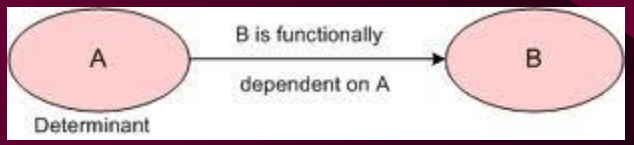
Normalization

Functional Dependencies

- Functional Dependency and Normalization were used as the primary relational database design process before E-R diagrams came about
- Functional dependency is a relationship between or among attributes
- Given two attributes X and Y, Y is functionally dependent upon X if the value of X determines the value of Y; X is called a determinant
- For example (when students have one major), major is functionally dependent upon StudentId, since knowing StudentId uniquely defines Major

Functional Dependencies (con't)

- Written as:
  - StudentId --> Major
- If StudentId determines Major, then a particular value of StudentId will be paired (with only one value of Major; the inverse is generally not true (a Major will be paired with many StudentId's)



Groups of Attributes

- Functional dependency can involve multiple attributes
- In the table: (SID, Class, Grade)
- (SID, Class) --> Grade
- The combination of SID and Class uniquely determine the Grade

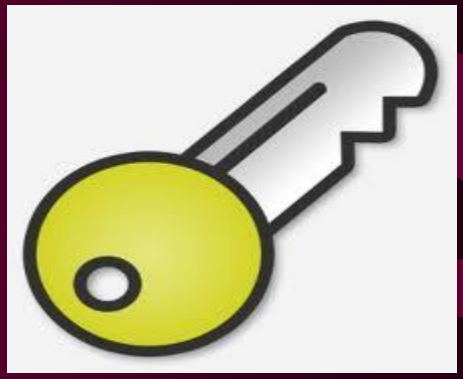
Groups of Attributes (con't)

- If X --> (Y , Z) then it is true that
  - X --> Y
  - X --> Z
- If (X, Y) --> Z than it is not true that
  - X --> Y
  - X --> Z
  - Y --> Z

Key (mathematically)

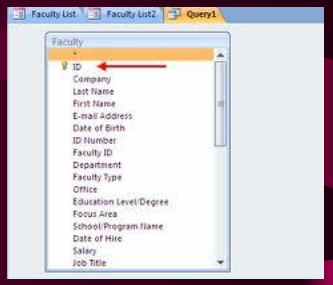
- A group of one or more attributes that uniquely identifies a row
- In the relation:
  - STUDENT (StudentId, Major, Name, Address,...)
  - StudentId is a key
- In the relation:
  - GRADE (StudentId, Course, Grade)
  - (StudentId, Course) is the key [assuming a student only completes the same course once, or only the last taking of the course is retained in the database]

Unique Key



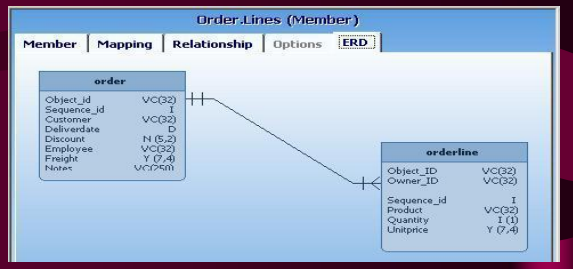
- The notion of key described on the previous slide is the mathematical term
- In practice this is usually called a *unique key*
- There may be other keys that are not unique (and these could be used as indexes)
- There may be more than one unique key (*candidate keys*), but only one is the main table key (or *primary key*)

Primary Key



- In an employee table, the primary key would probably be employeeNumber; another unique key would be social security number (a unique index would probably be set up on social security number to avoid duplicates)
- A table can only have one “primary key”
- A non-unique key may be “name”
- In some terminology the word primary has to do with the primary physical organization of the records
- Usually the table’s primary organization (hashed or b-tree data structure) is consistent with the main unique key

Foreign Key



- A Foreign Key in one table is a value that is the primary (main) key of another table
- A table can have none, one, or many foreign keys
- The value of Isbn in the COPY table is a foreign key, since the value of Isbn is the primary key in the TITLE table
- The value of StudentId in the COPY table is a foreign key, since the value of StudentId is the primary key of the STUDENT table

COPY Table [ISBN/Copy # is PK, Student & ISBN are FK's]

Title	CopyNumber	Student	Due Date
Isbn-T1	1	null	null
Isbn-T1	2	*SId-S1*	4/1/96
Isbn-T1	3	*SId-S2*	5/2/96
Isbn-T2	1	null	null
Isbn-T2	2	null	null
Isbn-T3	1	*SId-S1*	3/27/96
Isbn-T3	2	null	null

Primary Keys and Determinants

- Primary Key and Determinant are not necessarily the same in a table
- A determinant of a functional dependency may or may not be unique to the table
- If A--> B where both A and B are in the same table, we still do not know whether A is unique in that table

Primary Keys and Determinants (con't)

- Consider the table
  - ACTIVITY (StudentId, Activity, Fee)
- The primary key is the group:
  - (StudentId, Activity)
  - since students can have multiple activities, and an activity will typically have many students
- However, Activity is the determinant of Fee (assuming students all pay the same for the same activity)

Normalization

- There are many ways to model a real world situation with a relational model (involving multiple tables)
- Some models are better in others in that they minimize redundant data or are not subject to inconsistencies upon maintenance ("anomalies") (insertion, modification or deletion of information)
- *Normalization* is the process of restructuring a set of tables into a more consistent model

SERVICE Database

- Consider the relation (table):
  - SERVICE (Customer, Service, Charge)
- Customer is the unique name of someone to whom service is provided; a customer may have more than one service and a service may be provided to more than one customer
- Service is the unique name for a type of service, that is regularly scheduled
- Charge is what the customer pays for the service, every customer is charged the same for the same service

SERVICE Table

Customer	Service	Charge
Jones	Cut Grass	40
Smith	Rake Leaves	50
Doe	Shovel Driveway	60
Williams	Cut Grass	40

- What s wrong with this table ???

SERVICE Table

Customer	Service	Charge
Jones	Cut Grass	40
Smith	Rake Leaves	50
Doe	Shovel Driveway	60
Williams	Cut Grass	40

- Redundancy
  - Duplicate service name
  - Duplicate service rates
- Difficult to modify some info
  - Rate for cutting grass
- What else ???

Anomalies

- Insertion - if we want to store the fact that we are now going to offer a service to wash cars for $25, we cannot include it in the database until someone wants it
- Deletion - If Doe no longer wants us to shovel his driveway, when we delete that row, we will lose how much we charge for shoveling (since he is the last one using that service)

Single Concept per Relation

- *Tables should contain information about a single concept or theme;* each row should be an instance of one object or notion
- In the previous table we tried to express two concepts:
  - which customers want which service
  - how much each service costs

- How can we “fix” this table ???

SERVICE Table

Customer	Service	Charge
Jones	Cut Grass	40
Smith	Rake Leaves	50
Doe	Shovel Driveway	60
Williams	Cut Grass	40

Don’t look ahead !

Represent the Model with Two (or more) Tables

- If there is no customer attributes in database, then we could use two tables:
  - CUSTOMER (Name, *Service*)
  - SERVICE (Service, Charge)
- If customer attributes are included we need three tables:
  - CUSTOMER (Name, address)
  - SERVICE (Service, Charge)
  - CUSTOMER-SERVICE(*Name, Service*)

Anomalies

- For this data arrangement, what are the primary and foreign keys ?
  - CUSTOMER (Name, *Service*)
  - SERVICE (Service, Charge)
- For this data arrangement, what are the primary and foreign keys ?
  - CUSTOMER (Name, address)
  - SERVICE (Service, Charge)
  - CUSTOMER-SERVICE(*Name, Service*)

SERVICE Table

Service	Charge
Cut grass	40
Rake Leaves	50
Shovel	60

CUSTOMER Table

Customer	Service
Jones	Cut grass
Smith	Rake Leaves
Doe	Shovel Snow
Williams	Cut Grass

Anomalies (con’t)

- We now no longer have anomalies
- Here we now have 14 cells used instead of 12, but in general there will be a savings in storage as the number of customers would probably be much greater than the number of services
- But we have introduced a new potential problem

Referential Integrity

- If we decide we are no longer going to shovel driveways, we can not just delete the item from the service table; we must do something about the customers using that service
- If we want to sign up a new customer for a service not in the service table, we must either:
  - sign him up with a null entry in the service column, and come back and fix it later
  - determine a charge for it and add it to the service table first; constrain the customer insert
- These are "business rules", and implemented in different database schemas different ways (ie triggers)

Normal Forms

- Tables can be classified by the types of anomalies to which they are vulnerable, or what *normal form* they are in
- The normal forms are hierarchical:
  - Domain/Key
  - Fifth
  - Fourth
  - Boyce-Codd
  - Third
  - Second
  - First

Relation

- The relation in mathematics is similar to the two dimension table we think of in Excel or Access
- However the relation is not just any table !

First Normal Form

- Any table that meets the definition of a relation is in first normal form:
  - cells are singled value, no repeating groups or arrays
  - all entries in a column must be of the same kind (match domain of column)
  - no two rows can be identical
  - orders of rows and columns is not significant
- The table SERVICE(Customer, Service, Charge) is in first normal form

Single Value for Cells

- Character
- Text String - fixed length, or maximum length
- Text String (unspecified length) - Memo
- Number (integer, decimal, floating, money)
- Date/time
- BLOB (pointer to document, image, video, sound, etc.)

Puppy Example

- We want to store info about our puppies (and the tricks they know):
  - Single valued: Puppy number, puppy name, kennel code, kennel name, kennel location
  - Multi-valued: trick ID 1…N, trick name 1 …N, trick where learned 1…N, skill level 1…N

For 1st Normal Form – Two Tables [no repeating data in a cell]

- Puppy Table
  - Puppy Number
  - Puppy Name
  - Kennel Code
  - Kennel Name
  - Kennel Location
- Trick Table
  - Puppy Number
  - Trick ID
  - Trick Name
  - Trick Where Learned
  - Skill Level

What is the primary key in the Trick table ?

Second Normal Form (2NF)

- In the table SERVICE(Customer, Service, Charge), the charge is partially dependent on the primary key (Customer, Service); since it is fully dependent on only part of the key
- A relation is in the second normal form if all of its non-key attributes are dependent on ***all*** of the key
- Any table that has a primary key composed of one attribute is automatically in second normal form, but to be in second normal form a table does not have to have a primary key of one attribute
- The restructured model with two tables
  - (CUSTOMER & SERVICE) is in second normal form

What about our Trick Table ? How can we get to 2nd Normal form ?

A relation is in the second normal form if all of its non-key attributes are dependent on ***all*** of the key

- Trick Table
  - Puppy Number
  - Trick ID
  - Trick Name
  - Trick Where Learned
  - Skill Level

Don’t look ahead !

Trick Name was only partially dependent on the key

- Trick Table
  - Puppy Number
  - Trick ID
  - Trick Name
  - Trick Where Learned
  - Skill Level

- Trick Table
  - Trick ID
  - Trick Name
- Puppy Tricks
  - Puppy Number
  - Trick ID
  - Trick Where Learned
  - Skill Level

Transitive Dependency

- Consider the table ATTENDANCE (Person, Section, Seat, TicketPrice)
- Where each person is assigned to one seat, and each section has seats with all the same price
- The primary key is just one attribute: Person, so it is in second normal form
- However: Person --> Section --> TicketPrice
- This is called a *transitive dependency*

What problems are going to have maintaining this data ?

Person	Section	Seat	Price
Ed	C	1	10
Mary	B	2	15
Leroy	C	2	10
Thelma	A	2	20
Joe	B	1	15
Alice	A	1	20

Third Normal Form (3NF)

- A table is in third normal form if it is in second normal form and if it has no transitive dependencies
- The ATTENDANCE can be divided into two tables:
  - PEOPLE (Person, *Section*, Seat)
  - PRICE (Section, TicketPrice)
- Now the two tables are in 3NF
- What is the foreign key ?

What about our Puppy Table ?

How can we put our puppy table in 3rd normal form ?

- Puppy Table
  - Puppy Number
  - Puppy Name
  - Kennel Code
  - Kennel Name
  - Kennel Location

Don’t look ahead !

Eliminate Transitive Dependencies [puppy - > kennel -> name/loc]

- Puppy Table
  - Puppy Number
  - Puppy Name
  - Kennel Code
  - Kennel Name
  - Kennel Location

- Puppies
  - Puppy Number
  - Puppy Name
  - *Kennel Code*
- Kennels
  - Kennel Code
  - Kennel Name
  - Kennel Location

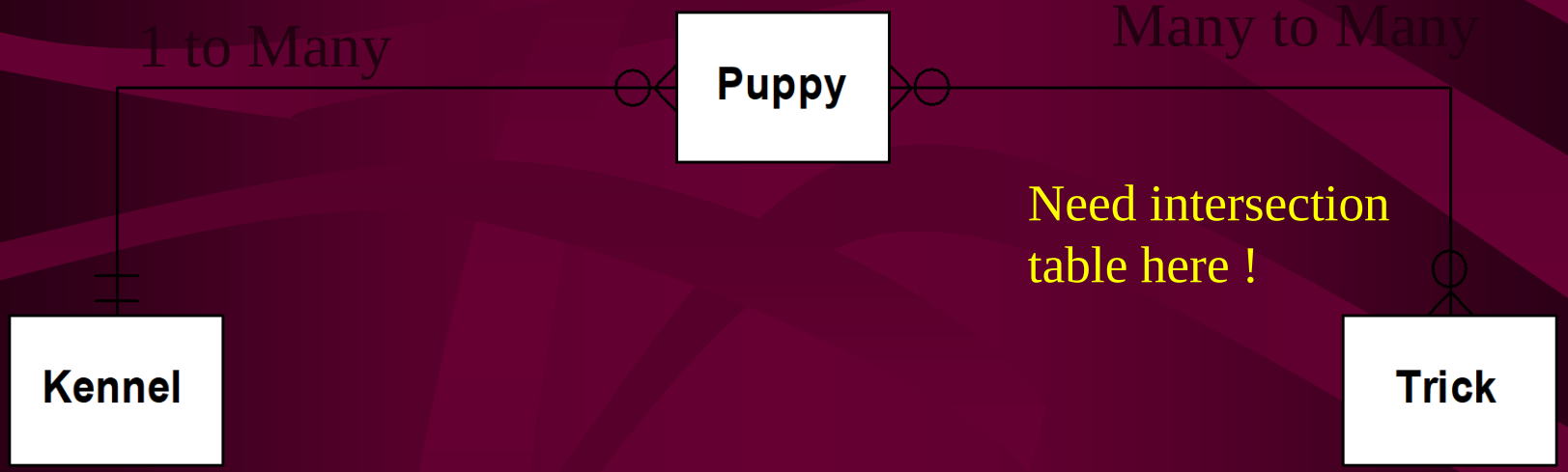
What is the FK ?

Class Exercise

- Draw the E-R diagram for the puppy model (puppies, kennels, tricks)

Don’t look ahead !

Puppy Model


1 to Many
Puppy
Many to Many
Need intersection table here !
Kennel
Trick

Multiple Primary Key Possibilities

- Even relations in third normal form can have anomalies
- Situation arises when there are multiple primary key possibilities
- Consider the table:
  - ASSIGNMENT (Customer, Car, SalesPerson)
  - A customer may be considering buying more than one type of car
  - A SalesPerson only sells one type of car, but there may be more than one salesperson for a type of car

ASSIGNMENT Table

[Brice & Adams sell Buicks, Jones is considering Buicks & Pontiacs, Dodd has 2 customers]

Customer	Car	SalesPerson
Jones	Buick	Adams
Jones	Pontiac	Dodd
Doe	Cadilac	Moore
Willams	Buick	Bryce
Smith	Pontiac	Dodd
Evans	Cadilac	Moore

What problems are going to have maintaining this data ?

Multiple Primary Key Possibilities (con't)

- Customer by itself is not a candidate key - why ?
- Salesperson by itself is not a candidate - why ?
- The candidate primary keys are:
  - (Customer, Car)
  - (Customer, Salesperson)
- But there is another functional dependency:
  - Salesperson --> Car
- If Jones is no longer a customer, we will lose the fact that Adams is a Buick salesperson

Boyce-Codd Normal Form (BCNF)

- A relation is in Boyce-Codd Normal form if every determinant is a candidate key
  - Salesperson --> Car ; but salesperson is not a candidate key
- Decompose ASSIGNMENT into two tables:
  - SALESPERSON (SalesPerson, Car)
  - CUSTOMER (Customer, *SalesPerson*)
  - What is the foreign key ?

Fourth Normal Form

- A relation is in Fourth Normal form if it is in BCNF and, if the file contains several multivalued dependencies, these are dependent upon each other

Fourth Normal Form (con't)

- A multivalued dependency holds between two attributes in a table if the second attribute can assume different values for a given value of the first
- For R(A,B,C), if A determines multiple values of B and A determines multiple values of C, and B and C are independent, then the table is not in FNF

Multivalued Dependency

- Consider the Table:
  - OUTFIT (Person, Shirt, Pants)
- Which describes the clothing outfits one can wear
- A person can wear many different shirts
- A person can wear many different pants
- The primary key for the outfitted person is all three columns

Multivalued Dependency (con't)

Person	Shirt	Pants
Joe	Blue	Yellow
Joe	Red	Green
Joe	Blue	Green
Joe	Red	Yellow
Mary	White	Black
Mary	White	Blue

Multivalued Dependency (con't)

- If one buys another shirt, you have to insert a row into the table for each pants you have
- If you delete a shirt, you have to delete many rows
- If Pants and Shirts are independent, then this table is not in 4NF
- Need to break into two tables:
  - PANTS (Person, Pants)
  - SHIRTS (Person, Shirts)

Multivalued Dependency (con't)

- If Pants and Shirts were in some way dependent (such as you do not wear red shirts with yellow pants), then you could not break up the original table
- Beware of a *join anomaly (when you join the two tables PANTS and SHIRTS) or join dependency*, since the OUTFITS table may have contained some of these constraints (such as this person cannot wear this pants with that shirt)
- Not covered in textbook, see Codd's original article

Puppy Example

- Puppy Tricks
  - Puppy Number
  - Trick ID
  - Trick Where Learned
  - Skill Level
  - Costume
- This is ok, if we mean “a puppy only does a certain trick while wearing a particular costume”

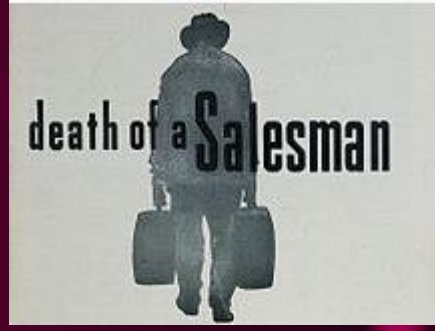
- Puppy Tricks
  - Puppy Number
  - Trick ID
  - Trick Where Learned
  - Skill Level
- Puppy Costumes
  - Puppy Number
  - Costume
- This formulation is necessary if we want to keep track of the costumes a puppy can wear

Other Normal Forms

- Fifth Normal Form - also deals with multivalue dependencies, particularly with higher order relationships
- Multirelation Normal forms - eliminating redundancy and anomalies between multiple but similar relations

Fifth Normal Form

- **Fifth normal form** (**5NF**), also known as **Project-join normal form** (**PJ/NF**) is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships
- A table is said to be in the 5NF if and only if every join dependency in it is implied by the candidate keys



Salesman Example
[3 concepts]

Lee J Cobb (Willy), seated, with Arthur Kennedy (Biff), left, and Cameron Mitchell (Happy) in the 1949 production of *Death of a Salesman*

Travelling Salesman	Brand	Product Type
Jack Schneider	Acme	Vacuum Cleaner
Jack Schneider	Acme	Breadbox
Willy Loman	Robusto	Pruning Shears
Willy Loman	Robusto	Vacuum Cleaner
Willy Loman	Robusto	Breadbox
Willy Loman	Robusto	Umbrella Stand
Louis Ferguson	Robusto	Vacuum Cleaner
Louis Ferguson	Robusto	Telescope
Louis Ferguson	Acme	Vacuum Cleaner
Louis Ferguson	Acme	Lava Lamp
Louis Ferguson	Nimbus	Tie Rack

Fifth Normal Form (con't)

- The table's predicate is: Products of the type designated by *Product Type*, made by the brand designated by *Brand*, are available from the travelling salesman designated by *Travelling Salesman*
- In the absence of any rules restricting the valid possible combinations of Travelling Salesman, Brand, and Product Type, the three-attribute table above is OK to model the situation correctly
- However, if a Salesman has certain Brands and certain Product Types in his repertoire -> if Brand B is in his repertoire, and Product Type P is in his repertoire, then (assuming Brand B makes Product Type P), the Salesman must offer only the products of Product Type P made by Brand B

Fifth Normal Form (con't)
Normalized Relation

Travelling Salesman	Product Type
Jack Schneider	Vacuum Cleaner
Jack Schneider	Breadbox
Willy Loman	Pruning Shears
Willy Loman	Vacuum Cleaner
Willy Loman	Breadbox
Willy Loman	Umbrella Stand
Louis Ferguson	Telescope
Louis Ferguson	Vacuum Cleaner
Louis Ferguson	Lava Lamp
Louis Ferguson	Tie Rack

Fifth Normal Form (con't) – Normalized Relations

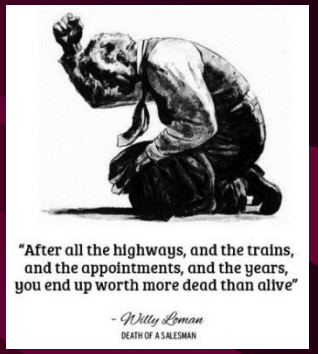
Travelling Salesman	Brand
Jack Schneider	Acme
Willy Loman	Robusto
Louis Ferguson	Robusto
Louis Ferguson	Acme
Louis Ferguson	Nimbus

For a salesman, there is no rock bottom to the life. He don't put a bolt to a nut, he don't tell you the law or give you medicine. He's a man way out there in the blue, riding on a smile and a shoeshine. And when they start not smiling back—that's an earthquake.

(Arthur Miller)

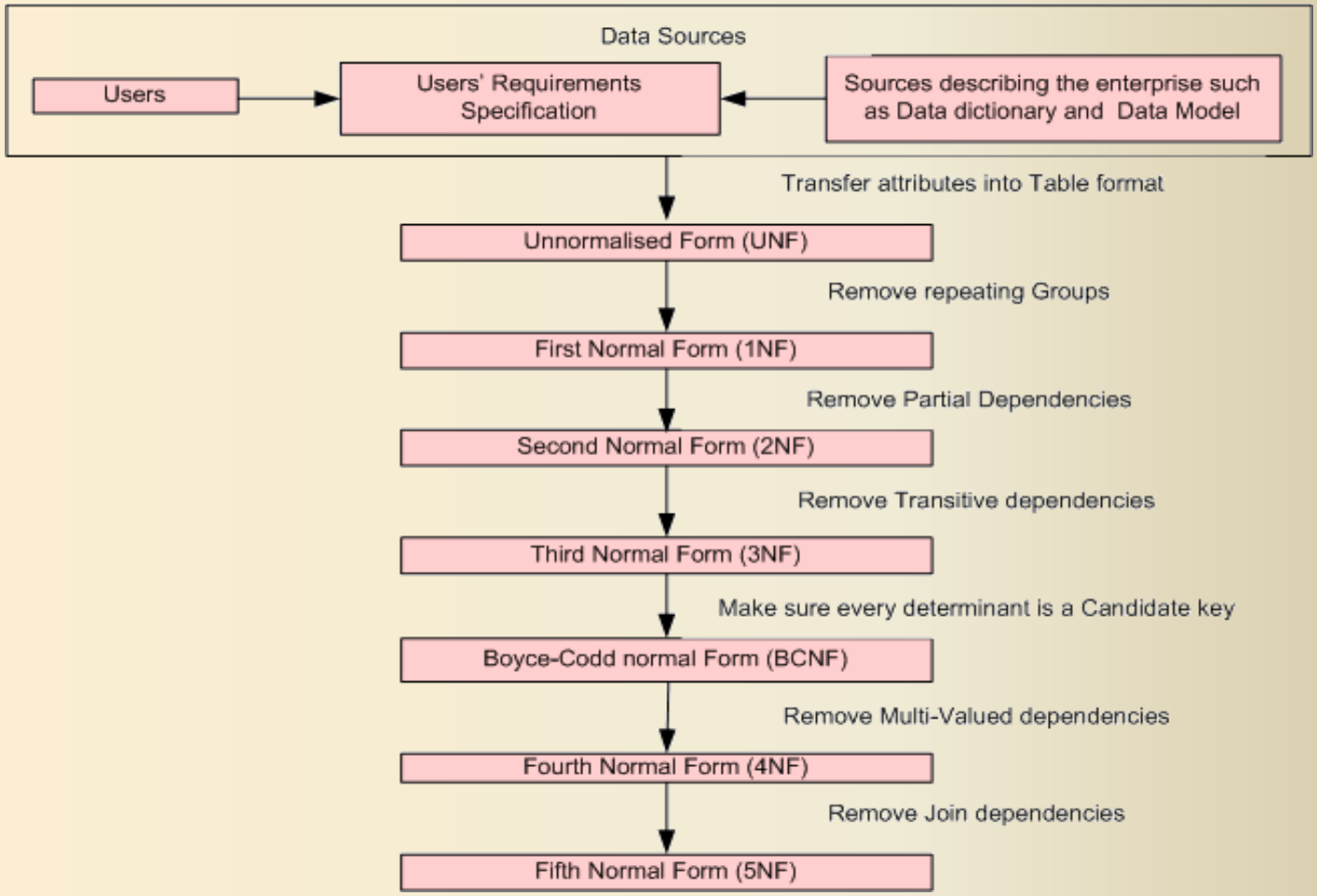
Fifth Normal Form (con't) – Normalized Relations

Brand	Product Type
Acme	Vacuum Cleaner
Acme	Breadbox
Acme	Lava Lamp
Robusto	Pruning Shears
Robusto	Vacuum Cleaner
Robusto	Breadbox
Robusto	Umbrella Stand
Robusto	Telescope
Nimbus	Tie Rack



Robusto makes 5 products, but Willi just sells 4 of these

Normal Forms



Normalization Example

First Normal Form (1NF):

Eliminate groups of repeating data.

NULL | Repeating Value

Partial Dependencies (PK) | Partial Dependencies (PK)

Company ID	Company Name	Member ID	Member Type	Member Name	Dues	License Number	Licence Type	License Active Date
101	BOZ Interiors, LLC	2001	Corporate	Cheryl Bozworth	500	IDB10017	Business	04/09/2004
		2002	Professional	Mike Hamley	300	IDP04235	Individual	10/11/1994
		2003	Associate	Dee LeBrau	150			
102	Shore Design, Inc.	2004	Corporate	Joyce Shore	500	IDB00423	Business	01/22/1987
		2005	Professional	Wanda Flueur	300	IDP00901	Individual	04/14/1989
103	Maslin Carpet Co.	2006	Industry Partner	Marty Highsmith	250			
104	Bath Magic, Ltd.	2007	Industry Partner	Sandi Goldberg	250			
105		2008	Professional	Cayman Kai	300	IDP00190	Individual	09/15/1986

PK — Partial Dependency

The amount charged for Member Dues is dependent upon the Member Type

Transitive Dependency

The Table above shows that the membership organization may have Companies that have one or many Members in the organization. It also shows columns that contain null data, such as Members with no Company Name, and Members that are not required to have a license (leaving null column rows in License Number, License Type, and License Active Date). The null data in the Company ID field makes it difficult to determine how many members belong to a certain Company, especially when the repeating information in the Company name filed is null as well.

Second Normal Form (2NF):

Remove Redundant Data

Separate key components to their own tables

Member Table
- ***Company ID***
- Company Name
- ***Member ID***
- Member Type
- Member Name
- Dues
- *License Number*
- License Type
- License Active Date

Company Table
- ***Company ID***
- Company Name

Member Table
- ***Member ID***
- Member Name
- Member Type
- Member Dues

Member Type is also a redundant value, but dues are dependent upon that value, so it will be dealt with in conversion to 3NF.

License Table
- ***License Number***
- License Type
- License Active Date

License Type is a redundant value and must be moved to a new License Type Table.

License Table
- ***License Number***
- *License Type ID*
- License Active Date

License Type Table
- ***License Type ID***
- License Type Label

Third Normal Form (3NF):

Remove transitive dependencies

The amount charged for Member Dues is dependent upon the Member Type, and needs to be placed in its own table

Company Table
- ***Company ID***
- Company Name

Member Table
- ***Member ID***
- Member Name
- Member Type
- Member Dues

Member Table
- ***Member ID***
- Member Name
- *Member Type ID*

Member Type Table
- ***Member Type ID***
- Member Type Label
- Member Type Dues

License Table
- ***License Number***
- *License Type ID*
- License Active Date

License Type Table
- ***License Type ID***
- License Type Label

- In all these normal forms, we have tried to separate tables containing multiple concepts!

Domain/Key Normal Form (DK/NF)

- For all the previous normal forms, no theory could guarantee that any of them would eliminate all anomalies
- However, a computer program can go thru a table (with sufficient data) and find normalization problems
- Fagin showed that a relation in DK/NF is free from all anomalies:
  - ACM Transactions on Database Systems, September, 1981

DK/NF (con't)

- A relation is in DK/NF if every constraint on the relation is a logical consequence of the definition of keys and domains
- Constraint - a rule governing static (non time dependent) values of the attributes
- Key - unique identifier
- Domain - the description of an attribute's allowed values (physical & semantic)

DK/NF (con't)

- ***In other words, a table is in DK/NF if enforcing key and domain restrictions causes all the constraints (business rules) to be met***
- No algorithm for converting a relation to DK/NF - experience and trail & error
- Define tables where constraints are logical consequences of keys and domains; constraints not so handled must be built into the application via procedures (code)

Attendance Example

- Consider the table ATTENDANCE (Person, Section, Seat, TicketPrice)
- Where each person is assigned to one seat, and each section has seats with all the same price
- The primary key is just one attribute: Person
- Constraint: Section --> TicketPrice
- This is not in DK/NF since this constraint cannot be enforced by the key (not in third normal form since it has a transitive dependency); ie, you can add a row that has a different ticket price from a row already in the table with the same section

Attendance Example (con't)

- The ATTENDANCE can be divided into two tables:
  - PEOPLE (Person, *Section*, Seat)
  - PRICE (Section, TicketPrice)
- Now the constraint can be enforced in the second table (you cannot have the same section with two or more prices)

Assignment Example

- Consider the table:
  - ASSIGNMENT (Customer, Car, SalesPerson)
  - not in BCNF (every determinant is not a candidate key)
  - Constraint: A SalesPerson only sells one type of car
- To enforce the constraint, we need to make the salesPerson a key in a table with the car he sells
  - SALESPERSON (SalesPerson, Car)
  - CUSTOMER (Customer, *SalesPerson*)

Optimization

- There are many optimization “tricks” and techniques
- They represent trade off’s in terms of:
  - speed for different access directions (lookup or reporting)
  - reducing difficulty of queries
  - storage space (data and indexes)
  - database and program maintenance
- Many of these beyond scope of this course

Optimization (con't)

- Denormalization
- Low Cardinality Optimization
  - One entities relates to two or three of another
- Controlled redundancy
  - Smaller record size for common data
- Query/SQL Optimization (later in course)
- Use of Subtypes
- Use of Indexes
- Table Contention
- Physical Media Contention
- Locking Contention (ie with auto-inc keys)

Denormalization

- Consider the table:
  - CUSTOMER (CID, CName, City, State, Zip)
  - This is not normalized since ZIP --> City,State
- The fully normalized form is:
  - CUSTOMER (CID, CNname, *Zip*)
  - ZIP (Zip, City, State)
  - less data duplication , easier maintenance, faster order filing (only ask for zip code)
  - however this causes more I/O work to output customer info, since two tables have to be read (joined)

Denormalization (con't)

- Another alternative:
  - CUSTOMER (CID, CName, City, State, *Zip*)
  - ZIP (Zip, City, State)
- Can still fill orders quicker by only asking for ZIP
- But extra since space used for city and state in each customer record
- Can use triggers to cascade changes in Zip to CUSTOMER and to automatically fill in CUSTOMER City and State fields from given Zip

Denormalization (con't)

- Consider the tables:
  - RX (RxNo, *PID*, *DID*, *NDC*, Date, ...
  - PATIENT (PID, FirstName, LastName, ...
  - DOCTOR (DID, FirstName, LastName, ...
  - DRUG (NDC, BrandName, ...
- However we need to frequently display and print patient's name with RX (such as on the label)
- How can we denormalize this for greater efficiency ?
- What problems may arise and how would we handle that ?

Don’t look ahead !

Denormalization (con't)

- RX (RxNo, *PID*, *DID*, *NDC*, Date, PName..
- Place index on PName

- But a person might change their name
- Trigger on changes to LastName and FirstName in PATIENT table to migrate changes to RX table

References

- E. F. Codd, “A Relational Model of Data for Large Shared Databanks,” Communications of the ACM, 1970
- Birth of the Relational Model
    - Intelligent Enterprise, October 1998, p 61, ff
- Fundamentals of Relational Data Organization
    - Byte, November 1981, p 48, ff
- Inside Relational Databases with Examples in Access by Mark Whitehorn and Bill Marklyn
- Beginning Relational Data Modeling, Second Edition by Sharon Allen and Evan Terry
- Relational Database Design and Implementation, Third Edition: Clearly Explained 3e (Morgan Kaufmann Series in Data Management Systems) by Jan L. Harrington

Homework

- Textbook Chapter 6
- Review Questions 1 thru 5
- Project Design Review
  - Requirements (info, queries/reports, …)
  - Entity Attributes
    - Major attributes
    - Unique identifiers
  - E-R Model