CZ2007
Introduction to Databases

Querying Relational Databases using SQL
Part--3

Dr. Quah T.S., Jon

School of Computer Science and Engineering
Nanyang Technological University, Singapore

Summary and roadmap

- Introduction to SQL
- SELECT
  FROM
  WHERE
- Eliminating duplicates
- Renaming attributes
- Expressions in SELECT Clause
- Patterns for Strings
- Ordering
- Joins
- Subquery
- Aggregations
- UNION, INTERSECT, EXCEPT

- Next
  - NULL
  - Outerjoin
  - Insert/Delete tuples
  - Create/Alter/Delete tables
  - Constraints (primary key)
  - Views
  - More constraints
  - Triggers
  - Indexes

NULL in SQL

- In SQL, whenever we want to leave a value blank, we set it as NULL
- The DBMS regards NULL as "an unknown value"
- This makes sense but it leads to a lot of complications…

Issues with NULL

- Any arithmetic operations involving NULL would result in NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT Price * 10
  FROM Product
  WHERE PName = 'iPad 2'

Price
6680

Issues with NULL (cont.)

- Any arithmetic operations involving NULL would result in NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT Price * 10
  FROM Product
  WHERE PName = 'iPhone xx'

Price
NULL

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT *
  FROM Product
  WHERE Price < 1000

PName	Price
iPhone 4	888
iPad 2	668

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT *
  FROM Product
  WHERE Price >= 1000

PName	Price
EOS 550D	1199

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE

- SELECT *
  FROM Product
  WHERE Price < 1000
  OR Price >= 1000

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

PName	Price
iPhone 4	888
iPad 2	668
EOS 550D	1199

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE

- SELECT *
  FROM Product
  WHERE Price < 1000
  OR Price >= 1000
  OR Price = NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

PName	Price
iPhone 4	888
iPad 2	668
EOS 550D	1199

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE
- Use IS NULL to check whether a value is NULL

- SELECT *
  FROM Product
  WHERE Price < 1000
  OR Price >= 1000
  OR Price IS NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE

- SELECT *
  FROM Product
  WHERE Price <> NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

PName	Price

Issues with NULL (cont.)

- Any comparison involving NULL results in FALSE
- Use IS NOT NULL to check whether a value is not NULL

- SELECT *
  FROM Product
  WHERE Price IS NOT NULL

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

⇩

PName	Price
iPhone 4	888
iPad 2	668
EOS 550D	1199

Issues with NULL (cont.)

- What about GROUP BY?
- NULLs are taken into account in group formation

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT Price,
  COUNT(*) AS Cnt
  FROM Product
  GROUP BY Price

Price	Cnt
NULL	1
668	1
888	1
1199	1

Issues with NULL (cont.)

- What about joins?

Phone

PName	Price
iPhone 4	888
iPhone xx	NULL

Tablet

PName	Price
ipad 2	668
IdeaPad	NULL

- SELECT P.PName, T.PName
  FROM Phone P, Tablet T
  WHERE P.Price > T.Price

PName	PName
iPhone 4	ipad 2

Issues with NULL (cont.)

- What about joins?

Phone

PName	Price
iPhone 4	888
iPhone xx	NULL

Tablet

PName	Price
ipad 2	668
IdeaPad	NULL

- SELECT P.PName, T.PName
  FROM Phone P, Tablet T
  WHERE P.Price = T.Price

PName	PName

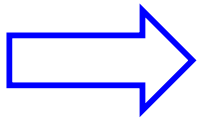
Issues with NULL (cont.)

- NULLs are ignored in
  - SUM,
  - MIN,
  - MAX

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT SUM(Price) as SumPrice
  FROM Product



SumPrice
2755

Issues with NULL (cont.)

- NULLs are ignored in AVG

Product

PName	Price
iPhone 4	888
iPad 2	668
iPhone xx	NULL
EOS 550D	1199

- SELECT AVG(Price) as AvgPrice
  FROM Product

AvgPrice
918

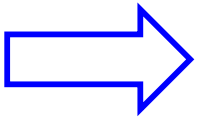
Issues with NULL (cont.)

- SQL ignores NULLs when counting the number of values in a column

Product

PName	Price
iPhone 4	888
iPad 2	668
NULL	NULL
EOS 550D	1199

- SELECT COUNT(Price)
  FROM Product



3

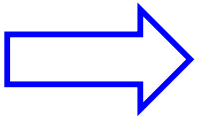
Issues with NULL (cont.)

- But NULLs are still counted in COUNT(*)

Product

PName	Price
iPhone 4	888
iPad 2	668
NULL	NULL
EOS 550D	1199

- SELECT COUNT(*)
  FROM Product



4

Issues with NULL (cont.)

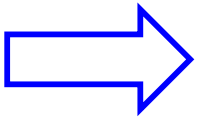
- But NULLs are still counted in COUNT(*)

Product

PName	Price
iPhone 4	888
iPad 2	668
Milestone	NULL
EOS 550D	1199

Avoid NULLs in tables whenever possible.
This can usually be achieved with proper schema design.

- SELECT COUNT(*)
  FROM Product



4

Summary and roadmap

- Introduction to SQL
- SELECT
  FROM
  WHERE
- Eliminating duplicates
- Renaming attributes
- Expressions in SELECT Clause
- Patterns for Strings
- Ordering
- Joins
- Subquery
- Aggregations
- UNION, INTERSECT, EXCEPT
- NULL

Reference: Chapter 6.1 of our TextBook

- Next
  - Outerjoin
  - Insert/Delete tuples
  - Create/Alter/Delete tables
  - Constraints (primary key)
  - Views
  - More constraints
  - Triggers
  - Indexes

Join

Product

PName	Price
iPhone 4	888

Sold

PName	Shop
iPhone 4	Suntec

- SELECT P.PName, Price, Shop
  FROM Product AS P, Sold AS S
  WHERE P.PName = S.PName

PName	Price	Shop
iPhone 4	888	Suntec

- SELECT P.PName, Price, Shop
  FROM Product AS P JOIN Sold AS S
  ON P.PName = S.PName

Join

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec

PName	Price	Shop
iPhone 4	888	Suntec

- SELECT P.PName, Price, Shop
  FROM Product AS P JOIN Sold AS S
  ON P.PName = S.PName

Outerjoins

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec

How?

PName	Price	Shop
iPhone 4	888	Suntec
iPad 2	668	NULL

- SELECT P.PName, Price, Shop
  FROM Product AS P JOIN Sold AS S
  ON P.PName = S.PName

Outerjoins (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec

Include the left tuples even when there is no match

PName	Price	Shop
iPhone 4	888	Suntec
iPad 2	668	NULL

- SELECT P.PName, Price, Shop
  FROM Product AS P LEFT OUTER JOIN Sold AS S
  ON P.PName = S.PName

Outerjoins (cont.)

Product

PName	Price
iPhone 4	888

Sold

PName	Shop
iPhone 4	Suntec
NULL	ION

Include the right tuples even when there is no match

PName	Price	Shop
iPhone 4	888	Suntec
NULL	NULL	ION

- SELECT P.PName, Price, Shop
  FROM Product AS P RIGHT OUTER JOIN Sold AS S
  ON P.PName = S.PName

Outerjoins (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec
NULL	ION

Include both left and right tuples even if there is no match

PName	Price	Shop
iPhone 4	888	Suntec
iPad 2	668	NULL
NULL	NULL	ION

- SELECT P.PName, Price, Shop
  FROM Product AS P FULL OUTER JOIN Sold AS S
  ON P.PName = S.PName

More join type: Inner Join

Syntax

- R INNER JOIN S USING (<attribute list>)
- R INNER JOIN S ON *R.column_name = S.column_name*

Example

TableA

Column1	Column2
1	2

TableB

Column1	Column3
1	3

The INNER JOIN of **TableA** and **TableB** on Column1 will return:

TableA.Column1	TableA.Column2	TableB.Column1	TableB.Column3
1	2	1	3

SELECT * FROM TableA INNER JOIN TableB USING (Column1)

SELECT * FROM TableA INNER JOIN TableB ON TableA.Column1 = TableB.Column1

Natural Join

Syntax

R NATURAL JOIN S

Example

TableA

Column1	Column2
1	2

TableB

Column1	Column3
1	3

The NATURAL JOIN of **TableA** and **TableB** will return:

Column1	Column2	Column3
1	2	3

SELECT * FROM TableA NATURAL JOIN TableB

- The repeated columns are avoided.
- One can not specify the joining columns in a natural join.

Summary and roadmap

- Introduction to SQL
- SELECT
  FROM
  WHERE
- Eliminating duplicates
- Renaming attributes
- Expressions in SELECT Clause
- Patterns for Strings
- Ordering
- Joins
- Subquery
- Aggregations
- UNION, INTERSECT, EXCEPT
- NULL
- Outerjoin

- Next
  - Insert/Delete tuples
  - Create/Alter/Delete tables
  - Constraints (primary key)
  - Views
  - More constraints
  - Triggers
  - Indexes

Reference: Chapter 6.3 of our TextBook

Inserting One Tuple

- INSERT INTO Product VALUES('iPhone 5', 999)
- Alternative approaches:
- INSERT INTO Product(PName, Price) VALUES('iPhone 5', 999)
- INSERT INTO Product(Price, PName,) VALUES(999, 'iPhone 5')

Product

PName	Price
iPhone 4	888
iPad 2	668

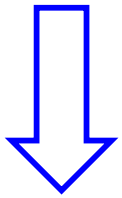
PName	Price
iPhone 4	888
iPad 2	668
iPhone 5	999

Partial Insertion

- INSERT INTO Product(PName) VALUES('iPhone 5')

Product

PName	Price
iPhone 4	888
iPad 2	668



PName	Price
iPhone 4	888
iPad 2	668
iPhone 5	NULL

Tuple Insertion via Subqueries

- The 'Sold' table is initially empty.

- INSERT INTO Sold
  SELECT PName, 'Suntec'
  FROM Product

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec

Tuple Insertion via Subqueries

- Assume that a new shop at the ION sells all products sold at the Suntec shop

- INSERT INTO Sold
  SELECT PName, 'ION'
  FROM Sold

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec

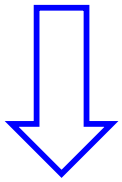
PName	Shop
iPhone 4	Suntec
iPad 2	Suntec
iPhone 4	ION
iPad 2	ION

Tuple Deletion

- DELETE FROM Sold
  WHERE PName = 'iPad 2'

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec



PName	Shop
iPhone 4	Suntec

Tuple Deletion (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec

- Remove from the Suntec shop all products over 800 dollars

PName	Shop
iPad 2	Suntec

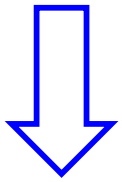
- DELETE FROM Sold
  WHERE Shop = 'Suntec'
  AND Sold.PName IN
  (SELECT P.PName FROM Product AS P
  WHERE Price > 800)

Deleting All Tuples

- DELETE FROM Sold;

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec



PName	Shop

Exercise

Sold

PName	Shop
iPhone 4	Suntec
iPad 2	Suntec
iPhone 4	ION

- Remove (i) any product from Suntec that is also sold at ION

  and (ii) any product from ION that is also sold at Suntec
- DELETE FROM Sold
  WHERE (Sold.Shop = 'Suntec' AND
  Sold.PName IN
  (Select S1.PName FROM Sold AS S1
  WHERE S1.Shop = 'ION'))
  OR (Sold.Shop = 'ION' AND Sold.PName IN
  (Select S2.PName FROM Sold AS S2
  WHERE S2.Shop = 'Suntec'))

Tuple Update

- Assume that the price of iPhone 4 should be reduced to 777

- UPDATE Product
  SET Price = 777
  WHERE PName = 'iPhone 4'

Product

PName	Price
iPhone 4	888
iPad 2	668

PName	Price
iPhone 4	777
iPad 2	668

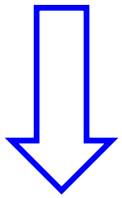
Tuple Update (cont.)

- Assume that Google buys Apple and reduces the prices of all its products by 100

- UPDATE Product
  SET Price = Price – 100,
  Company = 'Google'
  WHERE Company = 'Apple'

Product

PName	Price	Company
iPhone 4	888	Apple
iPad 2	668	Apple



PName	Price	Company
iPhone 4	788	Google
iPad 2	568	Google

Tuple Update (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668
Milestone	798

Maker

PName	Company
iPhone 4	Apple
iPad 2	Apple
Milestone	Motorola

- Reduce the price of all Apple products by 10%
- UPDATE Product
  SET Price = Price * 0.9
  WHERE Product.PName IN
  (SELECT Maker.PName FROM Maker
  WHERE Company = 'Apple')

Tuple Update (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668
Milestone	798

Maker

PName	Company
iPhone 4	Apple
iPad 2	Apple
Milestone	Motorola

- Reduce the price of all products by 10%
- UPDATE Product
  SET Price = Price * 0.9

Tuple Update (cont.)

Product

PName	Price
iPhone 4	888
iPad 2	668
Milestone	798

- Set the price of every product to half of the price of iPhone 4
- UPDATE Product
  SET Price =
  (SELECT P.Price / 2
  FROM Product AS P
  WHERE P.PName = 'iPhone 4')

Exercise

Beer

Name	Maker	Price

Wine

Name	Maker	Price

- Update the price of every beer to the average price of the wine by the same maker
- UPDATE Beer
  SET Beer.Price =
  (SELECT AVG(Wine.Price)
  FROM Wine
  WHERE Wine.Maker = Beer.Maker)

Exercise

Beer

Name	Maker	Price

Wine

Name	Maker	Price

- Delete any beer by a maker that does not produce any wine
- DELETE FROM Beer
  WHERE NOT EXISTS
  (SELECT *
  FROM Wine
  WHERE Wine.Maker = Beer.Maker)

Exercise

Beer

Name	Maker	Price

Wine

Name	Maker	Price

- For each beer, if there does not exist a wine with the same name, then create a wine with the same name and maker, but twice the price
- INSERT INTO Wine
  SELECT B.Name, B.Maker, B.Price * 2
  FROM Beer AS B
  WHERE NOT EXISTS
  (SELECT * FROM Wine
  WHERE Wine.Name = B.Name)

Table Creation

Product	
PName	Price

- CREATE TABLE Product(
  PName VARCHAR(30),
  Price INT);
- In general:
  CREATE TABLE <table name> (
  <column name 1> <column type 1>,
  <column name 2> <column type 2>,
  …);

Column Types

- INT or INTEGER (synonyms)
- REAL or FLOAT (synonyms)
- CHAR(n): fixed-length string of n characters
- VARCHAR(n): variable-length string of up to n characters
- DATE: = In 'yyyy-mm-dd' format
- …

Summary and roadmap

- Introduction to SQL
- SELECT
  FROM
  WHERE
- Eliminating duplicates
- Renaming attributes
- Expressions in SELECT Clause
- Patterns for Strings
- Ordering
- Joins
- Subquery
- Aggregations
- UNION, INTERSECT, EXCEPT
- NULL
- Outerjoin
- Insert/Delete tuples
- Create/Alter/Delete tables

- Next
  - Constraints (primary key)
  - Views
  - More constraints
  - Triggers
  - Indexes

Table Creation (cont.)

Product

PName	Price

- CREATE TABLE Product(
  PName VARCHAR(30), Price INT);

- What if we want to make sure that all products have distinct names?

- Solution: declare PName as PRIMARY KEY or UNIQUE

PRIMARY KEY

Product	
PName	Price

- CREATE TABLE Product(
  PName VARCHAR(30), Price INT);

- What if we want to make sure that all products have distinct names?

- CREATE TABLE Product(
  PName VARCHAR(30), Price INT,
  PRIMARY KEY (PName));

UNIQUE

Product

PName	Price

- CREATE TABLE Product(
  PName VARCHAR(30), Price INT);

- What if we want to make sure that all products have distinct names?

- CREATE TABLE Product(
  PName VARCHAR(30), Price INT,
  UNIQUE(PName));

PRIMARY KEY (cont.)

Catalog

PName	Shop	Price

- We want to make sure that there is no duplicate PName with the same Shop

- CREATE TABLE Catalog(
  PName VARCHAR(30),
  Shop VARCHAR(30),
  Price INT,
  PRIMARY KEY (PName, Shop));

UNIQUE (cont.)

Catalog

PName	Shop	Price

- We want to make sure that there is no duplicate PName with the same Shop

- CREATE TABLE Catalog(
  PName VARCHAR(30),
  Shop VARCHAR(30),
  Price INT,
  UNIQUE (PName, Shop));

PRIMARY KEY vs. UNIQUE

Catalog

PName	Shop	Price

- Difference 1
  - Only ONE set of attributes in a table can be declared as PRIMARY KEY
  - But we can declare multiple sets of attributes as UNIQUE

- CREATE TABLE Catalog(PName VARCHAR(30), Shop VARCHAR(30), Price INT,
  UNIQUE (PName, Shop), UNIQUE (Shop, Price)
  );

PRIMARY KEY vs. UNIQUE

Catalog

PName	Shop	Price

- Difference 2
  - If set of attributes are declared as PRIMARY KEY, then none of these attributes can be NULL
  - UNIQUE attributes still allow NULLs
- CREATE TABLE Catalog(
  PName VARCHAR(30),
  Shop VARCHAR(30),
  Price INT,
  UNIQUE (PName, Shop));

NOT NULL

Catalog

PName	Shop	Price

- We want to make sure that the price of each product is not NULL

- CREATE TABLE Catalog(
  PName VARCHAR(30),
  Shop VARCHAR(30),
  Price INT NOT NULL);

NOT NULL (cont.)

Catalog

PName	Shop	Price

- We want to make sure that the price as well as PName of each product is not NULL

- CREATE TABLE Catalog(
  PName VARCHAR(30) NOT NULL,
  Shop VARCHAR(30),
  Price INT NOT NULL);

NOT NULL (cont.)

Catalog

PName	Shop	Price

- CREATE TABLE Catalog(
  PName VARCHAR(30) NOT NULL,
  Shop VARCHAR(30),
  Price INT NOT NULL);

- NOT NULL may prevent partial insertions

- INSERT INTO Product(PName)
  Values('iPhone 5')

Error!

DEFAULT

Catalog

PName	Shop	Price

- We want to specify that, by default, the shop and price of a product is 'Suntec' and 1, respectively

- CREATE TABLE Catalog(
  PName VARCHAR(30) DEFAULT 'Suntec',
  Shop VARCHAR(30),
  Price INT DEFAULT 1);

Combination

Catalog

PName	Shop	Price

- We want to specify that, by default, the shop and price of a product is 'Suntec' and 1, respectively. In addition, the shop should not be NULL

- CREATE TABLE Catalog(
  PName VARCHAR(30) NOT NULL DEFAULT 'Suntec',
  Shop VARCHAR(30),
  Price INT DEFAULT 1);

Table Deletion

Catalog

PName	Shop	Price

- DROP TABLE Catalog

Table Modification

- Adding a new attribute
- ALTER TABLE Catalog
  ADD Price INT
- We can also add some declarations
- ALTER TABLE Catalog
  ADD Price INT NOT NULL
  DEFAULT 1

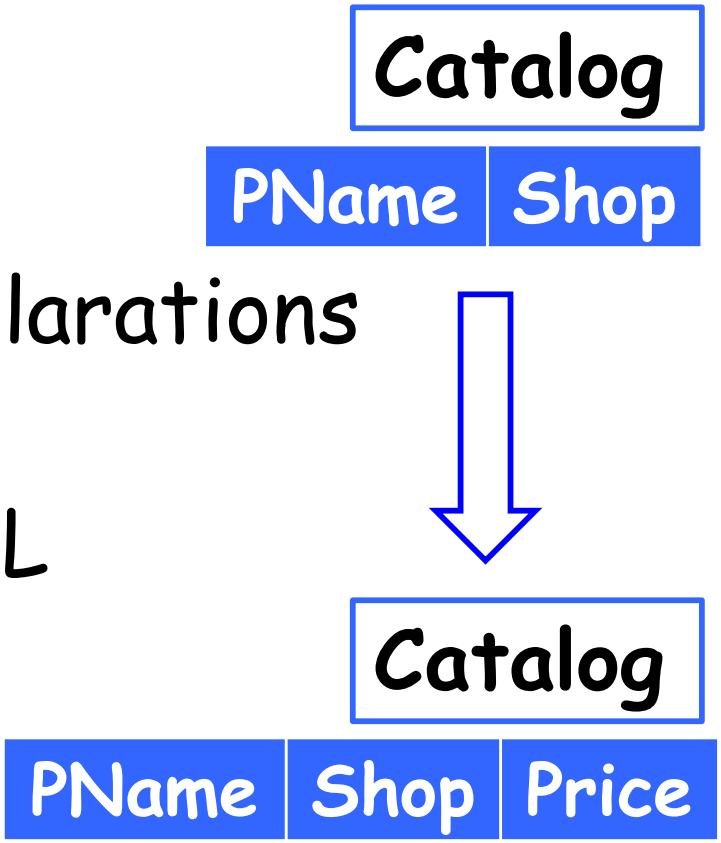
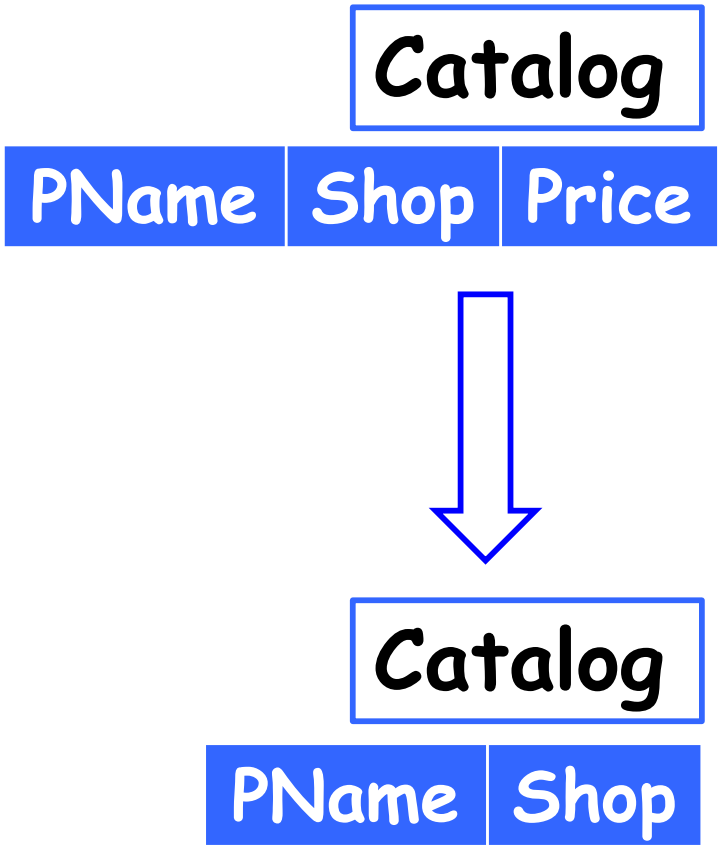


Table Modification

- Deleting an attribute
- ALTER TABLE Catalog DROP Price



Summary and roadmap

- Introduction to SQL
- SELECT
  FROM
  WHERE
- Eliminating duplicates
- Renaming attributes
- Expressions in SELECT Clause
- Patterns for Strings
- Ordering
- Joins
- Subquery
- Aggregations
- UNION, INTERSECT, EXCEPT
- NULL
- Outerjoin
- Insert/Delete tuples
- Create/Alter/Delete tables
- Constraints (primary key)

- Next
  - Views
  - More constraints
    - FOREIGN KEY
    - CHECK
    - ASSERTION
    - Triggers
  - Indexes

Reference: Chapter 6.5 of our TextBook