ER DIAGRAMS AND NORMAL FORMS

Some figures taken from the book "Concepts of Database Management"

Database Design

- **Database design: Why do we need it?**
  - Agree on structure of the database before deciding on a particular implementation
- **Consider issues such as:**
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs
- **Several formalisms exist**
  - We discuss one flavor of E/R diagrams

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

1. Requirements analysis

- What is going to be stored?
- How is it going to be used?
- What are we going to do with the data?
- Who should access the data?

Technical and non-technical people are involved

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

2. Conceptual Design

- A high-level description of the database
- Sufficiently precise that technical people can understand it
- But, not so precise that non-technical people can't participate

This is where E/R fits in.

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

3. More:

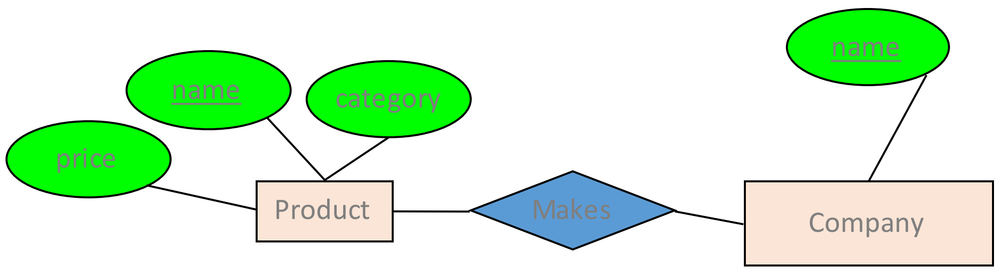
- Logical Database Design
- Physical Database Design
- Security Design

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

E/R Model & Diagrams used

This process is iterated **many** times



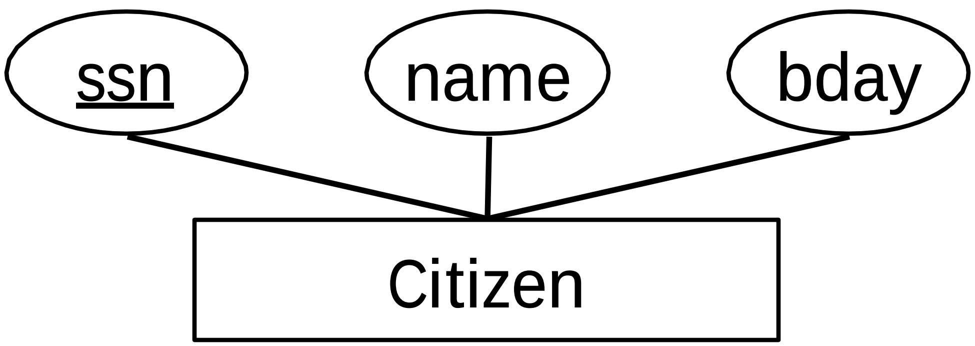
E/R is a *visual syntax* for DB design which is ***precise enough*** for technical points, but ***abstracted enough*** for non-technical people

ER Model Basics

- *Entity:* Distinguishable real-world object
  - Described by a set of *attributes,* Each attribute has a *domain*
- *Entity Set*: A collection of similar entities. E.g., all citizens.
  - All entities in an entity set have the same set of attributes.

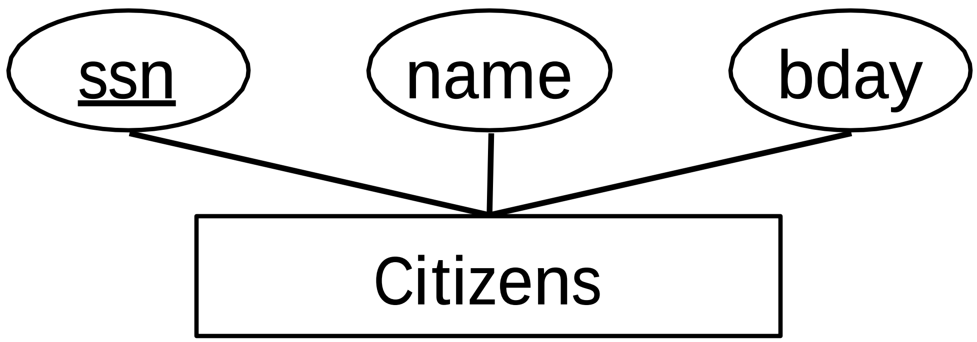
    Key : minimal set of attributes whose values uniquely identify an entity in an entity set

    - Primary key
    - Candidate key
- Pictorially ...



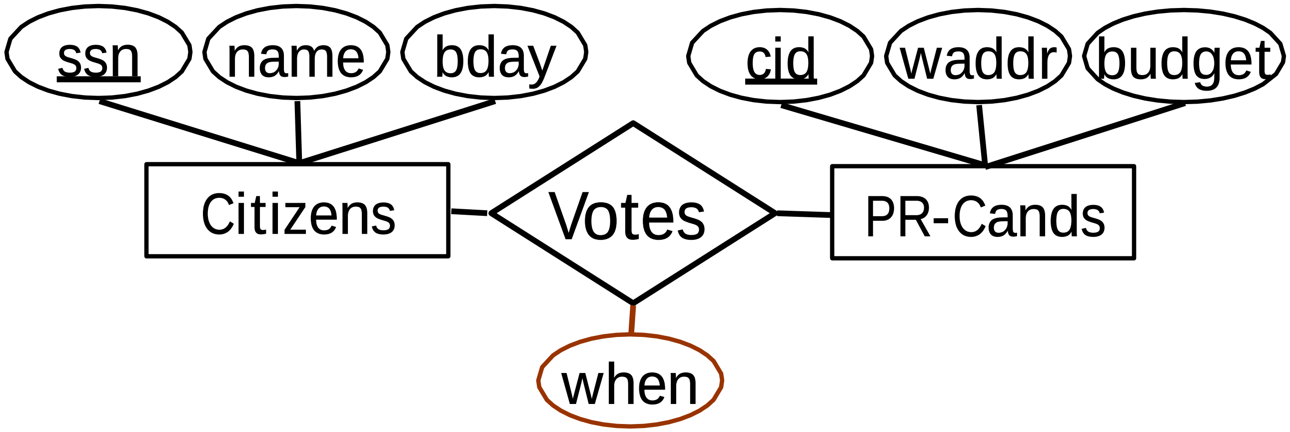
- *Relationship* : Association among two or more entities

Logical DB Design: ER to Relational




```
CREATE TABLE Citizens
        (ssn    CHAR(11),
         name   CHAR(20),
         bday   DATE,
         PRIMARY KEY  (ssn))
```

Relationship Sets to Tables



Relationship set -> Table

Attributes:

- Participating entity set primary keys
  - Foreign key
  - Superkey
- Descriptive attributes

```
CREATE TABLE Votes(
        ssn     CHAR(11),
        cid     INTEGER,
        when    DATE,
PRIMARY KEY (ssn, cid),
FOREIGN KEY (ssn) REFERENCES Citizens,
FOREIGN KEY (cid) REFERENCES PR-Cands)
```

Can ssn have a null value?

Can generalize to n-ary relationships

Another E/R diagram format

```
Department (DepartmentNum, Name, Location)
Employee (EmployeeNum, LastName, FirstName, Street, City,
    State, PostalCode, WageRate, SocSecNum, DepartmentNum)
    AK   SocSecNum
    SK   LastName, FirstName
    FK   DepartmentNum → Department
```

The E-R diagram for the preceding database design appears in Figure 6-2.

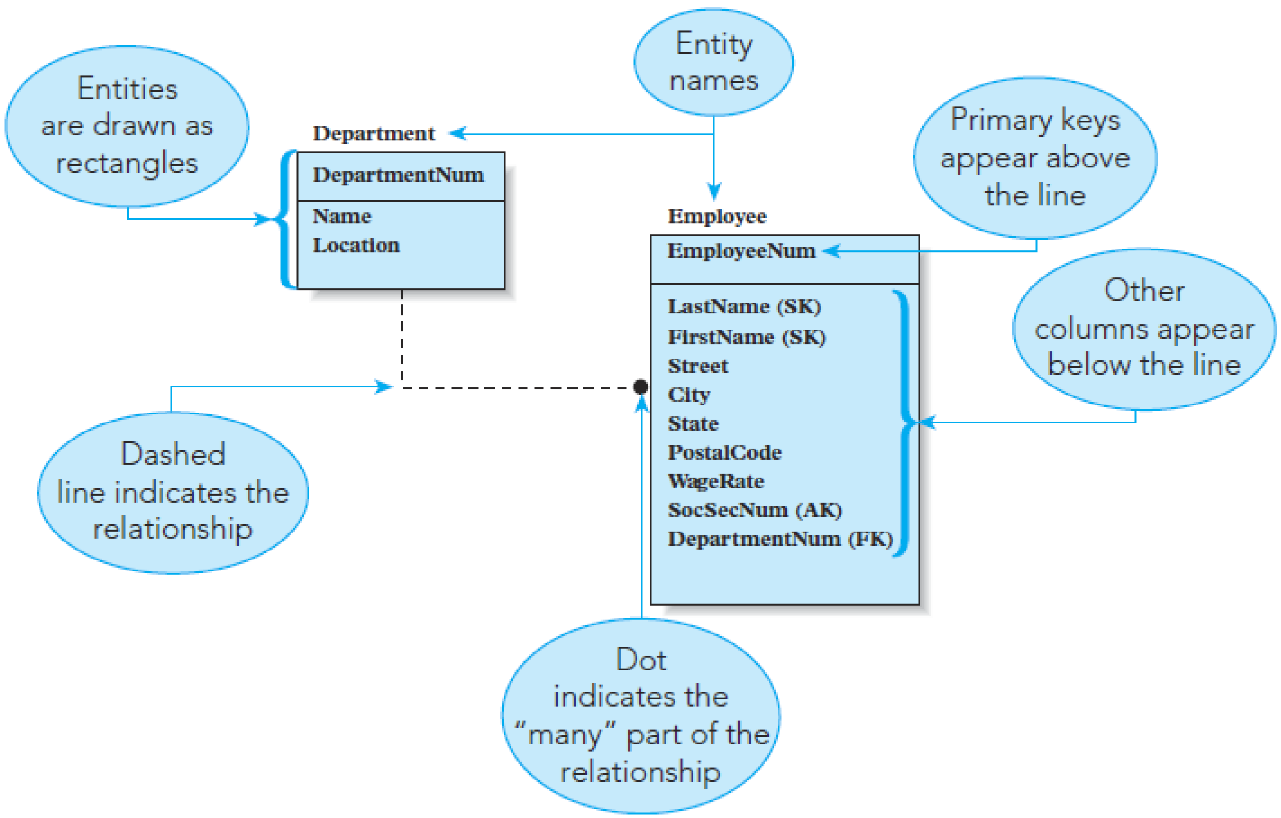


FIGURE 6-2 E-R diagram

Design theory

- Design theory is about how to represent your data to avoid ***anomalies***.

What is a good relational schema? How can we improve it?

- e.g.: Suppliers (<u>name, item</u>, desc, addr, price)

  <u>Redundancy</u> Problems:

  1. A supplier supplies two items: Redundant Storage
  2. Change address of a supplier: Update Anomaly
  3. Insert a supplier: Insertion Anomaly
     - What if the supplier does not supply any items (nulls?)
     - Record desc for an item that is not supplied by any supplier
  4. Delete the only supplier tuple: Delete Anomaly
     - Use nulls?
     - Delete the last item tuple. Can't make name null. Why?

Dealing with Redundancy

- Identify “bad” schemas
  - functional dependencies
- Main refinement technique: decomposition
  - replacing larger relation with smaller ones
- Decomposition should be used judiciously:
  - Is there a reason to decompose a relation?
    - Normal forms: guarantees against (some) redundancy
  - Does decomposition cause any problems?
    - Lossless join
    - Dependency preservation
    - Performance (must join decomposed relations)

Functional Dependency (FD)

Def: Let A,B be *sets* of attributes
We write A → B or say A ***functionally determines*** B if, for any tuples t_1 and t_2:

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call A → B a **<u>functional dependency</u>**

A->B means that
"whenever two tuples agree on A then they agree on B."

Consider, (X,Y) → Z

X	Y	Z	K
1	1	11	A
1	2	12	A
2	2	22	A
2	2	22	B

- An FD is a statement about all allowable relations.
  - Based only on application semantics, can't deduce from instances
  - Can simply check if an instance violates FD (and other ICs)

FDs for Relational Schema Design

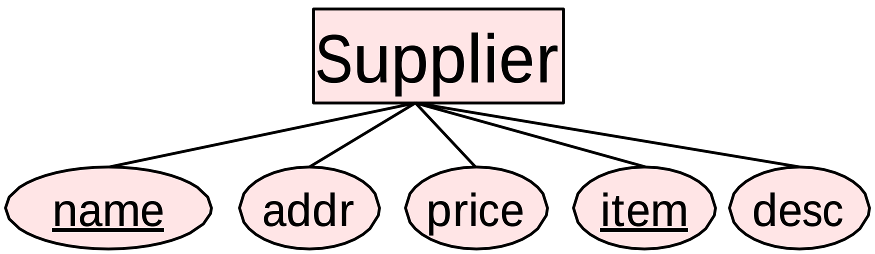
- High-level idea: **why do we care about FDs?**

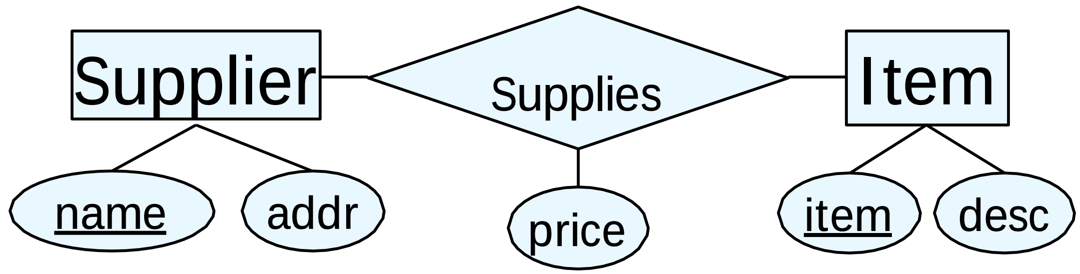
    1. Start with some relational *schema*

    2. Model its *functional dependencies (FDs)*

    3. Use these to *design a better schema*
        1. One which minimizes the possibility of anomalies

Example: Constraints on Entity Set





- S(name, item, desc, addr, price)
- FD: {n,i} → {n,i,d,a,p}
- FD: {n} → {a}
- FD: {i} → {d}
- Decompose to: NA, ID, INP

- Spl(name, item, price)
  - FD: {n,i} → {n, i, p}
- Sup(name, addr)
  - FD: {n} → {n, a}
- Item (item, desc)
  - FD: {i} → {i, d}

ER design is subjective and can have many E + Rs

FDs: sanity checks + deeper understanding of schema

Same situation could happen with a relationship set

Finding functional dependencies

Rep

RepNum	LastName	FirstName	Street	City	State	PostalCode	Commission	PayClass	Rate
15	Campos	Rafael	724 Vinca Dr.	Grove	CA	90092	$23,457.50	1	0.06
30	Gradey	Megan	632 Liatris St.	Fullton	CA	90085	$41,317.00	2	0.08
45	Tian	Hui	1785 Tyler Ave.	Northfield	CA	90098	$27,789.25	1	0.06
60	Sefton	Janet	267 Oakley St.	Congaree	CA	90097	$0.00	1	0.06

- RepNum -> LastName, FirstName ?
- LastName -> Street, City, State?
- PayClass-> Rate?

Find all functional dependencies

- StudentNum (student number)
- StudentLast (student last name)
- StudentFirst (student first name)
- HighSchoolNum (number of the high school from which the student graduated)
- HighSchoolName (name of the high school from which the student graduated)
- AdvisorNum (number of the student's advisor)
- AdvisorLast (last name of the student's advisor)
- AdvisorFirst (first name of the student's advisor)

Student numbers, high school numbers, and advisor numbers are unique; no two students have the same number, no two high schools have the same number, and no two advisors have the same number. Use this information to determine the functional dependencies in the Student relation.

Inferring FD

- ename $\rightarrow$ ejob, ejob $\rightarrow$ esal; $\Rightarrow$ ename $\rightarrow$ esal
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$
  - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Additional rules (derivable):
  - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- Set of all FD = closure of F, denoted as F^+
- AA sound: only generates FD in F^+
- AA complete: repeated application generates all FD in F^+

Keys and Superkeys

A **superkey** is a set of attributes $\mathbf{A_1, \ldots, A_n}$ s.t. for *any other* attribute **B** in R, we have $\{\mathbf{A_1, \ldots, A_n}\} \rightarrow \mathbf{B}$

I.e. all attributes are *functionally determined* by a superkey

A **key** is a *minimal* superkey

Meaning that no subset of a key is also a superkey

Decomposition

- Replace a relation with two or more relations
- Problems with decomposition
  1. Some queries become more expensive. (more joins)
  2. **Lossless Join**: Can we reconstruct the original relation from instances of the decomposed relations?
  3. **Dependency Preservation**: Checking some dependencies may require joining the instances of the decomposed relations.

Lossless Join Decompositions

- Relation R, FDs F: Decomposed to X, Y
- Lossless-Join decomposition if:

  $\Pi_X(r) \bowtie \Pi_Y(r) = r$ for **every** instance r of R
- Note, $r \subseteq \Pi_X(r) \bowtie \Pi_Y(r)$ is always true,

  not vice versa, unless the join is lossless
- Can generalize to three more relations

A	B	C
1	2	3
4	5	6
7	2	8

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

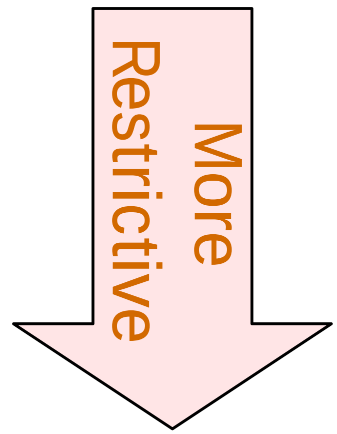
Lossless Join …

- Relation R, FDs F: Decomposed to X, Y
  - Test: lossless-join w.r.t. F if and only if the closure of F contains:
    - $X \cap Y \rightarrow X$, or
    - $X \cap Y \rightarrow Y$

    i.e. attributes common to X and Y contain a key for either X or Y
  - Also, given FD: $X \rightarrow Y$ and $X \cap Y = \varnothing$, the decomposition into R-Y and XY is lossless
    - **X is a key in XY, and appears in both**

Normal Forms

- Is any refinement is needed!
- Normal Forms: guarantees that certain kinds of problems won't occur
  - 1 NF : Atomic values
  - 2 NF : Historical
  - 3 NF : …
  - BCNF : Boyce-Codd Normal Form



- Role of FDs in detecting redundancy:
  - Relation R with 3 attributes, ABC.
    - No ICs (FDs) hold ⇒ no redundancy.
    - A → B ⇒ 2 or more tuples with the same A value, redundantly have the same B value!

First Normal Form (1NF)

- A relation (table) is in first normal form (1NF) when it does not contain repeating groups (multiple entries for a single record)

Orders

OrderNum	OrderDate	ItemNum	NumOrdered
51608	10/12/2015	CD33	5
51610	10/12/2015	KL78 TR40	25 10
51613	10/13/2015	DL51	5
51614	10/13/2015	FD11	1
51617	10/15/2015	NL89 TW35	4 3
51619	10/15/2015	FD11	2
51623	10/15/2015	DR67 FH24 KD34	5 12 10
51625	10/16/2013	MT03	8

Unnormalized

`Orders (OrderNum, OrderDate, (ItemNum, NumOrdered) )`

Orders

OrderNum	OrderDate	ItemNum	NumOrdered
51608	10/12/2015	CD33	5
51610	10/12/2015	KL78	25
51610	10/12/2015	TR40	10
51613	10/13/2015	DL51	5
51614	10/13/2015	FD11	1
51617	10/15/2015	NL89	4
51617	10/15/2015	TW35	3
51619	10/15/2015	FD11	2
51623	10/15/2015	DR67	5
51623	10/15/2015	FH24	12
51623	10/15/2015	KD34	10
51625	10/16/2015	MT03	8

1NF

`Orders (OrderNum, OrderDate, ItemNum, NumOrdered)`

Second Normal Form (2NF)

- A column is a nonkey column when it is not a part of the primary key (PK)
- A table (relation) is in second normal form (2NF) when it is in 1NF and no nonkey column is dependent on only a portion of the primary key

Orders

OrderNum	OrderDate	ItemNum	Description	NumOrdered	QuotedPrice
51608	10/12/2015	CD33	Wood Block Set (48 piece)	5	$86.99
51610	10/12/2015	KL78	Pick Up Sticks	25	$10.95
51610	10/12/2015	TR40	Tic Tac Toe	10	$13.99
51613	10/13/2015	DL51	Classic Railway Set	5	$104.95
51614	10/13/2015	FD11	Rocking Horse	1	$124.95
51617	10/15/2015	NL89	Wood Block Set (62 piece)	4	$115.99
51617	10/15/2015	TW35	Fire Engine	3	$116.95
51619	10/15/2015	FD11	Rocking Horse	2	$121.95
51623	10/15/2015	DR67	Giant Star Brain Teaser	5	$29.95
51623	10/15/2015	FH24	Puzzle Gift Set	12	$36.95
51623	10/15/2015	KD34	Pentominoes Brain Teaser	10	$13.10
51625	10/16/2015	MT03	Zauberkasten Brain Teaser	8	$45.79

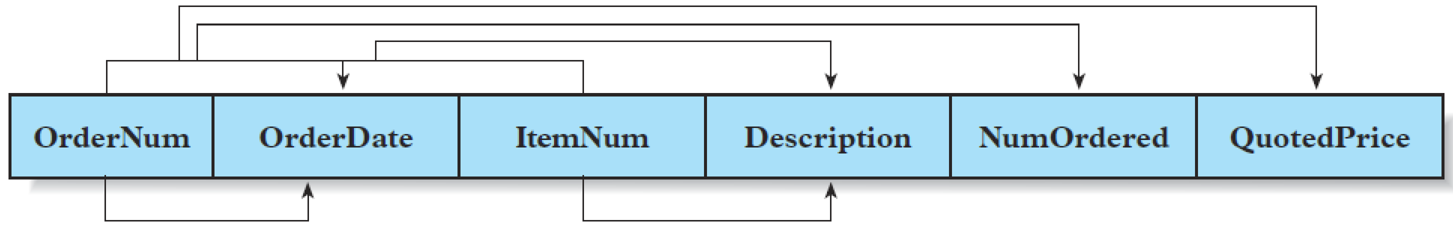
Orders (<u>OrderNum</u>, OrderDate, <u>ItemNum</u>, Description, NumOrdered, QuotedPrice)

This table has the following functional dependencies:

OrderNum → OrderDate
ItemNum → Description
OrderNum, ItemNum → NumOrdered, QuotedPrice, OrderDate, Description

2NF (cont)

Orders dependencies



Orders

OrderNum	OrderDate
51608	10/12/2015
51610	10/12/2015
51613	10/13/2015
51614	10/13/2015
51617	10/15/2015
51619	10/15/2015
51623	10/15/2015
51625	10/16/2015

Item

ItemNum	Description
AH74	Patience
BR23	Skittles
CD33	Wood Block Set (48 piece)
DL51	Classic Railway Set
DR67	Giant Star Brain Teaser
DW23	Mancala
FD11	Rocking Horse
FH24	Puzzle Gift Set
KA12	Cribbage Set
KD34	Pentominoes Brain Teaser
KL78	Pick Up Sticks
MT03	Zauberkasten Brain Teaser
NL89	Wood Block Set (62 piece)
TR40	Tic Tac Toe
TW35	Fire Engine

OrderLine

OrderNum	ItemNum	NumOrdered	QuotedPrice
51608	CD33	5	$86.99
51610	KL78	25	$10.95
51610	TR40	10	$13.99
51613	DL51	5	$104.95
51614	FD11	1	$124.95
51617	NL89	4	$115.99
51617	TW35	3	$116.95
51619	FD11	2	$121.95
51623	DR67	5	$29.95
51623	FH24	12	$36.95
51623	KD34	10	$13.10
51625	MT03	8	$45.79

(<u>OrderNum</u>, OrderDate)

(<u>ItemNum</u>, Description)

(<u>OrderNum</u>, <u>ItemNum</u>, NumOrdered, QuotedPrice)

Conversion to 2NF

Boyce-Codd Normal Form (BCNF)

- Relationn R with FDs F is in BCNF if, for all X → A in F^+
  - $A \in X$ (trivial FD), or
  - X is a super key

  i.e. all non-trivial FDs over R are key constraints.
- **No redundancy in R** (at least none that FDs detect)
- Most desirable normal form
- Costumer table is not in BCNF since RepNum is not a key

Customer

CustomerNum	CustomerName	Balance	CreditLimit	RepNum	LastName	FirstName
126	Toys Galore	$1,210.25	$7,500.00	15	Campos	Rafael
260	Brookings Direct	$575.00	$10,000.00	30	Gradey	Megan
334	The Everything Shop	$2,345.75	$7,500.00	45	Tian	Hui
386	Johnson's Department Store	$879.25	$7,500.00	30	Gradey	Megan
440	Grove Historical Museum Store	$345.00	$5,000.00	45	Tian	Hui
502	Cards and More	$5,025.75	$5,000.00	15	Campos	Rafael
586	Almondton General Store	$3,456.75	$15,000.00	45	Tian	Hui
665	Cricket Gift Shop	$678.90	$7,500.00	30	Gradey	Megan
713	Cress Store	$4,234.60	$10,000.00	15	Campos	Rafael
796	Unique Gifts	$124.75	$7,500.00	45	Tian	Hui
824	Kline's	$2,475.99	$15,000.00	30	Gradey	Megan
893	All Season Gifts	$935.75	$7,500.00	15	Campos	Rafael

Customer (<u>CustomerNum</u>, CustomerName, Balance, CreditLimit, RepNum, LastName, FirstName)

The functional dependencies in this table are as follows:

CustomerNum → CustomerName, Balance, CreditLimit, RepNum, LastName, FirstName
RepNum → LastName, FirstName

Customer (<u>CustomerNum</u>, CustomerName, Balance, CreditLimit, RepNum)
Rep (<u>RepNum</u>, LastName, FirstName)

Third Normal Form (3NF)

- Relation R with FDs F is in 3NF if, for all X → A in F^+

> If **$\{A_1, ..., A_n\}$ → B** is a *non-trivial* FD in R
>
> then $\{A_1, ..., A_n\}$ is a superkey for R **OR**
>
> B is part of some key of R ***(prime attribute)***

  - Minimality of a key (not superkey) is crucial!

- BCNF implies 3NF
- e.g.: Sailor (Sailor, Boat, Date, CreditCrd)
  - SBD -> SBDC, S -> C **(not 3NF)**
  - If C -> S, then CBD -> SBDC (i.e. CBD is also a key). Now in 3NF!
  - Note redundancy in (S, C); 3NF permits this
  - Compromise used when BCNF not achievable, or perf. Consideration
- Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible.

Multi-Value Dependencies (MVDs)

A multi-value dependency (MVD) is another type of dependency that could hold in our data, ***which is not captured by FDs***

Formal definition:

Given a relation **R** having attribute set **A**, and two sets of attributes $\mathbf{X}, \boldsymbol{Y} \subseteq \boldsymbol{A}$

The ***multi-value dependency (MVD)*** $\boldsymbol{X} \twoheadrightarrow \boldsymbol{Y}$ holds on R if

for any tuples $\boldsymbol{t_1}, \boldsymbol{t_2} \in \boldsymbol{R}$ s.t. $\boldsymbol{t_1}[\boldsymbol{X}] = \boldsymbol{t_2}[\boldsymbol{X}]$, there exists a tuple $\boldsymbol{t_3}$ s.t.:

$t_1[X] = t_2[X] = t_3[X]$

$t_1[Y] = t_3[Y]$

$t_2[A \backslash Y] = t_3[A \backslash Y]$

Where A \ B means "elements of set A not in set B"

Note that this can be avoided all together if each repeated group is put in its own table.

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
Cinema 1	Star Trek: The Wrath of Kahn	Burrito
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
Marcus	Star Wars: The Boba Fett Prequel	Ramen
Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

Are there any functional dependencies that might hold here?

No…

And yet it seems like there is some pattern / dependency…

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
Cinema 1	Star Trek: The Wrath of Kahn	Burrito
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
Marcus	Star Wars: The Boba Fett Prequel	Ramen
Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

For a given movie theatre…

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
Cinema 1	Star Trek: The Wrath of Kahn	Burrito
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
Marcus	Star Wars: The Boba Fett Prequel	Ramen
Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

For a given movie theatre…

Given a set of movies and snacks…

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
Cinema 1	Star Trek: The Wrath of Kahn	Burrito
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
Marcus	Star Wars: The Boba Fett Prequel	Ramen
Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

For a given movie theatre…

Given a set of movies and snacks…

Any movie / snack combination is possible!

MVDs: Movie Theatre Example

	Movie_theater	film_name	snack
t_1	Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
	Cinema 1	Star Trek: The Wrath of Kahn	Burrito
	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Marcus	Star Wars: The Boba Fett Prequel	Ramen
	Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write **{A}** $\twoheadrightarrow$ **{B}** if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$

MVDs: Movie Theatre Example

	Movie_theater	film_name	snack
t_1	Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Cinema 1	Star Trek: The Wrath of Kahn	Burrito
	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Marcus	Star Wars: The Boba Fett Prequel	Ramen
	Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write **{A}** $\twoheadrightarrow$ **{B}** if for any tuples t_1,t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$

MVDs: Movie Theatre Example

	Movie_theater	film_name	snack
t_1	Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Cinema 1	Star Trek: The Wrath of Kahn	Burrito
	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Marcus	Star Wars: The Boba Fett Prequel	Ramen
	Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write **{A}** $\twoheadrightarrow$ **{B}** if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$
- $t_3[B] = t_1[B]$

MVDs: Movie Theatre Example

	Movie_theater (A)	film name (B)	Snack (C)
t_1	Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Cinema 1	Star Trek: The Wrath of Kahn	Burrito
	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Marcus	Star Wars: The Boba Fett Prequel	Ramen
	Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write **{A} ↠ {B}** if for any tuples t_1,t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$
- $t_3[B] = t_1[B]$
- and $t_3[R\backslash B] = t_2[R\backslash B]$

Where R\B is "R minus B" i.e. the attributes of R not in B

MVDs: Movie Theatre Example

	Movie_theater (A)	film name (B)	Snack (C)
t_2	Cinema 1	Star Trek: The Wrath of Kahn	Kale Chips
	Cinema 1	Star Trek: The Wrath of Kahn	Burrito
t_3	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_1	Cinema 1	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Marcus	Star Wars: The Boba Fett Prequel	Ramen
	Marcus	Star Wars: The Boba Fett Prequel	Plain Pasta

This expresses a sort of dependency (= data redundancy) that we *can't* express with FDs

**Actually, it expresses <u>conditional independence</u> (between film and snack given movie theatre)!*

Summary of Normal Forms

Normal Form	Meaning/Required Conditions
1NF	No repeating groups
2NF	1NF and no nonkey column dependent on only a portion of the primary key
3NF	2NF and the only determinants are candidate keys (BCNF) or part of a candidate key
4NF	3NF and no multivalued dependencies

For FDs, BCNF is the normal form
A tradeoff for insert performance: 3NF

For today

- Convert the following table to third normal form. In this table, StudentNum determines Student-Name, NumCredits, AdvisorNum, and AdvisorName. AdvisorNum determines AdvisorName. CourseNum determines Description. The combination of StudentNum and CourseNum determines Grade.

```
Student (StudentNum, StudentName, NumCredits,
AdvisorNum, AdvisorName, (CourseNum, Description,
Grade))
```