result = ABCGH$ $(CG \rightarrow H$ and $CG \subseteq AGBC)$
4. $result = ABCGHI$ $(CG \rightarrow I$ and $CG \subseteq AGBCH)$

$$AG \rightarrow ABCGHI$$

Closure and Keys

The following questions are equivalent

- Is AG a superkey?
- Does $AG \rightarrow R$?
- Is $(AG)^+ \supseteq R$?

The following questions are equivalent:

- Is AG a candidate key?
- Is any subset of AG a superkey?
- Does $A \rightarrow R$ or $G \rightarrow R$?
- Is $A^+ \supseteq R$ or $G^+ \supseteq R$?

Outline

Normal Forms

1^{st} Normal Form

Regards the structure of each attribute in a relation

Normal Forms

1st Normal Form

Regards the structure of each attribute in a relation

2nd Normal Form

Regards the dependencies between key and non-key attributes

Normal Forms

1st Normal Form

Regards the structure of each attribute in a relation

2nd Normal Form

Regards the dependencies between key and non-key attributes

3rd Normal Form

Regards inter-dependencies between non-key attributes

Normal Forms

1st Normal Form

Regards the structure of each attribute in a relation

2nd Normal Form

Regards the dependencies between key and non-key attributes

3rd Normal Form

Regards inter-dependencies between non-key attributes

Boyce-Codd Normal Form

Regards inter-dependencies between key attributes

1st Normal Form

Definition

A relational schema R is in 1st normal form if the domains of all attributes of R are atomic

- Domain is atomic if its elements are considered to be indivisible units
- Atomicity is actually a property of how the elements of the domain are used
  - Example: Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
  - If the first two characters are extracted to find the department, the domain of roll numbers is not atomic
  - Doing so is a bad idea: leads to encoding of information in application program rather than in the database

Note: we will assume all relations are in first normal form.

Problems of the 1^{st} Normal Form

- Consider the relation

$$works(\underline{person\_id, project}, budget, time\_spent)$$

where

$$project \longrightarrow budget$$

- It is in 1NF, however it still has the insertion, update and deletion anomalies

2^{nd} Normal Form

- Some useful definitions:
  - An attribute A is completely dependent on a set of attributes α if $\alpha \rightarrow A$ and no subset of α determines A
  - We say an attribute is a key attribute is it belongs to a candidate key

Definition

A relational schema R is in 2^{nd} normal form if it is in 1NF and every non-key attribute is completely dependent on the candidate keys

- Every 1NF relation with only non-composite candidate keys are also 2NF
- Nothing is said regarding inter-dependencies between key attributes
- Nothing is said regarding inter-dependencies between non-key attributes

An Example

- Consider the relation

$$R(\underline{A, B}, C, D)$$

where $A \rightarrow D$ (note that $AB \rightarrow C$ and $AB \rightarrow D$ are implied)

- Is R in 2NF?

An Example

- Consider the relation

$$R(\underline{A, B}, C, D)$$

where $A \rightarrow D$ (note that $AB \rightarrow C$ and $AB \rightarrow D$ are implied)

- Is R in 2NF?

- Decomposing R yields

$$R1(\underline{A, B}, C) \text{ and } R2(\underline{A}, D)$$

Both relations are now in 2NF.

Another Example

- Consider the relation

$$exam(\underline{student, course}, teacher, grade)$$

  where $course \rightarrow teacher$

- Is R in 2NF?

Another Example

- Consider the relation

$$exam(\underline{student, course}, teacher, grade)$$

where $course \rightarrow teacher$

- Is R in 2NF?

- Decomposing $exam$ yields

$$exam(\underline{student, course}, grade) \text{ and } course(\underline{course}, teacher)$$

Both relations are now in 2NF.

Yet Another Example

- Consider the relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

  where $item\_id \longrightarrow unit\_price$
- Is R in 2NF?

Yet Another Example

- Consider the relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

  where $item\_id \longrightarrow unit\_price$

- Is R in 2NF?

Yes it is!

And yet...

Problems of the 2nd Normal Form

- 2NF does not guarantee that there is no redundancy
  - For instance, consider updating the unit price of an item in relation

$$order(\underline{order\_id}, item\_id, quantity, unit\_price)$$

  where $item\_id \rightarrow unit\_price$

- Redundancy arises because of dependencies between non-key attributes

Eliminating the Anomalies

- To eliminate the anomalies, we need to eliminate the dependencies between non-key attributes
  - I.e. $item\_id \longrightarrow unit\_price$
- We can do this by splitting the relation into

$$order(\underline{order\_id}, item\_id, quantity)$$
$$item(\underline{item\_id}, unit\_price)$$

- The dependencies still hold, but now there are no dependencies between non-key attributes

3rd Normal Form

Definition

A relational schema R is in 3rd normal form if it is in 2NF and there are no dependencies between non-key attributes

- All relations that have only one non-key attribute and are 2NF are also 3NF
- Nothing is said regarding inter-dependencies between key attributes

An Example

- Consider the relation

$$R(\underline{A}, B, C)$$

  where $B \rightarrow C$
- Is R in 3NF?

An Example

- Consider the relation

$$R(\underline{A}, B, C)$$

  where $B \rightarrow C$
- Is R in 3NF?
- Decomposing R yields

$$R1(\underline{B}, C), \text{ where } B \rightarrow C$$
$$R2(\underline{A}, B), \text{ where } A \rightarrow B$$

- Both relations are now in 3NF, i.e., there are no dependencies between non-key attributes

Another Example

- Consider the relation

$$employee(\underline{id}, name, department\_number, section\_number, manager)$$

  where $department\_number \longrightarrow section\_number$ and $section\_number \longrightarrow manager$

- Is *employee* in 3NF?

Another Example

- Consider the relation

$$employee(\underline{id}, name, department\_number, section\_number, manager)$$

where $department\_number \rightarrow section\_number$ and $section\_number \rightarrow manager$

- Is *employee* in 3NF?
- Decomposing *employee* yields

$$employee(\underline{id}, name, department\_number)$$
$$department(\underline{department\_number}, section\_number)$$
$$section(\underline{section\_number}, manager)$$

- All relations are now in 3NF

Yet Another Example

- Consider the relation

  $works(\underline{employee\_id, department\_id}, budget, time, project\_id)$

  where $project\_id \rightarrow budget$
- Is *works* in 3NF?

Yet Another Example

- Consider the relation

  $$works(\underline{employee\_id, department\_id}, budget, time, project\_id)$$

  where $project\_id \rightarrow budget$
- Is *works* in 3NF?
- Decomposing *works* yields

  $$works(\underline{employee\_id, department\_id}, time, project\_id)$$
  $$project(\underline{project\_id}, budget)$$
- Both relations are now in 3NF

A Final Example

- Consider the relation

$$uses(project, director, item, quantity)$$

where

$$\begin{aligned} project, item &\longrightarrow quantity \\ project &\longrightarrow director \\ director &\longrightarrow project \\ director, item &\longrightarrow quantity \end{aligned}$$

- In which normal form is R?

A Final Example

- Consider the relation

$$uses(project, director, item, quantity)$$

where

$$\begin{aligned} project, item &\rightarrow quantity \\ project &\rightarrow director \\ director &\rightarrow project \\ director, item &\rightarrow quantity \end{aligned}$$

- In which normal form is R?
- R is in 3NF, however...

Problems with the 3rd Normal Form

- The relation

$$uses(project, director, item, quantity)$$

still allows some anomalies

  - The same project director is stored several times
  - The director is stored only when items are used
  - The project is stored only when the director is known
  - Changing the project's director may imply changing several tuples

- This anomalies arise from the fact that there are dependencies between key attributes (*director* and *project*)

Boyce-Codd Normal Form

Definition

A relational schema R is in Boyce-Codd Normal Form if it is 3NF and every determinant is a superkey

- In other words, for every dependency $\alpha \rightarrow \beta$ in R^+, either it is a trivial dependency (i.e., $\beta \subseteq \alpha$), or α is a superkey
- In BCNF
  - There are no dependencies between non-key attributes (3NF)
  - There are no dependencies between subsets of key attributes
- BCNF is different from 3NF only when
  - There is more than one candidate key
  - Candidate keys are composed of several attributes

An Example

- Consider the relation

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

  where $AB \rightarrow CD$, $BC \rightarrow AD$, and $A \rightarrow C$
- Is R in BCNF?

An Example

- Consider the relation

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

  where $AB \rightarrow CD$, $BC \rightarrow AD$, and $A \rightarrow C$
- Is R in BCNF?
- Decomposing R yields

$$R1(A, C)$$
$$R2(A, B, D)$$

- Both relations are now in BCNF
  - Note that not all dependencies where preserved

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

  where $teacher \longrightarrow course$

- In which normal form is relation *class*?

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

  where $teacher \rightarrow course$

- In which normal form is relation *class*?
- What are the possible anomalies?

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

where $teacher \rightarrow course$

- In which normal form is relation *class*?
- What are the possible anomalies?
- What are the problems with the following decompositions?

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

where $teacher \rightarrow course$

- In which normal form is relation *class*?
- What are the possible anomalies?
- What are the problems with the following decompositions?
  - $class1(student, teacher)$ and $class2(course, teacher)$

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

  where $teacher \rightarrow course$
- In which normal form is relation *class*?
- What are the possible anomalies?
- What are the problems with the following decompositions?
  - $class1(student, teacher)$ and $class2(course, teacher)$
  - $class1(course, student)$ and $class2(course, teacher)$

Another Example

- Consider the relation

$$class(\underline{course, student}, teacher)$$

  where $teacher \rightarrow course$
- In which normal form is relation *class*?
- What are the possible anomalies?
- What are the problems with the following decompositions?
  - $class1(student, teacher)$ and $class2(course, teacher)$
  - $class1(course, student)$ and $class2(course, teacher)$
  - $class1(course, student)$ and $class2(student, teacher)$

Schema Decomposition

- When we decompose a schema to achieve normalization, some loss of information may occur

Example

1. Consider the relation $r(A, B, C)$, where $A \rightarrow B$ and $C \rightarrow B$

A	B	C
1	X	1
3	X	2
2	Y	3
4	Y	4

Schema Decomposition

- When we decompose a schema to achieve normalization, some loss of information may occur

Example

1. Consider the relation $r(A, B, C)$, where $A \rightarrow B$ and $C \rightarrow B$

A	B	C
1	X	1
3	X	2
2	Y	3
4	Y	4

2. We can decompose into to $r_1(A, B)$ and $r_2(B, C)$

A	B
1	X
3	X
2	Y
4	Y

B	C
X	1
X	2
Y	3
Y	4

Schema Decomposition

- When we decompose a schema to achieve normalization, some loss of information may occur

Example

1. Consider the relation $r(A, B, C)$, where $A \rightarrow B$ and $C \rightarrow B$

A	B	C
1	X	1
3	X	2
2	Y	3
4	Y	4

2. We can decompose into to $r_1(A, B)$ and $r_2(B, C)$

A	B
1	X
3	X
2	Y
4	Y

B	C
X	1
X	2
Y	3
Y	4

3. However, when we perform $r_1 \bowtie r_2$, we get

A	B	C
1	X	1
1	X	2
3	X	1
3	X	2
2	Y	3
2	Y	4
4	Y	3
4	Y	4

Lossless-Join Decomposition

Definition

Consider the relational schema $R = (R_1, R_2)$. A decomposition of $r(R)$ into R_1 and R_2 is lossless-join if

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r)$$

Theorem (Heath's Theorem)

A decomposition of R into R_1 and R_2 is lossless-join if and only if at least one of the following dependencies is in F^+:

$$R1 \cap R2 \rightarrow R1$$

$$R1 \cap R2 \rightarrow R2$$

Example

In the previous example, the decomposition into $r_1(A, B)$ and $r_2(A, C)$ is a lossless-join decomposition, since $\{A, B\} \cap \{A, C\} \rightarrow \{A, B\}$ (because $A \rightarrow B$).

Dependency Preservation

- Suppose we decompose $r(R)$ into R_1, R_2, ..., R_n

Definition

Let F_i be the set of dependencies in F^+ that include only attributes in R_i. A decomposition is dependency preserving if

$$(F_1, F_2, \ldots, F_n)^+ = F^+$$

- If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive.

Dependency Preservation (cont.)

- There are some situations where
  - BCNF is not dependency preserving, and
  - efficient checking for FD violation on updates is important
- That is why we need the 3rd Normal Form
  - Allows some redundancy (with resultant problems)
  - But functional dependencies can be checked on individual relations without computing a join.
  - There is always a lossless-join, dependency-preserving decomposition into 3NF

Note: It is always possible to decompose a relation into a set of relations that are in BCNF such that the decomposition is lossless, but it may not be possible to preserve dependencies.

Relational Design Goals

- Goal for a relational database design is:
  - BCNF
  - Lossless join
  - Dependency preservation
- If we cannot achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy due to use of 3NF
- Interestingly, SQL does not provide a direct way of specifying functional dependencies other than superkeys
  - Can specify FDs using assertions, but they are expensive to test
  - Even if we had a dependency preserving decomposition, using SQL we would not be able to efficiently test a functional dependency whose left hand side is not a key

4^{th} Normal Form

- Multivalued dependencies regard dependencies between the values present in tuples
  - For instance, in a relation *teaches*(*course*, *teacher*, *book*) we can say that *course* multidetermines *teacher* and *course* multidetermines *book* (denoted by $course \twoheadrightarrow teacher$ and $course \twoheadrightarrow book$)
  - In this case, if we were to add a new book to the course, we would have to add a new tuple for each of the course teachers
- The 4^{th} Normal Form requires that a database has no multivalued dependencies

Other Normal Forms

- The 5th Normal Form (a.k.a. project-join normal form) generalizes multivalued dependencies through the concept of join dependencies
- The Domain-key Normal Form (DKNF) requires that the database contains no constraints other than domain constraints and key constraints
- The 6th Normal Form defined when extending the relational model to take into account the temporal dimension (
- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists. Hence rarely used.

Outline

ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
  - Example: an *employee* entity with attributes *department_number* and *department_address*, and a functional dependency $department\_number \rightarrow department\_address$
  - Good design would have made department an entity
- Functional dependencies from non-key attributes of a relationship set possible, but rare

Other Design Issues

- Some aspects of database design are not caught by normalization
- Examples of bad database design, to be avoided:
  - Instead of *earnings*(*company_id*, *year*, *amount*), use *earnings_*2000, *earnings_*2001, *earnings_*2002, etc., all on the schema (*company*$_i$*d*, *earnings*)
    - Above are in BCNF, but make querying across years difficult and needs new table each year
  - Use *company_year*(*company_id*, *earnings_*2000, *earnings_*2001, *earnings_*2002)
    - Also in BCNF, but also makes querying across years difficult and requires new attribute each year
    - Is an example of a *crosstab*, where values for one attribute become column name
    - Used in spreadsheets and in data analysis tools

Denormalization for Performance

- May want to use non-normalized schema for performance
- For example, displaying *customer_name* along with *account_number* and *balance* requires join of *account* with *depositor*
- Alternative 1: Use denormalized relation containing attributes of *account* as well as *depositor* with all above attributes
  - faster lookup
  - extra space and extra execution time for updates
  - extra coding work for programmer and possibility of error in extra code
- Alternative 2: use a materialized view, defined as $account \bowtie depositor$
  - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

Modeling Temporal Data

- Temporal data have an association time interval during which the data are valid
- A snapshot is the value of the data at a particular point in time
- Adding a temporal component results in functional dependencies like

$$customer\_id \rightarrow customer\_street, customer\_city$$

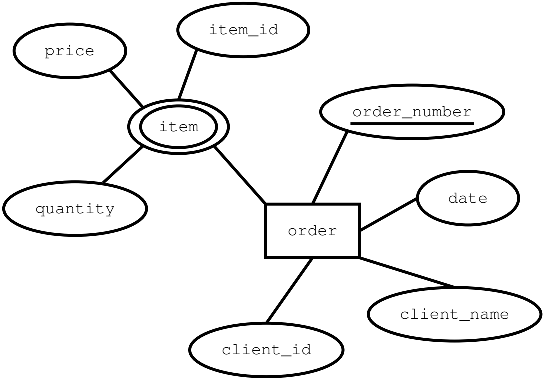
  not to hold, because the address varies over time

- A temporal functional dependency holds on schema R if the corresponding functional dependency holds on all snapshots for all legal instances $r(R)$

Outline

Example 1

- Consider the following E-R diagram



- If we convert it to relational "as is", in which NF will the resulting schema be?
- What anomalies can occur?

Example 2

- Consider the following relation

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

where the following dependencies hold

$$B \longrightarrow D$$
$$D \longrightarrow B$$
$$ABD \longrightarrow C$$

- What are the candidate keys?
- In which normal form is R?
- How can we decompose it into BCNF relations?
- Are any dependencies lost?

Example 3

- Consider the following relation

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

where the following dependencies hold

$$A \rightarrow BCD$$
$$B \rightarrow ACD$$
$$CD \rightarrow AB$$

- What are the candidate keys?
- In which normal form is R?

Example 4

- Consider the following relation

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

where the following dependencies hold

$$AB \longrightarrow CDE$$
$$D \longrightarrow E$$

- What are the candidate keys?
- In which normal form is R?
- If instead of $AB \longrightarrow CDE$ we had that $ABD \longrightarrow CE$, in which normal form would R be?

Example 5

- Consider the following relation

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

where the following dependencies hold

$$AB \rightarrow CD$$

- Indicate a functional dependency that would make R not be in the 2NF.
- Indicate a functional dependency that would make R be in 2NF, but not be in the 3NF.
- Indicate a functional dependency that would make R be in 3NF, but not be in the BCNF.

Example 6

- Consider the following relation

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

and its decomposition into

$$R1(B, C) \text{ and } R2(A, C)$$

- Which functional dependencies should hold to assure that this decomposition is lossless-join?
- Assuming that this is a lossless-join decomposition, write the relational algebra expression that can rebuild R from $R1$ and $R2$.

End of Chapter 7