

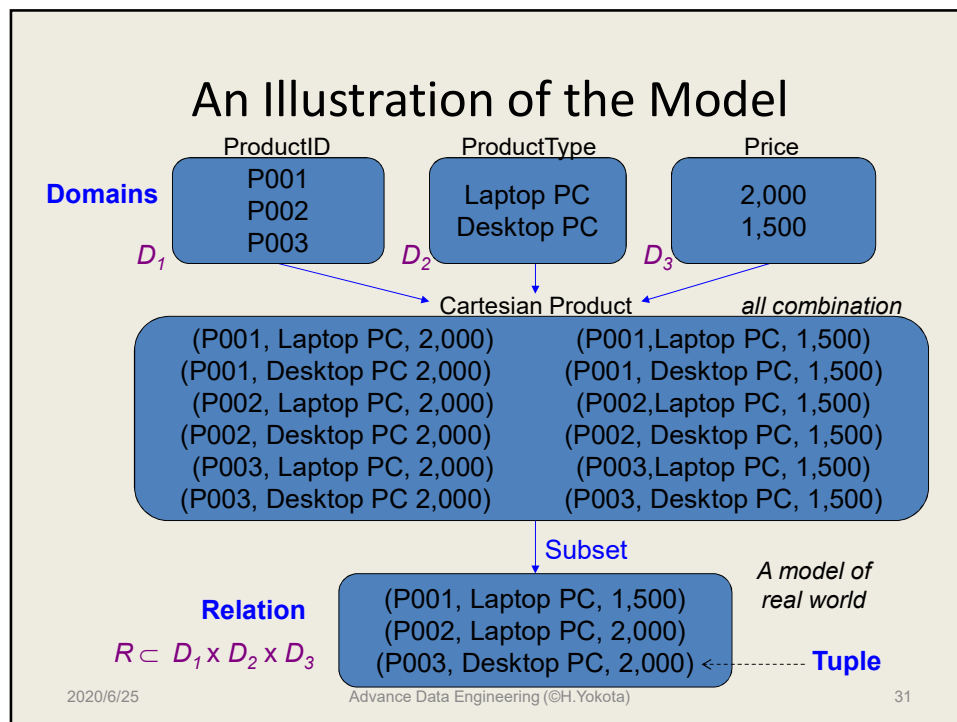
A Short Course for Relational Databases

Advanced Data Engineering

A Brief Introduction of Relational DB

- The **Relational Model**
 - Proposed by E.F. Codd (IBM) in 1970
 - 1981 ACM Turing Award
 - 2003.4.18 Died (78 years old)
 - Based on the Set Theory
- Basic Concept
 - A **relation** R is a subset of Cartesian product of **domain** D_i ($1 < j < n$)

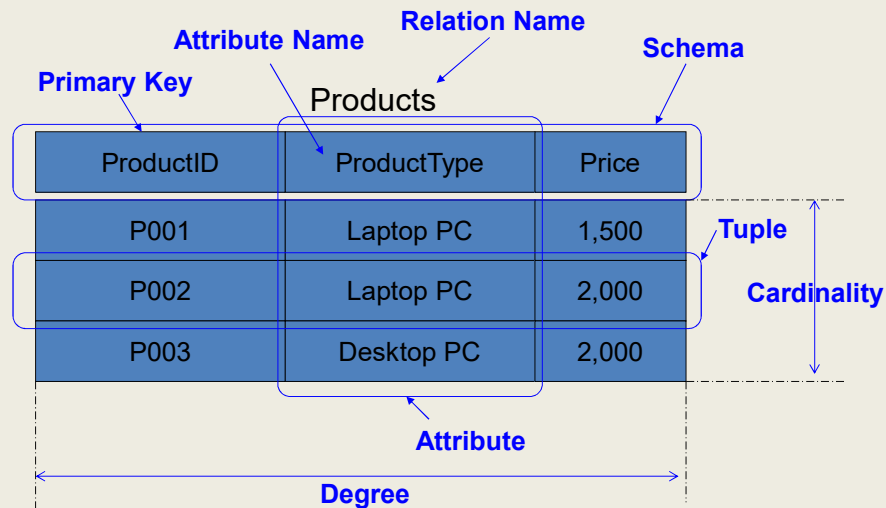
$$R \subset D_1 \times D_2 \times \dots \times D_j \times \dots \times D_n$$
 - A **tuple** t is an element of a relation: $t \in R$



Terminologies of Relational DB

- The number of tuples is called **cardinality**
- A domain in a relation is called an **attribute**
 - Each attribute has an **attribute name**
 - The collection attribute names is called a **schema**
 - n is called a **degree** of the relation
- A set of attributes which uniquely identifies each tuple in a relation is called a **key** of the relation
 - There are some **candidate keys**
 - A **primary key** and other **alternate keys**
- A relation can be seen as a table.
 - A tuple is a row of the table.
 - An attribute is a column of the table.

A Table Image of RDB



2020/6/25

Advance Data Engineering (©H.Yokota)

33

Relational Database Operations

- **Relational Algebra**
 - A collection of operators
 - take relations as their operands and return relations as their results
- **Relational Calculus**
 - Specification of requiring results
 - An example: $\{ (t_r, t_s) \mid R(t_r) \wedge S(t_s) \wedge t_r[X] \theta t_s[Y] \}$
- Algebra is *prescriptive* while Calculus is *descriptive*
 - Both expressive power is identical

2020/6/25

Advance Data Engineering (©H.Yokota)

34

Relational Algebra Operations

- Set Operations
 - Union ($R \cup S$)
 - Intersection ($R \cap S$)
 - Difference ($R - S$)
 - Cartesian Product ($R \times S$)
- Dedicated Relational Algebra Operations
 - Projection ($\pi_{\text{attribute-list}} R$)
 - Selection ($\sigma_{\text{conditions}} R$)
 - Join (θ -Join: $R \bowtie_{\theta} S$, eq-Join: $R \bowtie_{r=s} S$)
 - Division ($R \div S$)

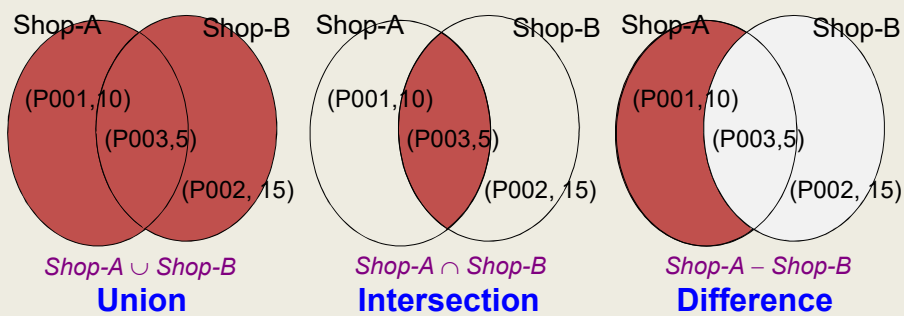
2020/6/25

Advance Data Engineering (©H.Yokota)

35

Set Operations on RDB

Shop-A		Shop-B	
ProductID	Stocks	ProductID	Stocks
P001	10	P002	15
P003	5	P003	5



2020/6/25


Advance Data Engineering (©H.Yokota)

36

Projection π

- Derive specified attributes with eliminating duplication

ProductID	ProductType	Price
P001	Laptop PC	1,500
P002	Laptop PC	2,000
P003	Desktop PC	2,000



ProductType
Laptop PC
Desktop PC

$\pi_{\text{ProductType}}$ $\text{Products} = \{ t[\text{ProductType}] \mid \text{Products}(t) \}$

- * Delta Projection: without eliminating duplication

2020/6/25


Advance Data Engineering (©H.Yokota)

37

Selection σ

- Derive all tuples satisfying specified conditions

ProductID	ProductType	Price
P001	Laptop PC	1,500
P002	Laptop PC	2,000
P003	Desktop PC	2,000



ProductID	ProductType	Price
P002	Laptop PC	2,000
P003	Desktop PC	2,000

$\sigma_{\text{Price} > 1,800}$ $\text{Products} = \{ t \mid \text{Products}(t) \wedge t[\text{Price}] > 1,800 \}$

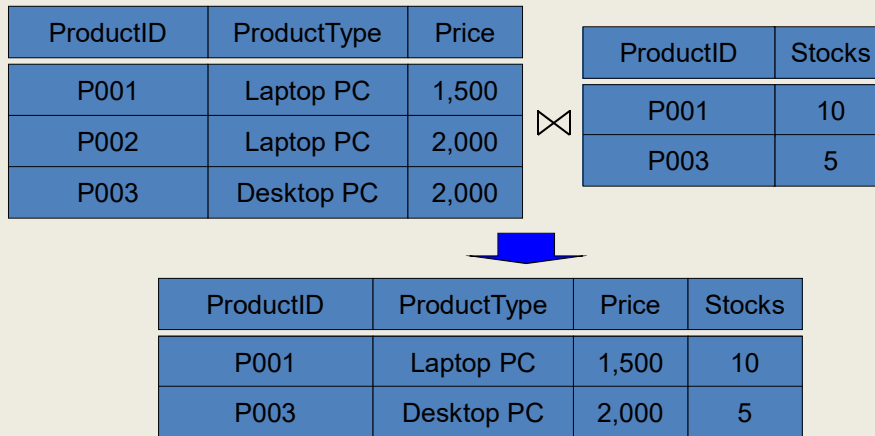
2020/6/25

Advance Data Engineering (©H.Yokota)

38

Join ⋈

- Combine two relations with specified conditions



Products ⋈_{ProductID = ProductID} ShopA

$= \{ (t1, t2) \mid \text{Products}(t1) \wedge \text{ShopA}(t2) \wedge t1[\text{ProductID}] = t2[\text{ProductID}] \}$

2020/6/25

Advance Data Engineering (©H. Yokota)

39

Query Example

- Consider a relational algebra query to derive “ProductName” and “Price” of products categorized as “Computer”, whose stocks are more than ten in ShopA or ShopB, under the following four relations:
 - Products (ProductID, ProductName, ProductType, Price)
 - Categories (ProductType, Category)
 - ShopA (ProductID, Stock)
 - ShopB (ProductID, Stock)

$\pi_{\text{ProductName, Price}} ($
 $((\text{Products} \bowtie_{\text{ProductType} = \text{ProductType}} (\sigma_{\text{Category} = \text{“Computer”}} \text{Categories})))$
 $\bowtie_{\text{ProductID} = \text{ProductID}} ((\sigma_{\text{Stocks} > 10} \text{ShopA}) \cup (\sigma_{\text{Stocks} > 10} \text{ShopB}))$
 $)$
 $)$

2020/6/25

Advance Data Engineering (©H. Yokota)

40

Write a query from Google Form

```

 $\pi$  ProductName, Price (
  ((Products  $\bowtie$  ProductType = ProductType ( $\sigma$  Category = "Computer" Categories))
     $\bowtie$  ProductID = ProductID (( $\sigma$  Stocks > 10 ShopA)  $\cup$  ( $\sigma$  Stocks > 10 ShopB))
  )
)

Projection[ProductName, Price] (
  Join[ProductID = ProductID] (
    Join[ProductType = ProductType] (
      Products, Selection[Category = "Computer"] (Categories)),
    Union (Selection[Stock > 10] (ShopA), Selection[Stock > 10] (ShopB))
  )
)

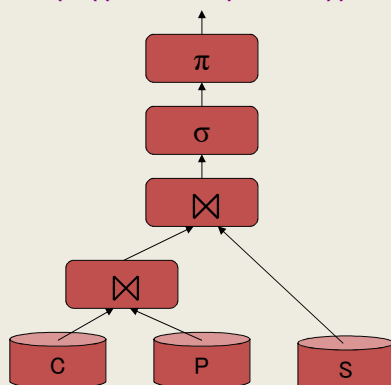
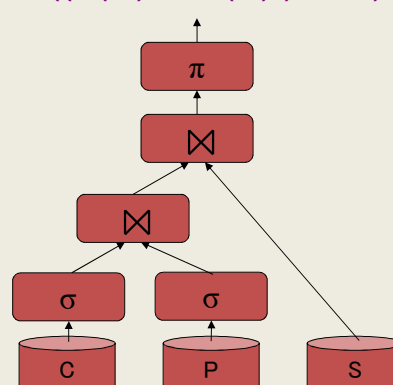
```

2020/6/25

Advance Data Engineering (©H.Yokota)

41

Examples of Query Trees

 $\pi(\sigma((C \bowtie P) \bowtie S))$

 $\pi((\sigma(C) \bowtie \sigma(P)) \bowtie S)$


C: Categories, P: Products, S: Shop-A

2020/6/25

Advance Data Engineering (©H.Yokota)

42

Question (2-1)

- Write a relational algebra query to derive “ProductID” and “Category” of products whose price is more than \$2,000 and whose stock are less than ten stocks in both ShopA and ShopB, under the following four relations:
 - Products (ProductID, ProductName, ProductType, Price)
 - Categories (ProductType, Category)
 - ShopA (ProductID, Stock)
 - ShopB (ProductID, Stock)

2020/6/25

Advance Data Engineering (©H.Yokota)

43

Query Languages

- **SQL** (Structured Query Language)
 - Developed in SystemR Project of IBM
 - Standardized (ANSI/ISO, JIS)
 - Data Definition Language (DDL)
 - + Data Manipulation Language (DML)
 - Directed or Embedded SQL
 - From some programming languages
- Basic Syntax (DML)
 - SELECT** List of Attribute Names
 - FROM** List of Relation Names
 - WHERE** Conditions;

2020/6/25

Advance Data Engineering (©H.Yokota)

44

SQL Examples (1)

- **SELECT DISTINCT ProductType**
FROM Products;
 - Projection for the Products Table
 - $\pi_{\text{ProductType}} \text{ Products}$
 - $\{ t[\text{ProductType}] \mid \text{Products}(t) \}$
- **SELECT ProductType**
FROM Products;
 - Delta Projection
 - Without eliminating duplication

2020/6/25

Advance Data Engineering (©H.Yokota)

45

SQL Examples (2)

- **SELECT ProductID, ProductType, Price**
FROM Products
WHERE Price > 1,800;
 - Selection for the Products Table
 - $\sigma_{\text{Price} > 1,800} \text{ Products}$
 - $\{ t \mid \text{Products}(t) \wedge t[\text{Price}] > 1,800 \}$
- **SELECT ***
FROM Products
WHERE Price > 1,800
AND ProductType = "Laptop PC";

2020/6/25

Advance Data Engineering (©H.Yokota)

46

SQL Examples (3)

- **SELECT ***
FROM Products, ShopA
WHERE Products.ProductID=ShopA.ProductID;
– Join between the Products and Shop-A Tables
 - $\text{Products} \bowtie_{\text{Product-ID} = \text{Product-ID}} \text{Shop-A}$
 - $\{(t1, t2) \mid \text{Products}(t1) \wedge \text{ShopA}(t2) \wedge t1[\text{ProductsID}] = t2[\text{ProductID}]\}$
- **SELECT ***
FROM Products
WHERE ProductID IN
(SELECT ProductID FROM ShopA);

2020/6/25

Advance Data Engineering (©H.Yokota)

47

A Query Example

- Consider another relation “Categories” having two attributes: “ProductType” and “Category”
 - Laptop PCs and Desktop PCs are categorized as PCs in the table
- Write an SQL query to derive stocks of PC sold in the ShopA at the price higher than \$1,800
- Consider Relational Algebra Expressions for the query

```
SELECT Stocks
FROM ShopA
WHERE ProductID IN
(SELECT ProductID
FROM Products
WHERE Price > 1,800
AND ProductType IN
(SELECT ProductType
FROM Categories
WHERE Category = PC));
```

2020/6/25

Advance Data Engineering (©H.Yokota)

48

Another Example

```
SELECT ShopA.Stocks
FROM   ShopA, Products, Categories
WHERE  ShopA.ProductID = Products.ProductID
AND    Products.ProductType =
        Categories.ProductType
AND    Products.Price > 1,800
AND    Categories.Category = PC;
```

2020/6/25

Advance Data Engineering (©H.Yokota)

49

Question (2-2)

- Write an SQL query to derive “ProductID” and “Category” of products whose price is more than \$1,500 and whose stock are less than ten stocks in both ShopA and ShopB, under the following four relations:
 - Products (ProductID, ProductName, ProductType, Price)
 - Categories (ProductType, Category)
 - Shop-A (ProductID, Stock)
 - Shop-B (ProductID, Stock)

2020/6/25

Advance Data Engineering (©H.Yokota)

50

Aggregate Functions (1)

- Count
 - SELECT COUNT(*)
FROM Products
 - SELECT DISTINCT COUNT(ProductType)
FROM Products
- Summation
 - SELECT SUM(Stocks)
FROM ShopA, Products
WHERE ShopA.ProductID=Products.ProductID
AND Products.Price > 1,800
- Other Functions: AVG(), MAX(), MIN()

2020/6/25

Advance Data Engineering (©H.Yokota)

51

Aggregate Functions (2)

- Group By
 - SELECT Product-Type, AVG(Price)
FROM Products
GROUP BY ProductType
- Having
 - SELECT ProductType
FROM Products
GROUP BY ProductType
HAVING AVG(Price) > 1,800

Product Type	AVG(Price)
Laptop PC	1,750
Desktop PC	2,000

2020/6/25

Advance Data Engineering (©H.Yokota)

52

Question (2-3)

- Assume the following four relations:
 - Products (ProductID, ProductName, ProductType, Price)
 - Categories (ProductType, Category)
 - ShopA (ProductID, Stock)
- A) Write an SQL query to derive average stocks in ShopA per category of products whose prices are less than \$2,000. The SQL query lists the average stock with the category name.
- B) Write an SQL query to derive category name of products whose prices are less than \$2,000, and where the average stocks of the category in ShopA is greater than 5.

2020/6/25

Advance Data Engineering (©H.Yokota)

53

Access Methods

- Sequential access vs. Direct access
- Indexing
 - To speed up retrieval
- Inverted File
 - Imagine index pages of a book
- Tree structured Index: B-trees
 - Common: most relational system support B-trees
 - B+tree support both direct & sequential accesses
- Hashing
 - Direct access by using a Hash function

2020/6/25

Advance Data Engineering (©H.Yokota)

54

Transaction

- A logic unit of work having ACID properties.
 - A: Atomicity
 - All-or-nothing (Commit: all, Rollback: nothing)
 - C: Consistency
 - A transaction transforms a consistent state of the database into another consistent state, without necessarily preserving consistency at all intermediate points.
 - I: Isolation
 - Transactions are isolated from one another. Any given transaction's update are concealed from all the rest transactions running concurrently, until that transaction commits.
 - D: Durability
 - Once a transaction commits, its update survive, even if there is a subsequent system crash.

2020/6/25

Advance Data Engineering (©H.Yokota)

55

Serializability

- Transactions should behave as if they were run **serially**
 - Even though their execution may overlap in time
- **Concurrency control** is required
 - Using Locks
 - Two Phase Lock (2PL) is essential
 - Deadlocks can be occurred
 - Using Timestamp
- We will consider distributed lock/commit

2020/6/25

Advance Data Engineering (©H.Yokota)

56

Recovery

- Transaction Recovery
 - Rollback by the user, or for serializability
 - States are kept on volatile memory
- System Recovery
 - Caused by system failures
 - States are kept on non volatile memory
 - Recovered by checkpoint and logs
- Media Recovery
 - Caused by media failures

2020/6/25

Advance Data Engineering (©H.Yokota)

57

Transaction control in SQL

- BEGIN WORK
 - Declaration of starting a transaction
- COMMIT WORK
 - Declaration of terminating a normal transaction
- ROLLBACK WORK
 - Abortion of a transaction
- We will also consider other controls for extended transaction models

2020/6/25

Advance Data Engineering (©H.Yokota)

58

OLTP / OLAP

- OLTP: On Line Transaction Processing
 - Typically handling simultaneous fixed form queries for data entry and retrieval
 - Banking system, Trading system, POS system, etc.
- OLAP: On Line Analytical Processing
 - Handling ad hoc complex queries for strategic decision support of managers

2020/6/25

Advance Data Engineering (©H.Yokota)

59

Recent Trend

- ACID properties are too strong for scalability
 - Some applications does not require strong consistency
 - Such as Twitter, Facebook, etc.
 - How about Amazon ?
- BASE transaction model (for KVS)
 - Basically Available
 - Soft state
 - Eventually consistent

2020/6/25

Advance Data Engineering (©H.Yokota)

60