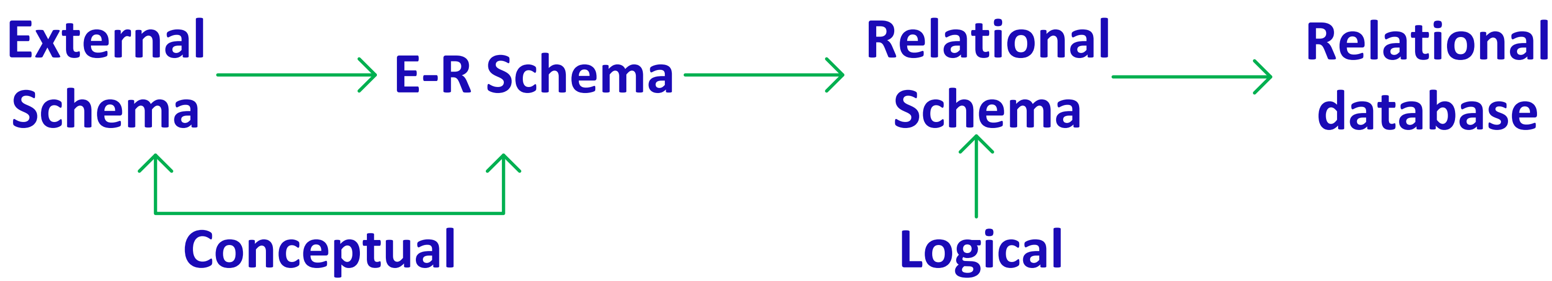
CS430: Database Systems

Dr. Chetan Jaiswal,
Department of Computer Science,
Truman State University
cjaiswal@truman.edu

Relational Database Design

The path



Chapter 14: Relational Database Design

- Informal Design Guidelines for Relational Schema
- Functional Dependencies
- Normal Forms

Relational Database Design

- **Informal design guidelines**
  - Levels at which we can discuss *goodness* of relational schemas
  - Approaches to database design
  - Making sure attribute semantics are clear
  - Reducing redundant information in tuples
  - Reducing NULL values in tuples
  - Disallowing possibility of generating spurious tuple

Relational Database Design

- **Informal design guidelines**
  - Minimize redundancy to save storage
  - Select attributes that are logically and semantically connected
  - Schema should not be too big (many attributes in one schema)

Relational Database Design

- **A good relational schema**
  - A good relational schema contains a set of relevant attributes of the entity it represents where every attribute is clearly related (directly or indirectly) to other attributes of the schema. A relation of the schema should require minimum storage space and have minimum data redundancy

Relational Database Design

- **An example**
  - **The attribute interest rate is not related (logically or semantically) to other attributes. A bad relational schema**

Department

Dnumber	Dlocation	Dname	Interest rate

  - **Make it a good schema by replacing it with "Dsize"**

Department

Dnumber	Dlocation	Dname	Dsize

Relational Database Design

- **An example**
  - **Consider the following Sport relation**

Assume one word is needed to store the value of an attribute. Sport will require 20 words.

Remove duplicate values and create S1 and S2. S1 + S2 requires 18 words.

Sport

Sname	Inst_id	Inst_name	Expertise	Fee
Football	1	Tom	Football	200
Tennis	1	Tom	Tennis	200
Baseball	2	Peter	Baseball	300
Golf	2	Peter	Golf	300

S1

Inst_name	Inst_id	Expertise 1	Expertise 2	Fee
Tom	1	Football	Tennis	200
Peter	2	Baseball	Golf	300

S2

Sname	Inst_id
Football	1
Tennis	1
Baseball	2
Golf	2

Relational Database Design

- **Modification problems**
  - **Consider the following Student relation**

Student

Stu_id	Activity	Fee
100	Skiing	200
100	Golf	65
150	Swimming	50
175	Squash	50
175	Swimming	50
200	Swimming	50
200	Golf	65

Relational Database Design

- ## Modification problems
  - Suppose Student 100 gave up skiing. The first record must be deleted from the database. This deletion has bad side effect which makes the database incomplete.
  - Effect: Removes more information than necessary. If a new student with id 190 wants to enroll in Skiing then he/she cannot because fee information is no longer available. Further, you want to add a new activity, say Basketball and charge $200. You cannot do that either.

Student

Stu_id	Activity	Fee
~~100~~	~~Skiing~~	~~200~~
100	Golf	65
150	Swimming	50
175	Squash	50
175	Swimming	50
200	Swimming	50
200	Golf	65

A bad relation

Relational Database Design

- **Dependency theory**
  - **These modification problems are called *Modification Anomalies*. They do not let you to modify a relation.**
  - **They are minimized (not easy to completely eliminate them)**
  - **To understand the ways of minimizing them, we need to understand the dependency theory. The theory illustrates how one attribute value (or a set of attribute values) depends on the value of another attribute value (or a set of attribute values). Thus, how *Ai* depends on *Aj* ($i \neq j$)**

Relational Database Design

- **Dependency theory**
  - To maintain consistency a relation must satisfy a set of *integrity constraints*. A consistent relation reflects the facts. For example, if an instructor teaches a database course then the database must reflect this information, i.e., the relation which stores this information must satisfy constraints related to instructor and course attributes. We need to formalize these concepts

Relational Database Design

- **Key Dependency**
  - **One of the most common dependency.**
  - **Formally, Given a relation scheme *R(U)* where ($U = A1, A2, \ldots, An$) a key dependency is expressed as *key*(*K*), (where $K \subseteq U$) and is satisfied by a relation *r*, if and only if $ti(K) \neq tj(K)$. When constraints encompass non-key attributes, then they are called as Functional Dependencies (FD) and key dependency becomes a subset of functional dependencies. These functional dependencies are the basis of relational database design.**

Relational Database Design

- **Functional Dependencies (FD)**
  - **An FD indicates how the value of an attribute determines the value of another attribute. For example, If the value of Ssn is given then it will identify the value of another attribute of the relation. Thus, if a value of Ssn = 123456789 is given then it will identify the value of Lname = Smith. This means that whenever you get Ssn = 123456789 then the value of Lname will only be “Smith”**

Relational Database Design

Functional Dependencies (FD)

- Consider the following relation (Schedule):
- An FD indicates how the value of an attribute determines another attribute

Schedule

Pilot	Flight	Date	Departs
Cushing	83	9 Aug	10:15a
Cushing	116	10 Aug	1:25p
Clark	281	8 Aug	5:50a
Clark	301	12 Aug	6:35p
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Chin	116	12 Aug	1:25p
Copley	281	9 Aug	5:50a
Copley	281	13 Aug	5:50a
Copley	412	15 Aug	1:25p

Relational Database Design

- **Functional Dependencies (FD)**
  - **Constraints on Schedule**
    1. **Exactly one time for one flight**
    2. **For a {pilot, date, time} there is one flight**
    3. **For a {flight, date} there is one pilot**

These restrictions indicate how this relation can be processed (modified, expand, contract etc.) These are examples of Functional Dependencies (FD).

Relational Database Design

Functional Dependency Notation

- "→" is used to indicate FD between two set of attributes. So X → Y will mean X functionally determines Y. X is called the Left side of FD and Y the Right side of FD. X is also called the determinant of the FD X → Y
- Example

  If we have Activity → Fee, then the value of Activity determines it Fee. If the value of the Activity changes then the value of Fee must also change

Relational Database Design

- **Formally**
  - **Let *r* be a relation on *R(X, Y)*. if *r* satisfies the FD $X \rightarrow Y$ then if $t1(X) = t2(X)$, we must have $t1(Y) = t2(Y)$. This means that the *Y* value of a tuple in *r(R)* is determined by the *X* value of that tuple in *r(R)*, i.e., *Y is functionally dependent on X or X functionally determines Y*.**

Relational Database Design

- **Algorithm to identify Flight → Depart in Schedule**
  - Result: FD exists because whenever Flight = 281 we have Depart = 5:50a, whenever Flight = 83, we have Depart = 10:15a. Note that FD is the relationship among attributes of a relation

Pilot	Flight	Date	Departs
Cushing	83	9 Aug	10:15a
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Cushing	116	10 Aug	1:25p
Chin	116	12 Aug	1:25p
Clark	281	8 Aug	5:50a
Copley	281	9 Aug	5:50a
Copley	281	13 Aug	5:50a
Clark	301	12 Aug	6:35p
Copley	412	15 Aug	1:25p

Relational Database Design

- Class exercise
  - Does Depart → Flight in Schedule?
  - Does Date → Flight in Schedule?

Pilot	Flight	Date	Departs
Cushing	83	9 Aug	10:15a
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Cushing	116	10 Aug	1:25p
Chin	116	12 Aug	1:25p
Clark	281	8 Aug	5:50a
Copley	281	9 Aug	5:50a
Copley	281	13 Aug	5:50a
Clark	301	12 Aug	6:35p
Copley	412	15 Aug	1:25p

Relational Database Design

- **Problem with this algorithm**
  - **Process intensive. If a relation has 1 million tuples and its degree is high then the search time is very large.**
- **Solution**
  - **Armstrong's axioms**
- **We first discuss graphical representation of FDs.**

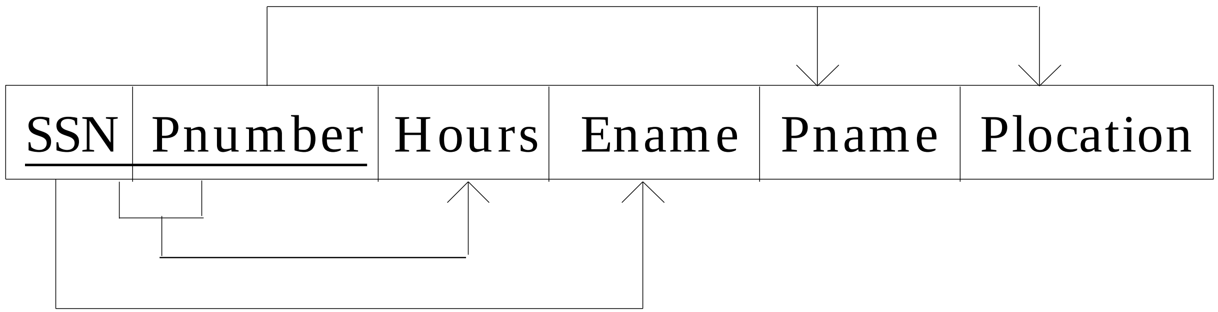
Relational Database Design

- **Graphical representation of FDs.**
  - **The head of the arrows pointing to the right side of FDs and the tails are connected to the left side of FDs.**
  - **Schema Emp_Dept. FDs are**

  - **SSN → {Ename, Bdate, Address, Dnumber, Dname, Dmgrssn}**
  - **Dnumber → {Dname, Dmgrssn}.**

Relational Database Design

- Graphical representation of FDs.
  - The head of the arrows pointing to the right side of FDs and the tails are connected to the left side of FDs.
  - Schema Emp_Proj Fds are



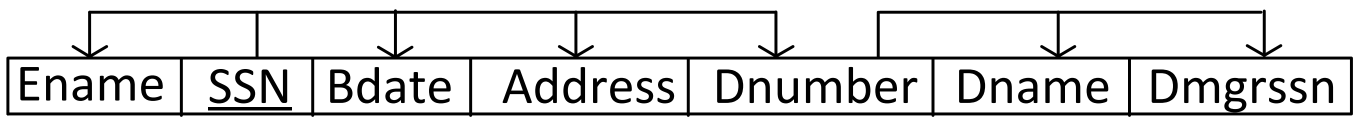
  - {SSN, Pnumber} → Hours
  - SSN → Fname
  - Pnumber → {Pname, Plocation}.

Relational Database Design

- **Important points to remember**
  - One must remember that FDs in a relation are defined by the database designer. In the above examples these FDs may not be valid if they have not been defined.
  - A relational schema R may have n instances. If an FD for R is identified then every instance of R must satisfy the FD. A FD on R is false if one instance of a relation satisfies it while another instance does not. To verify if a certain FD is true one has to check all possible instances of R.

Closure of an FD

- A database designer defines a set of FDs which is identified as F. Some additional FDs may be derived from F, that are called derived FDs. The set of all such FDs derived from F is called the closure of F and represented by F+. Consider schema EMP-DEPT:



- **Defined (F):** SSN → {Ename, Bdate, Address, Dnumber}
  Dnumber → {Dname, Dmgrssn}
- **Derived (F+):** SSN →{Dname, Dmgrssn}

Relational Database Design

Inference Rules: Augmentation rule

- Example; We want to show that $F = A \rightarrow B$ is satisfied by the following relation. We obtain $\Pi_B(\sigma_{A = a1}(r))$ from this relation. It gives only one tuple: b1. So whenever there is a1, there will only be b1.

r(A	B	C	D)
a1	b1	c1	d1
a2	b2	c1	d1
a1	b1	c1	d2
a3	b3	c2	d3

- We can see F+, which are

  $AB \rightarrow B$, $AC \rightarrow B$, $AD \rightarrow B$, $ABC \rightarrow B$, $ADB \rightarrow B$, $ACD \rightarrow B$ and $ABCD \rightarrow B$. It can also be seen that $AC \rightarrow BC$, $AD \rightarrow BD$, $ACD \rightarrow BCD$ and so on. $AC \rightarrow BC$ means whenever $t1(AC) = t2(AC)$, there will be $t1(BC) = t2(BC)$.

Relational Database Design

Inference Rules: Transitive rule

- It establishes that if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$
- Proof: Let r's F = {$X \rightarrow Y$, $Y \rightarrow Z$}. Let $t1 \in r$ and $t2 \in r$. We know that if $t1(X) = t2(X)$, then $t1(Y) = t2(Y)$ and also if $t1(Y) = t2(Y)$, then $t1(Z) = t2(Z)$. Therefore if $t1(X) = t2(X)$, then $t1(Z) = t2(Z)$. This is one of the most important axioms

Relational Database Design

Inference Rules: Transitive rule

- Example: To show that if $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

r(A	B	C	D)
a1	b1	c2	d1
a2	b2	c1	d2
a3	b1	c2	d1
a1	b1	c2	d3

- $\Pi_B(\sigma_{A\ =\ a1}\ (r))$ will give only one tuple: b1. $\Pi_C(\sigma_{b\ =\ b1}\ (r))$ will give only one tuple: c2. $\Pi_C(\sigma_{A\ =\ a1}\ (r))$ will give only one tuple: c2. This establishes that for each a1, there will be only b1 ($A \rightarrow B$) and for each b1 there will be only c2 ($B \rightarrow C$). Thus, whenever there is a1, there will be c2 ($A \rightarrow C$).

Relational Database Design

Inference Rules: Union or additive rule

- This axiom allows us to combine two or more FDs with the same left side. Thus, if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Proof

  If r satisfies $X \rightarrow Y$ and $X \rightarrow Z$ then $\Pi_Y(\sigma_{X=x}(r))$ and $\Pi_Z(\sigma_{X=x}(r))$ both have at most one tuple for any X-value x. If $\Pi_{YZ}(\sigma_{X=x}(r))$ had more than one tuple, then at least one of $\Pi_Y(\sigma_{X=x}(r))$ and $\Pi_Z(\sigma_{X=x}(r))$ would have more than one tuple. Thus $X \rightarrow YZ$

Relational Database Design

Inference Rules: Pseudotransitivity rule

- This rule allows us to extend the transitive rule further. Thus, if $\{X \rightarrow Y, WY \rightarrow Z\}$ then $WX \rightarrow Z$
- Proof

  Let r satisfy $X \rightarrow Y$, $WY \rightarrow Z$ and let t1 and t2 be tuples in r. We know that if t1(X) = t2(X), then t1(Y) = t2(Y) and also t1(WY) = t2(WY) then t1(Z) = t2(Z). From t1(WX) = t2(WX) we can deduce that t1(X) = t2(X) (because $X \subseteq WX$; from Reflexive rule) and so t1(Y) = t2(Y) and further t1(WY) = t2(WY), which implies t1(Z) = t2(Z). Thus, $WX \rightarrow Z$

Relational Database Design

- $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$
- Prove $BC \rightarrow ABCDE$?

$A \rightarrow BC$ given

$A \rightarrow B$, $A \rightarrow C$ decomposition

$B \rightarrow D$, so $A \rightarrow D$ given, transitive

$A \rightarrow CD$ union

$CD \rightarrow E$, so $A \rightarrow E$ transitive

$A \rightarrow ABCDE$ union of above steps

$E \rightarrow A$, so $E \rightarrow ABCDE$ given, transitive

$CD \rightarrow E$, so $CD \rightarrow ABCDE$ transitive

$B \rightarrow D$, so $BC \rightarrow CD$ augmentation

$BC \rightarrow ABCDE$ transitive

Relational Database Design

Normal Forms and Modification Anomalies

- **A relational database is a set of normalized relations. Each relation has either minimum or no modification anomalies. The database design, therefore, tries to minimize or eliminate modification anomalies from a relation. Thus, the database design process is as follows:**
- ***Normalize a non-normalize relation. Identify all modification anomalies that exist in this relation. Does it have modification anomalies?***
- ***NO: End of database design process.***
- ***YES: further normalize it. Continue this process iteratively until the relation either has no modification anomalies or they are minimized***

Relational Database Design

Non-Normalized relation

- It is a relation that has “repeating groups”
- A repeating group represent multiple values of an attribute for one value of another attribute
- Example

Degree (A non-normalized relation)

Student Name	Year	Degree
John	1990	MS
	2002	BS
Kumar	1967	BS
	1969	MS
	1983	Ph.D.

For one value of “Student Name”, there are two values of “Year” and two values of “Degree”. Thus, attributes “Year” and “Degree” are repeating groups. or
(John, (1990,2002), (MS,BS))
(John, 1990, MS, 2002, BS)

Relational Database Design

- **Non-Normalized → Normalized**
  - Degree relation must be normalize
  - How?
  - By repeating the values of Student name

Degree (A normalized relation)

Student Name	Year	Degree
John	1990	MS
John	2002	BS
Kumar	1967	BS
Kumar	1969	MS
Kumar	1983	Ph.D.

Relational Database Design

Normal Forms

- Normal forms (NF): A NF of a relation defines the type of modification anomalies it eliminates. There are First normal form (1NF), Second normal form (2NF), Third normal form (3NF), Boyce-Codd normal form (BCNF), Fourth normal form (4NF), Domain/Key normal form (DK/NF) and Fifth normal form (5NF). We will study only 1NF through 4NF

Relational Database Design

First Normal Form: 1NF

- A relation is in 1NF if it’s attributes does not contain repeating groups, i.e., all its attributes are atomic
- Example

Order

Ono	Date	Part_descrip	Pno	No_Ordered	Price
12489	90287	Iron	AX12	11	14.95
12491	90287	Stove	BT04	1	402.99
12491	90287	Washer	BZ66	1	311.99
12494	90487	Bike	CB03	4	175.00
12495	90487	Mixer	CX11	2	57.95
12498	90587	Skates	AZ52	2	22.95
12498	90587	Baseball	BA74	4	4.95
12500	90587	Stove	BT04	1	402.99

Order is in 1NF.
Degree: 6
Cardinality: 8
Primary Key: {Ono, Pno}
Superkey: many

Relational Database Design

Modification Anomalies

- Update: Yes. A change to the description of BT04 requires several changes
- Addition: Yes. In the absence of incomplete update, BT04 may have different values in other attributes.
- Deletion: By deleting BT04 we lose that BT04 represents Stove
- Inconsistent data: In the absence of incomplete update, BT04 may have different values in other attributes

Order

Ono	Date	Part_descrip	Pno	No_Ordered	Price
12489	90287	Iron	AX12	11	14.95
12491	90287	Stove	BT04	1	402.99
12491	90287	Washer	BZ66	1	311.99
12494	90487	Bike	CB03	4	175.00
12495	90487	Mixer	CX11	2	57.95
12498	90587	Skates	AZ52	2	22.95
12498	90587	Baseball	BA74	4	4.95
12500	90587	Stove	BT04	1	402.99

Conclusion

Order has modification Anomalies. They should be minimized or removed

Relational Database Design

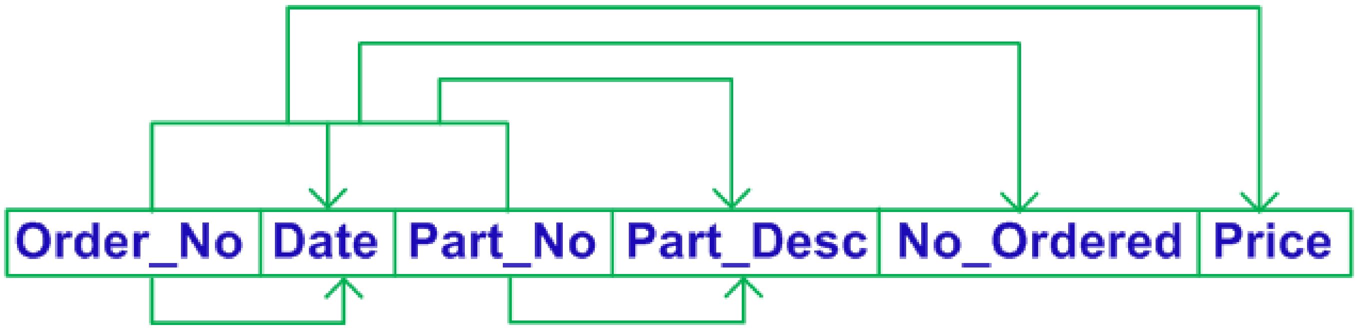
Minimization of Modification Anomalies

- A relation with modification anomalies is further normalized to higher normal form (1NF→2NF). Usually the normalization to the next higher normal form resolves the issue. If not then it is normalized to next higher normal form. Order must be normalize to 2NF

Relational Database Design

Normalization to 2NF

- **2NF:** ***A relation schema is in 2NF if it is in 1NF and every non-key attribute is dependent on the key - the whole key for composite keys***
- **Dependency diagram of Order**



Relational Database Design

Minimization of Modification Anomalies

- **Normalization procedure**
  - Identify the set of attributes that makes up the PK: {Order_no, Part_no}.
  - Create all subsets of the above set: {Order_no}, {Part_no} and {Order_no and Part_no}
  - Designate each of these subsets as the PK of a relation that contains those attributes, which are dependent on these PKs
  - Use Π to split the parent relation using these designated PKs.

Relational Database Design

Minimization of Modification Anomalies

- Order (1NF) to three relations Order, Part, and Order_Line (any more suggestions????)

Order

Order_No	Date
12489	90287
12491	90287
12494	90487
12495	90487
12498	90587
12500	90587

Part

Part_No	Part_Desc
AX12	Iron
AZ52	Skates
BA74	Baseball
BH22	Toaster
BT04	Stove
BZ66	Washer
CA14	Skillet
CB03	Bike
CX11	Mixer

Order_Line

Part_No	No-Ordered	Order_No	Price
AX12	11	12489	14.95
BZ66	1	12491	402.99
CB03	1	12491	311.95
CX11	4	12494	175.00
AZ52	2	12494	57.95
BA74	2	12498	22.95
BT04	4	12498	401.99

Order: Order_No → Date
Part: Part_No → Part-Desc
Order_Line: Part_No → all other attributes
All these relations are in 2NF.

Minimization of Modification Anomalies

- **Anomalies in Order, Part, and Order_Line**
  - Change: If BT04 is changed to something else then it requires only one change in Part relation
  - Add a new part and its description: If a new tuple is added in Part then there is no need to have an order exist for that part
  - Delete order 12489: This delete does not cause AX12 to be deleted from Part, thus we do not loose the description of AX12
  - Information loss: none.

Q. Does this imply that relations in 2NF do not have modification anomalies?

A. No. Relations in 2NF may suffer with all modification anomalies.

Relational Database Design

Modification Anomalies in 2NF relation

Example

Customer

Cust_no	Name	Address	Slsrep_no	Slsrep_name
124	Sally A	4747 Troost	2	Tom J
256	Ann S	215 Oak	6	Bill S
311	Don C	48 College	12	Sam B
315	Tom D	914 Cherry	6	Bill S
405	Al W	519 Watson	12	Sam B
412	Sally A	16 Elm	3	Mary J
522	Mary N	108 Pine	12	Sam B
567	Joe B	808 Ridge	6	Bill S
587	Judy R	512 Pine	6	Bill S
622	Dan M	419 Chip	3	Mary J

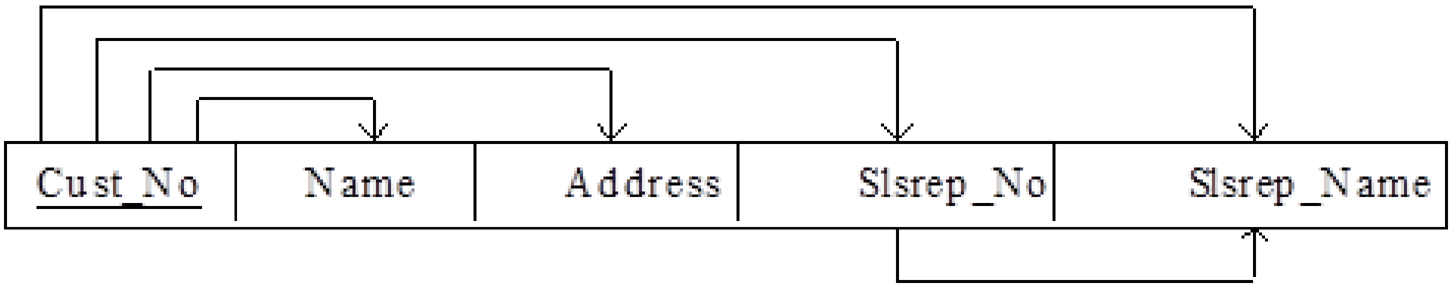
Relational Database Design

Modification Anomalies in 2NF relation

Dependency diagram of Customer

Customer

Cust_no	Name	Address	Slsrep_no	Slsrep_name
124	Sally A	4747 Troost	2	Tom J
256	Ann S	215 Oak	6	Bill S
311	Don C	48 College	12	Sam B
315	Tom D	914 Cherry	6	Bill S
405	Al W	519 Watson	12	Sam B
412	Sally A	16 Elm	3	Mary J
522	Mary N	108 Pine	12	Sam B
567	Joe B	808 Ridge	6	Bill S
587	Judy R	512 Pine	6	Bill S
622	Dan M	419 Chip	3	Mary J



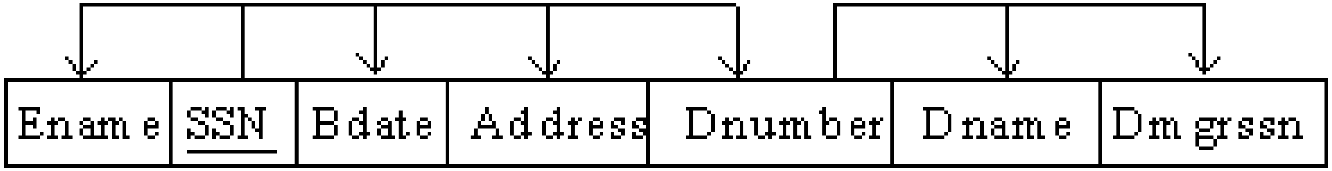
Relational Database Design

- **Modification Anomalies in 2NF relation**
  - Customer is in 2NF. It suffers with all the anomalies
  - Update: A change to Slsrep_name requires multiple changes
  - Inconsistent data: There is nothing in the design that would prohibit a Slsrep_name from having two different names
  - Additions: Need a customer to add Slsrep_no 47
  - Deletions: Delete all the customers of a sales rep then we lose the name of the Sales rep also
    - Reason for these anomalies: Slsrep_no, which is not a PK, determines Slsrep_name. As a result Slsrep_no can appear many times in the relation
    - Remedy: Normalize Customer relation by transforming it into 3NF relations

Relational Database Design

Normalization to 3NF

- *3NF: A relation scheme R is in 3NF if it is in 2NF and no non-prime attribute of R is transitively dependent on the primary key*
- A transitive dependency exists among 3 or more attributes
- Example of transitive dependency



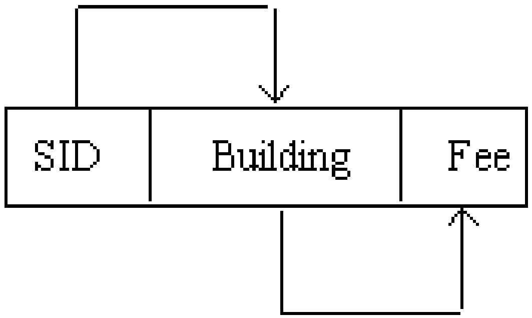
- SSN → Dnumber
- Dnumber → Dname and Dnumber → Dmgrssn
- Therefore SSN → Dname and transitively SSN → Dmgrssn

Relational Database Design

Normalization to 3NF

- Consider the following relation. It is in 2NF but not in 3NF because it has transitive dependency. Housing has all modification anomalies. To minimize them we normalize it to 3NF

SID	Building	Fee
100	Randolph	1200
150	Ingersol	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200



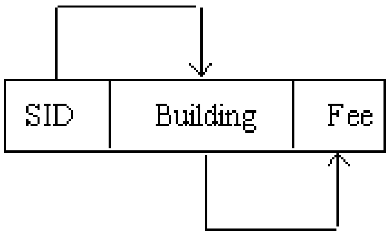
- SID → Building. SID is the key.
- Building → Fee and transitively SID → Fee

Relational Database Design

Normalization to 3NF

- 2NF → 3NF

SID	Building	Fee
100	Randolph	1200
150	Ingersol	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200



- SID → Building. SID is the key.
- Building → Fee and transitively SID → Fee

Housing

SID	Building
100	Randolph
150	Ingersol
200	Randolph
250	Pitkin
300	Randolph

Fee

Building	Fee
Randolph	1200
Ingersol	1100
Pitkin	1100

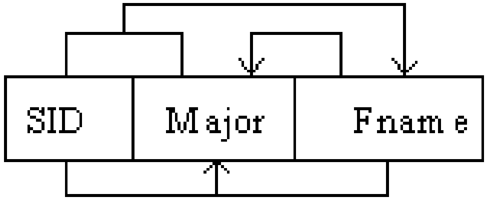
Relational Database Design

Normalization to BCNF

- We now take a 3NF relation and check out its anomalies

Advisor

SID	Major	Fname
100	Math	Cauchy
150	Psychology	Jung
200	Math	Riemann
250	Math	Cauchy
300	Psychology	Perls
300	Math	Riemann



- A student can have one or more majors
- A major can have several faculty as advisors.
- A faculty member advises in only one major area
- *SID* cannot be a key since a student can have many majors and therefore many advisors
- A a student cannot have many advisors in the same area
- This situation arises when we have more than one candidate keys

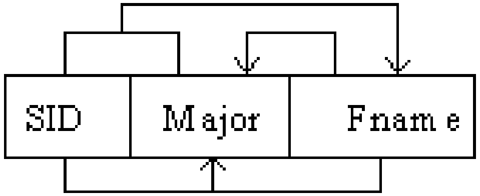
Relational Database Design

Normalization to BCNF

- We now take a 3NF relation and check out its anomalies

Advisor

SID	Major	Fname
100	Math	Cauchy
150	Psychology	Jung
200	Math	Riemann
250	Math	Cauchy
300	Psychology	Perls
300	Math	Riemann



- Keys are: {SID, Major} → Fname and {SID, Fname} → Major
- One of these can be selected as a primary key.
- Determinant: Fname → Major

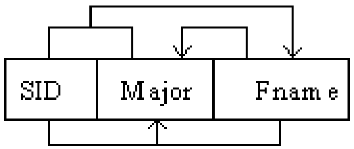
Relational Database Design

Normalization to BCNF

- This relation does not have transitive dependency. Advisor is in 3NF since there is no transitive dependency but it has modification anomalies.

Advisor

SID	Major	Fname
100	Math	Cauchy
150	Psychology	Jung
200	Math	Riemann
250	Math	Cauchy
300	Psychology	Perls
300	Math	Riemann



- Deletion: Cannot delete SID 300, we lose the information that Perls advises in Psychology
- Addition: Cannot add Keynes advises in Economics if there is no student
- Update: Multiple changes are required if Cauchy advises in Physics
- Inconsistency: Any change in Cauchy-Math will introduce inconsistency
- Solution: Normalize advisor to Boyce/Codd Normal form

Relational Database Design

Normalization to BCNF

- **BCNF (Boyce-Codd Normal Form)**
- ***A relation is in BCNF if every determinant is a candidate key***
- ***3NF → BCNF***

Advisor

SID	Fname
100	Cauchy
150	Jung
200	Riemann
250	Cauchy
300	Perls
300	Riemann

Major

Major	Fname
Math	Cauchy
Psychology	Jung
Math	Riemann
~~Math~~	~~Cauchy~~
Psychology	Perls
~~Math~~	~~Riemann~~

Normalization to 4NF

- Relations in BCNF are not entirely free from anomalies
- Consider Student relation

Student

SID	Major	Activity
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

Semantics: A student can enroll in more than one major and in more than one activity.

PK: All three attributes.

What is the relationship between Activity and Major?

It is not functional dependency, because students have several majors. There is some sort of relationship that can be illustrated by an example.

Relational Database Design

Anomalies

Example

Suppose: Student 100 wants to enroll in Skiing.
Add: tuple 100, Music, Skiing.
Resulting relation

SID	Major	Activity
100	Music	Skiing
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

Student

SID	Major	Activity
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

Semantics: It implies that Student 100 Skis as a Music major but he/she does not know to ski as an Accounting major. This does not make sense.

Relational Database Design

- **Anomalies**
  - **Solution**
  - **Add tuple: *100, Accounting, Skiing*. The resulting relation is consistent. The relationship between *SID* and *Major* is a *Multivalued dependency*. *SID* determines not a single value but several values. Thus (*SID 100*) determines *majors* (Music, Accounting) and *activities* (Skiing, Swimming, Tennis). The relation is in BCNF since all attributes make the primary key.**
  - **But it has anomalies**

Relational Database Design

Normalization to 4NF

- **4NF: A relation is in 4NF if it is in BCNF and it has no multivalue anomalies**
- **Solution: Split the relation**

S_Major

SID	Major
100	Music
100	Accounting
150	Math

S_Activity

SID	Activity
100	Sking
100	Swimming
100	Tennis
150	Jogging

Multivalued dependency always occur in pairs. For the Student relation SID →→Major because Major depends only on the value of SID and not on the value of Activity. Similarly, SID →→ Activity. Activity does not dependent on Major and this creates problem in the sense that whenever we add a new Major, we must add a tuple for every value of Activity.

Fourth normal form eliminates independent many-to-one relationships between columns.

To be in Fourth Normal Form,

a relation must first be in Boyce-Codd Normal Form.

a given relation may not contain more than one multi-valued attribute.

Example (Not in 4NF)

Scheme → {MovieName, ScreeningCity, Genre)

Primary Key: {MovieName, ScreeningCity, Genre)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Example 2 (Not in 4NF)

Scheme → {Manager, Child, Employee}

1. Primary Key → {Manager, Child, Employee}
2. Each manager can have more than one child
3. Each manager can supervise more than one employee
4. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	Seth	Adam

Example 3 (Not in 4NF)

Scheme → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

Example 1 (Convert to 4NF)

Old Scheme → {MovieName, ScreeningCity, Genre}

New Scheme → {MovieName, ScreeningCity}

New Scheme → {MovieName, Genre}

Movie	Genre
Hard Code	Comedy
Bill Durham	Drama
The Code Warrier	Horror

Movie	ScreeningCity
Hard Code	Los Angles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrier	New York

Example 2 (Convert to 4NF)

Old Scheme → {Manager, Child, Employee}

New Scheme → {Manager, Child}

New Scheme → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob
Mary	Seth

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

Example 3 (Convert to 4NF)

Old Scheme → {Employee, Skill, ForeignLanguage}

New Scheme → {Employee, Skill}

New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

First normal form -1NF

- 1NF : if all attribute values are atomic: no repeating group.

- The following table is not in 1NF

DPT_NO	MG_NO	EMP_NO	EMP_NM
D101	12345	20000 20001 20002	Carl Sagan Mag James Larry Bird
D102	13456	30000 30001	Jim Carter Paul Simon

Table in 1NF

DPT_NO	**MG_NO**	**EMP_NO**	**EMP_NM**
D101	12345	20000	Carl Sagan
D101	12345	20001	Mag James
D101	12345	20002	Larry Bird
D102	13456	30000	Jim Carter
D102	13456	30001	Paul Simon

- all attribute values are atomic because there are no repeating group and no composite attributes.

2) Second Normal Form

- Second normal form (2NF) further addresses the concept of removing duplicative data:
  - A relation R is in 2NF if
    - (a) R is 1NF , and
    - (b) all non-prime attributes are fully dependent on the primary key. Which is creating relationships between these new tables and their predecessors through the use of foreign keys.
  - There is no partial dependency in 2NF.

No dependencies on non-key attributes

Inventory			
<u>Description</u>	<u>Supplier</u>	Cost	Supplier Address

There are two non-key fields. So, here are the questions:

•If I know just Description, can I find out Cost? No, because we have more than one supplier for the same product.

•If I know just Supplier, and I find out Cost? No, because I need to know what the Item is as well.

Therefore, Cost is fully, functionally dependent upon the ENTIRE PK (Description-Supplier) (candidate key as well) for its existence.

Inventory		
<u>Description</u>	<u>Supplier</u>	Cost

CONTINUED…

Inventory			
<u>Description</u>	<u>Supplier</u>	Cost	Supplier Address

- **If I know just Description, can I find out Supplier Address? No, because we have more than one supplier for the same product.**

- **If I know just Supplier, and I find out Supplier Address? Yes. The Address does not depend upon the description of the item.**

Therefore, Supplier Address is NOT functionally dependent upon the ENTIRE PK (Description-Supplier) for its existence.

Supplier	
<u>Name</u>	Supplier Address

So putting things together

Inventory			
<u>Description</u>	<u>Supplier</u>	Cost	Supplier Address

↓

Supplier	
<u>Name</u>	Supplier Address

Inventory		
<u>Description</u>	<u>Supplier</u>	Cost

3NF Remove columns that are not dependent upon the primary key.

So for every nontrivial functional dependency X --> A,

(1) X is a superkey, or

(2) A is a prime (key) attribute.

Example of 3NF

Books

<u>Name</u>	Author's Name	Author's Nom-de Plume	# of Pages

- **If I know # of Pages, can I find out Author's Name? No. Can I find out Author's Non-de Plume? No.**
- **If I know Author's Name, can I find out # of Pages? No. Can I find out Author's Non-de Plume? YES.**

Therefore, Author's Nom-de Plume is functionally dependent upon Author's Name, not the PK for its existence. It has to go.

Books		
<u>Name</u>	Author's Name	# of Pages

Author	
<u>Name</u>	Non-de Plume

Example with first three forms

Suppose we have this Invoice Table

Invoice Table **Violate's Normalization Form 1**

	Customer Information											
Invoice#	**Cust#**	**Name**	**Addr**	**Quant1**	**Part1**	**Amt1**	**Quant2**	**Part2**	**Amt2**	**Quant3**	**Part3**	**Amt3**
1001	43	Jones	121 1st	200	Screw	2.00	300	Nut	2.25	100	Washr	0.75
1002	55	Smith	222 2nd	1	Motor	52.00	5	Brace	44.44			
1003	43	Jones	121 1st	10	Saw	121.00						

First Normal Form: No repeating groups.

•The above table violates 1NF because it has columns for the first, second, and third line item.

•Solution: you make a separate line item table, with it's own key, in this case the combination of invoice number and line number

Table now in 1NF

Complies with Normalization Form 1, Violate's Normalization Form 2

Invoice table

Invoice#
1001
1002
1003

Line item table

		Customer Information					
Invoice#	LIne#	Cust#	Name	Address	Quant1	Part1	Amt1
1001	1	43	Jones	121 1st	200	Screw	2.00
1001	2	43	Jones	121 1st	300	Nut	2.25
1001	3	43	Jones	121 1st	100	Washr	0.75
1002	1	55	Smith	222 2nd	1	Motor	52.00
1002	2	55	Smith	222 2nd	10	Saw	121.00
1003	1	43	Jones	121 1st	5	Brace	44.44

Second Normal Form:
Each column must depend on the *entire* primary key.

Complies with Normalization Form 2, Violate's Normalization Form 3

Invoice table

Invoice#	Customer Information		
	Cust#	Name	Address
1001	43	Jones	121 1st
1002	55	Smith	222 2nd
1003	43	Jones	121 1st

Line item table

Invoice#	LIne#	Quant1	Part1	Amt1
1001	1	200	Screw	2.00
1001	2	300	Nut	2.25
1001	3	100	Washr	0.75
1002	1	1	Motor	52.00
1002	2	10	Saw	121.00
1003	1	5	Brace	44.44

Third Normal Form:

Each column must depend on *directly* on the primary key.

Complies with Normalization Form 3

Invoice table

Invoice#	Cust#
1001	43
1002	55
1003	43

Line item table

Invoice#	Line#	Quant1	Part1	Amt1
1001	1	200	Screw	2.00
1001	2	300	Nut	2.25
1001	3	100	Washr	0.75
1002	1	1	Motor	52.00
1002	2	10	Saw	121.00
1003	1	5	Brace	44.44

Customer table

Cust#	Name	Address
43	Jones	121 1st
55	Smith	222 2nd

Examples

1NF A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only

2NF A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key

Example: 1NF but not 2NF

FIRST (supplier_no, status, city, part_no, quantity)

Functional Dependencies:

(supplier_no, part_no) → quantity

(supplier_no) → status

(supplier_no) → city

city → status (Supplier's status is determined by location)

Comments:

Non-key attributes are not mutually independent (city → status).

Non-key attributes are not fully functionally dependent on the primary key (i.e., status and city are dependent on just part of the key, namely supplier_no).

Anomalies:

INSERT: We cannot enter the fact that a given supplier is located in a given city until that supplier supplies at least one part (otherwise, we would have to enter a null value for a column participating in the primary key C a violation of the definition of a relation).

DELETE: If we delete the last (only) row for a given supplier, we lose the information that the supplier is located in a particular city.

UPDATE: The city value appears many times for the same supplier. This can lead to inconsistency or the need to change many values of city if a supplier moves.

Decomposition (into 2NF):

SECOND (supplier_no, status, city)

SUPPLIER_PART (supplier_no, part_no, quantity)

3NF A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. An attribute C is transitively dependent on attribute A if there exists an attribute B such that: A → B and B → C. Note that 3NF is concerned with transitive dependencies which do not involve candidate keys. A 3NF relation with more than one candidate key will clearly have transitive dependencies of the form: primary_key → other_candidate_key → any_non-key_column

Example (2NF but not 3NF):

SECOND (supplier_no, status, city)

Functional Dependencies:

supplier_no -> status

supplier_no -> city

city -> status

Comments:

Lacks mutual independence among non-key attributes.

Mutual dependence is reflected in the transitive dependencies: supplier_no ® city, city ® status.

Anomalies:

INSERT: We cannot record that a particular city has a particular status until we have a supplier in that city.

DELETE: If we delete a supplier which happens to be the last row for a given city value, we lose the fact that the city has the given status.

UPDATE: The status for a given city occurs many times, therefore leading to multiple updates and possible loss of consistency.

Decomposition (into 3NF):

SUPPLIER_CITY (supplier_no, city)

CITY_STATUS (city, status)

BCNF A relation R is in Boyce-Codd normal form (BCNF) if and only if every determinant is a candidate key

Example (3NF but not BCNF):

SUPPLIER_PART (supplier_no, supplier_name, part_no, quantity)

Functional Dependencies:

We assume that supplier_name's are always unique to each supplier. Thus we have two candidate keys:

(supplier_no, part_no) and (supplier_name, part_no)

Thus we have the following dependencies:

(supplier_no, part_no) $\rightarrow$ quantity

(supplier_no, part_no) $\rightarrow$ supplier_name

(supplier_name, part_no) $\rightarrow$ quantity

(supplier_name, part_no) $\rightarrow$ supplier_no

supplier_name $\rightarrow$ supplier_no

supplier_no $\rightarrow$ supplier_name

Comments:

Although supplier_name $\rightarrow$ supplier_no (and vice versa), supplier_no is not a non-key column — it is part of the primary key! Hence this relation technically satisfies the definition(s) of 3NF (and likewise 2NF, again because supplier_no is not a non-key column).

Anomalies:

INSERT: We cannot record the name of a supplier until that supplier supplies at least one part.

DELETE: If a supplier temporarily stops supplying and we delete the last row for that supplier, we lose the supplier's name.

UPDATE: If a supplier changes name, that change will have to be made to multiple rows (wasting resources and risking loss of consistency).

Decomposition (into BCNF):

SUPPLIER_ID (supplier_no, supplier_name)

SUPPLIER_PARTS (supplier_no, part_no, quantity)

Another BCNF Example

BCNF Example

- SCT (Student, Course, Teacher
  - Each student takes a given course from one teacher only
  - Each teacher teaches one course only

Student	Course	Teacher	
S1	C1	A	
S1	C2	B	
S2	C1	A	
S2	C2	C	<----- delete?

BCNF example

- (S,C) Primary key, SC -> T
- SCT relation is 3NF
- But also T -> C
- Decomposition: <u>S</u><u>T</u> and <u>T</u>C
- has a lossless join, but the functional dependency SC -> T is not preserved in this case

Sources:

- www.cs.sjsu.edu
- Dr. Kumar, Professor, CSEE, UMKC
- Elmasri/Navathe
- www.psu.edu