# BZAN 6354

# Lecture 13

## April 15, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences

# Agenda

- Administration
- Quick Overview of FD and Armstrong's Axioms
- Modules 8.1-8.2 (Normal forms and Normalization)
- Break
- Modules 8.3 – 8.4 (Examples/Demo)
  - Will put you in a good place for Assignment 4
- If time allows… Module 13.1: String Manipulation SQL

# Administration

- Assignment 4 is posted
  - Due Monday, April 22 @ midnight
  - We will do some examples in class tonight that are very similar to what you will be expected to do

- SQL Project
  - Please go ahead and get started
  - So far only about 1/3 of you have started (as far as I can tell…)

- Upcoming schedule:
  - April 22 – Advanced SQL / Wrap up and review
  - April 29 – Exam 2 (in this room during class time)

# Extra credit opportunity

- Course evaluations will be available in AccessUH on Thursday April 18.
  - If 60% of the class (18/30 students) completes the evaluation **everyone will get half a point** added to their final grade
  - If 80% of the class (24/20 students) completes the evaluations **everyone will get one full point** added to their final grade


Faculty/Course Evaluation

- A few things to note:
  - Course evaluations close on Monday, April 29 @ 11:59 PM
  - Course evaluations are anonymous
    - I only ever see the aggregate results – not what any one individual put
  - I can <u>never</u> see who has completed, only the percentage
    - You will get the extra credit even if you do not complete the evaluation, as long as enough of your classmates do
  - I see evaluation results only after final grades are submitted
  - What you put in the evaluation, and if you even complete it has no bearing on the extra credit – only the % that submit does

# What do course evaluations look like?

- This is all I see during the evaluation period:

## Faculty Course Evaluation Report

**Select Term:** Fall 2023

Search by Class No. | Type your search here | Search | Reset Search

| Term | Catalog | Class No. | Course Description | Instructor Name | Evaluation Start | Evaluation End | Student(s) Enrolled | Survey(s) Taken | Response Rate (%) | Available Actions |
|---|---|---|---|---|---|---|---|---|---|---|
| 2210 | BZAN 6354 | 15792 | DB Mgt Tools Bus Analytics | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 33 | 6 | 18.18 | |
| 2210 | BZAN 6354 | 18545 | DB Mgt Tools Bus Analytics | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 14 | 0 | 0.00 | |
| 2210 | BZAN 6356 | 15793 | Adv DB Mgt Tools Bus Analytics | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 16 | 4 | 25.00 | |
| 2210 | MIS 4397 | 20483 | Advanced Web Applications | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 33 | 5 | 15.15 | |
| 2210 | MIS 7373 | 17846 | Bus Appls Database Mgt Sys I | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 28 | 3 | 10.71 | |
| 2210 | MIS 8397 | 20061 | Survey of MIS Research | Grimes,George M | 11/20/2023 12:01:00 AM | 12/1/2023 11:59:00 PM | 2 | 0 | 0.00 | |

# What do course evaluations look like?

- A week or two after grades are posted I get a button to “View Reports”

## Faculty Course Evaluation Report

**Select Term:** Spring 2023

Search by: Class No. | Type your search here | Search | Reset Search

| Term | Catalog | Class No. | Course Description | Instructor Name | Evaluation Start | Evaluation End | Student(s) Enrolled | Survey(s) Taken | Response Rate (%) | Available Actions |
|---|---|---|---|---|---|---|---|---|---|---|
| 2190 | BZAN 6354 | 15670 | DB Mgt Tools Bus Analytics | Grimes,George M | 4/20/2023 12:01:00 AM | 5/1/2023 11:59:00 PM | 32 | 11 | 34.38 | View Reports |
| 2190 | BZAN 6354 | 18846 | DB Mgt Tools Bus Analytics | Grimes,George M | 4/20/2023 12:01:00 AM | 5/1/2023 11:59:59 PM | 30 | 5 | 16.67 | View Reports |
| 2190 | BZAN 6356 | 25915 | Adv DB Mgt Tools Bus Analytics | Grimes,George M | 4/20/2023 12:01:00 AM | 5/1/2023 11:59:00 PM | 20 | 7 | 35.00 | View Reports |
| 2190 | MIS 7373 | 18494 | Bus Appls Database Mgt Sys I | Grimes,George M | 4/20/2023 12:01:00 AM | 5/1/2023 11:59:59 PM | 34 | 5 | 14.71 | View Reports |

# What do course evaluations look like?

## Section 1 Instructor

| | Relative Frequency Distribution of Response | | | | | Section Statistics | | | Dept. Statistics | | | College Statistics | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | N | Mean | Std. Dev. | N | Mean | Std. Dev. | N | Mean | Std. Dev. |
| 1) The instructor was well prepared for class. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2017 | 4.49 | 0.88 | 9460 | 4.57 | 0.78 |
| 2) The instructor presented the subject matter for this course clearly by using effective teaching techniques. | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 11 | 5 | 0 | 2014 | 4.34 | 1 | 9441 | 4.43 | 0.91 |
| 3) The instructor was receptive to questions and alternate points of view of the subject material. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2011 | 4.44 | 0.89 | 9430 | 4.49 | 0.84 |
| 4) The instructor stimulated my mental curiosity. | 81.8 | 9.1 | 9.1 | 0.0 | 0.0 | 11 | 4.73 | 0.65 | 2013 | 4.31 | 1 | 9442 | 4.36 | 0.97 |
| 5) The instructor treated the students in an appropriate manner in the classroom. | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 11 | 5 | 0 | 2013 | 4.56 | | | | |
| 6) The instructor generally was available to consult with and assist students. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2013 | 4.45 | | | | |
| 7) I will recommend this instructor to other students for the same course. | 90.0 | 10.0 | 0.0 | 0.0 | 0.0 | 10 | 4.9 | 0.32 | 2013 | 4.37 | | | | |
| 8) The grading system seemed fair. | 81.8 | 18.2 | 0.0 | 0.0 | 0.0 | 11 | 4.82 | 0.4 | 2018 | 4.38 | | | | |
| 9) The procedures for providing feedback information were such that I knew how I was progressing in the course. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2013 | 4.34 | | | | |
| 10) The instructor was enthusiastic in encouraging students to focus on the subject material. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2010 | 4.51 | | | | |
| 11) The instructor set forth course requirements clearly and relatively early in the semester. | 81.8 | 18.2 | 0.0 | 0.0 | 0.0 | 11 | 4.82 | 0.4 | 2008 | 4.48 | | | | |
| 12) The instructor was an effective teacher compared to other instructors. | 90.9 | 9.1 | 0.0 | 0.0 | 0.0 | 11 | 4.91 | 0.3 | 2010 | 4.33 | | | | |
| Mean | | | | | | | 4.89 | | | 4.42 | | | | |
| Score | | | | | | | 58.72 | | | 52.9 | | | | |

## Section 3 Students' Comments

**Comments on what went well in the course**

1) Everything went well and as planned by Intsructor

2) The instructor was always well prepared for the class.

3) Great presence and well spoken. Course slides are very informative

4) The overall material of the course was very well taught.

5) The professor is very supportive and helps the students whenever needed. I am happy in taking the course as I learned a lot.

**Comments on what needs to be improved in the course**

1) I don't think any, Everything was just perfect.

2) Course work needs to be improved.

3) There could be more preparation material for exams. I practice exam or prep notes could have been beneficial.

4) As far as I knew everything in course is pretty well organized

**General comments, opinions, and suggestions**

1) Instructor has a very clear picture in mind about the course and the coursework. Honestly, I just listen to his lectures and no other preparation is needed . He is outstanding.

2) Everything is good.

3) N/A

4) I strongly recommend students take the course as the professor is very supportive.

# Review: Functional Dependencies

- What is a functional dependency?
  - FDs specify a relationship between attributes in a relation
  - May be semantically obvious or inferred

- In the expression A $\rightarrow$ B, what do we call A and B?
  - A: Determinant
  - B: Dependent

- Where do functional dependencies come from?
  - Business Rules

- How do you know a functional dependency is undesirable?
  - The determinant is not a candidate key

# Review: Armstrong's Axioms

- Three primary Axioms
  - Reflexivity
    - If Y is a subset of X, then X → Y (a trivial dependency)
  - Augmentation
    - If X → Y, then {X,Z} → Y and {X,Z}→{Y,Z}
  - Transitivity
    - If X → Y and Y → Z, then X → Z

- Four inference rules
  - Union
    - If X → Y and X → Z, then X → {Y,Z}
  - Decomposition
    - If X → {Y,Z}, then X → Y and X → Z
  - Composition
    - If A → B and C → D, then {A,C} → {B,D}
  - Pseudotransitivity
    - If X → Y and {Y,W} → Z, then {X,W} → Z

# Review: Attribute Closure

- Similar to closure of a set of FDs for a relation, but at a lower level

- A set of all attributes functionally determined by a determinant

- If we are looking for closure for a set of attributes, Z, from the set of FDs, F, from relation R:
  - Expressed as $Z^+$ or Closure [Z | F]
  - "Z under F"

- This can be very useful for identifying candidate keys!

# Review: Synthesis Approach to find Candidate Keys

- Given a relation schema, R, and a set of FDs, F, that holds for R, find a subset, Z, of attributes of R such that $Z^+$ (closure [Z | F]) includes all attributes of R
  - Basically, find a FD where the dependent includes ALL attributes of R

- Given R (A, B, C, D, E, G, H) and F [fd1, fd2, fd3] where
  FD1: B → {G, H}
  FD2: A → B
  FD3: C → D

  What is a candidate key of R?

# Review: Synthesis Approach to find Candidate Keys

- Given R (A, B, C, D, E, G, H) and F [fd1, fd2, fd3] where
FD1: B $\rightarrow$ {G, H}    FD2: A $\rightarrow$ B    FD3: C $\rightarrow$ D
  - $A^+ = \{A,B,G,H\}$
  - $B^+ = \{B,G,H\}$
  - $C^+ = \{C,D\}$
  - $\{A,C\}^+ = \{A,B,C,D,G,H\}$

- Can all attributes of R be determined by {A,C}?
  - No, we are missing E
  - {A,C} is not a candidate key!

- {A,C,E} $\rightarrow$ {A,B,C,D,E,G,H}  Therefore, {A,C,E} is a candidate key
  - Augmentation allows us to add E to both sides!

# Review: Decomposition Approach to find Candidate Keys

- Given the universal relation schema R $\{A_1, A_2, A_3, . . . , A_n\}$
  - Step 1: Set superkey, K of R = {A1, A2, A3, . . . , An}
  - Step 2: Remove an attribute $A_i$, (i = 1, 2, 3, . . . . ., n) from R such that $\{K – A_i\}$ is still a superkey, K’, of R
    - Note: In order for K’ to be a superkey of R, the FD: (K’ → $A_i$) should persist in F+
  - Step 3: Repeat step 2 above recursively until K’ is further irreducible
- The irreducible K’ is a candidate key of R under the set of FDs, F.

# Module 8.1 – 8.2

## Normalization

- Normalization – 1NF, 2NF, 3NF, BCNF

# Normalization

- Data redundancy resulting in modification anomalies are due to "undesirable FDs"
- FDs are derived from business rules – we can't just get rid of them, we must resolve via….NORMALIZATION!
- Normalization is a technique that facilitates systematic validation of the participation of attributes in a relation schema from a perspective of data redundancy.
- Normal Forms (NFs) provide a stepwise progression towards attaining a design that is guaranteed to be free of data redundancies that cause modification anomalies from a functional dependency perspective.

# The prime directive for FD based Normalization

- A relation schema R is fully normalized with regard to functional dependencies if for every **non-trivial** FD in R, the determinant is a candidate key of R.
  - Remember: A FD in R is trivial if and only if the dependent is a subset of the determinant.

- The Goal:
  - Decompose R such that in the decomposition D[R1, R2,..Rn] for every FD in D the determinant is a candidate key of the respective relation schema in D.

# Desirable Versus Undesirable FDs

- Desirable FDs in a relation schema R are those where the determinant is a candidate key of R – no exceptions.

- **Undesirable FDs in a relation schema R are those where the determinant is not a candidate key of R.**
    - That is, the FDs will cause data redundancy and the consequent modification anomalies in R.

# A simple algorithm for Normalization

- Pull out the undesirable FD(s) from the target relation schema R as separate relation schema(s) [R1, R2, etc.]

- Retain the determinant of the pulled-out relation schema (say, R1) as an attribute(s) in the leftover target relation schema, R0, to facilitate reconstruction of the original target relation schema.

# A simple algorithm for Normalization

- This is what we did earlier

| Store | Product | Price | Quantity | Location | Discount | Sq_ft | Manager |
|---|---|---|---|---|---|---|---|
| 15 | Refrigerator | 1850 | 120 | Houston | 5% | 2300 | Metzger |
| 15 | Dishwasher | 600 | 150 | Houston | 5% | 2300 | Metzger |
| 13 | Dishwasher | 600 | 180 | Tulsa | 10% | 1700 | Metzger |
| 14 | Refrigerator | 1850 | 150 | Tulsa | 5% | 1900 | Schott |
| 14 | Television | 1400 | 280 | Tulsa | 10% | 1900 | Schott |
| 14 | Humidifier | 55 | 30 | Tulsa | | 1900 | Schott |
| 17 | Television | 1400 | 10 | Memphis | | 2300 | Creech |
| 17 | Vacuum Cleaner | 300 | 150 | Memphis | 5% | 2300 | Creech |
| 17 | Dishwasher | 600 | 150 | Memphis | 5% | 2300 | Creech |
| 11 | Computer | | 180 | Houston | 10% | 2300 | Creech |
| 11 | Refrigerator | 1850 | 120 | Houston | 5% | 2300 | Creech |
| 11 | Lawn Mower | 300 | | Houston | | 2300 | Creech |

FD1: Store → {Location, Sq_Ft, Manager}
FD2: {Store, Product} → Quantity
FD3: Product → Price
FD4: Quantity → Discount

FD1: Store → {Location, Sq_Ft, Manager}

STORE

| Store | Location | Sq_ft | Manager |
|---|---|---|---|
| 15 | Houston | 2300 | Metzger |
| 13 | Tulsa | 1700 | Metzger |
| 14 | Tulsa | 1900 | Schott |
| 17 | Memphis | 2300 | Creech |
| 11 | Houston | 2300 | Creech |

FD3: Product → Price

PRODUCT

| Product | Price |
|---|---|
| Refrigerator | 1850 |
| Dishwasher | 600 |
| Television | 1400 |
| Humidifier | 55 |
| Vacuum Cleaner | 300 |
| Computer | |
| Lawn Mower | 300 |
| Washing Machine | 750 |

INVENTORY

| Store | Product | Quantity |
|---|---|---|
| 15 | Refrigerator | 120 |
| 15 | Dishwasher | 150 |
| 13 | Dishwasher | 180 |
| 14 | Refrigerator | 150 |
| 14 | Television | 280 |
| 14 | Humidifier | 30 |
| 17 | Television | 10 |
| 17 | Vacuum Cleaner | 150 |
| 17 | Dishwasher | 150 |
| 11 | Computer | 180 |
| 11 | Refrigerator | 120 |
| 11 | Lawn Mower | |

FD2: {Store, Product} → Quantity

DISC_STRUCTURE

| Quantity | Discount |
|---|---|
| 120 | 5% |
| 150 | 5% |
| 180 | 10% |
| 280 | 10% |
| 30 | |
| 10 | |

FD4: Quantity → Discount

# Normal Forms: An Overview

- A relation schema is said to be in a particular normal form if it satisfies certain prescribed criteria for that normal form.

- First normal form (1NF) reflects the properties of a relation schema – i.e., by definition a relation schema is in 1NF.

- Normal forms associated with functional dependencies are second (2NF), third (3NF), and Boyce-Codd (BCNF) normal forms.

- The violations of each of these normal forms signal the presence of a specific type of 'undesirable' FD.
  - violation of a normal form, can be interpreted as equivalent to an inadvertent mixing up of entity types belonging to two different entity classes in a single entity type.

# First Normal Form (1NF)

- First normal form (1NF) imposes conditions so that a base relation that is physically stored as a file does not contain records with a variable number of fields.
  - This is accomplished by prohibiting multi-valued and composite attributes in a relation schema.

- Such a constraint, in effect, prevents relations from containing other relations.

- In essence 1NF requires that the domain of an attribute must include only atomic values and that the value of an attribute in a relation's tuple must be a single value from the domain of that attribute.

# Example of 1NF violation

**ALBUM**

| <u>**Album_no**</u> | **Artist_nm** | **Price** | **Stock** |
|---|---|---|---|
| BS123 | Britney Spears | 17.95 | 1000 |
| JT111 | Justin Timberlake | 17.95 | 1200 |
| BTL007 | {John Lennon, Paul McCartney, George Harrison, Ringo Star} | 23.95 | |
| MJ100 | Michael Jackson | 17.95 | |
| JM456 | John Mayer | 16.95 | 1000 |
| JM151 | John Mayer | 16.95 | 1000 |
| MX789 | Madonna | 11.95 | 500 |
| DJM237 | {John Denver, Michael  Jackson, Madonna} | 11.95 | 2000 |
| DR711 | Diana Ross | 12.95 | 1000 |
| PM137 | Paul McCartney | 19.95 | |

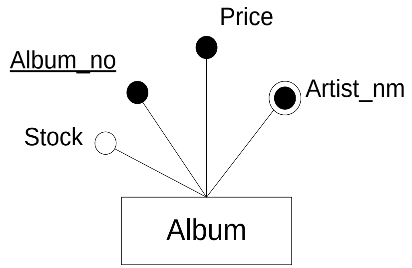
**<u>Note 1</u>: Album_no is the primary key of ALBUM**

**<u>Note 2</u>: Artist_nm is a multi-valued attribute causing a first normal form violation.**

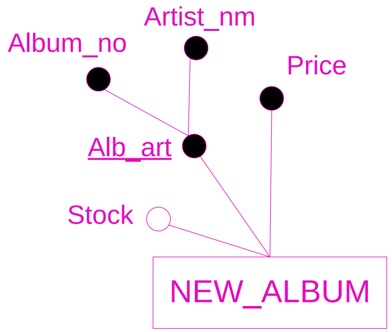
# Resolution of 1NF violation

**NEW_ALBUM**

| **Album_no** | **Artist_nm** | **Price** | **Stock** |
|---|---|---|---|
| BS123 | Britney Spears | 17.95 | 1000 |
| JT111 | Justin Timberlake | 17.95 | 1200 |
| BTL007 | John Lennon | 23.95 | |
| BTL007 | Paul McCartney | 23.95 | |
| BTL007 | George Harrison | 23.95 | |
| BTL007 | Ringo Star | 23.95 | |
| MJ100 | Michael Jackson | 17.95 | |
| JM456 | John Mayer | 16.95 | 1000 |
| JM151 | John Mayer | 16.95 | 1000 |
| MX789 | Madonna | 11.95 | 500 |
| DJM237 | John Denver | 11.95 | 2000 |
| DJM237 | Michael Jackson | 11.95 | 2000 |
| DJM237 | Madonna | 11.95 | 2000 |
| DR711 | Diana Ross | 12.95 | 1000 |
| PM137 | Paul McCartney | 19.95 | |




ERD for ALBUM




ERD for NEW_ALBUM (No MVA)

**Note 1**: Artist_nm is no longer a multi-valued attribute

**Note 2**: Album_no is no longer a superkey of NEW_ALBUM either {Album_no, Artist_nm} is the primary key of ALBUM thus rendering Artist_nm a single-valued attribute and achieving 1NF in NEW_ALBUM.

# Desirable & Undesirable FDs

- R: NEW_ALBUM (Album_no, Artist_nm, Price, Stock)
- F: fd1: {Album_no, Artist_nm} $\rightarrow$ Price
  fd2: {Album_no, Artist_nm} $\rightarrow$ Stock
  fd3: Album_no $\rightarrow$ Price;
  fd4: Album_no $\rightarrow$ Stock
- Fc = {fd3, fd4}

- Candidate Key of NEW_ALBUM: (Album_no, Artist_nm);
- Primary Key: (Album_no, Artist_nm)

- fd1 and fd2 are desirable FDs in NEW_ALBUM
- fd3 and fd4 are undesirable FDs in NEW_ALBUM **Why?**

# Second Normal Form (2NF)

**Note: There is a confusing definition in the book!**
On page 398 (and other places) the definition of 2NF says "non-prime" when it should say "non-key"!

- A relation schema R is in 2NF if every non-key attribute in R is fully functionally dependent on the primary key of R.

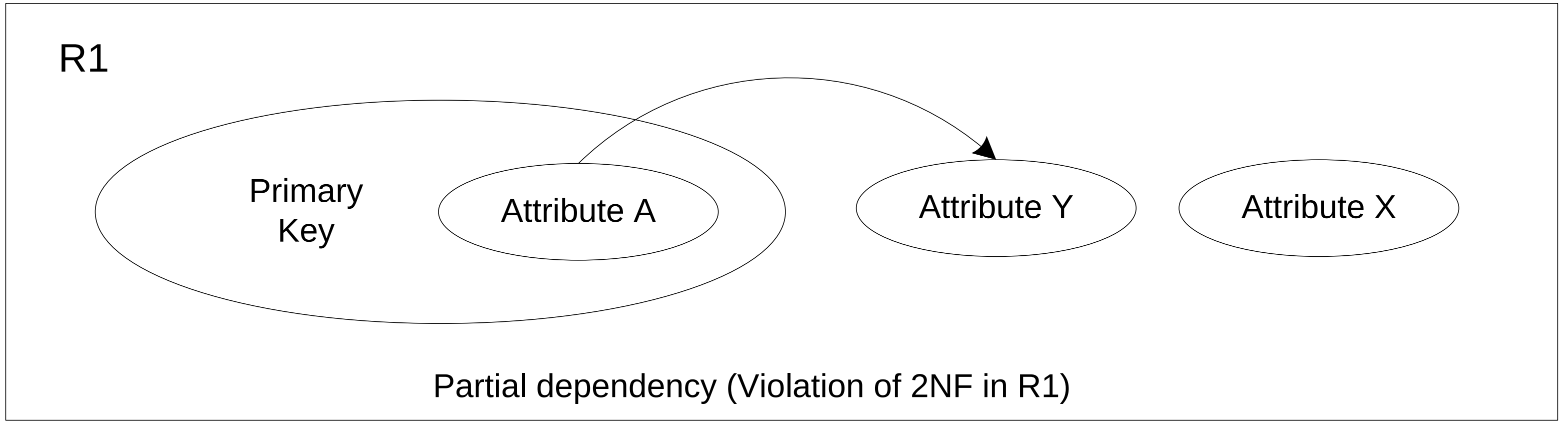
- This means a non-key attribute is not functionally dependent on a proper subset of the primary key of R.
  - Partial dependency

- The Second Normal Form (2NF) is based on a concept known as full functional dependency.

- A functional dependency of the form Z → A is a 'full functional dependency' if and only if no proper subset of Z functionally determines A.

- In other words, if Z → A and X → A, and X is a proper subset of Z, then Z does not fully functionally determine A, i.e., Z → A is not a full functional dependency; it is a partial dependency.

# Violation of 2NF




Note: The primary key and the attributes A, Y, and X can be atomic or composite. A is a prime attribute, whereas Y and X are non-key attributes. In order for a partial dependency to exist here, Attribute A must be a proper subset of the primary key of R1.

# Violation of 2NF - Example

**NEW_ALBUM**

| **Album_no** | **Artist_nm** | **Price** | **Stock** |
|---|---|---|---|
| BS123 | Britney Spears | 17.95 | 1000 |
| JT111 | Justin Timberlake | 17.95 | 1200 |
| BTL007 | John Lennon | 23.95 | |
| BTL007 | Paul McCartney | 23.95 | |
| BTL007 | George Harrison | 23.95 | |
| BTL007 | Ringo Star | 23.95 | |
| MJ100 | Michael Jackson | 17.95 | |
| JM456 | John Mayer | 16.95 | 1000 |
| JM151 | John Mayer | 16.95 | 1000 |
| MX789 | Madonna | 11.95 | 500 |
| DJM237 | John Denver | 11.95 | 2000 |
| DJM237 | Michael Jackson | 11.95 | 2000 |
| DJM237 | Madonna | 11.95 | 2000 |
| DR711 | Diana Ross | 12.95 | 1000 |
| PM137 | Paul McCartney | 19.95 | |

# 2NF Violation Explained

- F: fd1: {Album_no, Artist_nm} → Price
  fd2: {Album_no, Artist_nm} → Stock
  fd3: Album_no → Price
  fd4: Album_no → Stock

- Fc = {fd3, fd4}
- Candidate Key of NEW_ALBUM: (Album_no, Artist_nm);
- Primary Key: (Album_no, Artist_nm)
- fd3 and fd4 violate 2NF in NEW_ALBUM
  - The determinant in FD3 and FD4 is not a CK
  - Partial dependency of {Price, Stock} on one key attribute in the composite candidate key

# Resolution of 2NF Violation

- The resolution of 2NF violation is a two-step process that decomposes the target relation schema with the undesirable FDs into multiple relation schemas such that the undesirable FDs are rendered desirable.

  - Pull out the undesirable FD(s) from the target relation schema as separate relation schema(s).
  - Retain the determinant of the pulled-out relation schema as an attribute(s) in the leftover target relation schema to facilitate reconstruction of the original target relation schema.

# Resolution of 2NF Violation

ALBUM_ARTIST. Album_no $\subseteq$ ALBUM_INFO.Album_no

R1: ALBUM_INFO (Album_no, Price, Stock};

R2: ALBUM_ARTIST (Album_no, Artist_nm)

**ALBUM_INFO**

| Album_no | Price | Stock |
|---|---|---|
| BS123 | 17.95 | 1000 |
| JT111 | 17.95 | 1200 |
| BTL007 | 23.95 | |
| MJ100 | 17.95 | |
| JM456 | 16.95 | 1000 |
| JM151 | 16.95 | 1000 |
| MX789 | 11.95 | 500 |
| DJM237 | 11.95 | 2000 |
| DR711 | 12.95 | 1000 |
| PM137 | 19.95 | |

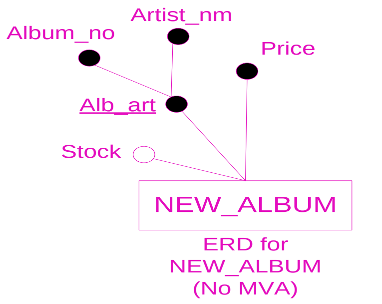
Note: Album_no → Price
Album_no → Stock

**ALBUM_ARTIST**

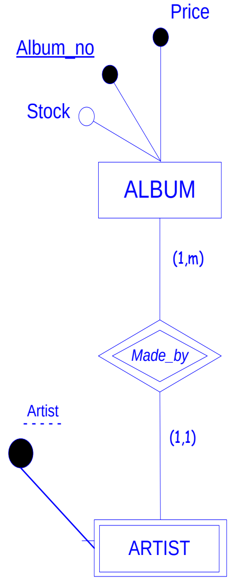
| Album_no | Artist_nm |
|---|---|
| BS123 | Britney Spears |
| JT111 | Justin Timberlake |
| BTL007 | John Lennon |
| BTL007 | Paul McCartney |
| BTL007 | George Harrison |
| BTL007 | Ringo Star |
| MJ100 | Michael Jackson |
| JM456 | John Mayer |
| JM151 | John Mayer |
| MX789 | Madonna |
| DJM237 | John Denver |
| DJM237 | Michael Jackson |
| DJM237 | Madonna |
| DR711 | Diana Ross |
| PM137 | Paul McCartney |

{Album_no, Artist_nm} → {Album_no, Artist_nm}

Note: No non-trivial FD present




Old Model




New Model

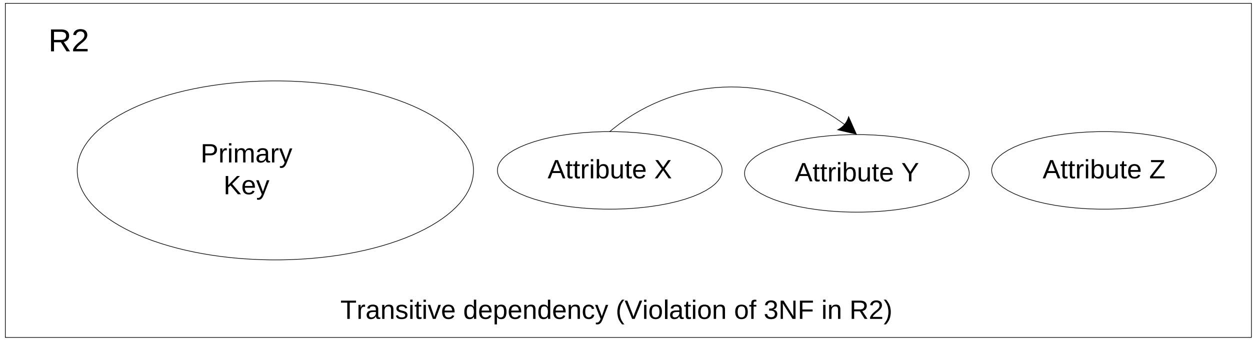
# Third Normal Form – 3NF

**Note: There is a confusing definition in the book!**
On page 402 (and other places) the definition of 3NF says "non-prime" when it should say "non-key"!

- A relation schema R is in 3NF if it is in 2NF and no non-key attribute is functionally dependent on another non-key attribute in R.
  - i.e., there are no transitive dependencies

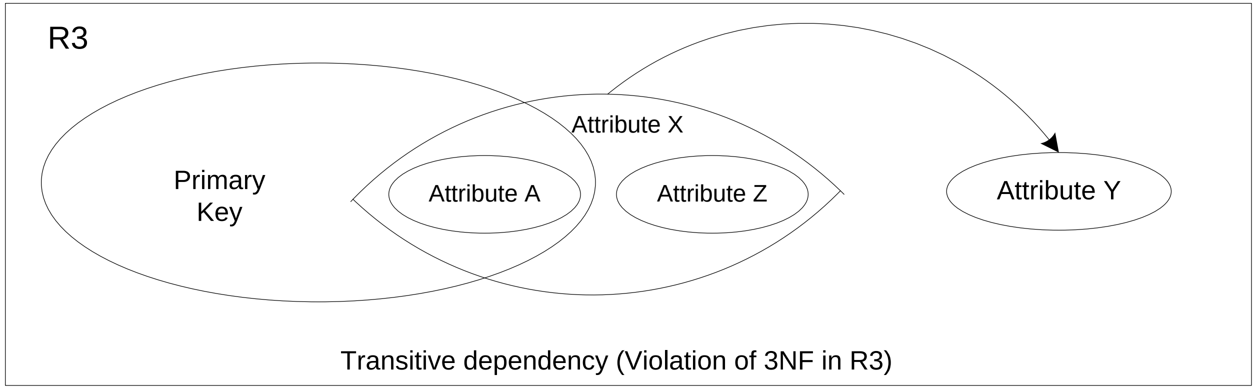
- The Third Normal Form (3NF) is based on the concept of transitive dependency.
  - Given a relation schema R (X, A, B) where
    - X, A, and B are pair-wise disjoint atomic or composite attributes,
    - X is the primary key of R, and
    - A and B are non-key attributes
    - If A $\rightarrow$ B (or B $\rightarrow$ A) in R, then B (or A) is said to be 'transitively dependent' on X, the primary key of R.

# Violation of 3NF




Note: X, Y, and Z can be atomic or composite attributes. Each is a non-key attribute.

# Another possible violation of 3NF




Note: The primary key and the attributes A, X, Y, and Z can be atomic or composite. A is a key attribute, whereas X, Y, and Z are non-key attributes. The fact that the non-key attribute Y is functionally dependent on the non-key attribute X constitutes the third normal form violation.

# 3NF Violation – An Example

FLIGHT

| Flight# | Origin | Destination | Mileage |
|---|---|---|---|
| DL507 | Seattle | Denver | 1537 |
| DL123 | Chicago | Dallas | 1058 |
| DL723 | Boston | St. Louis | 1214 |
| DL577 | Denver | Los Angeles | 1100 |
| DL5219 | Minneapolis | St. Louis | 580 |
| DL357 | Chicago | Dallas | 1058 |
| DL555 | Denver | Houston | 1100 |
| DL5237 | Cleveland | St. Louis | 580 |
| DL5271 | Chicago | Cleveland | 300 |

- Fc: [fd1, fd2, fd3], where:

  fd1: Flight# → Origin;

  fd2: Flight# → Destination;

  fd3: (Origin, Destination) → Mileage

# Resolution of 3NF Violation

- The resolution of a 3NF violation is accomplished by applying the same two-step process used earlier to resolve the 2NF violation. The two-step process is:
  - Pull out the undesirable FD(s) from the target relation schema as separate relation schema(s).
  - Retain the determinant of the pulled-out relation schema as an attribute(s) in the leftover target relation schema to facilitate reconstruction of the original target relation schema.

# Resolution of 3NF Violation

FLIGHT.{Origin, Destination} $\subseteq$ DISTANCE.{Origin, Destination}

R1: FLIGHT (Flight#, Origin, Destination)

R2: DISTANCE (Origin, Destination, Mileage}

FLIGHT

| Flight# | Origin | Destination |
|---|---|---|
| DL507 | Seattle | Denver |
| DL123 | Chicago | Dallas |
| DL723 | Boston | St. Louis |
| DL577 | Denver | Los Angeles |
| DL5219 | Minneapolis | St. Louis |
| DL357 | Chicago | Dallas |
| DL555 | Denver | Houston |
| DL5237 | Cleveland | St. Louis |
| DL5271 | Chicago | Cleveland |

DISTANCE

| Origin | Destination | Mileage |
|---|---|---|
| Seattle | Denver | 1537 |
| Chicago | Dallas | 1058 |
| Boston | St. Louis | 1214 |
| Denver | Los Angeles | 1100 |
| Minneapolis | St. Louis | 580 |
| Denver | Houston | 1100 |
| Cleveland | St. Louis | 580 |
| Chicago | Cleveland | 300 |

Note: fd1: Flight# → Origin
fd2: Flight# → Destination

Note: fd3: (Origin, Destination) →Mileage

# A set of heuristics you can use
(thanks to Dr. Scamell)

- A relation is in 1NF if there are no multivalued attributes.

- A relation is in 2NF if it is in 1NF and there are no <u>partial dependencies</u> – that is, it satisfies at least one of the following criteria:
  - The primary key consists of only one attribute
  - There are no non-key attributes
  - No non-key attribute is functionally dependent on part of the primary key (partial dependency)

- A relation is in 3NF if it is in 2NF and there are no <u>transitive dependencies</u>- that is, no relationships between non-key attributes

# Boyce-Codd Normal Form (BCNF)

- While 3NF removes most data redundancy issues, there is still potential for problems, particularly when:
  - A relation schema has at least two candidate keys,
  - Any two candidate keys are composite attributes, and
  - There is an attribute overlap between the two candidate keys

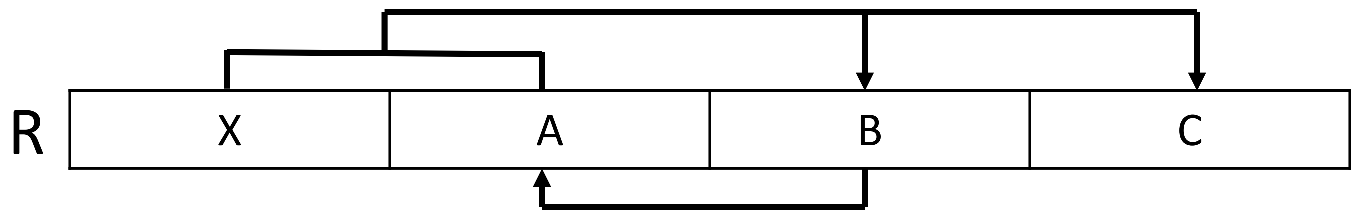
- BCNF to the rescue!
  - Kind of… there is still a trade off between 3NF and BCNF

- A relation schema R is in BCNF if for every non-trivial FD in R, **the determinant is a superkey of R**
  - By this definition of BCNF, violation of 2NF or 3NF also implies violation of BCNF

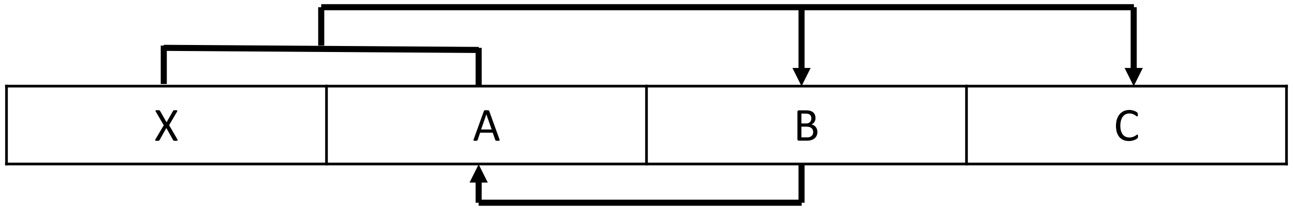
# Boyce-Codd Normal Form (BCNF)

- Consider the following relation and FDs:
  - R(X,A,B,C)
  - FD1: {X, A} → B
  - FD2: {X, A} → C
  - FD3: B → A

- What is/are the candidate key(s)?
  - {X,A} and {X,B}



# Boyce-Codd Normal Form (BCNF)



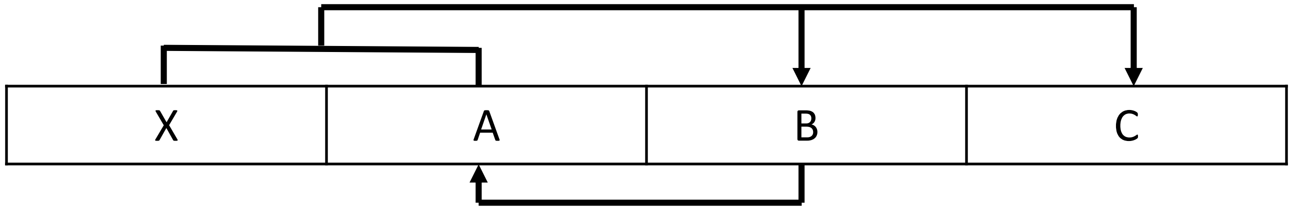

- {XA} → {BC}
- B → A

- Candidate keys: {XA} and {XB}

- What NF Is this?

- Is there a 2NF violation? No!
  - R is in 2NF if it is in 1NF and no **<u>non-key</u>** attribute is FD by **<u>part</u>** of a candidate key
  - No violation! B determines A, but A is a key attribute, so not a 2NF violation. This is in at least 2NF.

# Boyce-Codd Normal Form (BCNF)



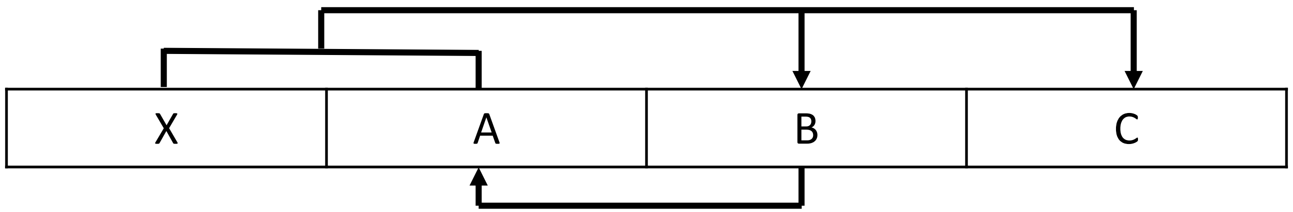

- {XA} → {BC}
- B → A

- Candidate keys: {XA} and {XB}

- What NF Is this?

- Is there a 3NF violation? No!
  - R is in 3NF if it is in 2NF and no **non-key** attribute is FD on another non-key attribute in R
  - No violation! B determines A, but A is a key attribute, so not a 3NF violation. This is in at least 3NF.

# Boyce-Codd Normal Form (BCNF)




- {XA} → {BC}
- B → A

- Candidate keys: {XA} and {XB}

- What NF Is this?

- Is there a BCNF violation? YES!
    - R is in BCNF if for every non-trivial FD in R, **<u>the determinant is a superkey of R</u>**
    - Yes, there is a violation! B (the determinant in B→A) is not a superkey, so we are not in BCNF.

# Resolution of a BCNF Violation

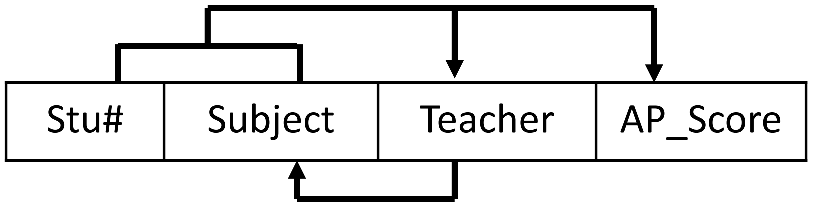
- The resolution of a BCNF violation is accomplished by applying the same two-step process used earlier to resolve the 2NF & 3NF violations:
  - Pull out the undesirable FD(s) from the target relation schema as separate relation schema(s).
  - Retain the determinant of the pulled-out relation schema as an attribute(s) in the leftover target relation schema to facilitate reconstruction of the original target relation schema.

# Let’s work through a BCNF violation

- Stu_Sub(Stu#, Subject, Teacher, Ap_score)
  - FD1: {Stu#, Subject} → Teacher
  - FD2: {Stu#, Subject} → Ap_score
  - FD3: Teacher → Subject
- We can also infer:
  - FD4: {Stu#, Teacher} → Subject
  - FD5: {Stu#, Teacher} → Ap_score

STU_SUB

| Stu# | Subject | Teacher | Ap_score |
|---|---|---|---|
| IH123 | Chemistry | Raturi | 4 |
| IH123 | English | Stephan | 4 |
| IH235 | History | Walker | 5 |
| IH357 | English | Campbell | 4 |
| IH571 | Chemistry | Raturi | 3 |
| IH235 | English | Campbell | 4 |




- Is there a 3NF violation?
  - No! We are in 3NF
- Is there data redundancy?
  - Yes! We are repeating values for Subject and Teacher
- Is there a BCNF violation?
  - Yes! In FD3, Teacher is not a superkey

# BCNF Violation: Modification Anomalies

STU_SUB

| Stu# | Subject | Teacher | Ap_score |
|---|---|---|---|
| IH123 | Chemistry | Raturi | 4 |
| IH123 | English | Stephan | 4 |
| IH235 | History | Walker | 5 |
| IH357 | English | Campbell | 4 |
| IH571 | Chemistry | Raturi | 3 |
| IH235 | English | Campbell | 4 |

- Suppose we want to add a new Teacher for a Subject (e.g., Teacher: ‘Salter’, Subject: ‘English’)
  - Insertion Anomaly: Not possible without a Stu# associated with this insertion since Stu# is part of the primary key of STU_SUB

- Suppose we want to replace Campbell with Smith
  - Update Anomaly: Multiple tuples require update and failure to update even one changes the semantics of the scenario

- Suppose we fire Raturi
  - Deletion Anomaly: we lose the AP scores for IH123 and IH571

# Resolution of BCNF Violation Demonstrated

- Stu_Sub (Stu#, Subject, Teacher, Ap_score)
  - FD1: {Stu#, Subject} → Teacher
  - FD2: {Stu#, Subject} → Ap_score
  - FD3: Teacher → Subject

- What are the candidate Key(s) of Stu_Sub?
  - {Stu#, Subject}, {Stu#, Teacher}
  - Let’s use {<u>Stu#, Teacher</u>} as PK for now

# Resolution of BCNF Violation Demonstrated

- Possible resolution:
  - R1: Teach_Sub (<u>Teacher</u>, Subject)
  - R2: Stu_AP (<u>Stu#, Teacher</u>, Ap_Score

**TEACH_SUB**

| <u>Teacher</u> | Subject |
|---|---|
| Raturi | Chemistry |
| Stephan | English |
| Walker | History |
| Campbell | English |

Teacher → Subject

**STU_AP**

| <u>Stu#</u> | <u>Teacher</u> | Ap_score |
|---|---|---|
| IH123 | Raturi | 4 |
| IH123 | Stephan | 4 |
| IH235 | Walker | 5 |
| IH357 | Campbell | 4 |
| IH571 | Raturi | 3 |
| IH235 | Campbell | 4 |

(Stu#, Teacher) → Ap_score

# An alternate (problematic) approach

- STU_SUB (Stu#, Subject, Teacher, Ap_score)
  - FD1: {Stu#, Subject} → Teacher;
  - FD2: {Stu#, Subject} → Ap_score;
  - FD3: Teacher → Subject

- What are the candidate Key(s) of Stu_Sub?
  - {Stu#, Subject}, {Stu#, Teacher}
  - Let's use {<u>Stu#, Subject</u>} as PK this time

# A problematic resolution of BCNF Violation

- A second (bad) possible resolution:
  - R1: Teach_Sub (Teacher, Subject)
  - R2: Stu_AP (Stu#, Subject, Ap_Score

TEACH_SUB

| Teacher | Subject |
|---|---|
| Raturi | Chemistry |
| Stephan | English |
| Walker | History |
| Campbell | English |

Teacher → Subject

STU_AP

| Stu# | Subject | Ap_score |
|---|---|---|
| IH123 | Chemistry | 4 |
| IH123 | English | 4 |
| IH235 | History | 5 |
| IH357 | English | 4 |
| IH571 | Chemistry | 3 |
| IH235 | English | 4 |

(Stu#, Subject) → Ap_score

- Meets BCNF, but DOES NOT preserve all functional dependencies – we no longer know who taught IH123, Ih357, and IH235 English!

# The problem with BCNF

- In many cases BCNF will clear up lingering redundancy from 3NF, however…

- If we are forced to choose between BCNF without preserving dependencies and 3NF with preserving dependencies, it is generally preferable to opt for the latter (3NF)
    - After all, if one can't test for dependency preservation efficiently, one either pays a high penalty in system performance or risks the integrity of the data in the database.

- Neither is an attractive alternative, but the limited amount of redundancy allowed under 3NF is regarded as the lesser of the two evils.

# Modules 8.1 – 8.2

## Normalization

- Normalization – 1NF, 2NF, 3NF, BCNF

# Break

10 Minutes

# Module 8.3 – 8.4

## Demo / Examples

- A comprehensive treatment of normalization

# First Normal Form (1NF)

- First normal form (1NF) imposes conditions so that a base relation that is physically stored as a file does not contain records with a variable number of fields.
  - This is accomplished by prohibiting multi-valued and composite attributes in a relation schema.

- Such a constraint, in effect, prevents relations from containing other relations.

- In essence, 1NF, by definition, requires that the domain of an attribute must include only atomic values and that the value of an attribute in a relation’s tuple must be a single value from the domain of that attribute.

# Second Normal Form (2NF)

- A relation schema R is in 2NF if every non-key attribute in R is fully functionally dependent on the primary key of R.

- This means a non-key attribute is not functionally dependent on a proper subset of the primary key of R.
  - Partial dependency

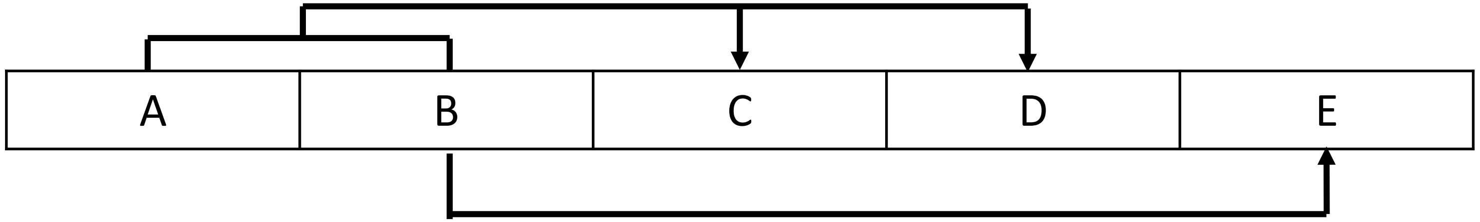
- The Second Normal Form (2NF) is based on a concept known as full functional dependency.

- A functional dependency of the form Z $\rightarrow$ A is a ‘full functional dependency’ if and only if no proper subset of Z functionally determines A.

- In other words, if Z $\rightarrow$ A and X $\rightarrow$ A, and X is a proper subset of Z, then Z does not fully functionally determine A, i.e., Z $\rightarrow$ A is not a full functional dependency; it is a partial dependency.

# Review: Violation of 2NF

- {A,B} is the primary key
- {AB} → {CD} is fine
    - C and D are FULLY FUNCTIONALLY DEPENDENT on the PK
- B → E is a partial dependency
    - A non-prime attribute (E) is being determined by PART of the primary key




- The FD of B→E should be moved to a new relation
    - The determinant (B) will be left in the source relation

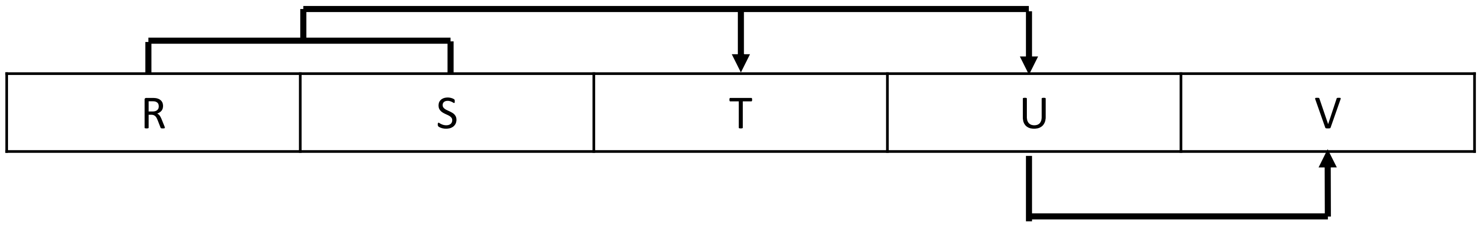
# Third Normal Form – 3NF

- A relation schema R is in 3NF if it is in 2NF and no non-key attribute is functionally dependent on another non-key attribute in R.
  - i.e., there are no transitive dependencies

- The Third Normal Form (3NF) is based on the concept of transitive dependency.
  - Given a relation schema R (X, A, B) where
    - X, A, and B are pair-wise disjoint atomic or composite attributes,
    - X is the primary key of R, and
    - A and B are non-prime attributes
    - If A → B (or B → A) in R, then B (or A) is said to be 'transitively dependent' on X, the primary key of R.

# Review: Violation of 3NF

- {R,S} is the primary key
- {RS} → {TU} is fine
  - T and U are FULLY FUNCTIONALLY DEPENDENT on the PK
- U → V is a transitive dependency
  - A non-key attribute (V) is being determined by another non-key attribute (U)




- The FD of U→V should be moved to a new relation
  - The determinant (U) will be left in the source relation

# A simple algorithm for Normalization

- This is what we did last time

| Store | Product | Price | Quantity | Location | Discount | Sq_ft | Manager |
|---|---|---|---|---|---|---|---|
| 15 | Refrigerator | 1850 | 120 | Houston | 5% | 2300 | Metzger |
| 15 | Dishwasher | 600 | 150 | Houston | 5% | 2300 | Metzger |
| 13 | Dishwasher | 600 | 180 | Tulsa | 10% | 1700 | Metzger |
| 14 | Refrigerator | 1850 | 150 | Tulsa | 5% | 1900 | Schott |
| 14 | Television | 1400 | 280 | Tulsa | 10% | 1900 | Schott |
| 14 | Humidifier | 55 | 30 | Tulsa | | 1900 | Schott |
| 17 | Television | 1400 | 10 | Memphis | | 2300 | Creech |
| 17 | Vacuum Cleaner | 300 | 150 | Memphis | 5% | 2300 | Creech |
| 17 | Dishwasher | 600 | 150 | Memphis | 5% | 2300 | Creech |
| 11 | Computer | | 180 | Houston | 10% | 2300 | Creech |
| 11 | Refrigerator | 1850 | 120 | Houston | 5% | 2300 | Creech |
| 11 | Lawn Mower | 300 | | Houston | | 2300 | Creech |

FD1: Store → {Location, Sq_Ft, Manager}
FD2: {Store, Product} → Quantity
FD3: Product → Price
FD4: Quantity → Discount

FD1: Store → {Location, Sq_Ft, Manager}

STORE

| Store | Location | Sq_ft | Manager |
|---|---|---|---|
| 15 | Houston | 2300 | Metzger |
| 13 | Tulsa | 1700 | Metzger |
| 14 | Tulsa | 1900 | Schott |
| 17 | Memphis | 2300 | Creech |
| 11 | Houston | 2300 | Creech |

FD3: Product → Price

PRODUCT

| Product | Price |
|---|---|
| Refrigerator | 1850 |
| Dishwasher | 600 |
| Television | 1400 |
| Humidifier | 55 |
| Vacuum Cleaner | 300 |
| Computer | |
| Lawn Mower | 300 |
| Washing Machine | 750 |

INVENTORY

| Store | Product | Quantity |
|---|---|---|
| 15 | Refrigerator | 120 |
| 15 | Dishwasher | 150 |
| 13 | Dishwasher | 180 |
| 14 | Refrigerator | 150 |
| 14 | Television | 280 |
| 14 | Humidifier | 30 |
| 17 | Television | 10 |
| 17 | Vacuum Cleaner | 150 |
| 17 | Dishwasher | 150 |
| 11 | Computer | 180 |
| 11 | Refrigerator | 120 |
| 11 | Lawn Mower | |

FD2: {Store, Product} → Quantity

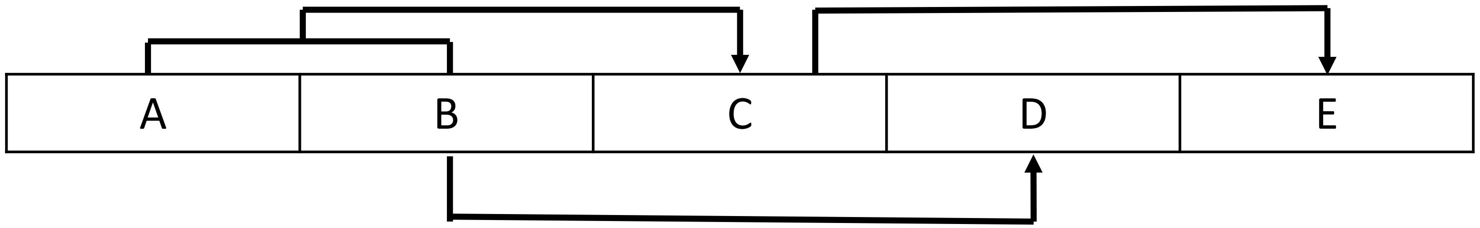
DISC_STRUCTURE

| Quantity | Discount |
|---|---|
| 120 | 5% |
| 150 | 5% |
| 180 | 10% |
| 280 | 10% |
| 30 | |
| 10 | |

FD4: Quantity → Discount

# Simplest all inclusive example I can come up with

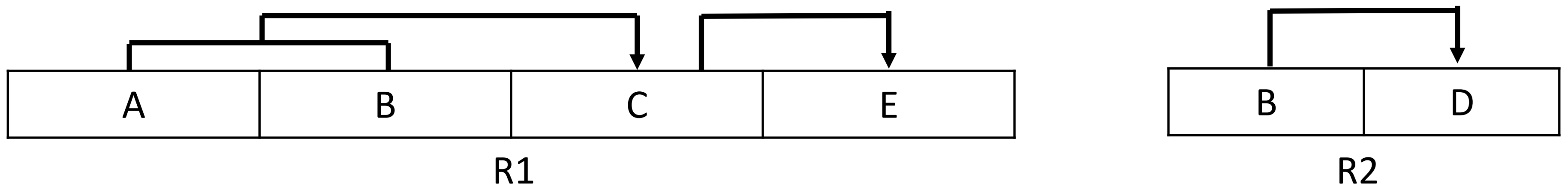
- R1(A, B, C, D, E)
  - FD1: {A,B} → C
  - FD2: B → D
  - FD3: C → E




- {A,B} is the only CK, and is thus the PK
  - You can figure this out via decomposition or synthesis
- FD2 is a partial dependency, violates 2NF
- FD3 is a transitive dependency, violates 3NF

# Simplest all inclusive example I can come up with

- To get to 2NF, decompose R into two relations:
  - R1(A,B,C,E)
    - FD1: {A,B} → C
    - FD2: C → E
  - R2(B, D)
    - FD3: B → D



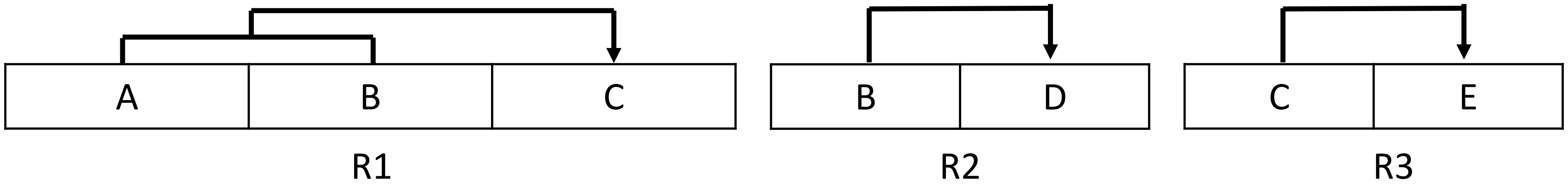

- R1 has no partial dependencies, but still has a transitive dependency (FD2), thus is in 2NF
- R2 has no partial nor transitive dependencies, thus is in 3NF

# Simplest all inclusive example I can come up with

- To get to 3NF, decompose into three relations:
  - R1(A,B,C)
    - FD1: {A,B} → C

  - R2(B, D)
    - FD3: B → D

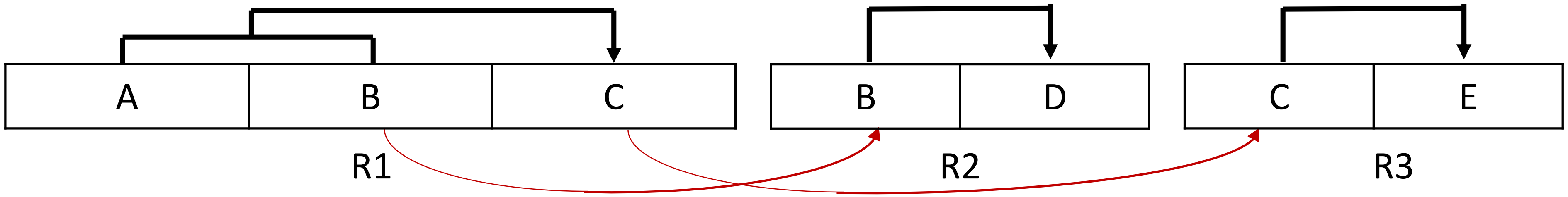
  - R3(C, E)
    - FD2: C → E




R1, R2, and R3 are all in 3NF now – Yipee!

# Simplest all inclusive example I can come up with

- Note that all attributes and dependencies have been preserved, and we can recreate the original relation be joining the relations back together using the attributes that overlap in the resulting relations
  - B is now a FK in R1 and a CK in R2
  - C is now a FK in R1 and a CK in R3



# Identifying candidate keys

- We can derive the candidate keys in the universal relation schema (URS; the set of attributes in the FDs of F) using two methods:
  - Synthesis
  - Decomposition

# Let’s work though a few

- I suggest writing these out for your notes and for practice!

# Let’s work though a few (#1)

- R(A, B, C, D, E)
  - FD1: A → B
  - FD2: {B,C} → E
  - FD3: (E,D} → A

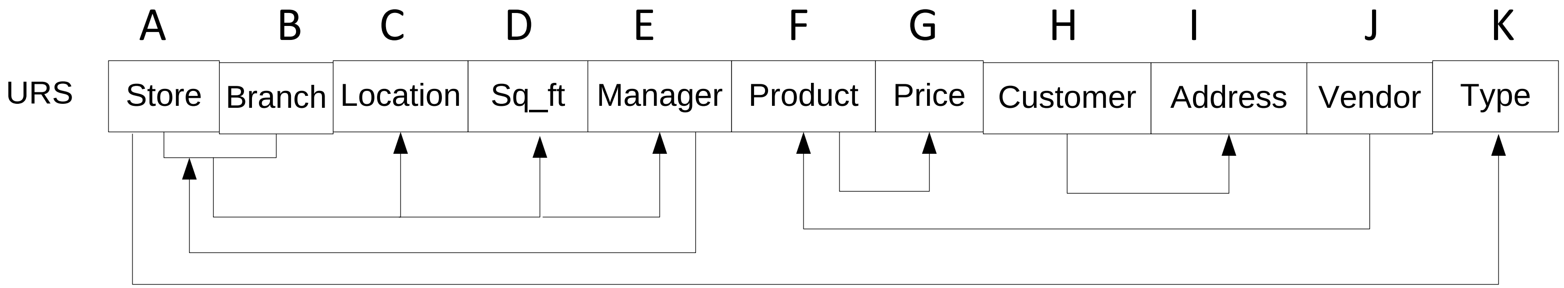
- Does {A,B,C,D,E} → {A,B,C,D,E}? Yes
- Does {A,B,C,D} → E? Yes
- Does {A,B,C} → D? **No**
- Does {A,B,D} → C? **No**
- Does {A,C,D} → B? Yes
- Does {C,D} → A? **No**
- …So {A,C,D} → {A,B,C,D,E}
- Any others?
- {E,D} → A, so through pseudotransitivity, {E,C,D} is a CK
- {B,C} → E, so through pseudotransitivity, {B,C,D} is a CK

# Let's work though a few (#2)

- Given R (A, B, C, D, E, X, Y) and:
  - FD1: A → B
  - FD2: A → C
  - FD3: B → C
  - FD4: E → A
  - FD5: {A, D} → {X, E}

- CK: {A,D,Y} and {E,D,Y}

# Let's work though a few (#3)




Given F [fd1, fd2, fd3, fd4, fd5, fd6, fd7, fd8] that prevails over URS where

fd1: {Store, Branch} → Location;

fd2: Customer → Address;

fd3: Vendor → Product;

fd4: {Store, Branch} → Sq_ft;

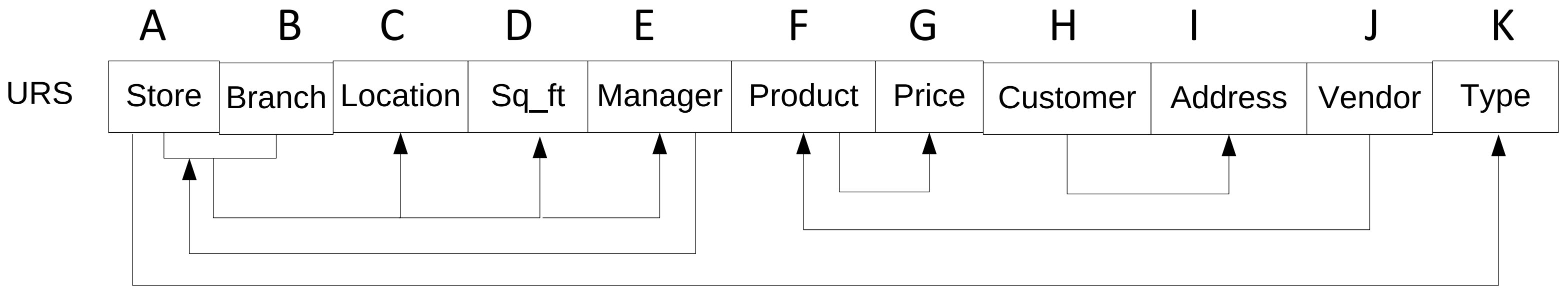
fd5: Product → Price;

fd6: {Store, Branch} → Manager;

fd7: Manager → {Store, Branch};

fd8: Store → Type

# Let’s work though a few (#3)




Given F [fd1, fd2, fd3, fd4, fd5, fd6, fd7, fd8] that prevails over URS where

FD1: {A, B} → C;   FD2: H→ I;

FD3: J → F;   FD4: {A,B} → D;

FD5: F → G;   FD6: {A, B} → E;

FD7: E → {A,B};   FD8: A → K

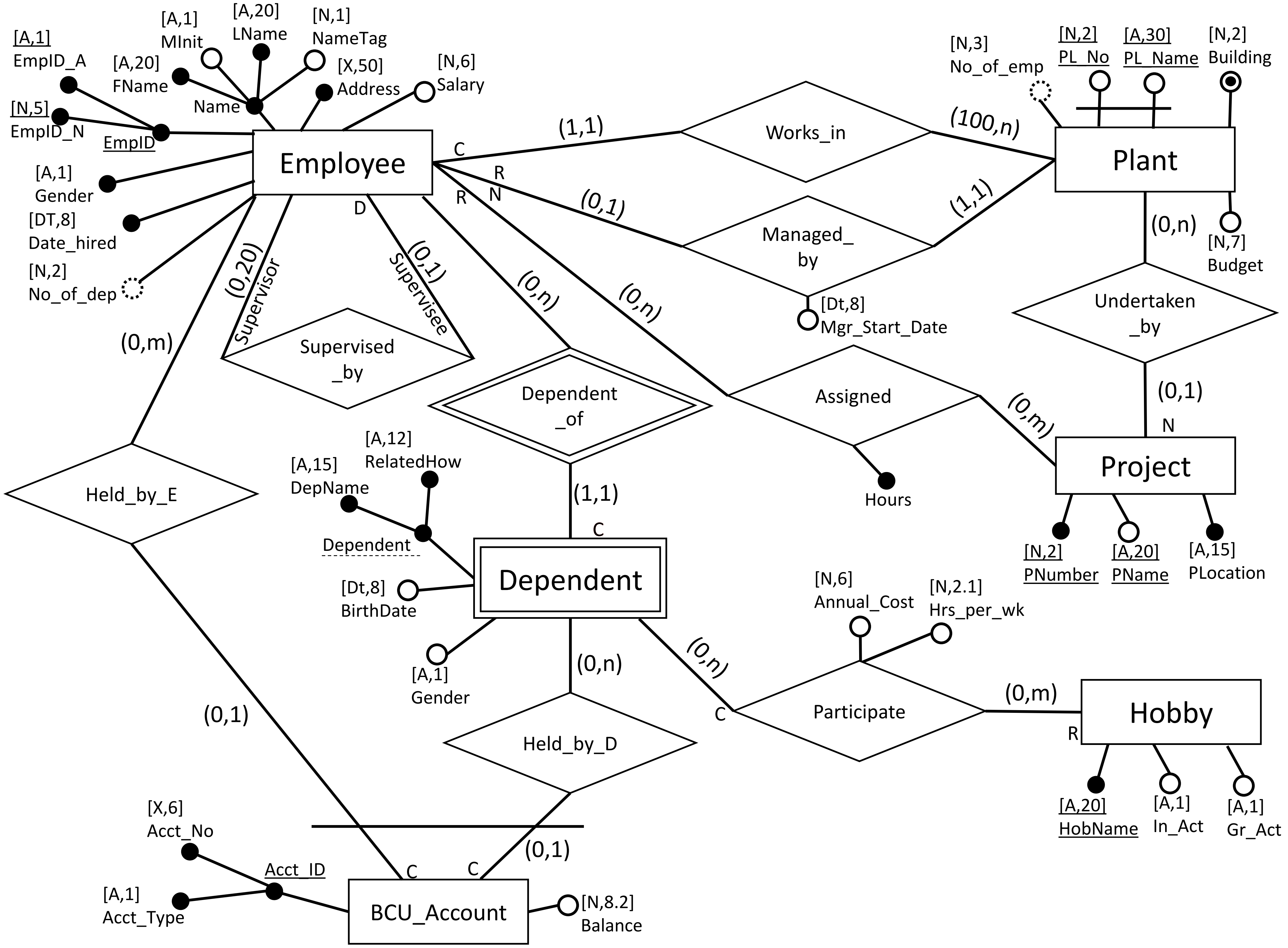
What are the candidate keys?

CK: {A, B, H, J} and {E, H, J}

Also known as:
{Store, Branch, Customer, Vendor} and
{Manager, Customer, Vendor}

# A long, long time ago, we created this model…

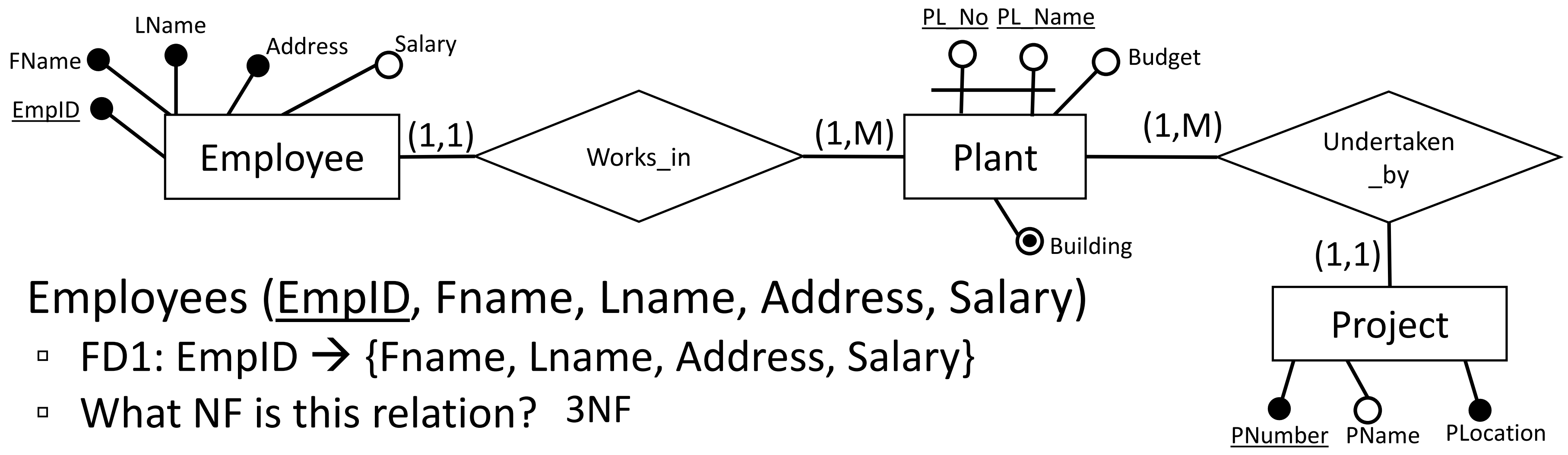
The Bearcat



[A,1]
EmpID_A
[N,5]
EmpID_N
EmpID
[A,20]
FName
[A,1]
MInit
[A,20]
LName
[N,1]
NameTag
Name
[X,50]
Address
[N,6]
Salary
[A,1]
Gender
[DT,8]
Date_hired
[N,2]
No_of_dep
Employee
C
R
R
N
D
(1,1)
Works_in
(100,n)
[N,3]
No_of_emp
[N,2]
PL_No
[A,30]
PL_Name
[N,2]
Building
Plant
[N,7]
Budget
(0,1)
Managed_
by
(1,1)
[Dt,8]
Mgr_Start_Date
(0,n)
Undertaken
_by
(0,1)
N
(0,20)
Supervisor
(0,1)
Supervisee
Supervised
_by
(0,m)
(0,n)
(0,n)
Dependent
_of
(1,1)
Assigned
(0,m)
Hours
Project
[N,2]
PNumber
[A,20]
PName
[A,15]
PLocation
Held_by_E
[A,12]
RelatedHow
[A,15]
DepName
Dependent
C
Dependent
[Dt,8]
BirthDate
[A,1]
Gender
(0,n)
(0,n)
[N,6]
Annual_Cost
[N,2.1]
Hrs_per_wk
C
Participate
(0,m)
Hobby
R
[A,20]
HobName
[A,1]
In_Act
[A,1]
Gr_Act
(0,1)
Held_by_D
(0,1)
[X,6]
Acct_No
Acct_ID
[A,1]
Acct_Type
C
C
BCU_Account
[N,8.2]
Balance

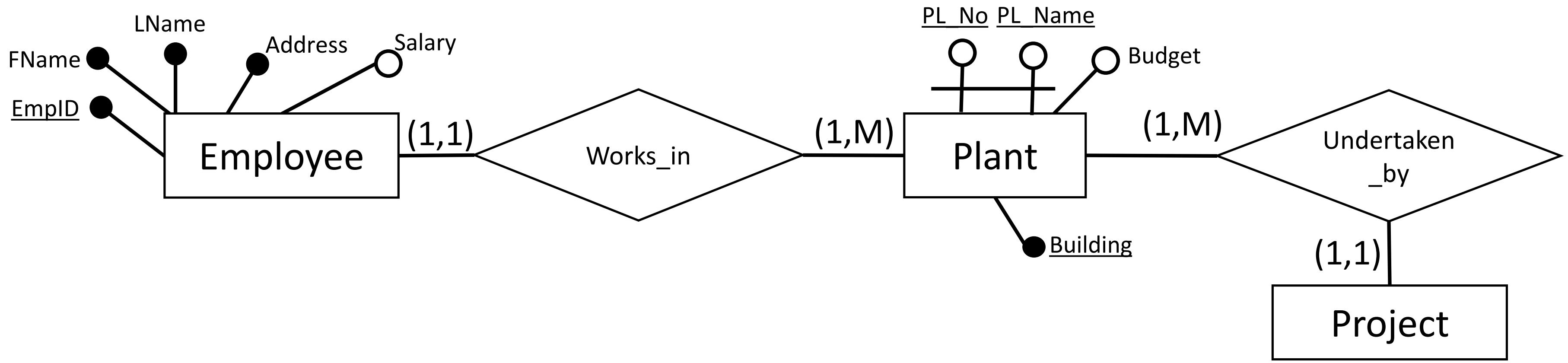
# Let's take just a small piece of it…

- …and simplify it just a little more
    - Get rid of a few attributes, just to save some time




- Employees (<u>EmpID</u>, Fname, Lname, Address, Salary)
    - FD1: EmpID → {Fname, Lname, Address, Salary}
    - What NF is this relation? 3NF
- Project (<u>Pnumber</u>, Pname, Plocation)
    - FD1: Pnumber → {Pname, Plocation}
    - What NF is this relation? 3NF
- Plant (<u>PLNo</u>, <u>PLName</u>, Budget, Building)
    - FD1: {PLNo, PLName} → {Budget,Building}
    - What NF is this relation? Not even 1NF, we have a multi-valued attribute!

# Let’s take just a small piece of it…




- To resolve the multi-valued issue, we can put each value for building in its own tuple, but then we have to make building part of the candidate key…
  - Employees (<u>EmpID</u>, Fname, Lname, Address, Salary)
  - Project (<u>Pnumber</u>, Pname, Plocation)
  - Plant (<u>PLNo</u>, <u>PLName</u>, <u>Building</u>, Budget)

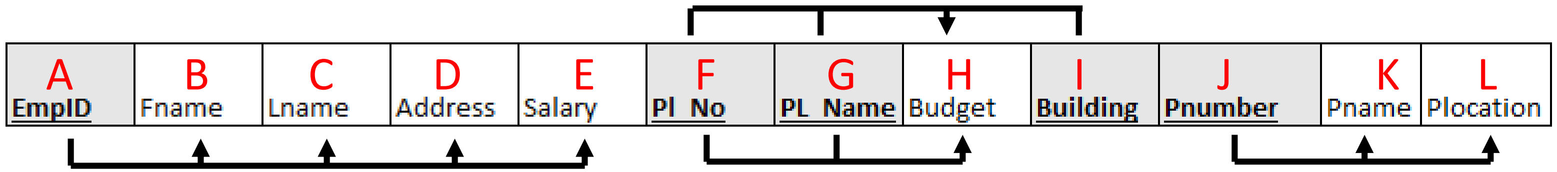
# What if all this were in one big relation?

| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EmpID | Fname | Lname | Address | Salary | Pl_No | PL_Name | Budget | Building | Pnumber | Pname | Plocation |

- To save some writing, let's use letters…
- A → {B, C, D, E}
- {F, G} → {H}
- J → {K, L}
- {F, G, I} → {H}

- Find a candidate key:
  - {A, F, G, I, J} → {A, B, C, D, E, F, G, H, I, J, K, L}
  - {Empid, PL_No, PL_Name, Building, Pnumber}

# What if all this were in one big relation?




- What NF is this big relation in?
  - 1NF – LOTS of partial dependencies

- Why is this a problem?
  - Let's see!

# Insertion Anomaly

| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **EmpID** | Fname | Lname | Address | Salary | **Pl_No** | **PL_Name** | Budget | **Building** | **Pnumber** | Pname | Plocation |
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 102 | Jane | Doe | 55 South | 55000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 103 | John | Jones | 10 N Codd | 42000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 104 | Sally | Harper | 15 W 23rd | 60000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 106 | Joe | Smith | 10 S Park | 57000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 107 | Jim | Wilson | 99 Down s | 72000 | 3 | Gamma | 7000000 | A | 14 | Cars | Atlanta |

- If we want to add a new project called Phones, can we just do this?

| **EmpID** | Fname | Lname | Address | Salary | **Pl_No** | **PL_Name** | Budget | **Building** | **Pnumber** | Pname | Plocation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 102 | Jane | Doe | 55 South | 55000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 103 | John | Jones | 10 N Codd | 42000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 104 | Sally | Harper | 15 W 23rd | 60000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 106 | Joe | Smith | 10 S Park | 57000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 107 | Jim | Wilson | 99 Down s | 72000 | 3 | Gamma | 7000000 | A | 14 | Cars | Atlanta |
| | | | | | | | | | 15 | Phones | Houston |

- No! This would have NULL values for key attributes

# Insertion Anomaly

| EmpID | Fname | Lname | Address | Salary | Pl_No | PL_Name | Budget | Building | Pnumber | Pname | Plocation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 102 | Jane | Doe | 55 South | 55000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 103 | John | Jones | 10 N Codd | 42000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 104 | Sally | Harper | 15 W 23rd | 60000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 106 | Joe | Smith | 10 S Park | 57000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 107 | Jim | Wilson | 99 Down s | 72000 | 3 | Gamma | 7000000 | A | 14 | Cars | Atlanta |
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 15 | Phones | Houston |

- We would have to “randomly” associate an employee with this project in order to insert the record
- Similarly, if we want to add a new employee, a plant and project has to be entered as well….
  - And we might incorrectly enter the budget for the plant, the name of the project, etc…

# Deletion Anomaly

| EmpID | Fname | Lname | Address | Salary | Pl_No | PL_Name | Budget | Building | Pnumber | Pname | Plocation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 102 | Jane | Doe | 55 South | 55000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 103 | John | Jones | 10 N Codd | 42000 | 1 | Alpha | 5000000 | A | 10 | Robot | Houston |
| 104 | Sally | Harper | 15 W 23rd | 60000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 106 | Joe | Smith | 10 S Park | 57000 | 2 | Beta | 2000000 | A | 13 | Guns | Tucson |
| 107 | Jim | Wilson | 99 Down s | 72000 | 3 | Gamma | 7000000 | A | 14 | Cars | Atlanta |
| 101 | Bill | Smith | 123 Main | 50000 | 1 | Alpha | 5000000 | A | 15 | Phones | Houston |

- What if we delete Jim Wilson?
  - We lose all our data about Plant 3 and Project 14 – not good!

# How to decompose this big relation?

| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EmpID | Fname | Lname | Address | Salary | Pl_No | PL_Name | Budget | Building | Pnumber | Pname | Plocation |

FD1: A → {B, C, D, E}
FD2: {F, G} → {H}
FD3: J → {K, L}
FD4: {F, G, I} → {H}

## A first attempt at resolving partial dependencies....

Employees

| A | B | C | D | E |
|---|---|---|---|---|
| EmpID | Fname | Lname | Address | Salary |

$A \rightarrow \{B, C, D, E\}$

Plants

| F | G | H | I |
|---|---|---|---|
| Pl_No | PL_Name | Budget | Building |

$\{F,G\} \rightarrow H$

$\{F, G, I\} \rightarrow H$

Projects

| J | K | L |
|---|---|---|
| Pnumber | Pname | Plocation |

$J \rightarrow \{K, L\}$

# All Normalized to 3NF

- No partial dependencies
- No transitive dependencies

Employees

| A | B | C | D | E |
|---|---|---|---|---|
| **EmpID** | Fname | Lname | Address | Salary |

$A \rightarrow \{B, C, D, E\}$

Plants

| F | G | H |
|---|---|---|
| **Pl No** | **PL Name** | Budget |

$\{F,G\} \rightarrow H$

Projects

| J | K | L |
|---|---|---|
| **Pnumber** | Pname | Plocation |

$J \rightarrow \{K, L\}$

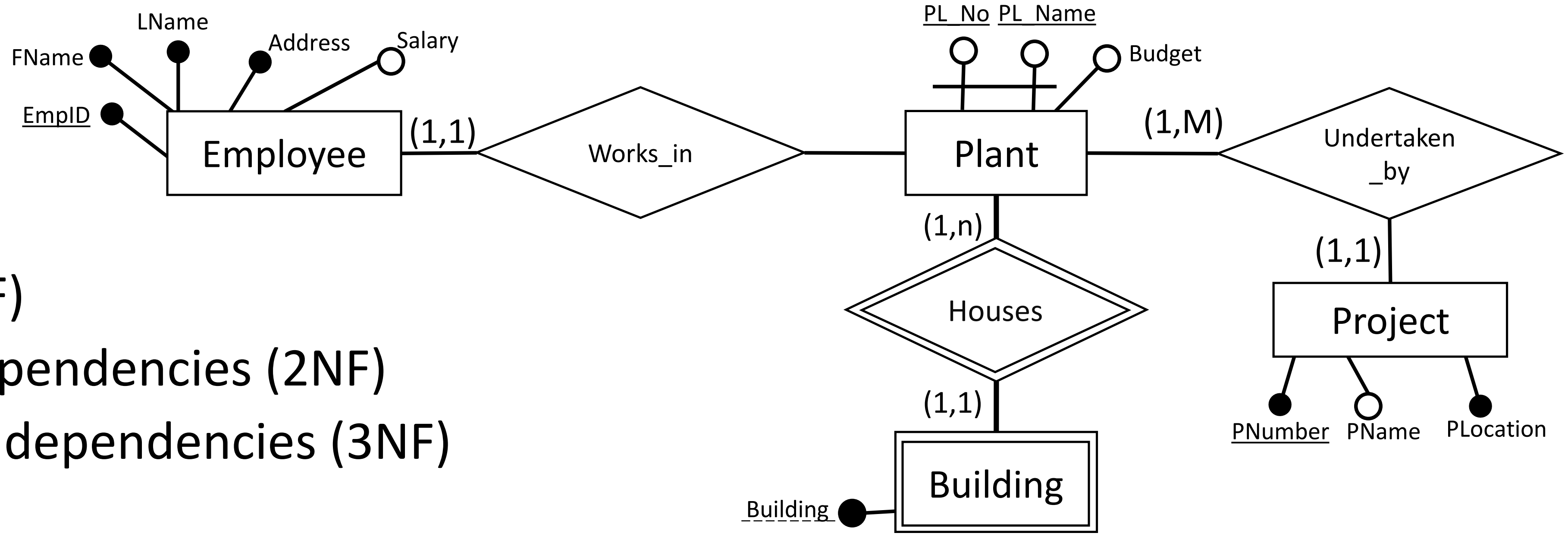
Buildings

| F | G | I |
|---|---|---|
| **Pl No** | **PL Name** | **Building** |

No non-trivial FD

$\{F,G,I\} \rightarrow \{F,G,I\}$

# Now all relations are in 3NF




- No MVA (1NF)
- No partial dependencies (2NF)
- No transitive dependencies (3NF)

- Employees (<u>EmpID</u>, Fname, Lname, Address, Salary)
- Project (<u>Pnumber</u>, Pname, Plocation)
- Plant (<u>PLNo</u>, <u>PLName</u>, Budget)
- Building <u>(Building</u>, <u>PLNo</u>, <u>PLName</u>)

# Review: Normalization

- What is the criteria for 1NF?
  - No multivalued or composite attributes
  - In order to be a relation at all, data must be in at least 1NF

- What is the criteria for 2NF?
  - 1NF + no partial dependencies

- What is the criteria for 3NF?
  - 2NF + no transitive dependencies

# Homework 4

- Very similar to what we did in class tonight
- Provided with relations and a set of FDs
  - Will need to identify candidate keys using both synthesis and decomposition
  - Will need to identify the normal form of the relation
  - Will need to normalize up to 3NF

# Module 8.3 – 8.4

## Demo / Examples

- A comprehensive treatment of normalization

# How are we on time?

- I recently realized you will need some of the following commands for part 7 of the SQL project…
  - If we have time today we’ll cover these
  - If not, we’ll cover them next week, but you have them in your notes if you want to push ahead on the SQL project!

# Module 13.1

## String manipulation SQL functions

- Concatenation
- SUBSTRING
- LENGTH
- TRIM (RTRIM / LTRIM)
- PAD (RPAD / LPAD)
- INSTR
- DECODE
- CASE

# Before we get started: What is DUAL?

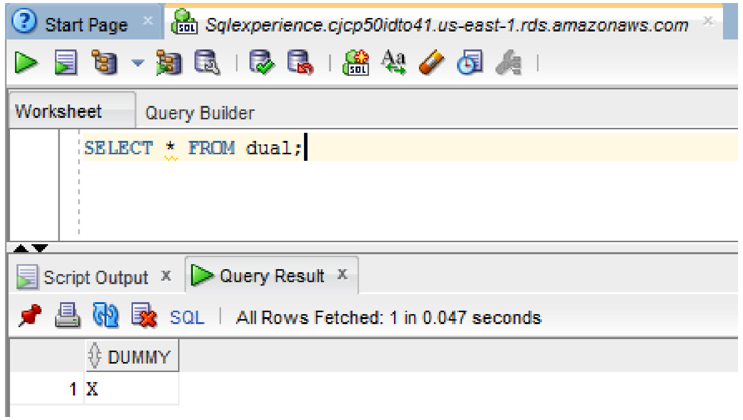
- DUAL is a table in Oracle (and some other DBMS) that you can query when you are executing queries that do not require a table, but you must specify one for parsing the query:
  - SELECT 2 + 4 From DUAL
  - Returns: 6

- Many DMBS do not require the use of dual. For example, in Microsoft SQL Server, this works fine:
  - SELECT 2 + 4
  - Returns: 6

# Before we get started: What is DUAL?

- DUAL has one tuple, with one attribute named “DUMMY” and one value, “X”

- `SELECT * FROM dual`
- `SELECT dummy FROM dual`




- Fun history lesson: the original implementation of dual had two tuples (hence the name “dual”) and was primarily used to double the number of records returned from a “real” table by doing the Cartesian product.

# The Concatenation Operator (||)

- Allows you to join multiple strings together using two vertical bars (AKA pipes)
  - Can be values from text attributes or a literal string
- In the SQL project database, you might try (for fun):

```
SELECT name || ' works here!' AS WorkerName FROM workers;
```

| | EMPID | NAME | PHONE | TITLE | SALARY | HIREDATE |
|---|---|---|---|---|---|---|
| 1 | 101 | James Smith | 2815550101 | Owner | 250,000 | 2010-01-01 00:00:00.000 |
| 2 | 102 | Matthew Martinez | 2815551919 | Director of Facilities | 125,000 | 2010-01-15 00:00:00.000 |
| 3 | 103 | Kimberly Hall | 7135551818 | Director of Racing | 160,000 | 2010-01-15 00:00:00.000 |
| 4 | 402 | Daniel Taylor | 2815551515 | Director of Education | 140,000 | 2014-10-02 00:00:00.000 |
| 5 | 414 | Michael Johnson | 7135550202 | Race Coordinator | 82,000 | 2023-12-01 00:00:00.000 |
| 6 | 436 | Sarah Rodriguez | 7135551010 | Ranch Hand | 55,000 | 2017-09-22 00:00:00.000 |
| 7 | 557 | Karen Lewis | 2815551111 | Race Coordinator | 78,000 | 2017-01-11 00:00:00.000 |
| 8 | 599 | Thomas Moore | 7135551414 | Health and Nutrition Specialist | 135,000 | 2015-02-20 00:00:00.000 |
| 9 | 670 | Christopher Anderson | 7135551616 | Ranch Hand | 52,000 | 2014-03-17 00:00:00.000 |

| | WORKERNAME |
|---|---|
| 1 | James Smith works here! |
| 2 | Matthew Martinez works here! |
| 3 | Kimberly Hall works here! |
| 4 | Daniel Taylor works here! |
| 5 | Michael Johnson works here! |
| 6 | Sarah Rodriguez works here! |
| 7 | Karen Lewis works here! |
| 8 | Thomas Moore works here! |
| 9 | Christopher Anderson works here! |

```
SELECT name || ' was hired on ' || hiredate as WorkerName FROM workers;
```

| | EMPID | NAME | PHONE | TITLE | SALARY | HIREDATE |
|---|---|---|---|---|---|---|
| 1 | 101 | James Smith | 2815550101 | Owner | 250,000 | 2010-01-01 00:00:00.000 |
| 2 | 102 | Matthew Martinez | 2815551919 | Director of Facilities | 125,000 | 2010-01-15 00:00:00.000 |
| 3 | 103 | Kimberly Hall | 7135551818 | Director of Racing | 160,000 | 2010-01-15 00:00:00.000 |
| 4 | 402 | Daniel Taylor | 2815551515 | Director of Education | 140,000 | 2014-10-02 00:00:00.000 |
| 5 | 414 | Michael Johnson | 7135550202 | Race Coordinator | 82,000 | 2023-12-01 00:00:00.000 |
| 6 | 436 | Sarah Rodriguez | 7135551010 | Ranch Hand | 55,000 | 2017-09-22 00:00:00.000 |
| 7 | 557 | Karen Lewis | 2815551111 | Race Coordinator | 78,000 | 2017-01-11 00:00:00.000 |
| 8 | 599 | Thomas Moore | 7135551414 | Health and Nutrition Specialist | 135,000 | 2015-02-20 00:00:00.000 |
| 9 | 670 | Christopher Anderson | 7135551616 | Ranch Hand | 52,000 | 2014-03-17 00:00:00.000 |

| | WORKERNAME |
|---|---|
| 1 | James Smith was hired on 01-JAN-10 |
| 2 | Matthew Martinez was hired on 15-JAN-10 |
| 3 | Kimberly Hall was hired on 15-JAN-10 |
| 4 | Daniel Taylor was hired on 02-OCT-14 |
| 5 | Michael Johnson was hired on 01-DEC-23 |
| 6 | Sarah Rodriguez was hired on 22-SEP-17 |
| 7 | Karen Lewis was hired on 11-JAN-17 |
| 8 | Thomas Moore was hired on 20-FEB-15 |
| 9 | Christopher Anderson was hired on 17-MAR-14 |

# The SUBSTR Function

- SUBSTR (char, m [,n]) returns a portion of char, beginning at character m, n characters long (if n is omitted, to the end of char). The first position of char is 1.

- Example:
  - `SELECT SUBSTR('ABCDEFG',3,4) AS "Substring" FROM DUAL;`
  - Returns: CDEF

- The SUBSTR function is often used in conjunction with the concatenation operator (||).

# Use of the Concatenation Operator (||)

- Display the name and phone number of all workers with phone formatted as (xxx)xxx-xxxx.

- ```
  SELECT name, '(' ||
  SUBSTR(phone,1,3) || ') ' ||
  SUBSTR(phone,4,3) || '-' ||
  SUBSTR(phone,7,4) AS "Phone No"
  FROM workers;
  ```

- We wouldn't be able to do these string manipulations (like substr) if we had stored phone as a numeric attribute!

| | NAME | Phone No |
|---|---|---|
| 1 | James Smith | (281) 555-0101 |
| 2 | Matthew Martinez | (281) 555-1919 |
| 3 | Kimberly Hall | (713) 555-1818 |
| 4 | Daniel Taylor | (281) 555-1515 |
| 5 | Michael Johnson | (713) 555-0202 |
| 6 | Sarah Rodriguez | (713) 555-1010 |
| 7 | Karen Lewis | (281) 555-1111 |
| 8 | Thomas Moore | (713) 555-1414 |
| 9 | Christopher Anderson | (713) 555-1616 |
| 10 | Angela Young | (281) 555-1717 |
| 11 | Charles Wilson | (281) 555-1313 |
| 12 | Robert Williams | (281) 555-0505 |
| 13 | Richard Davis | (281) 555-0909 |
| 14 | Jennifer Anderson | (281) 555-0303 |
| 15 | Anthony Thompson | (713) 555-2020 |
| 16 | Mary Garcia | (713) 555-0404 |
| 17 | Lisa Martinez | (713) 555-0606 |
| 18 | William Brown | (713) 555-0808 |
| 19 | Joseph Miller | (713) 555-1212 |
| 20 | David Jones | (281) 555-0707 |
  ```

# The LENGTH Function

- The LENGTH (char) function returns the length of the character string char.

- Example:

```
SELECT LENGTH('Jones, John') FROM DUAL;
```

  Returns: 11

- Note: attributes of datatype char (as opposed to varchar) return a length that includes all trailing blank spaces

# The RTRIM Function

- The RTRIM (char [, set]) function returns char, with final characters removed after the last character not in set.

- If no set of characters is specified, set defaults to ' ' (a blank space) and the function trims off trailing blanks.

- The RTRIM function operates on the rightmost characters in a string in the same way that the LTRIM function operates on the leftmost characters in a string.

- Example:
  - `SELECT RTRIM('STINSONxxXxx','x') AS "Right Trim Example" FROM DUAL;`
  - Returns: STINSONxxX

  - `SELECT RTRIM('Houston          ') AS "Right Trim Example" FROM DUAL;`
  - Returns: Houston

- Note: char (as opposed to varchar) deliver different results since char data type contains embedded trailing blanks.

# The LTRIM Function

- The LTRIM (char [, set]) function removes unwanted characters from the left of char, with initial characters removed up to the first character not in set.

- If no set of characters is specified, set defaults to ' ' (a blank space) and the function trims off leading blank spaces.

- Example:
  - `SELECT LTRIM('xxxXxxLAST WORD', 'x') AS "Left Trim Example" FROM DUAL;`
  - Returns: XxxLASTWORD

- Note: LTRIM is case-sensitive

# The LPAD and RPAD Functions

- The LPAD and RPAD functions allow you to “pad” the left (and right) side of a column or character string with a set of characters.

- Syntax: LPAD/RPAD (string, length [,'set'])
  - string is the name of the character column (or a literal string),
  - length is the total number of characters long that the result should be (i.e., its width), and
  - set is the set of characters that do the padding

# LPAD and RPAD Examples

- `SELECT LPAD('Page 1', 14, '*') AS "LPAD Example" FROM DUAL;`
  - Returns: ********Page 1

- `SELECT RPAD('Page 1', 14, '*.') AS "RPAD Example" FROM DUAL;`
  - Returns: Page 1*.*.*.*.

# The INSTR Function

- The INSTR function is used to return the numeric value of the location of a character string within a character column or character literal

- Syntax: INSTR (char1, char2 [,n[,m]])

- Its purpose is to locate the position of the $m^{th}$ occurrence of char2 in char1, beginning the search at position n.
  - If m is omitted, 1 is assumed.
  - If n is omitted, 1 is assumed.
  - The position is given relative to the first character of char1, even when n > 1.

# INSTR Function Examples

- Example:

  12345678901
  ```
  SELECT INSTR('MISSISSIPPI','S',5,2) AS "In String Example" FROM DUAL;
  ```
- Returns: 7

- Example:

  12345678901
  ```
  SELECT INSTR('MISSISSIPPI','S',5,1) AS "In String Example" FROM DUAL;
  ```
- Returns: 6

# Use of INSTR and SUBSTR Functions

```
SELECT TEXTBOOK.TITLE,                        7
INSTR(SUBSTR(TEXTBOOK.TITLE, INSTR(TEXTBOOK.TITLE, ' ')+1),'ing') "ing in word 2",
INSTR(SUBSTR(TEXTBOOK.TITLE,1),'ing') "ing in overall title"
FROM TEXTBOOK
WHERE INSTR(SUBSTR(TEXTBOOK.TITLE,
INSTR(TEXTBOOK.TITLE, ' ')+1), 'ing') > 0;
```

```
TITLE                       ing in word 2   ing in overall title
--------------------------- --------------  ------------------------
1234567890123456789
Linear Programming                      9                         16
Simulation Modeling                     6                         17
Data Modeling                           6                         11
```

**FOR YOUR NOTES: The numbers in red above the book titles are NOT part of the database – this is just something I added for demonstration purposes during class!!!**

# The DECODE Function

- The DECODE (value, search_value, result, default_value) function is used to compare value with search_value. If the values are equal, the DECODE function returns result; otherwise, default_value is returned. The DECODE function allows you to perform if-then-else logic in SQL within a row.

- Example: Note how the DECODE Function allows students to be listed in descending order by grade level (GR, SR, …, FR)

- ```
  SELECT SID, NAME, GRADELEVEL
  FROM STUDENT
  ORDER BY DECODE (GRADELEVEL, 'FR', '1', 'SO', '2', 'JR', '3', 'SR', 4, 'GR', 5) DESC;
  ```
  ```

# Case Expression in SQL

- The Oracle/PLSQL CASE expression has the functionality of an IF-THEN-ELSE statement. You can use the CASE expression within a SQL statement.

- Searched Case Expression Format

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    …
    WHEN conditionN THEN result
    ELSE default_result
END
```

- where condition1, condition2, …, conditionN are the conditions to be evaluated
- result1, result2, …, resultN are the returned results (one for each possible condition)
- default_result is the default result returned when no true condition is found

# Case Expression in SQL

- Can be used in a similar manner to decode:

```
SELECT SID, NAME, GRADELEVEL FROM STUDENT ORDER BY CASE
WHEN gradelevel='FR' THEN '1', WHEN gradelevel='SO' THEN '2',
WHEN gradelevel='JR' THEN '3', WHEN gradelevel='SR' THEN '4',
WHEN gradelevel='GR' THEN '5' END DESC;
```

- Or to replace values inline – for example, you could do something like:
  - `SELECT CASE WHEN ST_sex='M' then 'Mr. ' ELSE 'Ms. ' END || ST_name FROM staff;`

| ST_STAFFNO | ST_NAME | ST_POSITION | ST_SEX | ST_BIRTHDATE | ST_HIREDATE |
|---|---|---|---|---|---|
| S021 | Blake Ives | Manager | M | 01-OCT-58 | 03-OCT-02 |
| S037 | Randolph B. Cooper | Manager | M | 10-NOV-60 | 11-NOV-93 |
| S014 | Wynne Chin | Manager | M | 24-MAR-68 | 26-AUG-00 |
| S009 | Iris Junglas | Manager | F | 19-FEB-76 | 29-FEB-96 |
| S005 | Dennis A. Adams | Manager | M | 03-JUN-65 | 15-MAR-91 |
| S041 | Kathy M. Cossick | Manager | F | 13-JUN-68 | 23-JUN-98 |
| S023 | Manjari Mehta | Supervisor | F | 25-SEP-80 | 01-JUL-01 |
| S033 | Shince Francis | Supervisor | M | 12-DEC-81 | 18-NOV-03 |
| S055 | Amy Li | Assistant | F | 11-AUG-75 | 27-JAN-03 |

| STAFFERS |
|---|
| Mr. Blake Ives |
| Mr. Randolph B. Cooper |
| Mr. Wynne Chin |
| Ms. Iris Junglas |
| Mr. Dennis A. Adams |
| Ms. Kathy M. Cossick |
| Ms. Manjari Mehta |
| Mr. Shince Francis |
| Ms. Amy Li |

# Review: Single-Row Character Functions

- SUBSTR(char, m [,n])
  - Returns the portion of *char* starting at *m* and continuing for *n* characters
- LENGTH(char)
  - Returns the number of characters in *char*
- LTRIM(char [, set]) and RTRIM(char [, 'set'])
  - Removes characters in *set* from the left or right of *char* (default is space)
- LPAD/RPAD(char, length [,'set'])
  - Adds *length* characters in *set* to the left or right of *char* (default is space)
- INSTR (char1, char2 [,n[,m]])
  - Returns the position of the $m^{th}$ occurrence of *char2* in *char1*, starting at position *n*, by default, the first occurrence starting at position 1

# Module 13.1

## String manipulation SQL functions

- Concatenation
- SUBSTRING
- LENGTH
- TRIM (RTRIM / LTRIM)
- PAD (RPAD / LPAD)
- INSTR
- DECODE
- CASE

# BZAN 6354

# Lecture 13

## April 15, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences