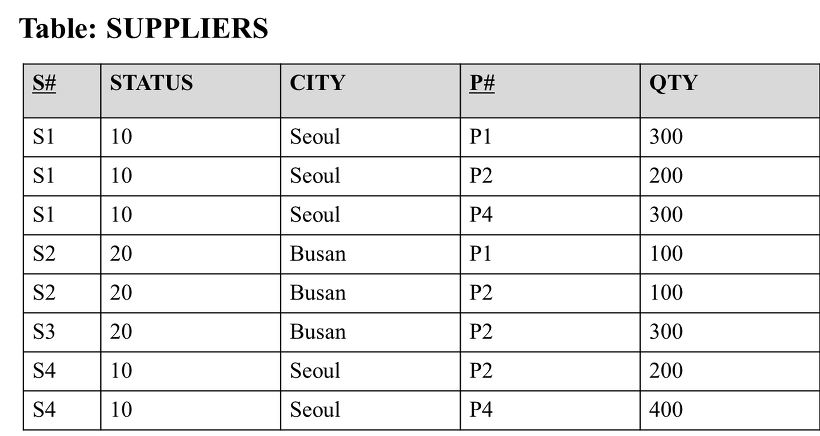
1. Database Normalization

In the design stage of a relational database, the task of determining the structure of data to minimize data redundancy is called normalization. In general, database normalization changes large, poorly organized tables and relationships between tables into smaller, well-organized ones.

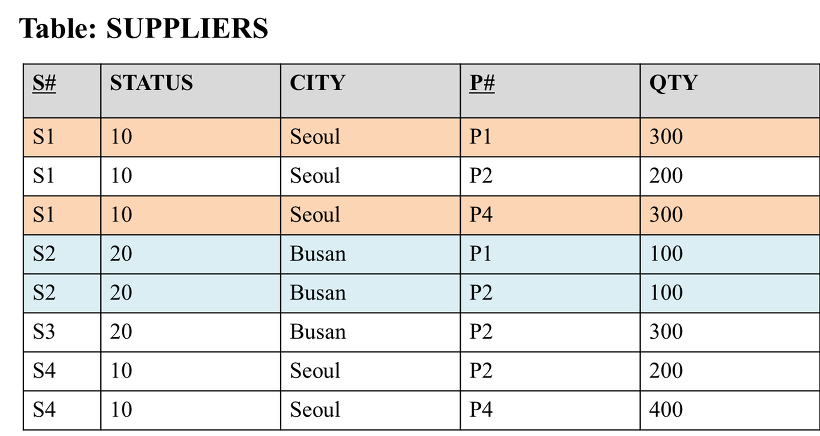
First Normal Form (1NF)

When all fields in the table have only scalar values ​​and all field values ​​are atomic, it is said to be 1NF. Atomic here means that there should be no duplicate entries in the table. Since the definition of "there is no overlapping item" in 1NF is not clear, several definitions of 1NF may also exist.



[Figure 1] Example table SUPPLIERS

Table SUPPLIERS in [Figure 1] above has {S#, P#} as primary keys, and shows a table that does not satisfy the definition of 1NF. Since the definition of 1NF is not satisfied, two records with different primary keys have the same value as shown in [Figure 2] below. That is, in the table of [Figure 2], there are duplicate entries of [STATUS=10, CITY=Seoul, QTY=300] and [STATUS=20, CITY=Busan, QTY=100].



[Figure 2] Table with duplicate records

In a table like [Figure 2] above, anomalies may appear during INSERT, DELETE, and UPDATE.

1) INSERT anomaly

If you want to store the information that the supplier S5 is  located in London in the table of [Figure 2], you need to input dirty data in the field corresponding to P#, which is a component of the primary key.

2) DELETE anomaly

In the table, the supplier named S3 no longer supplies P2, so if S#  deletes the record corresponding to S3, the information that the supplier named S3 is located in Busan is also lost.

3) UPDATE anomaly

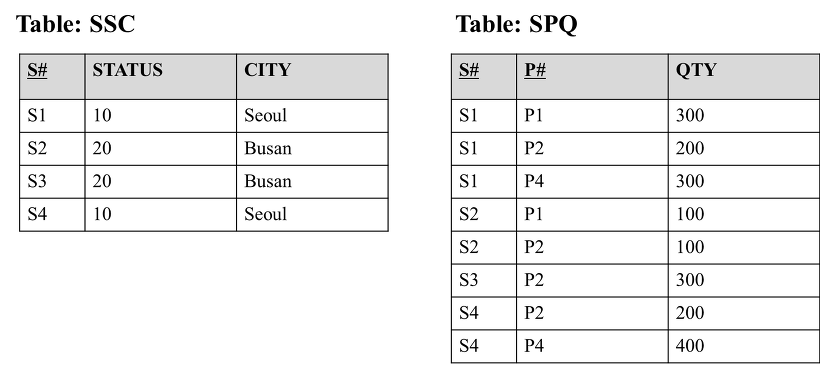
To change the city with the provider S1 from Seoul to Paris in the table, a total of 3 update operations are required. That is, an unnecessary update operation must be executed two more times to change one piece of information. By decomposing the table in [Figure 1] into two tables consisting of {S#, STATUS, CITY}, {S#, P#, QTY}, it can be changed to a new table that satisfies the definition of 1NF. The anomaly mentioned above does not occur in the new table decomposed to satisfy the definition of 1NF.

Second Normal Form (2NF)

2NF satisfies the properties of 1NF, and all functional dependencies in the table must be fully functional. The definition of complete functional dependency used when defining 2NF is  as follows.

what tableRRfield ofYYSet of autumn fields XXwhile being functionally dependent onXXexcept yourselfXXIf it is not functionally dependent on any subset of YYIsXXis said to be fully functionally dependent on

For example, in the table in [Figure 1], the primary key is {S#, P#}, but the field CITY is not {S#, P#} but is functionally dependent on S#. Therefore, the table in [Figure 1] is not 2NF because there is a functional dependency relationship that is not a complete functional dependency.  If the table in [Figure 1] above is transformed to satisfy the definition of 2NF, it can be decomposed into  two tables expressed as SSC and SPQ as shown in [Figure 3] below .



[Figure 3] Table that satisfies the definition of 2NF

However, even in the SSC and SPQ tables that satisfy the definition of 2NF, the following anomaly occurs.

1) INSERT anomaly

The SSC table  cannot store the information  "The STATUS of a city called Paris is 40" . Since the primary key of the SSC table is S#, information on the state of any city cannot be stored in the SSC table. Alternatively, dirty data  must be added  to S# to save the information .

2) DELETE anomaly

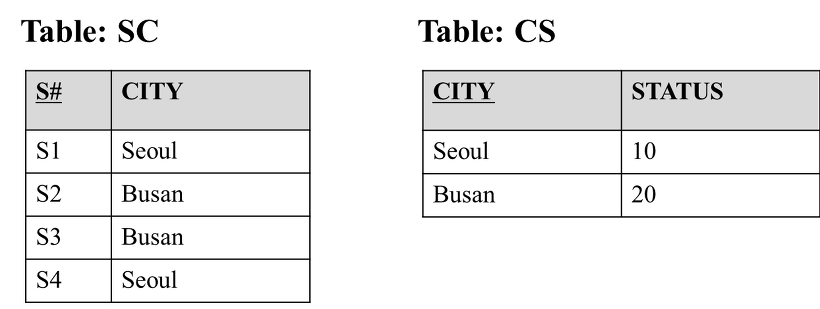
If the two providers S2 and S3 are removed from the SSC table, the information "The STATUS of the city called Busan is 20" is also lost.

3) UPDATE anomaly

 If the STATUS of a city called Seoul is changed from 10 to 30, in the SSC table, the task of changing the STATUS of Seoul to 30 must be performed twice, not once .

Third Normal Form (3NF)

When the non-key fields in a table are independent of each other while satisfying the definition of 2NF, it is called 3NF. For example, in the SSC table in [Figure 3], the SSC table  does not satisfy the definition of 3NF  because the STATUS field is functionally dependent on the CITY field, not the key of the table . If the SSC table of [Figure 3] is decomposed to satisfy the definition of 3NF, [Figure 4] is shown below.

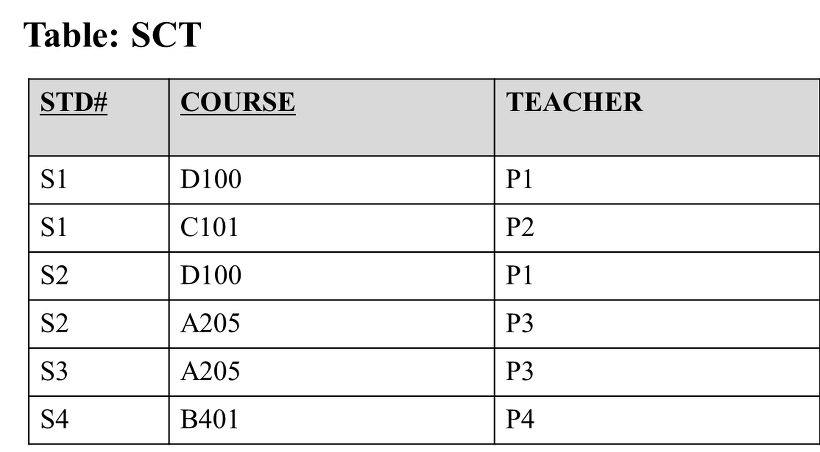


[Figure 4] SSC table decomposed to satisfy the definition of 3NF

In the SSC table of [Figure 3] above,  STATUS was dependent on CITY, not key, but in the SC and CS tables of [Figure 4], it can be seen that there is no dependency relationship between non-key fields.

Boyce-Codd Normal Form (BCNF)

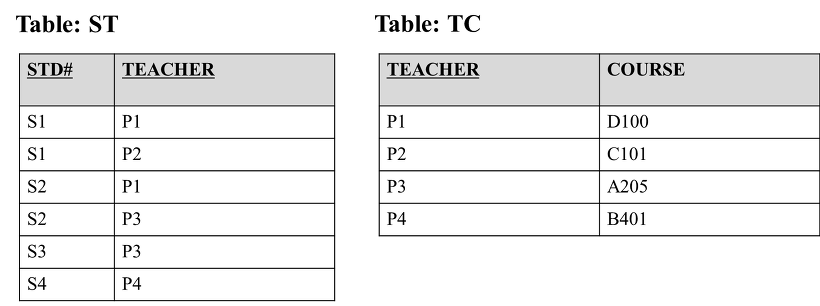
For a table, if the determinant of  all functional dependencies existing in the table is the  candidate key, it is called BCNF .



[Figure 5] Example table SCT

 [Figure 5] above shows a table that does not satisfy the definition of BCNF  . In the table of [Figure 5], the candidate key is {STD#, COURSE}. Thus, one record can be uniquely identified through {STD#, COURSE} .

However, in table SCT, there is a functional dependency in which COURSE is determined by TEACHER. In other words, the table SCT does not satisfy the definition of BCNF because there is a functional dependency relationship in which the determinant does not correspond to the candidate key.  If this table SCT is changed to satisfy the definition of BCNF , the table SCT is converted into two tables as shown in [Figure 6] below.



[Figure 6] Tables ST and TC modified to satisfy the definition of BCNF

In [Figure 6] above, the functional dependency relationship exists only in table TC, and the determinant is TEACHER, which corresponds to the candidate key of table TC. Therefore, the two tables in [Figure 6] satisfy the definition of BCNF.

Fourth Normal Form (4NF)

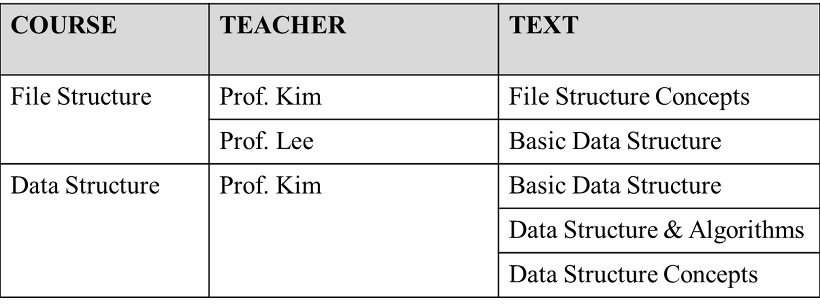
4NF is defined through the concept of MVD (multivalued dependency). MVD is a generalized concept of functional dependency. In functional dependency, if only one Y value is determined by determinant X, in MVD, multiple Y values ​​are determined by determinant X. The above MVD relationship is expressed as X → → Y.

MVD is divided into trivial MVD and nontrivial MVD. Each definition is as follows. In addition, all MVDs mentioned to define 4NF in this article refer to the following nontrivial MVDs.

* trivial MVD : if Y is a subset of X, or the sum of X and Y is the table itself (X ∪ Y = R)
* nontrivial MVD : Any MVD that is not a trivial MVD as defined above.

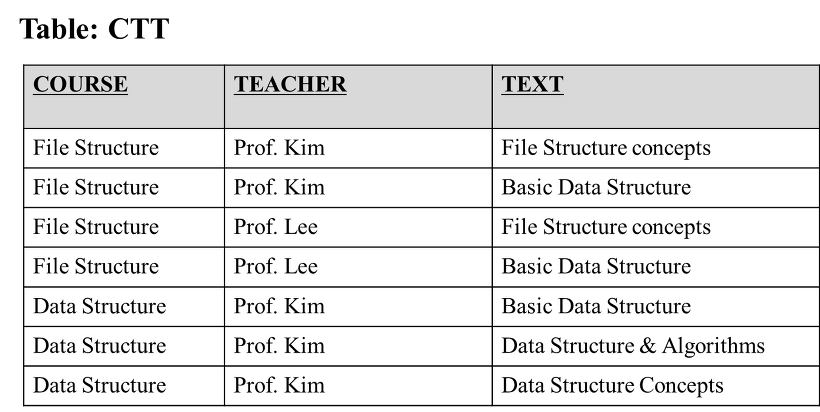
If fields A and B existing in a certain table R are A → → B, when all other fields in R have values ​​determined functionally dependent by A, it is said to be 4NF. For example, for table R(A, B, C, D), if A → → B, then A → C and if A → D, then it can be said that R satisfies the definition of 4NF.

As shown in [Figure 7] below, there is information including two fields with COURSE and MVD relationship. Each of the two fields has a value determined by the value of the COURSE field, and multiple TEACHER and TEXT values ​​may exist for one COURSE value.



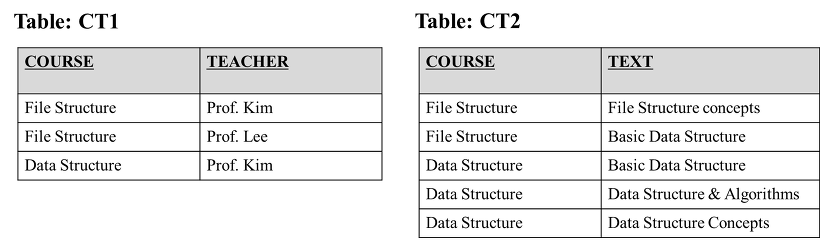
[Figure 7] Data including MVD relationship

If the data in [Figure 7] above is normalized, it is shown in [Figure 8] below, and the table CTT in [Figure 8] satisfies the definition of BCNF. However, since COURSE and TEACHER are MVD relations, and COURSE and TEXT are also MVD relations, it is not a 4NF normalized table.



[Figure 8] Table normalized by BCNF

In the table CTT of [Figure 8] above, if the person teaching the data structure is changed, an UPDATE anomaly occurs that requires changing a total of three records instead of one. To solve this UPDATE anomaly, it is necessary to change the table in [Figure 8] to satisfy the definition of 4NF. If the table CTT is divided to fit the definition of 4NF, it is changed to two tables, CT1 and CT2 in [Figure 9].



[Figure 9] Table partitioned to satisfy the definition of 4NF

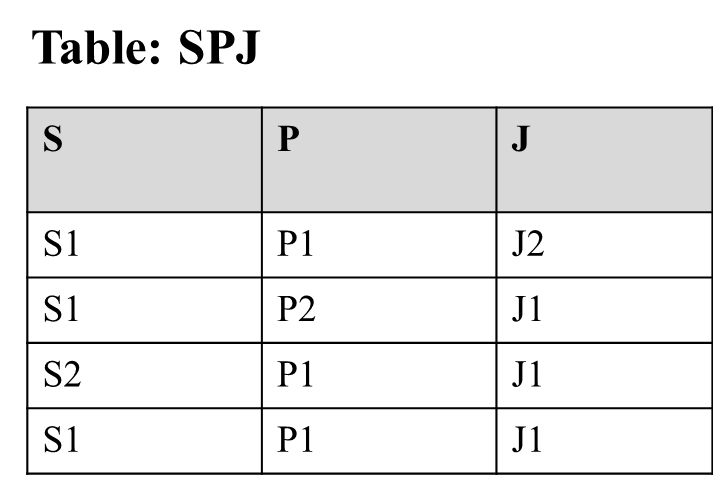
If the table CTT in [Figure 8] above  is divided into two tables CT1 and CT2 in [Figure 9], an UPDATE anomaly does not occur any more when the person who teaches Data Structure is changed.

Fifth Normal Form (5NF)

To define 5NF, we need to define join dependency first. The definition of join dependency used when defining 5NF is as follows.

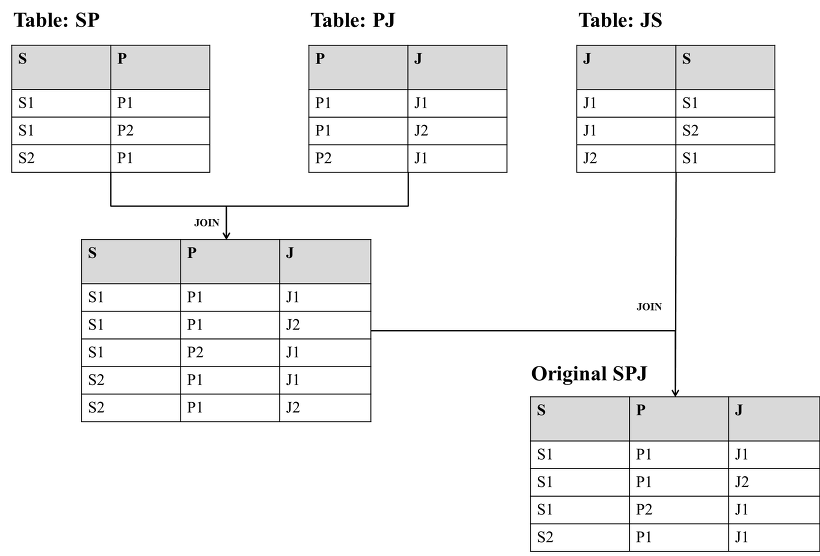
If any table T can be created by JOINing tables  including a  subset of fields included in T, table T has a join dependency.

5NF means that a  join dependency of a table is established only by a subset including the table's candidate key. For example, a table like [Figure 10] below does not satisfy the definition of 5NF.



[Figure 10] Table that does not satisfy the definition of 5NF

In the table SPJ in [Figure 10] above, SP , PJ, and JS  are not all keys of the table SPJ. However, SPJ can be projected into three tables (SP, PJ, JS) as shown in [Figure 11] below, and when the three tables are joined,  the original table, SPJ, is created. Therefore, SPJ does not satisfy the definition of 5NF because join dependency is established by  field other than candidate key .



 [Figure 11] Partition of table SPJ and JOIN operation on partitioned tables

Even if a table satisfying the definition of 5NF  is decomposed into a smaller table including the candidate key of the  original table, there is no data loss.  However, this property of a table satisfying the definition of 5NF does not necessarily mean that the table must be decomposed.

2. A strategy for applying semi-normalization is needed when a large number of joins occur in SQL statements that are engaged, resulting in performance degradation. Denormalization is one of the data modeling techniques for integrating and separating normalized entities, attributes, and relationships to improve system performance and simplify development and operation. Semi-normalization is performed when query performance is expected to decrease due to large amount of disk I/O; . In general, when it is judged that processing performance for a query is important, partial denormalization is considered.

When the number of processes accessing frequently used tables is the largest and always querying only a certain range.

If there is a large amount of data in the table and you frequently process large ranges, if there are performance issues

When it is technically difficult to query data due to excessive use of joins in the table

If denormalization is applied excessively, data integrity may be broken. Also, the response time for input, correction, and deletion queries may be delayed. In other words, the benefit from regularization is lost. There is a trade-off between the performance of input/deletion/modification and the performance of inquiry, two measures of database performance.

For complex query statements, wouldn't it be possible to solve the inexperience of the query through VIEW. Before proceeding with denormalization for query performance, we first review whether it can be solved by applying clustering or indexing.

3. relational database (Oracle, MySQL, MS-SQL, SQLite Etc..)is**named relational DB because it allows you to establish relationships between entities.**  
*Entity: The* type you want to represent in The DataBase, which means that they are distinguished from each other as intangible objects.

relationshiprefers **to when two entities are related**, which can be associated with 1:1, 1:N**,**and N:M.

# 1:1 relationship (one-to-one relationship)

a one-to-one relationship is a one-way relationship in which entity **side must have only one relationship with the other** entity.

in our country, for example, the institution of marriage is monogamous, one man can only marry a woman and a woman can only be married to a man.

you can't have more than one husband or wife, and this is the relationship.1:1 relationship



# 1:N RELATIONSHIP (ONE-TO-MANY RELATIONSHIPS)

A 1:N relationship means that one entity **can have multiple objects on the side of the entity in which the relationship was made**.

THERE ARE MANY 1:N RELATIONSHIPS IN THE REAL WORLD, WHICH ARE OFTEN USED TO DESIGN REAL DB.

**A 1:N RELATIONSHIP DOES NOT CREATE A NEW TABLE LIKE THE N:M RELATIONSHIP.**

for example, considering a parent-child relationship, a parent can have one, two, three, or more children.  
this is expressed as the parent owning the child (has a relationship).

on the contrary, a child must have only one parent (father, mother pair).

these relationships are called hierarchical structures.1:N relationship



Because it must be expressed in the position of multiple children (N) which parent of a pair (1) belongs to the parent, the PK**of the parent table is inserted into the child table as**FK**to represent the relationship**.

*Primary Key (PK):* A representative key that can identify each entity, a ***unique value that is not duplicated in the table, cannot be null.***

*Foreign Key (FK: Refers to* the primary key of another table, and all fields have the ***same domain (type & range of values) as the primary key*** referenced.All ***field values can be the same or null as the default key referenced.***

THAT IS, IN THE PARENT TABLE (1), THERE IS NO NEED TO PUT INFORMATION ABOUT WHO MY CHILDREN ARE, AND ONLY IN THE CHILD TABLE (N) CAN EACH CHILD EXPRESS THE RELATIONSHIP BY PUTTING THEIR OWN PARENT INFORMATION (FK).

# N:M RELATIONSHIP (MANY-TO-MANY RELATIONSHIP)

An N:M relationship is the one-to-one relationship**between both entities with a relationship**.

In other words,**each other sees each other as a**one-to-n relationship.

for example, considering the relationship between a school and a student, a school can have several students.1:N relationship

on the contrary, students can take multiple schools, so they also have a "students" and a "school".1:M relationship

therefore, schools and students can be said to have .N:M relationship



N:M relationshipBecause each other has each other, each **other's**PKs**must be their own foreign key columns.**1:N relationship1:M relationship

In general, the N: M relationship**is managed by creating another table having a representative key of the two tables as a column.**

**4. Transactions guarantee the completeness of work. In other words, it is a function that prevents the phenomenon that only a part of the work is applied by restoring the original state when all logical work sets are completely processed or cannot be processed. From the user's point of view, it can be understood as a logical unit of work, and from the system's point of view, it becomes the unit of a program that accesses or changes data.**

**A lock and a transaction are similar concepts, but in fact, a lock is a function to control concurrency, and a transaction is a function to ensure data consistency. A lock serves to allow only one connection to be changed at a time in sequence when multiple connections request the same resource at the same time. Resources here refer to records or tables. Contrary to this, a transaction is not necessarily a meaningful concept only when a query that performs multiple change operations is combined. A transaction is to guarantee that the logical working set itself is 100% applied or nothing is applied, regardless of whether there is one query in one logical working set or two or more queries. For example, if there is a failure in operation due to a problem such as a HW error or a SW error, special measures are required to solve these problems.**

What characteristics must a transaction satisfy?

Transaction must satisfy the following 4 characteristics of ACID.

Atomicity

If any problem occurs in the middle of a transaction, no work corresponding to the transaction should be performed, and all work should be performed only when no problem occurs.

Consistency

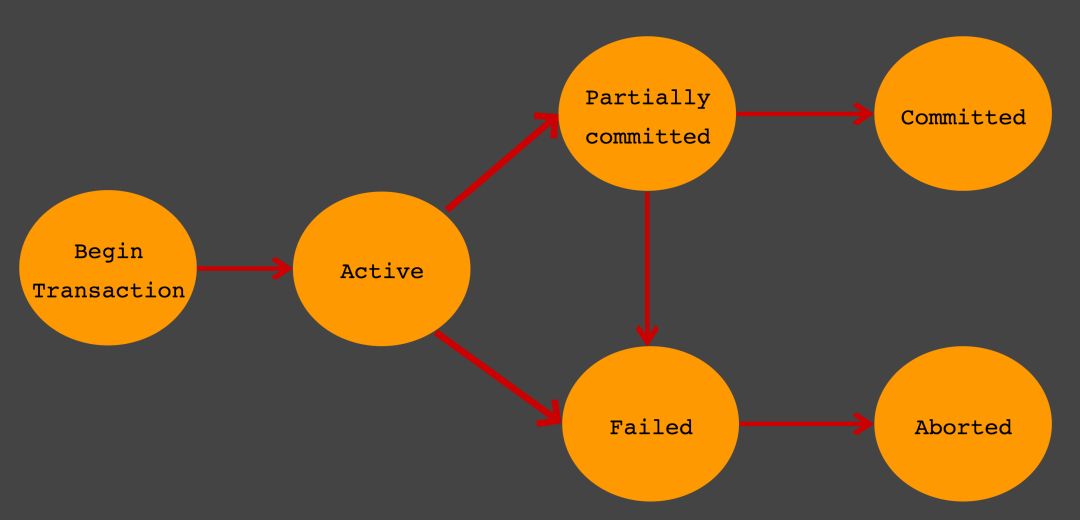
Even in the state after the transaction is completed, data consistency must be guaranteed as in the state before the transaction.

Isolation

Each transaction must be performed independently without interfering with each other.

Durability

After the transaction is normally terminated, the result of the operation should be permanently stored in the database.

state of the transaction

Active

The activity state of the transaction. A transaction is running and is in operation.

Failed

Transaction failure status. It is a state in which a transaction can no longer proceed normally.

Partial Committed

The state in which the transaction's command has arrived. It refers to the state that remains only after the previous statement of the transaction has been executed . Commit commit sql commit

Committed

Transaction completion status. This indicates that the transaction has been successfully completed.

Aborted

The transaction has been canceled. A state in which the transaction is canceled and the data returned to before the execution of the transaction.

Difference between Partial Committed and Committed

Commit When a request comes in, a state becomes a state. If the subsequent steps can be performed without any problem, it is transferred to the state, and if an error occurs, it is transferred to the state. In other words, indicates when a request is received, and indicates the state in which the is normally completed. Partial Commited Commit Committed Failed Partial Commited Commit Commited Commit

Points to note when using transactions

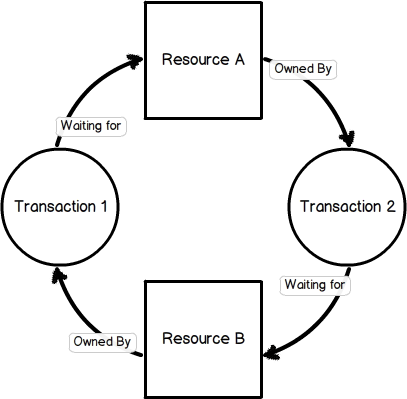
It is recommended that transactions be applied only to the minimum code that is absolutely necessary. In other words, it means to minimize the scope of the transaction. In general, the number of database connections is limited. However, if the time each unit program owns a connection increases, the number of available free connections decreases. Then, at some point, each unit program may have to wait to get a connection.

deadlock

When using multiple transactions, a deadlock can occur. Deadlock is a state in which two or more transactions acquire a lock on a specific resource (table or row) and request a lock owned by another transaction. do.

Deadlock Example (MySQL)

Due to the characteristics of MySQL MVCC, a lock is acquired when an update operation (Insert, Update, Delete) is executed in a transaction. (default lock on row)



Assume that transaction 1 acquires a lock on the first row of table B and transaction 2 also acquires a lock on the first row of table A.

Transaction 1 > create table B (i1 int not null primary key ) engine = innodb;

Transaction 2 > create table A (i1 int not null primary key ) engine = innodb;

Transaction 1 > start transaction ; insert into B values ( 1 );

Transaction 2 > start transaction ; insert into A values ( 1 );

If you request a lock on each other's first row without committing the transaction,

Transaction 1 > insert into A values ( 1 );

Transaction 2 > insert into B values ( 1 );

ERROR 1213 ( 40001 ): Deadlock found when trying to get lock; try restarting transaction

Deadlock occurs. A typical DBMS independently detects and reports deadlocks.

How to reduce the frequency of deadlocks

Commit transactions frequently.

Tables are accessed in a specified order. Above, transaction 1 accessed table A -> B in the order, and transaction 2 accessed table B -> A in the order. Make it access in the same order as table A -> B.

Avoid using read lock acquisition (SELECT ~ FOR UPDATE).

If multiple rows of a table are updated out of sequence in multiple connections, a deadlock is likely to occur.

5. importance of having a backup and restore process in a database.

BACK UP MYSQL DB

1. options for use

# mysqldump -u [ID] -p [Password] [OG DATA] > [Creat Backup DB].sql

2. how to use it

# mysqldump -u test\_user -p test\_db > backup\_test\_db.sql

passowrd :

RESTORE MYSQL DB

1. options for use

# mysql -u [ID] -p [password] [Restoration target DB] < [Back up DB].sql

2. how to use it

# mysql -u test\_user -p test\_db < backup\_test\_db.sql

passowrd :

BACK UP MYSQL TABLES

1. options for use

# mysqldump -u [ID] -p [password] [DB] [OG\_Name of the table to be backed up.] > [Name of the table to be backed up.].sql

2. how to use it

# mysqldump -u test\_user -p test\_db test\_table > backup\_test\_table.sql

passowrd :

RESTORING A MYSQL DB TABLE

1. options for use

# mysql -u [ID]-p [password] [Target restoration DB ] < [back up table].sql

2. how to use it

# mysql -u test\_user -p test\_db < backup\_test\_table.sql

passowrd :

BACK UP ALL MYSQL DATABASES

1. options for use

# mysqldump --all-databases -u [ID] -p --default-character-set=euckr < [back up DB].sql

2. how to use it

# mysqldump --all-databases -uroot -p --default-character-set=euckr > all.sql

RESTORE ALL MYSQL DATABASES

1. options for use

mysql --all-databases -u [ID] -p < [backup DB].sql

2. how to use it

# mysql -uroot -p < all.sql