BZAN 6354

Lecture 11

April 1, 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 Review
- Module 7.1
- Break
- Module 7.2
- Module 7.3

Administration

- Assignment 3 is due at 11:59 PM tonight

- SQL Project
  - Credentials for Oracle AND the submission site were sent out on Wednesday
    - Make sure you can connect to Oracle
    - Make sure you can connect to the submission system

  - A total of 122 points
    - Out of the 100 described on the syllabus – so you can earn an EXTRA/BONUS 2.2%

  - All due on April 29 (last day of class) at 11:59 PM
    - Don’t wait - this is a time consuming project!
    - This will be good practice for writing SQL on exam 2!

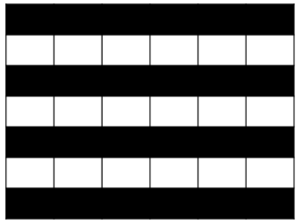
Review: relational algebra

- Two unary operations
  - Selection (σ)
  - Projection (π)
- Six binary operators
  - Union (∪)
  - Intersection (∩)
  - Difference (-)
  - Join (⋈)
  - Cartesian product (X)
  - Division (÷)

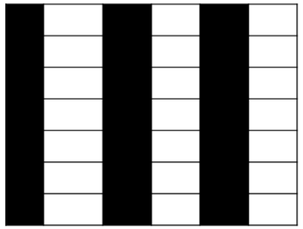
Review: Unary operators

- Select: $\sigma_{\text{<section condition>}}R$
- Project: $\pi_{\text{<attribute list>}}R$

Selection (σ)



Projection (π)



- The basic “Select-From-Where” block:

```
SELECT <attributes>
        FROM <table>
        WHERE <condition>
```

```
SELECT * FROM students;
```

```
SELECT empid, fname, lname FROM employees WHERE state=‘TX’;
```

Review: Additions to select-from-where

- Group by
  - Groups row based on a value of a particular attribute
  - Necessary for aggregate functions (count, min, max, avg, etc…)
    - SELECT college, AVG(hrs) FROM course GROUP BY college;

- Having
  - Similar to WHERE, but used for aggregate functions
    - SELECT college, AVG(hrs) FROM course
      GROUP BY college HAVING AVG(hrs) > 3;

- Order by
  - Changes the order in which tuples are returned
    - SELECT college, AVG(hrs) FROM course
      GROUP BY college ORDER BY AVG(hrs);

Review: Comparison operations

- Simple ones:
  - >, <, >=, <=, <>, =
    - SELECT * FROM employees WHERE salary > 50000;

- The LIKE operator allows for wildcards
  - % for many characters (* in MS Access)
    - SELECT * FROM employees WHERE lname LIKE ‘G%’;
  - _ for a single character (? In MS Access)
    - SELECT * FROM employees WHERE fname LIKE ‘_I_A’;

- The IS operator used to work with NULL values
  - SELECT * FROM employees WHERE fired_date IS NULL;

- All can be negated with the NOT operator

Fun Example #1

Show all attributes for employees with a salary greater than $20,000. Order the results by last name.

```
SELECT <attributes>    SELECT *
FROM <table>           FROM employees
WHERE <condition>      WHERE salary > 20000
GROUP BY <attribute>
HAVING <condition>
ORDER BY <attribute>   ORDER BY lname;
```

Fun Example #2

List each department name and the department’s average salary for departments that have an average salary of $50,000 or above. Do not include the ACCT or ECON departments. Order the results alphabetically by department

```
SELECT <attributes>     SELECT department, AVG(salary)
FROM <table>            FROM employees
WHERE <condition>       WHERE department NOT IN ('ACCT', 'ECON')
GROUP BY <attribute>    GROUP BY department
HAVING <condition>      HAVING AVG(salary) >= 50000
ORDER BY <attribute>    ORDER BY department;
```

Fun Example #3

List each department and the number of faculty in each department that have a salary over $100,000 when the average salary in the department is less than $80,000. Order the results so the department with the largest number of faculty meeting this criteria is first.

```
SELECT <attributes>   SELECT department, COUNT(*) as Num_faculty
FROM <table>          FROM employees
WHERE <condition>     WHERE salary > 100000
GROUP BY <attribute>  GROUP BY department
HAVING <condition>    HAVING AVG(salary) < 80000
ORDER BY <attribute>  ORDER BY COUNT(*) DESC;
```

Fun Example #4

Given this data, what value will be returned by this SQL statement?

```
SELECT count(*) FROM employees WHERE emp_name LIKE '_a%e%';
```

A. Null
B. 2
C. Dave, James
D. Dave, James, David, Saul

Answer: B

Employees

EMP_ID	EMP_Name	EMP_HrRate	EMP_Title	ST_ID
101	Kirk Hammett	12	Jr. Associate	1
102	Dave Mustaine	22	Manager	2
103	Stone Gossard	15	Sr. Associate	2
104	Kim Gordon	26	Manager	3
105	Thomas Morello	20	Sr. Associate	3
106	Kim Thayil	20	Sr. Associate	1
107	James Page	25	Sr. Associate	2
108	David Gilmour	13	Jr. Associate	1
109	Saul Hudson	14	Jr. Associate	3
110	Chris Impellitteri	24	Manager	1

SQL on the exam

- I will give you a few small datasets like this and ask you to write SQL by hand
  - You won't have Oracle to do "trial and error"
  - Not SUPER complicated, but will use WHERE, LIKE, GROUP BY, HAVING, aggregate functions, INNER and OUTER joins, subqueries, string manipulation, etc.
  - Like the SQL project, I will show you what the expected results should look like

- I may also show you some SQL queries and ask what they will return (just like on the previous slide)

Employees

EMP_ID	EMP_Name	EMP_HrRate	EMP_Title	ST_ID
101	Kirk Hammett	12	Jr. Associate	1
102	Dave Mustaine	22	Manager	2
103	Stone Gossard	15	Sr. Associate	2
104	Kim Gordon	26	Manager	3
105	Thomas Morello	20	Sr. Associate	3
106	Kim Thayil	20	Sr. Associate	1
107	James Page	25	Sr. Associate	2
108	David Gilmour	13	Jr. Associate	1
109	Saul Hudson	14	Jr. Associate	3
110	Chris Impellitteri	24	Manager	1

Review: Joining tables

- Cartesian product:
  - `SELECT * FROM H CROSS JOIN C;`
- Equijoin:
  - `SELECT * FROM H INNER JOIN C ON H.owner = C.username;`
- Self Join (for unary/recursive realtionships)
  - ```
    SELECT E.EmpID, E.Name AS EmpName, M.Name AS MgrName
    FROM Employee E  INNER JOIN Employee M
    ON E.MgrID = M.EmpID;
    ```
- Multi-table join
  - ```
    SELECT * FROM
     (employee E INNER JOIN work_in W ON E.empid = W.empid)
    INNER JOIN plants P ON P.plantid = W.plantid;
    ```

Review: Cartesian project + equality

- Cartesian product with conditional statement (equivalent to an equijoin)
  - ```
    SELECT * FROM employees E, plants P
    WHERE E.PlantID = P.PlantID;
    ```

- Multi-table join (same result as previous slide)
  - ```
    SELECT * FROM Employees E, Plants P, Work_in W
    WHERE E.EmpID = W.EmpID AND P.PlantID = W.PlantID;
    ```

Review: Outer Joins

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

- SELECT * FROM H LEFT OUTER JOIN C ON H.owner = C.username;
- SELECT * FROM H RIGHT OUTER JOIN C ON H.owner = C.username;
- SELECT * FROM H FULL OUTER JOIN C ON H.owner = C.username;

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	canderson	Christine	Anderson
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	tswift	Tina	Swift
6	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
7	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	tswift	Tina	Swift
14	Shamrock	Black	1,400	[NULL]	[NULL]	[NULL]	[NULL]
15	[NULL]	[NULL]	[NULL]	[NULL]	ssimpson	Sam	Simpson

Returned with Left outer join →
Returned with Right outer join→

Returned with Full outer join

Review: Set Operations

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

What must be true for set theory operations to work?

- `SELECT * FROM F UNION SELECT * FROM S;`
- `SELECT * FROM F INTERSECT SELECT * FROM S;`
- `SELECT * FROM F MINUS SELECT * FROM S;`
- `SELECT * FROM S MINUS SELECT * FROM F;`

The operator “minus” is only for Oracle, use “except” in other DBMS
Set operations are not available in MS Access

Review: Uncorrelated Subqueries

- The subquery is executed first and passes one or more values to the outer query
  - Three operators may be used: IN, ANY, ALL
  - May be negated with the NOT operator

- The IN operator evaluates if rows processed by the outer query are equal to any of the values returned by the subquery (i.e., it creates an OR condition).

Review: Uncorrelated Subqueries (IN)

- `SELECT * FROM horses`
  `WHERE name IN (SELECT ord_Horsename FROM orders);`

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift
14	Shamrock	Black	No	M	1,400	[NULL]

	RXNUM	ORD_MEDCODE	ORD_HORSENAME	DOSE	FREQUENCY
1	108	Dexa	Betty	2	1
2	107	Dexa	Shamrock	2	1
3	105	Doxy	Trotty	1	3
4	110	Enro	Erica	3	2
5	101	Flux	Sam	1	2
6	104	Iver	Trotty	2	1
7	103	Iver	John	2	1
8	102	Pen	Sam	1	1
9	106	Pres	Shamrock	1	1
10	109	Xyl	Hulk	1	1

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	John	Grey	No	M	1,800	mgrimes
3	Trotty	Brown	Yes	M	1,300	mgrimes
4	Shamrock	Black	No	M	1,400	[NULL]
5	Betty	White	Yes	F	1,250	tswift
6	Hulk	Grey	No	M	2,050	mgrimes
7	Erica	Yellow	Yes	F	920	canderson

To reiterate:

- There are many ways to do the same thing
  - Sometimes there is no benefit to one way over another
  - Sometimes knowing multiple ways will save you

- Generally, relational algebra expressions can be translated into any number of other equivalent expressions

- Much like we did when creating ERDs – DECOMPOSE the query into each subquery or operation and UNDERSTAND what each piece is doing

Module 7.1

Functional Dependencies

- What are functional dependencies, and what makes a FD "undesirable"?
- What is redundancy?
- What are modification anomalies?
- Decomposing relations to alleviate redundancy

Functional Dependencies

- So far we’ve done a lot of “guessing” about how databases should be designed
  - Grouping of attributes has so far been an intuitive process and requires rigorous validation to ensure design quality.

- Understanding **functional dependencies** gives us a systematic way of understanding:
  - What attributes should be grouped together in relations
  - What attributes are valid candidate keys
  - Where data redundancy problems exist

Functional Dependencies

- Remember early in the semester we created the Horses table, but added the owners phone number?
  - Made sense at first, but then we realized “Phone” was really an attribute of the customer that owns the horse, not the horse itself…
  - …which led to data redundancy (though we didn’t understand why back then…)

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	(555) 424-1313

Functional Dependencies

- Abbreviated as FD
- FDs specify a relationship between attributes in a relation schema
- An attribute **A** (atomic or composite) in a relation schema **R functionally determines** another attribute **B** (atomic or composite) in R if for a given value of A there is a single, specific value of B
  - Said another way: B is **functionally dependent** on A
- Expressed as A → B
  - A is called the **determinant**
  - B is called the **dependent**

Functional Dependencies

- A core part of database design
  - Empid → Name
  - SectionID → CourseName
  - Etc…

- Or in the case of our horse model →
  - Medcode→ {Medname, Classification, Cost, QoH, QoO}
  - Name → {Color, Spots, Sex, Weight, Owner}

- <u>Undesirable</u> function dependencies are the 'seeds' of data redundancy leading to <u>modification anomalies</u>.

- "Undesirable" FDs are ones where the determinant is not a candidate key!

	MEDCODE	MEDNAME	CLASSIFICATION	COST	QOH	QOO
1	Flux	Flunixin Meglumine	NSAID	27	43	0
2	Bute	Phenylbutazone	NSAID	19	84	0
3	Oxi	Oxibuzone	NSAID	12	55	0
4	Mel	Meloxicam	NSAID	6	72	0
5	Pen	Penicillin	Antibiotic	38	73	0
6	Doxy	Doxycycline	Antibiotic	81	89	0
7	TMPS	Trimethoprim Sulfa	Antibiotic	27	50	0
8	Enro	Enrofloxacin	Antibiotic	36	78	0
9	Xyl	Xylazine	Sedative	22	28	0
10	Acep	Acepromazine	Sedative	33	50	0
11	Fen	Fenbendazole	Dewormer	52	15	25
12	Meb	Mebendazole	Dewormer	81	25	0
13	Pyr	Pyrantel Embonate	Dewormer	11	29	0
14	Iver	Ivermectin	Dewormer	39	21	0
15	Clen	Clenbuterol	Respiratory	132	11	20
16	Pres	Prednisone	Corticosteroid	47	48	7
17	Dexa	Dexamethasone	Corticosteroid	47	18	82

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift
14	Shamrock	Black	No	M	1,400	[NULL]

Functional Dependencies

- Functional dependencies are essentially technical translations of user-specified business rules expressed as constraints in a relation schema
  - We cannot just ignore them when undesirable – we must still accommodate.

- Functional dependencies are the building blocks of “normalization” principles

The issues of quality of design: an illustration

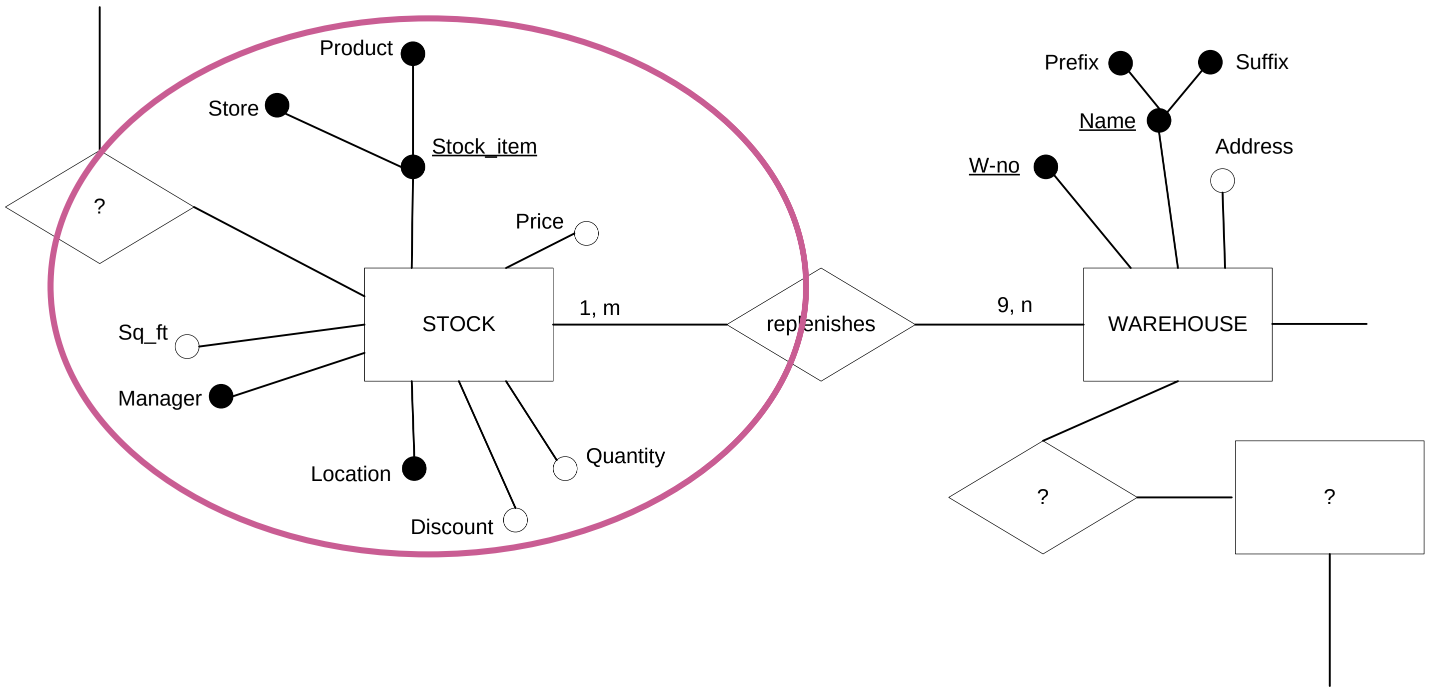


Figure 7.1a An excerpt from an ER diagram
(Page 359 in the optional textbook)

STOCK (<u>Store, Product</u>, Price, Quantity, Location, Discount, Sq_ft, Manager)

The Stock relation

How can we uniquely identify every instance in this relation?

STOCK

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

Figure 7.1c An instance of the relation schema, STOCK

Store? No
Product? No
Price? No
Quantity? No
Location? No
Discount? No
SqFt? No
Manager? No

The Stock relation

How can we uniquely identify every instance in this relation?

STOCK

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

For now, take my word that all instances can be uniquely identified by {Store, Product}. This is a: Super Key (Unique)

Candidate Key (Irreducible)

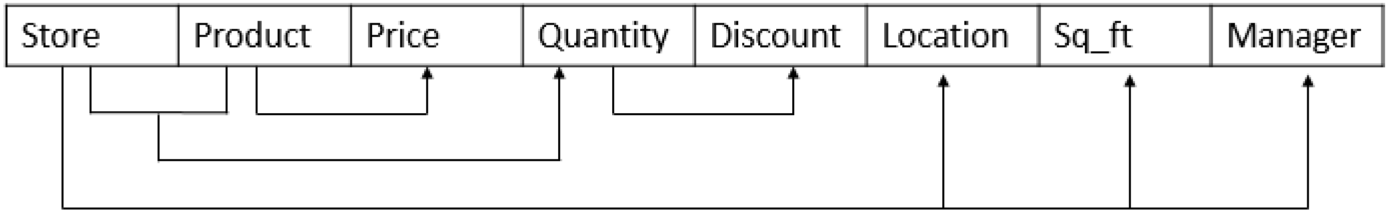
Primary Key (Not NULL)

FDs in the Stock relation

STOCK

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

- We can represent FD using a dependency diagram



FDs in the Stock relation

STOCK

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

- Does Product → Price?
  - Yes! For a given value of **Product** there is consistently a single, specific value of **Price**
  - Refrigerator is 1850, Dishwasher is 600, Television is 1400, etc…

- Does Price → Product?
  - No! While Product → Price, the reverse is not true (i.e., Price ↛ Product)
  - If I ask for the product that is $300 – do I mean the Vacuum or the Lawn Mower?

- Given A → B, we cannot infer B → A

FDs in the Stock relation

STOCK

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

- Does Store → Location?
  - Yes! If I ask for store 15, what is the location?
- Does Location → Store?
  - No! If I ask for the store in Houston, is it 11 or 15?
  - Location ↛ Store
- Again, given A → B, we cannot infer B → A

FDs in the Stock relation

STOCK

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

- If Store → Location, does {Store, Product} → Location?
  - Yes! think back to our discussion of super keys
- If Quantity → Discount, does {Quantity, Manager} → Discount ?
  - Yes!
- If a determinant determines a dependent, a determinant plus any other attribute will also determine that dependent

The Stock relation – Data Redundancy?

STOCK

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

Figure 7.1c An instance of the relation schema, STOCK

Is there data redundancy for the attribute:

Price? Location? Quantity? Discount?

Just because there is repeated data, does it mean the data is redundant? **No**

What is Data Redundancy?

- Repeated appearance of same data value for an attribute does <u>not</u> automatically mean data redundancy.
- Superfluous repetition that <u>does not add new meaning</u> constitutes data redundancy.
- Error in what attributes are assigned to what entity type(s) leads to data redundancy.
- Data redundancy leads to <u>modification anomalies</u>!

The Stock relation – Data Redundancy?

STOCK

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

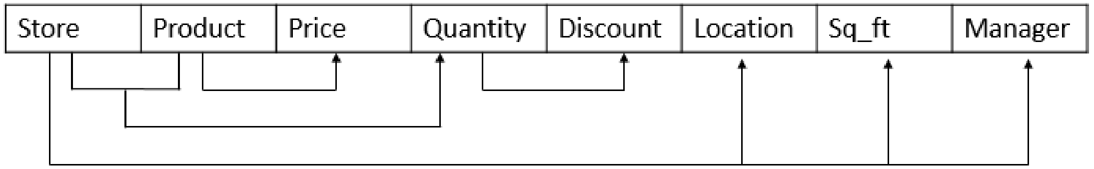
Figure 7.1c An instance of the relation schema, STOCK

Is there data redundancy for the attribute:

Price? **Yes** Location? **Yes** Quantity? **No** Discount? **Yes**

The problem

- When there are FDs in which the determinant is not a candidate key of R – this is an “undesirable” FD



- Recall that {Store, Product} is a Candidate Key (and the Primary Key)
  - Product is not a candidate key, but price is determined by Product – bad!
  - Quantity is not a candidate key, but discount is determined by Quantity – bad!
  - Store is not a candidate key, but Location, Sq_ft, and Manager are determined by store – bad!
  - In {Store, Product} → Quantity, {Store, Product} is a candidate key, which is why there is no redundancy!

STOCK

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

Modification Anomalies

- Three types of anomalies:
  - Insert Anomalies
  - Delete Anomalies
  - Update Anomalies

- We can collectively refer to these as <u>modification anomalies</u>

Insertion Anomaly

Suppose we want to add “Washing Machine” to our stock with a price of $600

STOCK

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
	Washing Machine	600					

We cannot do this, since Store (part of the primary key) cannot have a null value.
Without assigning a store for the Washing Machine, it is not possible to add this information.
Should Store be an attribute of Washing Machine? No – this is an undesirable FD!

Deletion Anomaly

Suppose we want to close store 17

STOCK

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

Not only does the action required entails deletion of multiple rows, but also can there be inadvertent loss of information - in this case that vacuum cleaner is priced at $300 is inadvertently lost.

Update Anomaly

Suppose store 11 is moved from Houston to Cincinnati

STOCK

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	Cincinnati	10%	2300	Creech
11	Refrigerator	1850	120	Cincinnati	5%	2300	Creech
11	Lawn Mower	300		Houston		2300	Creech

Figure 7.1c An instance of the relation schema, STOCK

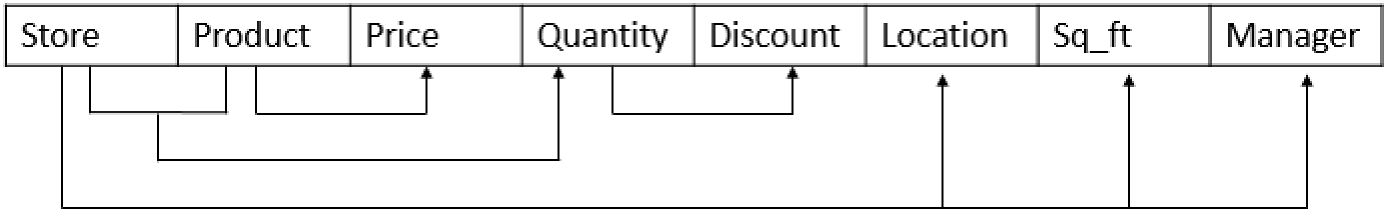
In addition to the necessity to update multiple rows, inadvertent failure to update all the relevant rows changes the semantics of the scenario – i.e., store 11 is located in both Houston and Cincinnati is an inadvertent change in semantics.

How to get rid of anomalies?

- What do we need to do to get rid of these anomalies?

STOCK

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



Decomposing to get rid of anomalies

STOCK

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

Figure 7.1c An instance of the relation schema, STOCK

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

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	

DISC_STRUCTURE

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

Decomposing to get rid of anomalies

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

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	

DISC_STRUCTURE

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

Figure 7.3 A redundancy-free decomposition of the STOCK instance in Figure 7.1c

No data redundancy or modification anomalies in the tables Store, Product, Inventory, or Disc_Structure

- Can add a product without assigning to a store
- Can delete a store without impacting products
- Can move a store by updating a single record

The design that is free from data redundancies/modification anomalies is said to be “normalized”

Design-specific ER Diagram Reverse-Engineered from the Relational Schema

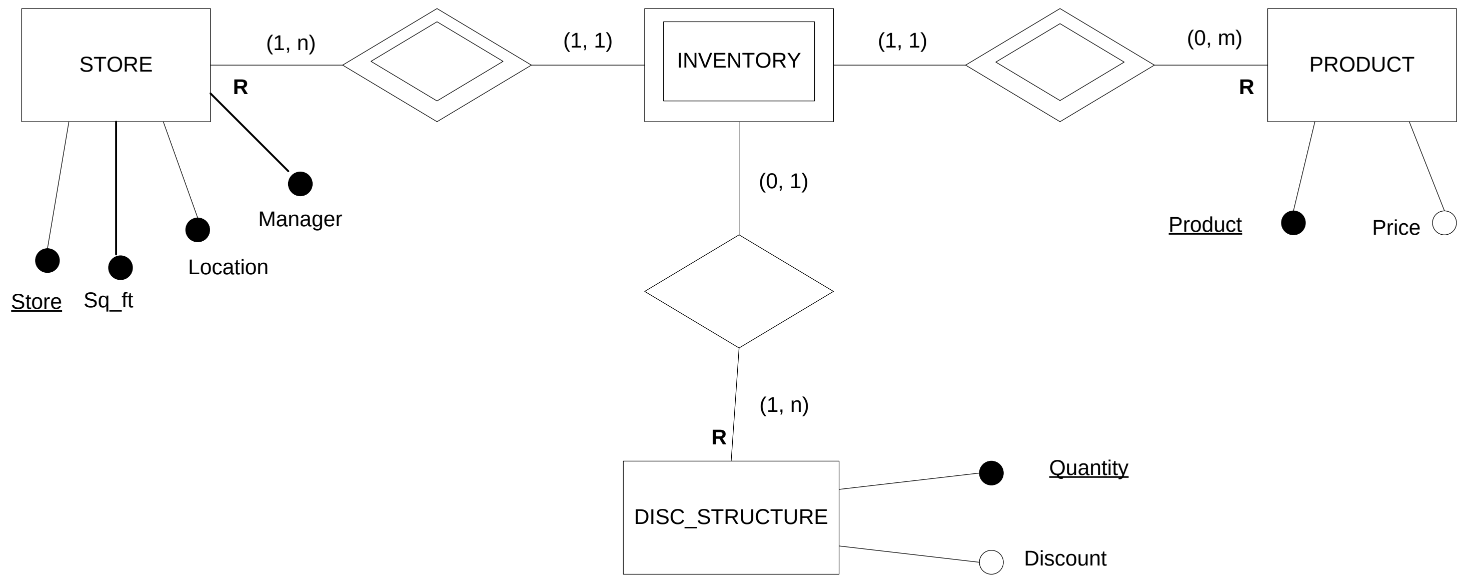
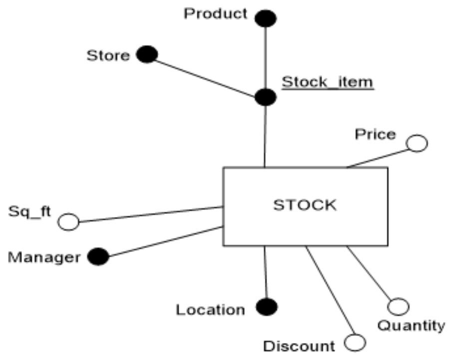


Figure 7.4b Design-specific ER diagram reversed-engineered from the logical schema in Figure 7.4a

A (much) better model

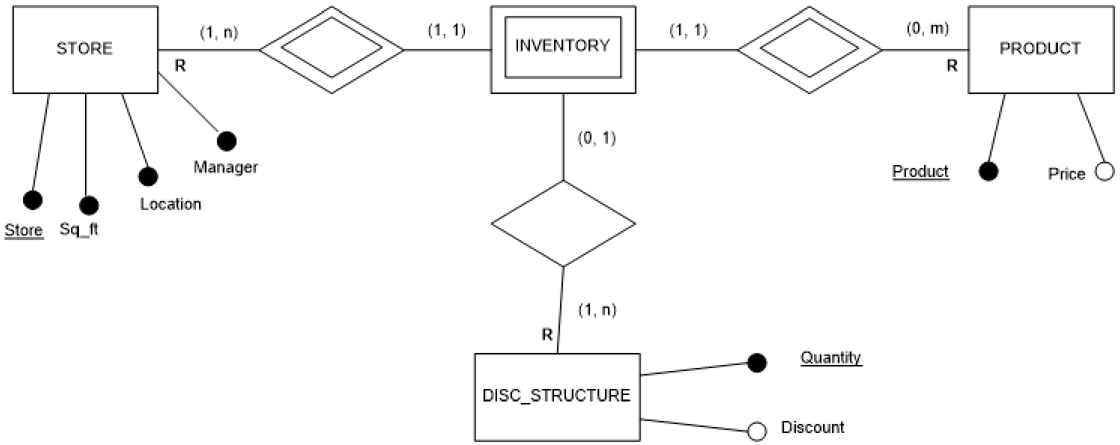


STOCK

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

Old and busted, full of data redundancy, bad FDs, and modification anomalies

Undesirable FDs resolved, ready to reliably serve the business



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

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	

DISC_STRUCTURE

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

Module 7.1

Functional Dependencies

- What are functional dependencies, and what makes a FD “undesirable”?
- What is redundancy?
- What are modification anomalies?
- Decomposing relations to alleviate redundancy

Break

Module 7.2
Closure and Cover of FDs

- Using Armstrong’s Axioms to find the closure and cover of F
- Finding the closure of individual attributes

Working with functional dependencies

- Armstrong's Axioms
  - Rules for systematically understanding what attributes are determined by what other attributes

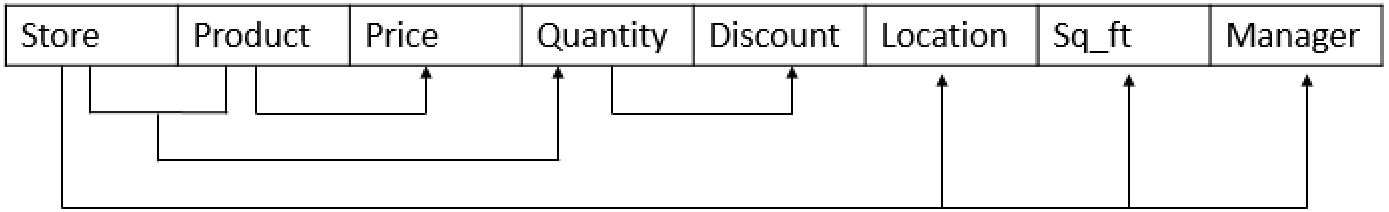
- Cover and Closure (F_C and F^+) for FDs
  - Cover: Minimal FDs that express all dependencies in F
  - Closure: All possible FDs in F (F^+)

- Helps us know that the decomposition is correct and complete <u>without</u> needing to look at the data

FDs in the Stock relation

- Typically we think of all the FDs in a relation as a set of dependencies called "F"
  - $F\{FD_1, FD_2, FD_3, \ldots, FD_n\}$
- Some FDs are *semantically obvious* based on the business rules
- Other FDs must be *inferred*
- The set of all FDs in F is referred to as F^+ ("F Closure")

FDs in the Stock relation



- The Semantically Obvious FDs:
  - FD1: Store → {Location, Sq_Ft, Manager}
  - FD2: {Store, Product} → Quantity
  - FD3: Product → Price
  - FD4: Quantity → Discount

- These are "obvious" because this is how the business people described the data to us.

Armstrong’s Axioms (Page 368)

- Three primary Axioms
  - Reflexivity
  - Augmentation
  - Transitivity

- Four inference rules
  - Union
  - Decomposition
  - Composition
  - Pseudotransitivity

Reflexivity

- Trivial Dependencies
  - Impossible not to satisfy and don’t tell us anything new

- If Y is a subset of X, then X → Y
  - If Y={A,C} and X={A,B,C,D} then {A,B,C,D} → {A,C}

- {PSID, FName, Lname} → {PSID}
- {PSID, FName, Lname} → {Fname, Lname}

- This also means attributes functionally determine themselves
  - X → X
  - Y → Y

Augmentation (works two ways)

1. We can add **anything** to the determinant and it will still determine the dependent
   - If X → Y, then:
     - {X,Z} → Y
     - {X, A, B, C, D, E, F, G} → Y
   - Remember our discussion of super keys?
2. We can add the **same attribute** to **both** the determinant and the dependent and it will still determine the dependent
   - If X → Y, then:
     - {X,A} → {Y,A}
   - Basically the reflexivity axiom

- If EmpID → Lname, then:
  - {EmpID, DateOfBirth} → Lname
  - {EmpID, HairColor} → {Lname, HairColor}

Transitivity

- If X → Y and Y → Z, then X → Z
- If ZipCode → City and City → State, then…
  - ZipCode → State
- If 77018 → Houston and Houston → Texas, then…
  - 77018 → Texas
- Is the inverse true? Does State → City, or City → Zip?
  - No!

(Assuming Houston is a unique city name)

Union

- If a determinant determines two dependents independently, it also determines the union of those dependents

- If X → Y and X → Z, then…
  - X → {Y, Z}

- If EmpID → Fname and EmpID → Lname, then…
  - EmpID → {Fname, Lname}

Decomposition

- Basically the opposite of Union – if a determinant determines two dependents together, it also determines them individually

- If X → {Y, Z} then…
  - X → Y and X → Z

- If EmpID → {Fname, Lname}, then…
  - EmpID → Fname and EmpID → Lname

Composition

- Similar to the Union rule

- If A → B and C → D, then…
  - {A,C} → {B,D}

- If EmpID → Lname and Dept → College then:
  - {EmpID, Dept} → {Lname, College}

Pseudotransitivity (my favorite)

- Basically the transitivity rule applied to individual attributes within a set

- If X → Y and {Y, W} → Z then…
  - {X, W} → Z

- If Dept → College and {College, YearsWorked} → Salary, then:
  - {Dept, YearsWorked} → Salary

- If MIS→ Bauer and {Bauer, 10} → 90000, then:
  - {MIS, 10} → 90000

- **In words**: If we know people that work for the MIS department are in Bauer college, **and** we know that people who have worked for Bauer college for 10 years make $90,000, **then** we also know that people who have worked for the MIS department for 10 years make $90,000

F and Closure of F (F^+)

- The set of semantically obvious FDs specified on a relation schema R is denoted as F

- The set that includes F and <u>all other FDs inferred from F</u> is called the **closure of F**, denoted as F^+

- Having specified F from the semantics of the attributes of a relation schema R, the designer can develop F^+
  - Armstrong’s axioms are useful in deriving F^+

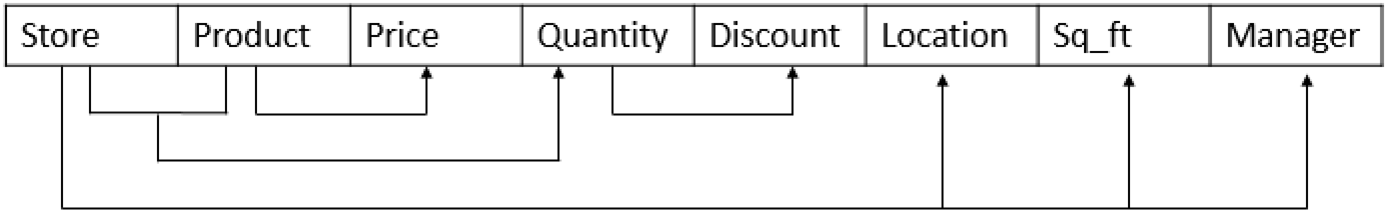
F and Closure of F (F^+)

- Given a set of semantically obvious FDs, F{fd1, fd2} where
  - fd1: {Store, Product} → Quantity
  - fd2: Quantity → Discount

- Using transitivity, one can infer the presence of:
  - fd3: {Store, Product} → Discount in the closure of F (F^+)

- Note: Since trivial dependencies do not provide any additional information, they are usually excluded from F^+.

Finding Closure (F^+) for Stock



- The Semantically Obvious FDs:
  - FD1: Store → {Location, Sq_Ft, Manager}
  - FD2: {Store, Product} → Quantity
  - FD3: Product → Price
  - FD4: Quantity → Discount

- These are “obvious” because they were specifically stated in what the business people told us

Finding Closure (F^+) for Stock

- Apply Armstrong's Axioms until no more FD can be derived (very inefficient)

- The Semantically Obvious FDs:
  - FD1: Store → {Location, Sq_Ft, Manager}
  - FD2: {Store, Product} → Quantity
  - FD3: Product → Price
  - FD4: Quantity → Discount

- A few derived FDs (there will be MANY of these in F^+):
  - Store → Location, Store → Sq_Ft, Store → Manager (**Decomposition of FD1**)
  - {Store, Product} → Discount (**Transitivity on FD2 and FD4**)
  - {Store, Product} → {Location, Price} (**Composition of FD1 and FD3**)
  - {Product, Quantity} → {Price, Discount} (**Composition of FD3 and FD4**)

- The **problem** is when there are FDs in which the determinant in that FD is not a candidate key of R – this is an "undesirable" FD
  - Fixed with "normalization"

Minimal Cover

- Often the semantically obvious FDs (F) have some redundancy or extraneous attributes, as they are derived from the business rules

- The minimal cover of F (often called G_c) is a simplified set of FDs that is equivalent to F
  - Expressed as $G_c \equiv F$

Minimal Cover: Example 1

- R{Tenant, Apartment, Rent}
  - FD1: Tenant → {Apartment, Rent}
  - FD2: Apartment → Rent
  - FD3: {Tenant, Apartment} → Rent
  - FD4: {Tenant, Rent} → Apartment

Tenant	Apartment	Rent
Alice	10A	$400
Bob	10B	$450
Charlie	20A	$425
Dave	20B	$375
Eugene	30A	$400

Minimal Cover: Example 1

- R{Tenant, Apartment, Rent}
  - FD1: Tenant → {Apartment, Rent}
  - FD2: Apartment → Rent
  - FD3: {Tenant, Apartment} → Rent
  - FD4: {Tenant, Rent} → Apartment

- We can decompose FD1:
  - FD1a: Tenant → Apartment
  - FD1b: Tenant → Rent

Minimal Cover: Example 1

- R{Tenant, Apartment, Rent}
  - FD1a: Tenant → Apartment
  - FD1b: $\boxed{\text{Tenant → Rent}}$
  - FD2: Apartment → Rent
  - FD3: {Tenant, Apartment} → Rent
  - ~~FD4: {Tenant, Rent} → Apartment~~

- We can use the Pseudotransitivity axiom to show that since Tenant → Rent (FD1b), the Rent attribute in FD4 is superfluous
  - FD4 becomes {Tenant, Tenant} → Apartment
    - or just Tenant → Apartment
  - Identical to FD1a, so we can remove FD4

Minimal Cover: Example 1

- R{Tenant, Apartment, Rent}
  - FD1a: $\boxed{\text{Tenant} \rightarrow \text{Apartment}}$
  - FD1b: Tenant → Rent
  - FD2: Apartment → Rent
  - ~~FD3: {Tenant, Apartment} → Rent~~
- We can use the Pseudotransitivity axiom to show that since Tenant → Apartment (FD1a), the Apartment attribute in FD3 is superfluous
  - FD3 becomes Tenant → Rent
  - Identical to FD1b, so we can **remove** FD3

Minimal Cover: Example 1

- R{Tenant, Apartment, Rent}
  - FD1a: $\boxed{\text{Tenant} \rightarrow \text{Apartment}}$
  - ~~FD1b: Tenant → Rent~~
  - FD2: $\boxed{\text{Apartment} \rightarrow \text{Rent}}$

- Using the transitivity axiom, we know that since Tenant → Apartment (FD1a) and Apartment → Rent (FD2), then Tenant → Rent
  - Identical to FD1b, so FD1b can be removed

Minimal Cover: Example 1

- We are left with the minimal cover (G_c)
  - FD1: Tenant → Apartment
  - FD2: Apartment → Rent

- We started here (F):
  - FD1: Tenant → {Apartment, Rent}
  - FD2: Apartment → Rent
  - FD3: {Tenant, Apartment} → Rent
  - FD4: {Tenant, Rent} → Apartment

- And have shown that $G_c \equiv F$
  - A smaller/similar set of FD to make sure we match

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!

Attribute Closure

- Given R(A,B,C,D,E,G,H), where F:
  - FD1: B → {G,H}
  - FD2: A → B
  - FD3: C → D

- What is A^+ (closure [A | F])?

- By applying Armstrong's Axioms, we can infer:
  - Transitivity of FD1 and FD2: A → {G,H}
  - Union of FD2 and FD1: A → {B,G,H}
  - Remember to include reflexivity: A → {A,B,G,H}
  - We can go no further, so A^+ = {A,B,G,H}

- Would it be helpful to find the closure of {A,B}?
- $\{A,C\}^+$ = {A,B,C,D,G,H}

No! If we know A, we already know everything B can tell us

Module 7.2

Closure and Cover of FDs

- Functional Dependencies and Normalization

Module 7.3

Deriving Candidate Keys

- Deriving candidate keys using the synthesis method
- Deriving candidate keys using the decomposition method

Synthesis Approach

- 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?

Synthesis Approach

- 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!

The Decomposition Approach

- 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: $(K' \rightarrow A_i)$ 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.

The Decomposition Approach

- 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

The Decomposition Approach

fd1: B → {G, H} fd2: A → B fd3: C → D

- Step 1: Set superkey (K) of URS {A, B, C, D, E, G, H}
  - K = {A, B, C, D, E, G, H}
  - It should be obvious that K → {A, B, C, D, E, G, H} at this point due to reflexivity (trivial dependency)

- Step 2: Remove an attribute (H) from K, call it K'.
  - Does (K' → H) persist in F^+, where K' = {A, B, C, D, E, G}?
  - Answer: **Yes**, K' → H since B → H
  - K' becomes K = {A, B, C, D, E, G}, and K → {A, B, C, D, E, G, H}

- Step 3. Remove another attribute (G) from K, call it K'
  - Does (K' → G) persist in F^+, where K' = {A, B, C, D, E}?
  - Answer: **Yes**, K' → G since B → G
  - K' becomes K = {A, B, C, D, E}, and K → {A, B, C, D, E, G, H}

- Step 4. Remove another attribute (E) from K, call it K'
  - Does (K' → E) persist in F+, where K' = {A, B, C, D}?
  - Answer: NO! We have no way of determining E based on {A,B, C, D}: {A,B, C, D} ↛ E
  - K remains as it was, K={A, B, C, D, E} , and K → {A, B, C, D, E, G, H}

The Decomposition Approach

fd1: B $\rightarrow$ {G, H}　　fd2: A $\rightarrow$ B　　fd3: C $\rightarrow$ D

- Step 5. Remove another attribute (D) from K, call it K'
  - Does (K' $\rightarrow$ D) persist in F+, where K' = {A, B, C, E}?
  - Answer: Yes, K' $\rightarrow$ D since C $\rightarrow$ D
  - K' becomes K = {A, B, C, E}, and K $\rightarrow$ {A, B, C, D, E, G, H}

- Step 6: Remove another attribute (C) from K, call it K'.
  - Does (K' $\rightarrow$ C) persist in F^+, where K' = {A, B, E}?
  - Answer: NO! We have no way of determining C based on {A, B, E}: {A, B, E} $\nrightarrow$ C
  - K remains as it was, K={A,B,C, E} , and K $\rightarrow$ {A, B, C, D, E, G, H}

- Step 7. Remove another attribute (B) from K, call it K'
  - Does (K' $\rightarrow$ B) persist in F^+, where K' = {A, C, E}?
  - Answer: Yes, K' $\rightarrow$ B since A $\rightarrow$ B
  - K' becomes K = {A, C, E}, and K $\rightarrow$ {A, B, C, D, E, G, H}

The Decomposition Approach

fd1: $B \rightarrow \{G, H\}$ fd2: $A \rightarrow B$ fd3: $C \rightarrow D$

- Step 8. Remove another attribute (A) from K, call it K'
  - Does (K' → A) persist in F+, where K' = {C, E}?
  - Answer: NO! We have no way of determining A based on {C, E}: $\{C, E\} \nrightarrow A$
  - K remains as it was, K={A, C, E} , and K → {A, B, C, D, E, G, H}

- At this point, we have determined that K = {A, C, E} is a superkey that cannot be further reduced and thus becomes a candidate key of the URS

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

Derivation of Other Candidate Keys of [R | F]

- If F_c contains an FD (FDx) where a candidate key of R is a dependent, then the determinant of FDx is also a candidate key of R.

- When a candidate key of R is a composite attribute, for each key attribute (atomic or composite), evaluate if the key attribute is a dependent in an FD (FDy) in F_c.
  - If so, then the determinant of FDy can, via pseudotransitivity, replace the key attribute under consideration, thus yielding additional candidate key(s) of R.

- Repetition of the above two steps for every candidate key of R will systematically reveal all the other candidate key(s), if any, of R.

Choosing the Primary Key from the CKs

- While we have noted that the choice of primary key from among the candidate keys is essentially arbitrary, some rules of thumb are often helpful in this regard:
  - A candidate key with the least number of attributes may be a good choice.
  - A candidate key whose attributes are numeric and/or of small sizes may be easier to work with from a developer's perspective.
  - A candidate key that is a determinant in a functional dependency in F rather than F^+ may be a good choice because it is probably semantically obvious from the user's perspective.
  - Surrogate (i.e., artificial) key should only be used as a last resort.
    - … but IRL they are frequently used!

Key Versus Non-Key Attributes

- To review (and remember back to our lecture from modules 6.5-6.7):
- An attribute is a key attribute if it is a proper subset of any **candidate key** of R.
- Any attribute that is not a subset of any candidate key is a non-key attribute
- A candidate key is neither a key nor a non-key attribute
- Based on the above discussion, we have an alternative definition for a candidate key from this point forward:
  - A candidate key of a relation schema R fully functionally determines all attributes of R.

Module 7.3

Deriving Candidate Keys

- Deriving candidate keys using the synthesis method
- Deriving candidate keys using the decomposition method

Progress Quiz Time!

- The Progress Quiz is available in Canvas
  - You MUST complete the quiz on Canvas by 5:00 on Friday – This in-class activity does not count for points!

  - Each week we will discuss the questions, so for those of you that are in class and keeping up with things, you'll have an extra easy time with it!

- Go to http://kahoot.it and we'll get started momentarily!

Go forth and do great things!

- Remember – No class next week (April 8)

- Get started on the SQL project – it will likely take a while to complete

- Assignment 3 is due tonight at 11:59 PM!

BZAN 6354

Lecture 11

April 1, 2024

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

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