- **Database**

  A collection of data or information that is used and managed by a group or individual person

- **Data**

  A value for describing real-world objects. An event, a phenomenon, or an idea can be data.

- **Data vs. Information**

  Data is primary description about a certain fact. Information is description about a fact derived from data.

- **Transient data**

  Dependent on a process. Exist only when a process is operated. ex) data store in RAM.

- **Persistent data**

  Independent on a process. Exist regardless of a process's life cycle.

  Standard database means a collection of persistent data.

- **Database with Data**

  Database is a collection of all related(associated) data.

  Data in database should have common property and should be stored in a computer to be easily managed and processed.

- **General operations using data in database**

  Computation / Insertion and deletion / Search / Sort / Analysis etc.

- **Database Management System (DBMS)**

  DBMS contains data and information about a particular enterprise.

  DBMS consists of Collection of interrelated data / Set of programs to access the data / An environment that is both convenient and efficient to use.

- **Persons involved in DBMS**

  - User (End User)

    Accesses database by using application.

  - Application Programmer

    Develops application programs that utilize database.

  - Database Administrator (DBA)

    Manages and supervises database and DBMS.

  - DBMS Developer

    Designs and develops DBMS.

- **File System**

  System program as a part of OS. Support to read and write data (or programs) by accessing hard disc. Data managed in a program are stored in an individual file.

- **Drawbacks of Using FS to Store Data**

  - Data redundancy and inconsistency

    Data are scattered. Duplication of information in different files.

  - Difficulty in accessing data

    Need to write a new program to carry out new tasks.

  - Integrity problems

    Restrict to make integrity constraints. Hard to add new constraints or change existing ones.

  - Atomicity problems

    Failures may leave database in an inconsistent state.

  - Security problems

    Providing users with all data is dangerous.

- **Integrity**

  DB내의 데이터에 대한 정확성, 일관성, 유효성, 신뢰성을 보장하기 위해 데이터 변경 혹은 수정 시 여러 가지 제한을 두어 데이터의 정확성을 보증하는 것.

- **Integrity Constraints**

  DB의 완전성을 높이기 위해서 DBMS가 체크하는 데이터의 조건. DBMS가 체크 정도를 강하게 하면 데이터의 정확함, 완전성은 높아짐.

- **Atomicity**

  트랜잭션이 지녀야 할 성질의 하나. 시스템의 어떤 상황 하에서도 한 트랜잭션에 대한 모든 연산들의 결과가 DB에 모두 반영되든가 아니면 전혀 반응되지 않아야 함을 의미하는 성질.

- **View of Data: Levels of Abstraction**

  DBMS does not reveal ways of recording and managing data to users.

  - Physical level: Information of data storage

    Describes how a record is stored. Abstracted to programmers and users. Mainly, DBMS developer, sometimes DBA.

  - Logical level: Information of data type

    Describes data stored in DB, and the relationships among the data. Abstracted to users. Mainly, DBA.

  - View level: Information of user application

    Describes data information opened to users Mainly, application programmer, sometimes user requests.

- **View of Data: Schemas**

  Similar to types and variables in programming languages. Structure or design of DB.

  Types of schemas with regard to level of abstraction.

  - Internal schema (Physical schema)

    DB design at the physical level. Define physical structure of DB such as location of data in disc by DBMS developer

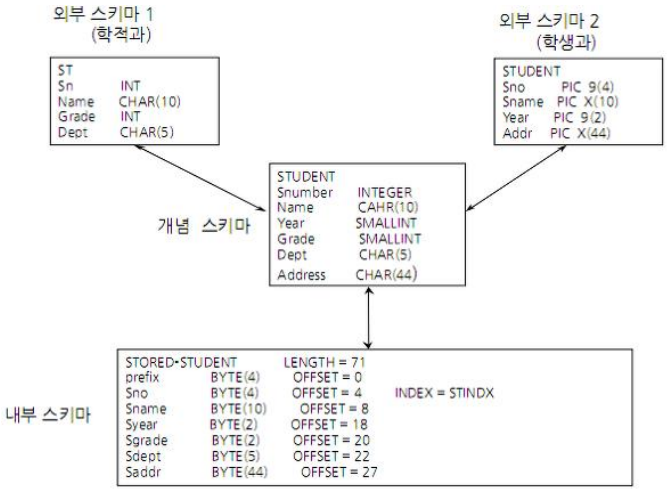
  - Conceptual schema (Logical schema)

    DB design at the logical level. Define logical structure of DB such as relation between data by DBA.

  - External schema (Sub schema)

    DB design at the view level. Define views that user or programmer requires by DB user or application programmer.

  In general, the database schema is conceptual level.



- **View of Data: Instance**

  The actual content of the database at a particular point in time.

  In general, instance varies but schema is fixed.

- **View of Data: Physical Data Independence**

  The ability to modify the physical schema without changing the logical schema.

- **Relational Model**

  Describe data and relations among data using tables and record-based model. Construct a DB from a set of record. Each record consists of one or more field (also called attribute)

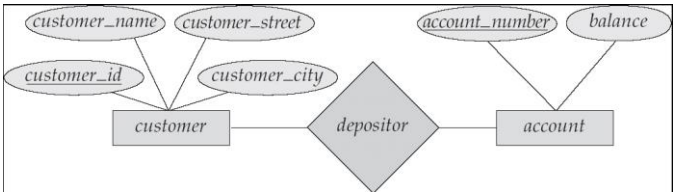
- **The Entity-Relationship Model**

  Models real-world data as a collection of entities and relationships.

  Entity is a "thing" or "object" that is distinguishable from other objects. Described by a set of attributes. e.g.) Bank account entity having account number and balance as attributes.

  Relationship is an association among several entities. e.g) Relationship between bank account and customer: Depositor (or Owner).

  Represented diagrammatically by an entity-relationship diagram



  타원: entity의 attribute / 사각형: entity / 마름모: relationship

- **Data Definition Language (DDL)**

  Used to define schema. Defined the DB schema / Generate tables and update a data dictionary / Data dictionary contains metadata (i.e. data about data) / Define storage structure and access methods / Define integrity constraints: Domain, Authorization

- **Data Manipulation Language (DML, Query Language)**

  Used to search for data and update (renew) DB. Language for accessing and manipulating the data. Searching, deleting, updating the stored data or adding a new data.

  SQL is the most widely used query language

- **Procedural DML**

  User specifies what data is required and how to get those data.

- **Declarative (non-procedural) DML**

  User specifies what data is required without specifying how to get those data.

- **DB administrator**

  Definition of storage structure and access method / Definition and modification of schema / Granting users authority to access the DB / Backing up data / Conducting performance monitoring and DB tuning.

- **DB System in Functional View: Storage management**

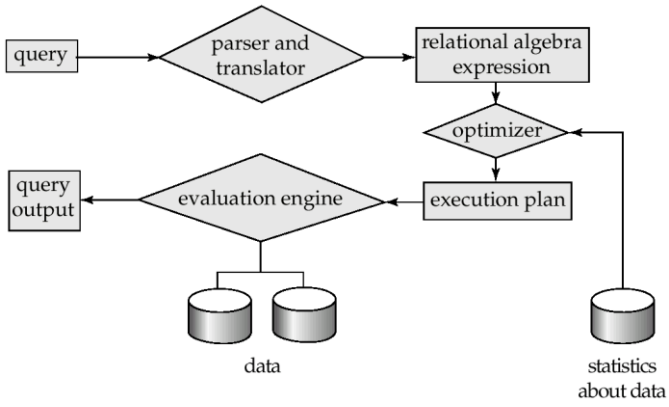
  A program that provides the interface between the low-level data stored in the database and the DB and the application programs and query processor.

  Maintain the DB to efficiently store, retrieve, and update data.

  Issues: Storage access / File organization / Indexing and hashing

- **DB System in Functional View: Query processing**

  Parsing and translation / Optimization / Evaluation



- **DB System in Functional View Transaction processing**

  Ensure that the database remains in a consistent state despite system failures and transaction failures.

  Control the interaction among the concurrent transactions, to ensure the consistency of the DB.

- **Key**

  Key is one or more attributes to classify tuples

- **Super Key**

  An attribute or a set of attributes that sufficiently identify a unique tuple in a relation r(R).

  An attribute that allows same values in different tuples *cannot* be a super key.

- **Candidate Key**

  A super key that has minimal number of attributes among all possible super keys.

- **Primary Key**

  A candidate key, if there are two or more candidate keys. Defined by DBA to distinguish tuples. *Not* allowed to have null value.

- **Relational Algebra**

  Procedural language. Take one or two relations as inputs and produce a new relation as a result.

- **Relational Algebra: Select Operation**

  Select all tuples that satisfy a given condition (predicate). Horizontal partitioning of a relation.

$$\sigma_p(r) = \{t | t \in r \text{ and } p(t)\}$$

  p is called the selection predicate. Use comparison operators ($=, \neq, >, \geq, <, \leq$) or logical operators (and(∧), or(∨), not(¬))

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

A	B	C	D
α	α	1	7
β	β	23	10

$$\sigma_{A=B \text{ and } D>5}(r)$$

- **Relational Algebra: Project Operation**

  Vertical partitioning of a relation

$$\Pi_{A_1, A_2, \cdots A_k}(r)$$

  where $A_1, A_2, \cdots A_k$ are attribute names and r is a relation name.

  The result is defined as the relation of k columns obtained by erasing the columns that are *not* listed.

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

A	C
α	1
β	1
β	2

$$\Pi_{A,C}(r)$$

- **Relational Algebra: Composition of Operations**

  Result of an algebra operation is a relation. A result can be used as an input of another operation.

$$\Pi_{\text{customer\_name}}\left(\sigma_{\text{customer\_city="Harrison"}}(\text{customer})\right)$$

- **Relational Algebra: Union Operation**

  Union of two results

$$r \cup s = \{t | t \in r \text{ or } t \in s\}$$

  r and s must have the same number of attributes. The attribute domain must be compatible.

  ex) $\Pi_{\text{customer\_name}}(\text{depositor}) \cup \Pi_{\text{customer\_name}}(\text{borrower})$

A	B
α	1
α	2
β	1

A	B
α	2
β	3

A	B
α	1
α	2
β	1
β	3

$$r \cup s$$

- **Relational Algebra: Set Difference Operation**

$$r - s = \{t | t \in r \text{ and } t \notin s\}$$

Conditions for $r - s$ to be valid are same as the conditions for union operation.

ex) $\Pi_{\text{customer\_name}}(\text{depositor}) \cup \Pi_{\text{customer\_name}}(\text{borrower})$

A	B
α	1
α	2
β	1

A	B
α	2
β	3

A	B
α	1
β	1

$r - s$

- **Relational Algebra: Cartesian-Product Operation**

Join information of two relations

$$r \times s = \{t, q | t \in r \text{ and } q \in s\}$$

If the number of tuples in r is n and the number of tuples in s is m, the number of tuples in $r \times s$ is $n \cdot m$.

Assume that attributes of $r(R)$ and $s(S)$ are disjoint ($R \cap S = \emptyset$). If not disjoint, then renaming must be used.

A	B
α	1
β	2

B	D	E
1	10	a
2	10	a
2	20	b
3	10	b

A	r.B	s.B	D	E
α	1	1	10	a
α	1	2	10	a
α	1	2	20	b
α	1	3	10	b
β	2	1	10	a
β	2	2	10	a
β	2	2	20	b
β	2	3	10	b

$r \times s$

- **Relational Algebra: Rename Operation**

A result of an operation is a new relation that has no name. Rename operation allows the relation to have more than one name.

If E has n attributes, then

$$\rho_{X(A_1, A_2, \cdots, A_n)}(E)$$

returns the result of E under the name X, and with the attributes renamed to $A_1, A_2, \cdots, A_n$.

- **Q. Search for the highest balance in account relation.**

$$\Pi_{\text{balance}}(\text{account}) - \Pi_{\text{account.balance}}(\sigma_{\text{account.balance} < \text{d.balance}}(\text{account} \times \rho_d(\text{account})))$$

- **Q. Find all customers who gas loan from Perryridge branch**

$$\Pi_{\text{customer\_name}}(\sigma_{\text{account.loan\_name}=d.\text{loan\_name}}(\text{borrower} \times \rho_d(\sigma_{\text{branch\_name}=\text{"Perryridge"}}(\text{loan}))))$$

- **Q. Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank**

$$\Pi_{\text{customer\_name}}(\sigma_{\text{account.loan\_name}=d.\text{loan\_name}}(\text{borrower} \times \rho_d(\sigma_{\text{branch\_name}=\text{"Perryridge"}}(\text{loan})))) - \Pi_{\text{customer\_name}}(\text{depositor})$$

- **Relational Algebra: Set-Intersection Operation**

$$r \cap s = \{t | t \in r \text{ and } t \in s\} = r - (r - s)$$

Assume that r, s have the same number of attributes and attributes of r and s compatible.

A	B
α	1
α	2
β	1

A	B
α	2
β	3

A	B
α	2

$r \cap s$

- **Relational Algebra: Natural-Join Operation**

Let r and s is be relations on schemas R and S respectively. Then, $r \bowtie s$ is a relation on $R \cup S$ obtained as follows:

$$r \bowtie s = \Pi_{R \cup S}(\rho_{r.A_1=s.A_1 \text{ and } r.A_2=s.A_2 \text{ and } \cdots \text{ and } r.A_n=s.A_n}(r \times s))$$

where $R \cap S = \{A_1, A_2, \cdots, A_n\}$

if $R \cap S = \emptyset$, $r \bowtie s = r \times s$.

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	10	γ
δ	2	β	b	δ

$r \bowtie s$

- **Relational Algebra: Assignment Operation**

The assignment operation (←) provides a convenient way to express complex queries.

Write query as a sequential program consisting of a series of assignments.

ex) Given $r(R)$ and $s(S)$, in which $S \subset R$,

$$\text{temp1} \leftarrow \Pi_{R-S}(r)$$
$$\text{temp2} \leftarrow \Pi_{R-S}((\text{temp1} \times s) - \Pi_{R-S,S}(r))$$
$$\text{result} = \text{temp1} - \text{temp2}$$

- **Relational Algebra: Generalized Projection**

Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{F_1,F_2,\cdots,F_n}(E)$$

E is any relational-algebra expression. Each of $F_1, F_2, \cdots, F_n$ are arithmetic expression involving constants and attributes in the schema of E.

ex) Given relation credit_info, find how much more each person can spend:

customer_name	limit	credit_balance
Curry	2000	1750
Hayes	1500	1500
Jones	6000	700
Smith	2000	400

customer_name	credit_available
Curry	250
Jones	5300
Smith	1600
Hayes	0

$$\Pi_{\text{customer\_name,(limit}-\text{credit\_balance) as credit\_available}}(\text{credit\_info})$$

- **Null and Unknown**

Null signifies an unknown value or that a value does not exist. The result of any arithmetic expression involving null is null.

Comparisons with null values return the unknown.

ex)

$$\text{null} > 5,\ \text{null} \leq 5 \rightarrow \text{unknown}$$
$$\text{unknown or true} = \text{true}$$
$$\text{unknown or false} = \text{unknown}$$
$$\text{unknown or unknown} = \text{unknown}$$
$$\text{unknown and true} = \text{unknown}$$
$$\text{unknown and false} = \text{false}$$
$$\text{unknown and true} = \text{unknown}$$
$$\text{not unknown} = \text{unknowns}$$

- **Modification of the DB: Deletion**

  The selected tuples are removed from the DB. Delete only whole tuples; not delete values on particular attributes.

  $$r \leftarrow r - E$$

  where r is a relation and E is a relational algebra query.

  ex) Delete all account records in the Perryridge branch

  $$\text{account} \leftarrow \text{account} - \sigma_{\text{branch\_name="Perryridge"}}(\text{account})$$

  ex) Delete all loan records with amount in the range of 0 to 1000

  $$\text{loan} \leftarrow \text{loan} - \sigma_{\text{amount} \geq 0 \text{ and } \text{ammount} \leq 1000}(\text{loan})$$

- **Modification of the DB: Insertion**

  To insert new tuple(s) into a relation

  $$r \leftarrow r \cup E$$

  where r is relation and E is a new data set.

  ex) Insert information in the DB specifying that Smith has $1200 in account A-973 at the Perryridge branch.

  $$\text{account} \leftarrow \text{account} \cup \{(\text{"Perryridge"}, \text{A-973}, 1200)\}$$

  $$\text{depositor} \leftarrow \text{depositor} \cup \{(\text{"Smith"}, \text{A-973})\}$$

- **Modification of the DB: Updating**

  To change a value in a tuple. Use the generalized projection operator to do this task

  $$r \leftarrow \Pi_{F_1, F_2, \cdots F_n}(r)$$

  F_i is the i-th attribute of r, if the i-th attribute is not updated, or, and expression, involving only constants and the attributes of r, if the attribute is to be updated.

  ex) Pay all accounts with balances over $10,000 by 6 percent interest and pay others by 5 percent

$$\text{account} \leftarrow \Pi_{\text{account\_number, branch\_name, balance} * 1.06}(\sigma_{\text{balance} > 1000}(\text{account})) \cup \Pi_{\text{account\_number, branch\_name, balance} * 1.05}(\sigma_{\text{balance} \leq 1000}(\text{account}))$$

- Structured Query Language (SQL)

  A special-purpose programming language designed for managing data stored in a relational DB management system (RDBMS). Consists of DDL and DML.

-