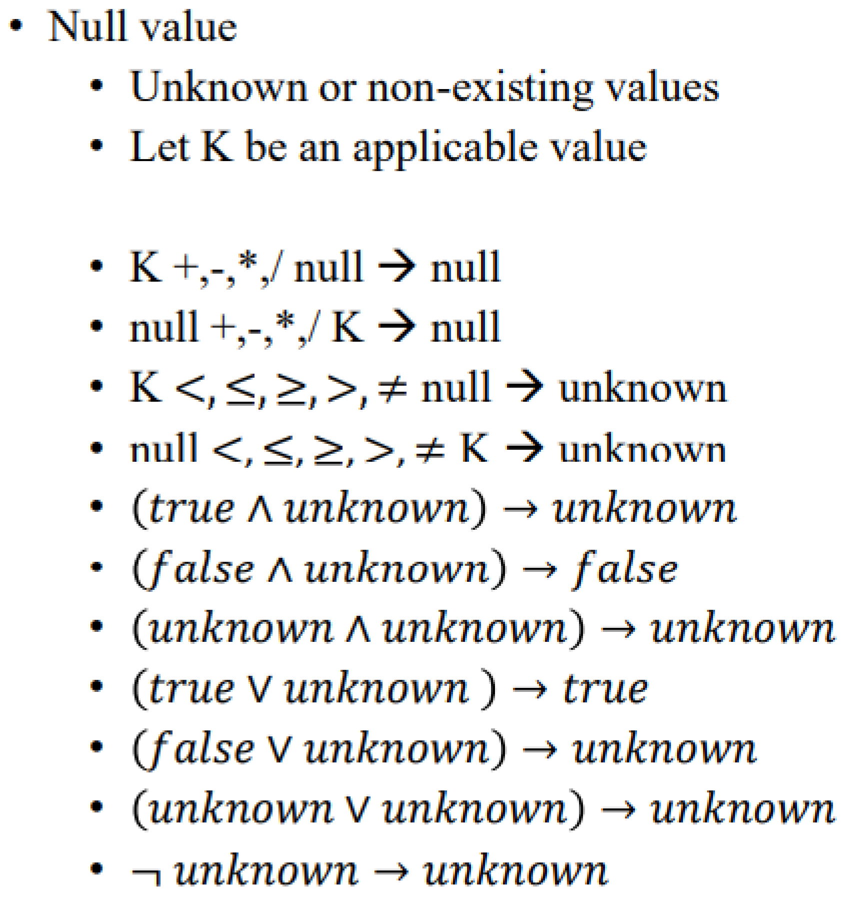
데이터베이스 시험

null과 **unknown** 에 대해서



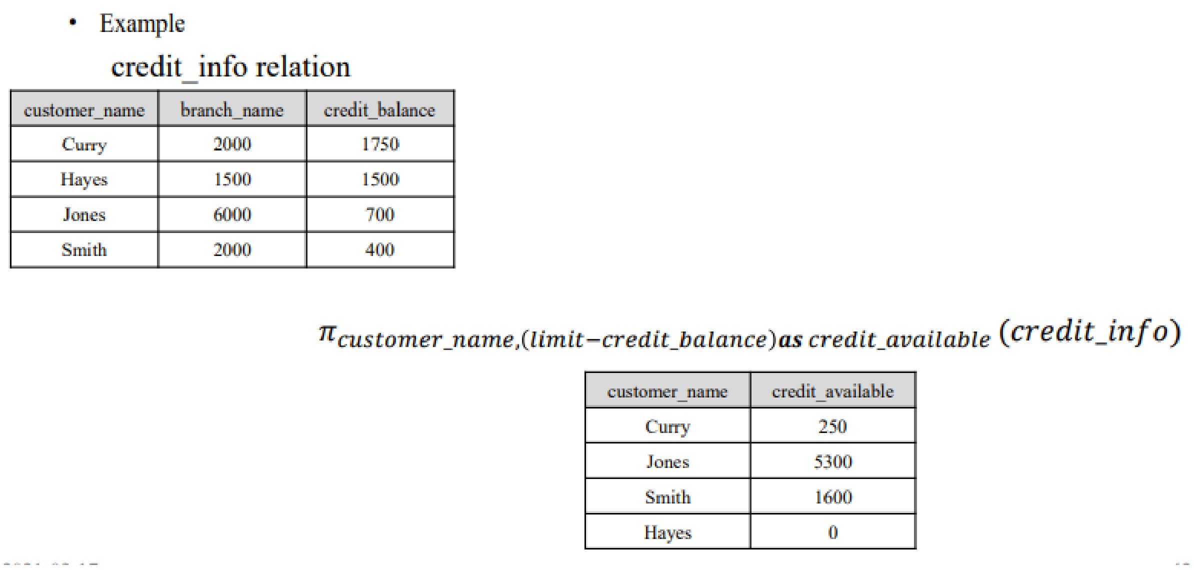
- Null value
  - Unknown or non-existing values
  - Let K be an applicable value

  - K +,-,*,/ null → null
  - null +,-,*,/ K → null
  - K <, ≤, ≥, >, ≠ null → unknown
  - null <, ≤, ≥, >, ≠ K → unknown
  - $(true \wedge unknown) \rightarrow unknown$
  - $(false \wedge unknown) \rightarrow false$
  - $(unknown \wedge unknown) \rightarrow unknown$
  - $(true \vee unknown\ ) \rightarrow true$
  - $(false \vee unknown) \rightarrow unknown$
  - $(unknown \vee unknown) \rightarrow unknown$
  - $\neg\, unknown \rightarrow unknown$

SELECT: $\sigma_{predicate}(relation)$

PROJECT: $\pi_{attr1,\ attr2...}(relation)$’

계산 가능 + as로 칼럼이름을 바꿀 수 도 있당



- Example

credit_info relation

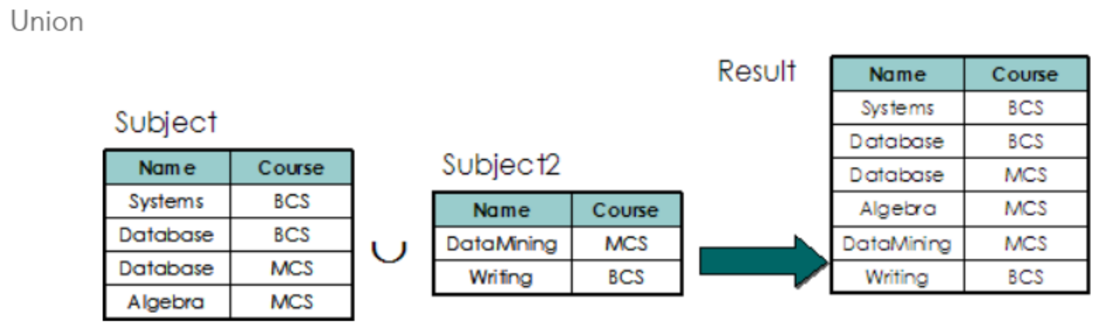
customer_name	branch_name	credit_balance
Curry	2000	1750
Hayes	1500	1500
Jones	6000	700
Smith	2000	400

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

customer_name	credit_available
Curry	250
Jones	5300
Smith	1600
Hayes	0

UNION: $relation_1 \cup relation_2$

중복 재외시켜준다!!



Union

Subject

Name	Course
Systems	BCS
Database	BCS
Database	MCS
Algebra	MCS

$\cup$

Subject2

Name	Course
DataMining	MCS
Writing	BCS

Result

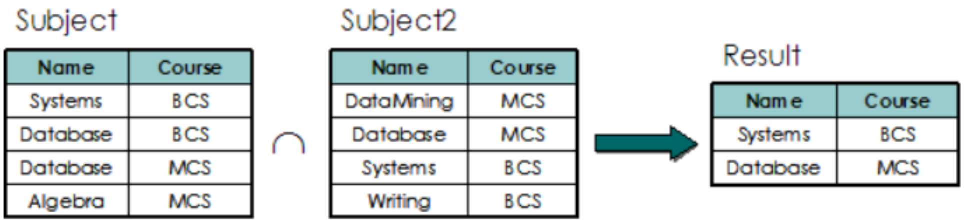
Name	Course
Systems	BCS
Database	BCS
Database	MCS
Algebra	MCS
DataMining	MCS
Writing	BCS

결과 사이즈	차수	특징
|R U S| <= |R| + |S|	R과 S의 차수가 같다	교환/결합법칙 성립

OUTER UNION: $r \cup^{+} s$

INTERSECTION: $relation_1 \cap relation_2$

Intersection



Subject

Name	Course
Systems	BCS
Database	BCS
Database	MCS
Algebra	MCS

∩

Subject2

Name	Course
DataMining	MCS
Database	MCS
Systems	BCS
Writing	BCS

Result

Name	Course
Systems	BCS
Database	MCS

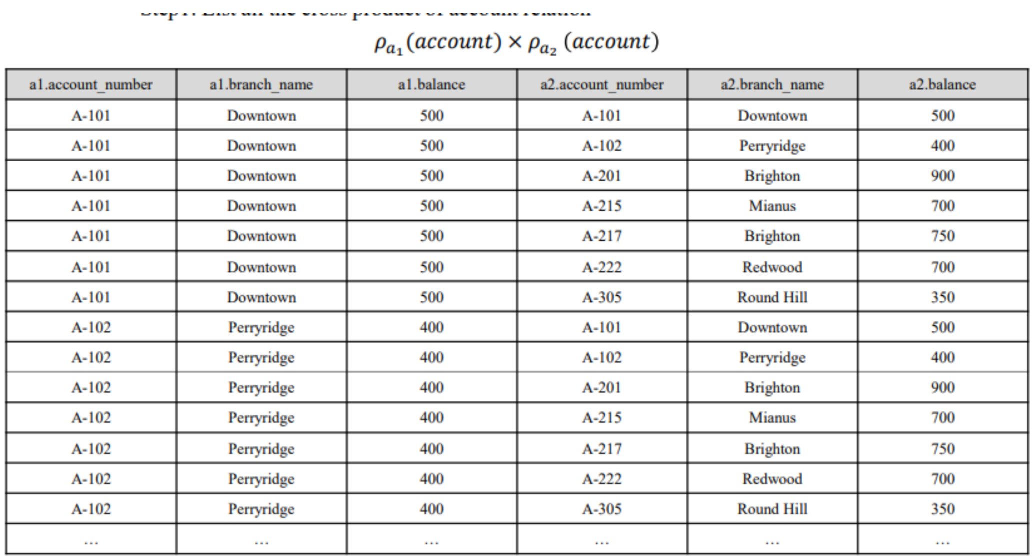
결과 사이즈	차수	특징
|R ∩ S| <= MIN{|R|, |S|}	R과 S가 차수가 같다	교환/결합법칙 성립

DIFFERENCE: $relation_1 - relation_2$

CARTESIAN PRODUCT: $relation_1 \times relation_2$

RENAME: $\rho_{renamed\_relation}(original\_relation)$

데이터 베이스의 이름이 바뀐다



Step1: List all the cross product of account relation

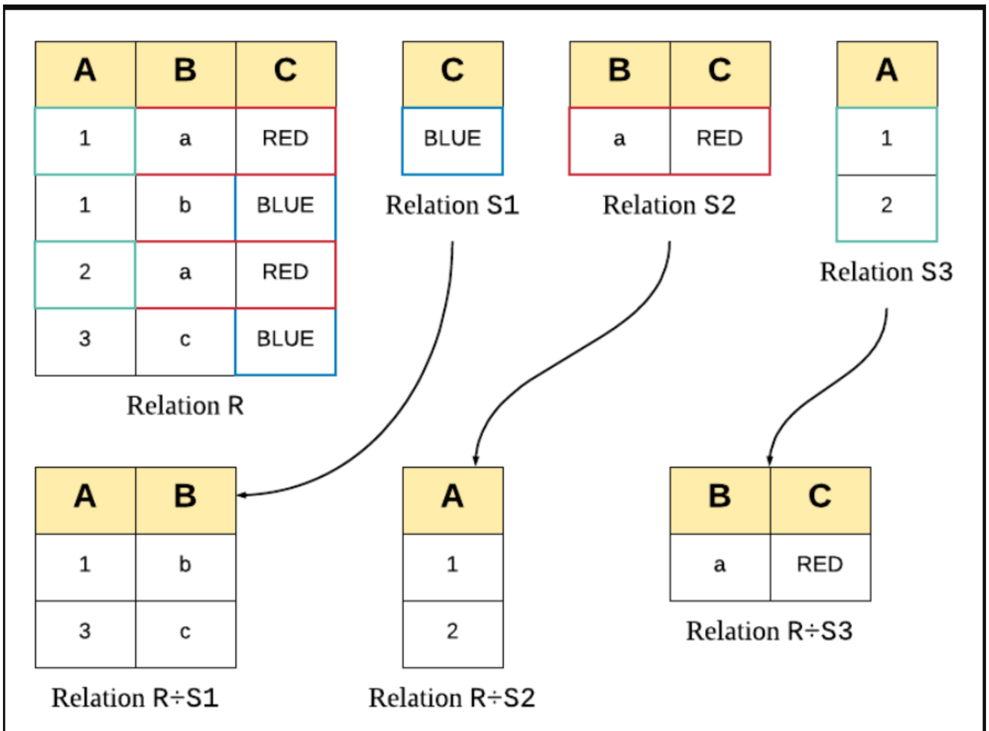
$$\rho_{a_1}(account) \times \rho_{a_2}(account)$$

a1.account_number	a1.branch_name	a1.balance	a2.account_number	a2.branch_name	a2.balance
A-101	Downtown	500	A-101	Downtown	500
A-101	Downtown	500	A-102	Perryridge	400
A-101	Downtown	500	A-201	Brighton	900
A-101	Downtown	500	A-215	Mianus	700
A-101	Downtown	500	A-217	Brighton	750
A-101	Downtown	500	A-222	Redwood	700
A-101	Downtown	500	A-305	Round Hill	350
A-102	Perryridge	400	A-101	Downtown	500
A-102	Perryridge	400	A-102	Perryridge	400
A-102	Perryridge	400	A-201	Brighton	900
A-102	Perryridge	400	A-215	Mianus	700
A-102	Perryridge	400	A-217	Brighton	750
A-102	Perryridge	400	A-222	Redwood	700
A-102	Perryridge	400	A-305	Round Hill	350
…	…	…	…	…	…

DIVISION: $relation_1 \div relation_2$

- division을 풀어서 쓴 식

- $r \div s$
- $= \pi_{R-S}(r) - \pi_{R-S}\left((\pi_{R-S}(r) \times s) - \pi_{R-S,S}(r)\right)$
  - $where\ S \subseteq R$



A	B	C
1	a	RED
1	b	BLUE
2	a	RED
3	c	BLUE

Relation R

C
BLUE

Relation S1

B	C
a	RED

Relation S2

A
1
2

Relation S3

A	B
1	b
3	c

Relation R÷S1

A
1
2

Relation R÷S2

B	C
a	RED

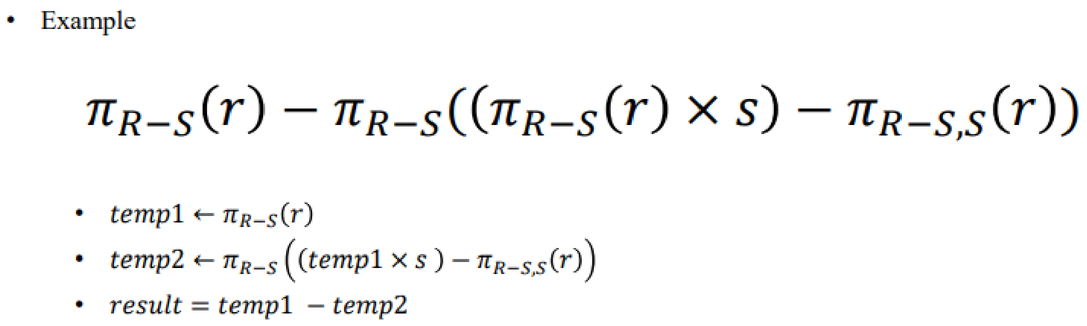
Relation R÷S3

Figure 8: Division R S1, S2 and S3

R÷S

릴레이션 R중에서 S와 관련되어 있는 모든 튜플을 추출 단 릴레이션의 S의 칼럼은 제외하고 릴레이션을 보여준다

ASSIGNMENT: $r_2 \leftarrow r_1$



- Example

$$\pi_{R-S}(r) - \pi_{R-S}((\pi_{R-S}(r) \times s) - \pi_{R-S,S}(r))$$

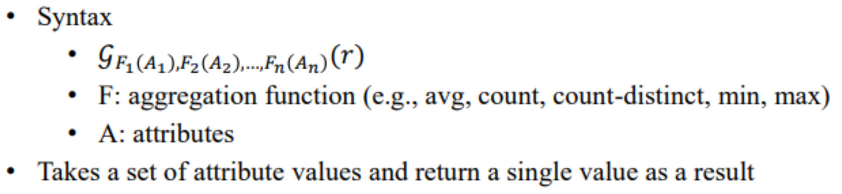
- $temp1 \leftarrow \pi_{R-S}(r)$
- $temp2 \leftarrow \pi_{R-S}\left((temp1 \times s\ ) - \pi_{R-S,S}(r)\right)$
- $result = temp1\ - temp2$

프로그래밍 언어의 변수같은 느낌, 복잡한 관계대수 식을 보기 편하게 해준다

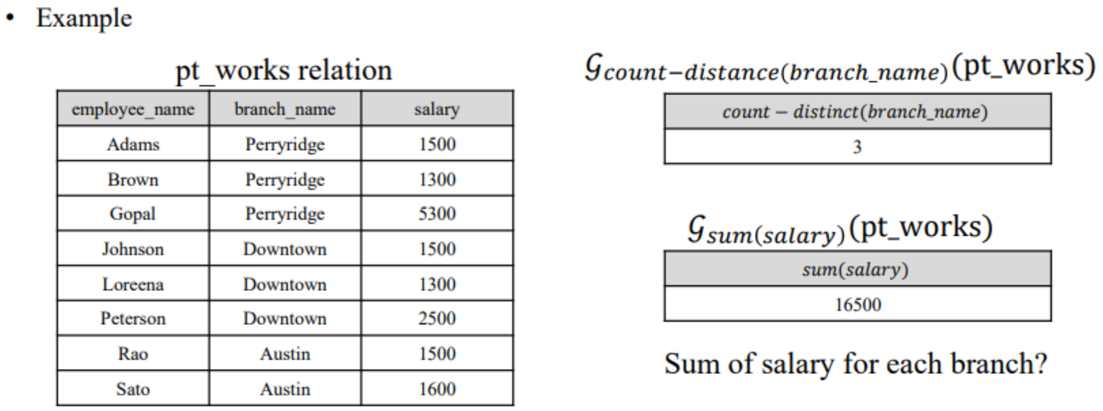
AGGREGATION: $_{G_1,G_2,\ldots,G_m}G_{F_1(A_1),\,F_2(A_2),\,\ldots,F_n(A_n)}(r)$

SQL의 집계함수의 역할을 한다.

avg(평균), count(개수 카운트), count-distinct(종류 개수), min(최솟값), max(최댓값), sum 등이 있다



- Syntax
  - $\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(r)$
  - F: aggregation function (e.g., avg, count, count-distinct, min, max)
  - A: attributes
- Takes a set of attribute values and return a single value as a result



- Example

pt_works relation

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

$\mathcal{G}_{count-distance(branch\_name)}$(pt_works)

$count - distinct(branch\_name)$
3

$\mathcal{G}_{sum(salary)}$(pt_works)

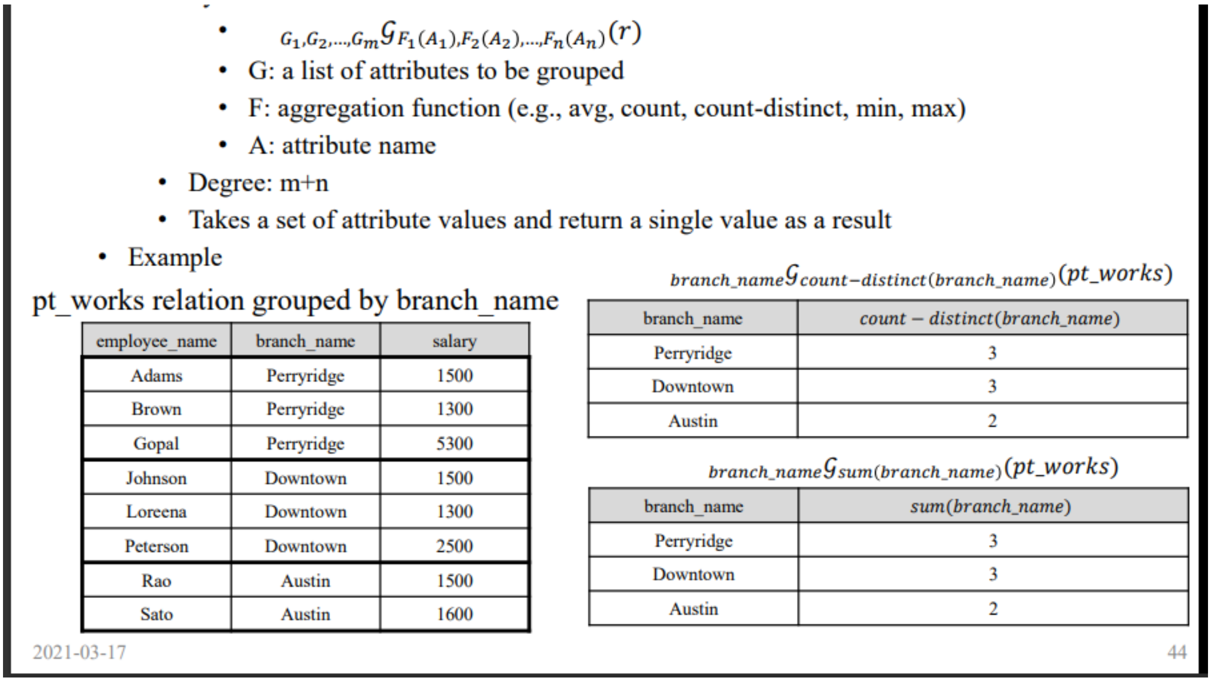
$sum(salary)$
16500

Sum of salary for each branch?

이를 통해 나오는 릴레이션의 차수는 m+n이다.

집계함수를 여러개 쓸 수 있다.

g기호 뒤에 출력할 칼럼명을 적으므로써 집계함수가 아닌 평범함 칼럼?도 출력할 수 있다.



- ${}_{G_1,G_2,\dots,G_m}\mathcal{G}_{F_1(A_1),F_2(A_2),\dots,F_n(A_n)}(r)$
- G: a list of attributes to be grouped
- F: aggregation function (e.g., avg, count, count-distinct, min, max)
- A: attribute name

- Degree: m+n
- Takes a set of attribute values and return a single value as a result

- Example

pt_works relation grouped by branch_name

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

${}_{branch\_name}\mathcal{G}_{count-distinct(branch\_name)}(pt\_works)$

branch_name	$count - distinct(branch\_name)$
Perryridge	3
Downtown	3
Austin	2

${}_{branch\_name}\mathcal{G}_{sum(branch\_name)}(pt\_works)$

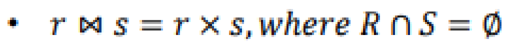
branch_name	$sum(branch\_name)$
Perryridge	3
Downtown	3
Austin	2

2021-03-17

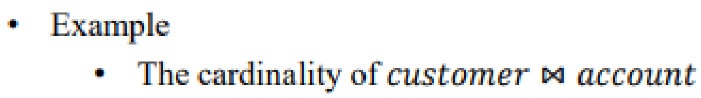
44

JOIN: $relation_1 \bowtie relation_2$

- 조인은 카디션 프로덕트(교차곱) + 셀릭트(select)의 연산이 합쳐진거다
- 세타조인, 동일조인, 자연조인에 대해서 알아두기



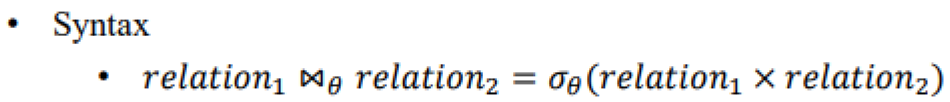
- $r \bowtie s = r \times s, where\ R \cap S = \emptyset$



- Example
  - The cardinality of $customer \bowtie account$

THETA JOIN: $relation_1 \bowtie_\theta relation_2$

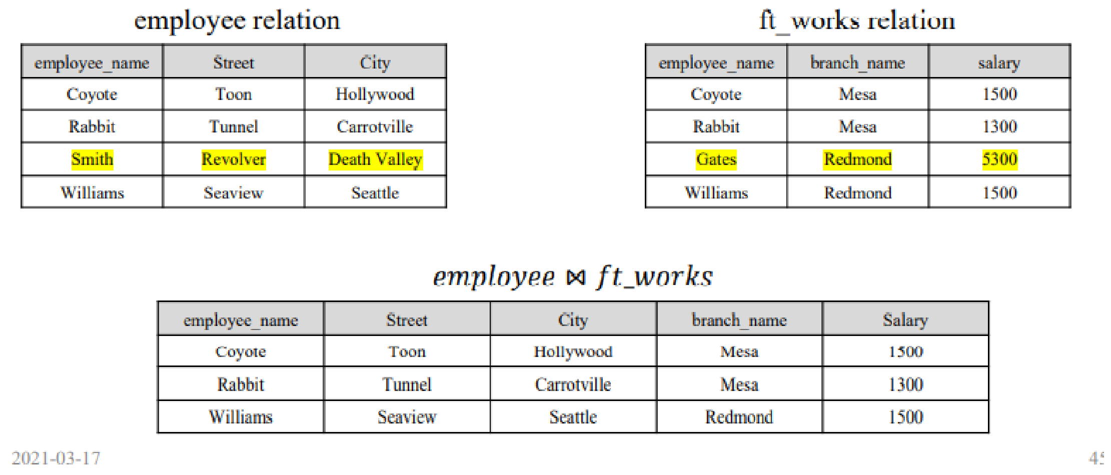
세타조인은 자연 조인보다 조금 더 확실하게 표현해주는거



- Syntax
  - $relation_1 \bowtie_\theta relation_2 = \sigma_\theta(relation_1 \times relation_2)$

시그마의 세타 (조건 관계식) = 조인의 세타

평범한 inner join



employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

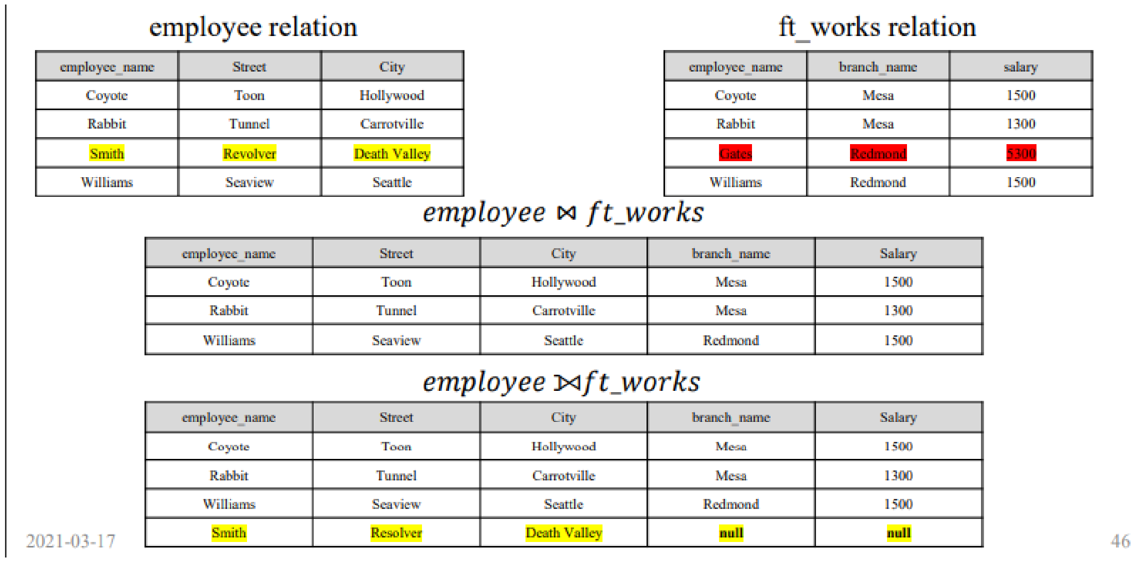
$employee \bowtie ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

2021-03-17

45

LEFT OUTER JOIN: $r ⟕ s$



employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee \bowtie ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

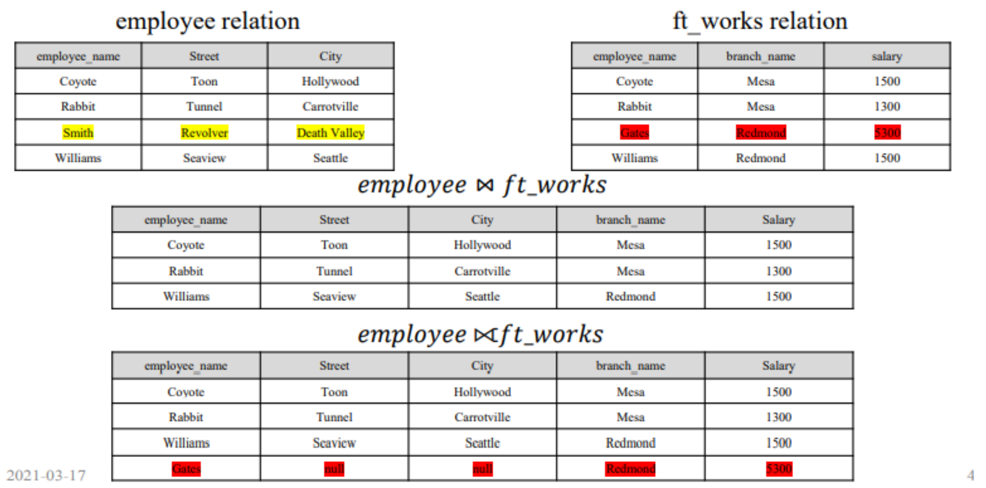
$employee ⟕ ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Resolver	Death Valley	null	null

2021-03-17

46

RIGHT OUTER JOIN: $r ⟖ s$



employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee \bowtie ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

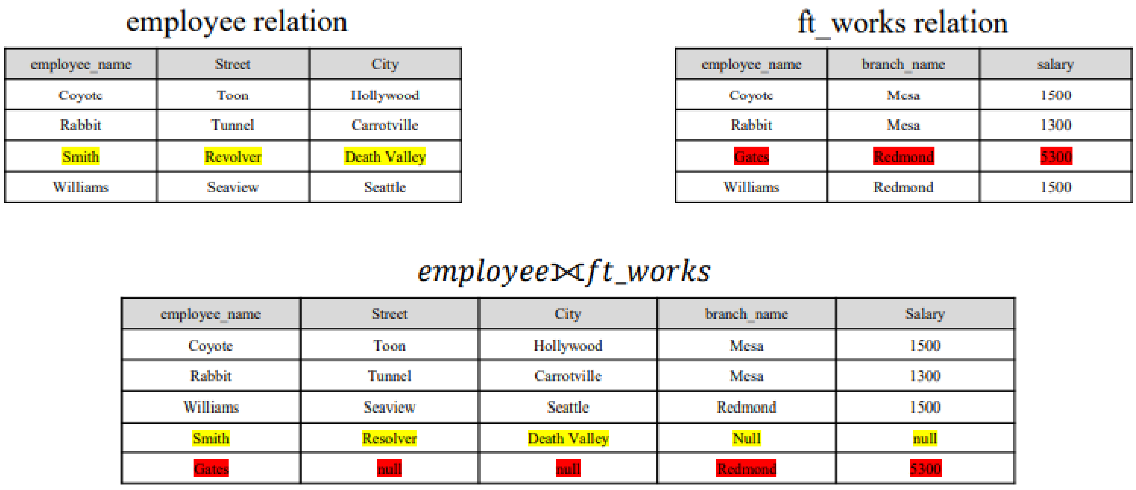
$employee ⟖ ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Gates	null	null	Redmond	5300

2021-03-17

4

FULL OUTER JOIN: r ⟗ s



employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee$⟗ft_works

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Resolver	Death Valley	Null	null
Gates	null	null	Redmond	5300

DDL

- USE mydb
- CREATE DATABASE mydb
- CREATE TABLE mytable (idx INT, name VARCHAR(10), ...)

  기본키 지정을 안해줘도 제대로 생성이 될까? 되네? ㅎㅎ

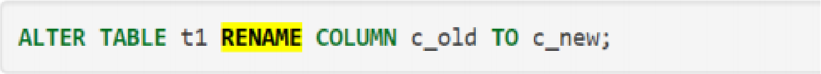
- DESCRIBE mytable -> 테이블 구조 보여줌
- INSERT INTO mytable VALUES (1,’yunsuu’,’man’)
- CREATE OR REPLACE DATABASE mydb

  = DROP DATABASE IF EXISTS mydb; CREATE DATABASE mydb db가 이미 있으면 삭제하고 다시 ㄱㄱ 없으면 그냥 평범하게 생성

- CREATE DATABASE IF NOT EXISTS mydb
- SHOW WARNINGS; -> 에러 로그를 보여준다
- CREATE OR REPLACE DATABASE mydb CHARACTER SET = latin1 COLLATE = latin1_german2_ci ;

각 나라마다 문자언어가 다르므로 필요한 바이트 수도 각각 다르다 이를 해결하기 위해

Character set이 존재한다

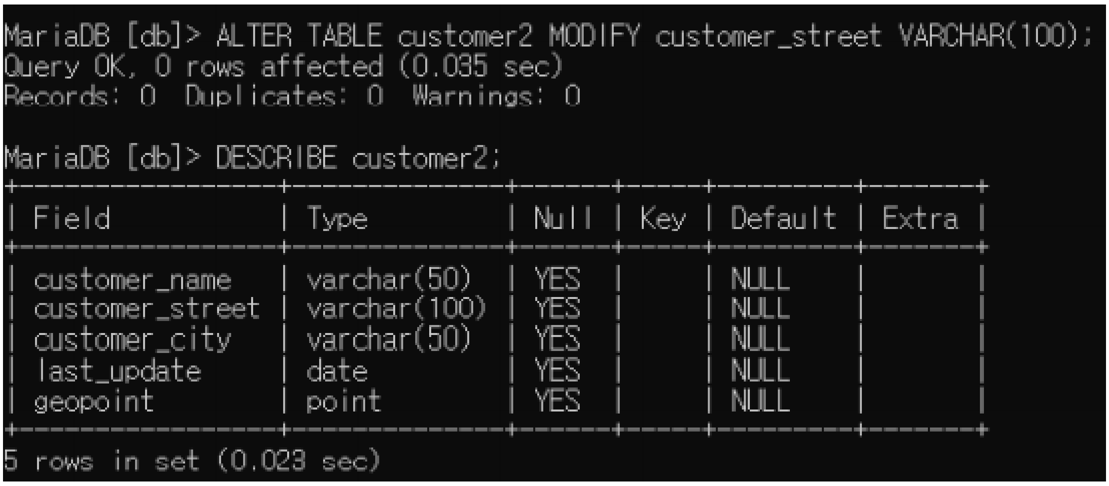
- SHOW DATABASES; -> 생성한 db의 종류를 보여준다
- DROP DATABASE mydb ;
- DROP DATABASE IF EXISTS mydb; -> db가 존재할때만 삭제
- DROP TABLE mytable; + DROP TABLE IF EXISTS customer2;
- ALTER TABLE mytable DROP COLUMN latitude, DROP COLUMN longitude;
- 

```
ALTER TABLE t1 RENAME COLUMN c_old TO c_new;
```

칼럼이름 다시 정의할때 (마리아 db 10.5.2 기준)

- ALTER TABLE mytable MODIFY idx VARCHAR(100)

예시



```
MariaDB [db]> ALTER TABLE customer2 MODIFY customer_street VARCHAR(100);
Query OK, 0 rows affected (0.035 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]> DESCRIBE customer2;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| customer_name   | varchar(50)  | YES  |     | NULL    |       |
| customer_street | varchar(100) | YES  |     | NULL    |       |
| customer_city   | varchar(50)  | YES  |     | NULL    |       |
| last_update     | date         | YES  |     | NULL    |       |
| geopoint        | point        | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
5 rows in set (0.023 sec)
```

DML

- SELECT * FROM mytable
- INSERT INTO mytable VALUES (615453, 'J.B.', 10,30,30,30), (123, 'J.C.', 9,27,25,23);

  여러개 insert 하는 sql 문

- DELETE FROM mytable

  `DELETE FROM [테이블] WHERE [조건]`

  db 지우는 구문 : DROP DATABASE IF EXISTS mydb;

  테이블에 모든 정보 지우는 문

- ALTER TABLE mytable ADD COLUMN total DOUBLE ;

  테이블 업데이트 문, mytable에 total(double) 칼럼을 추가한다

- UPDATE mytable SET last_update=CURDATE();

  curdate : 현재 날짜를 출력하는 빌트인 함수

빌트인 꿀받네 ㅎㅎ;;

DQL

- SELECT 문 순서

  SELECT

  FROM

  WHERE

  GROUP BY

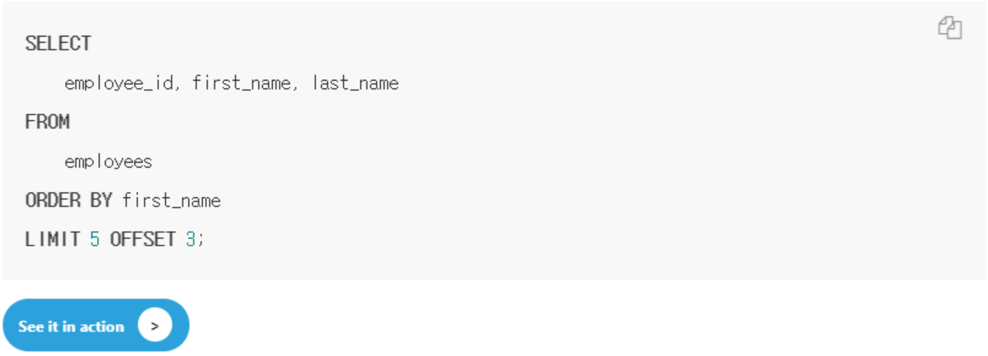
  HAVING

  ORDER BY

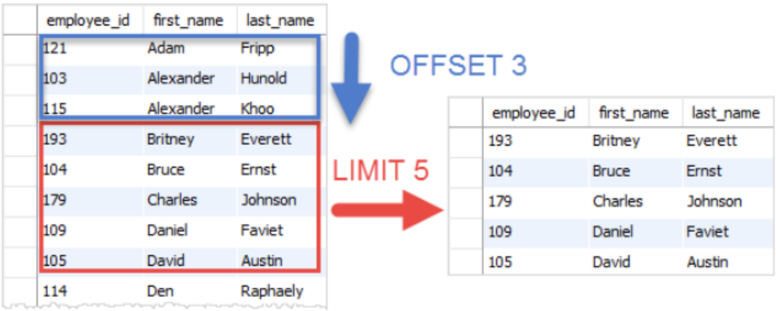
  LIMIT

- LIMIT 사용법

  LIMIT 4 -> 맨 위에서 부터 4개 추출



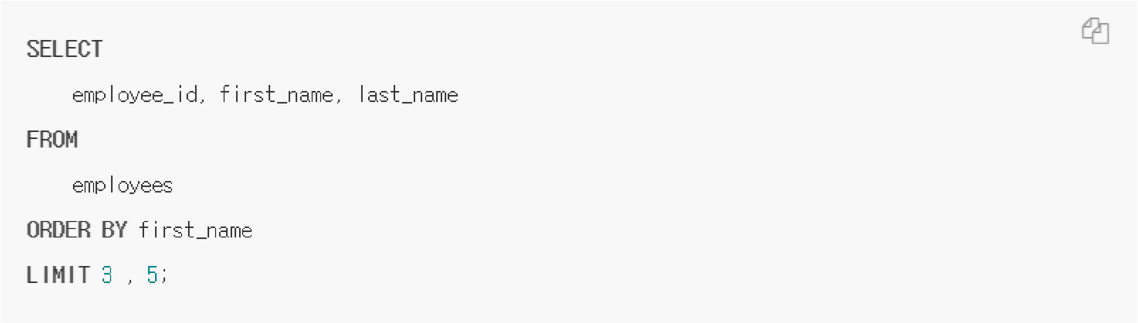
```
SELECT
    employee_id, first_name, last_name
FROM
    employees
ORDER BY first_name
LIMIT 5 OFFSET 3;
```



employee_id	first_name	last_name
121	Adam	Fripp
103	Alexander	Hunold
115	Alexander	Khoo
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
109	Daniel	Faviet
105	David	Austin
114	Den	Raphaely

employee_id	first_name	last_name
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
109	Daniel	Faviet
105	David	Austin

같은 표현으로는



```
SELECT
    employee_id, first_name, last_name
FROM
    employees
ORDER BY first_name
LIMIT 3 , 5;
```

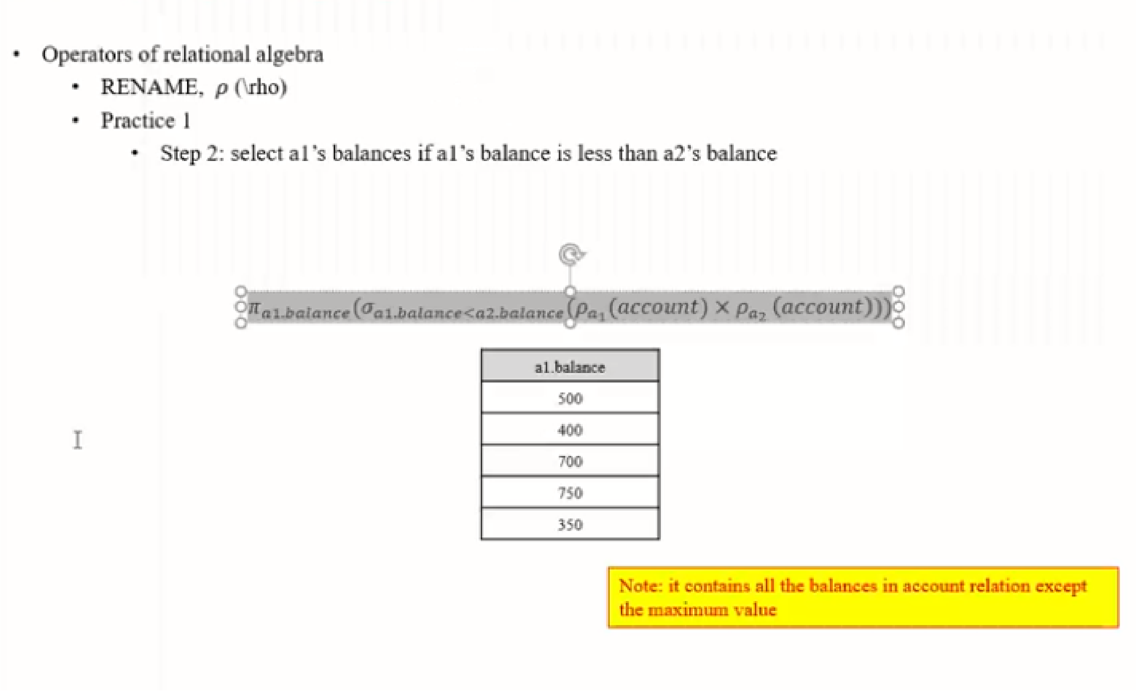
- INSERT INTO table (col_name,…) SELECT …

- CROSS JOIN

  SELECT * FROM x1 CROSS JOIN x2; -> x1이랑 x2 곱연산 한거

- A.a 할때 A.’a’ 이렇게 안해도됨
- SELECT (a+b) AS c FROM~ 이런식으로 AS로 rename 할수도 있다 혹은 SELECT * FROM a AS b 이런식으로도 가능
- SELECT DISTINCT a1.blance FROM account AS a1 CROSS JOIN loan AS a2 WHERE a1.blance < a2.blance

이 문제 답



- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 1
    - Step 2: select a1's balances if a1's balance is less than a2's balance

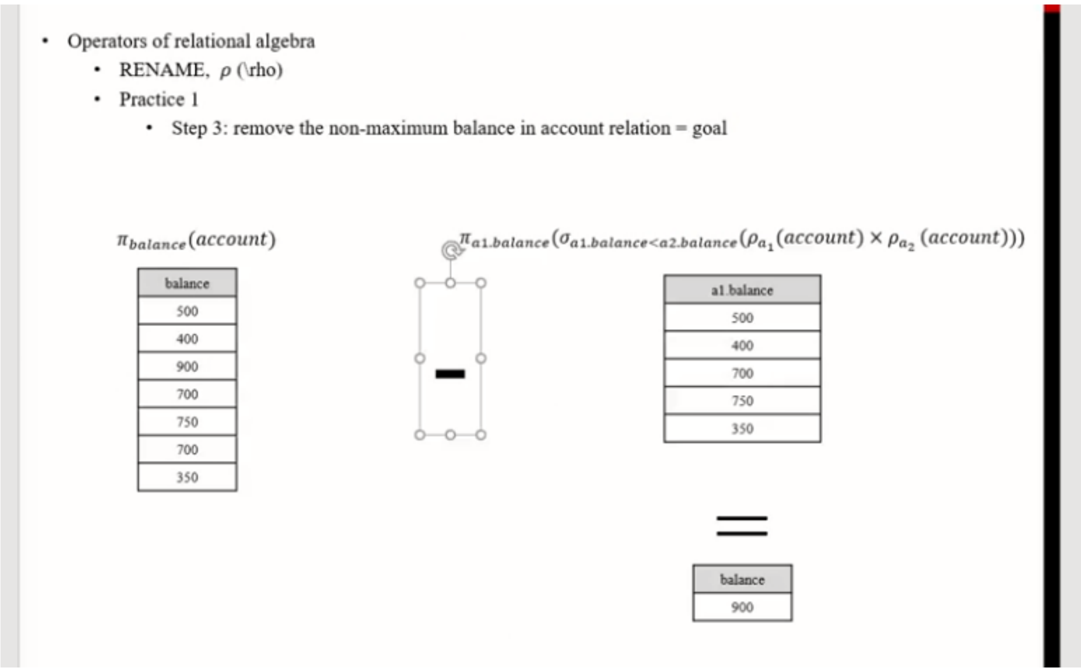
$$\pi_{a1.balance}(\sigma_{a1.balance<a2.balance}(\rho_{a_1}(account) \times \rho_{a_2}(account)))$$

a1.balance
500
400
700
750
350

Note: it contains all the balances in account relation except the maximum value

- SELECT DISTINCT blance FROM account WHERE blance EXCEPT SELECT DISTINCT a1.blance FROM account AS a1 CROSS JOIN loan AS a2 WHERE a1.blance < a2.blance

이문제 답, DISTINCT 붙여주는거 잊지말기



- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 1
    - Step 3: remove the non-maximum balance in account relation = goal

$\pi_{balance}(account)$

balance
500
400
900
700
750
700
350

$-$

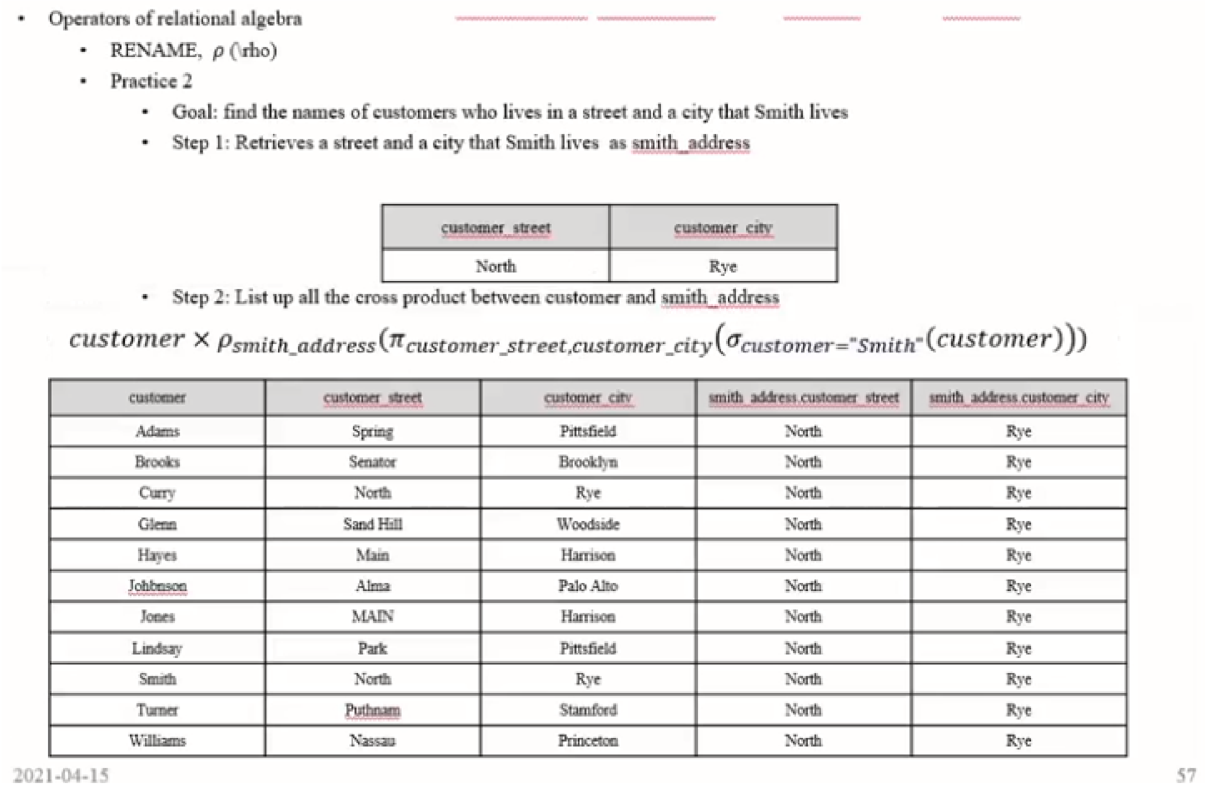
$\pi_{a1.balance}(\sigma_{a1.balance<a2.balance}(\rho_{a_1}(account) \times \rho_{a_2}(account)))$

a1.balance
500
400
700
750
350

$=$

balance
900

- 이문제 답



- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 2
    - Goal: find the names of customers who lives in a street and a city that Smith lives
    - Step 1: Retrieves a street and a city that Smith lives as smith_address

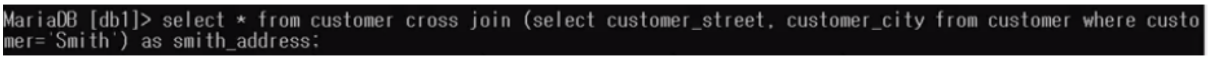
customer_street	customer_city
North	Rye

    - Step 2: List up all the cross product between customer and smith_address

$$customer \times \rho_{smith\_address}(\pi_{customer\_street,customer\_city}(\sigma_{customer="Smith"}(customer)))$$

customer	customer_street	customer_city	smith_address.customer_street	smith_address.customer_city
Adams	Spring	Pittsfield	North	Rye
Brooks	Senator	Brooklyn	North	Rye
Curry	North	Rye	North	Rye
Glenn	Sand Hill	Woodside	North	Rye
Hayes	Main	Harrison	North	Rye
Johbnson	Alma	Palo Alto	North	Rye
Jones	MAIN	Harrison	North	Rye
Lindsay	Park	Pittsfield	North	Rye
Smith	North	Rye	North	Rye
Turner	Puthnam	Stamford	North	Rye
Williams	Nassau	Princeton	North	Rye

2021-04-15 57



```
MariaDB [db1]> select * from customer cross join (select customer_street, customer_city from customer where custo
mer='Smith') as smith_address;
```

다른 테이블의 값을 이용해서 뭔가를 만들어 내려면 (SELECT ~) AS a1 이런식으로 이용해주기

- NATURAL JOIN

SELECT * FROM employee **NATURAL LEFT OUTER JOIN** ft_works;

- OUTER JOIN

SELECT * FROM employee **LEFT OUTER JOIN** ft_works ON employee.name = ft_wroks.name

자연조인이 아니면 on이 붙어서 조건을 써줘야 한다.

- 다른 테이블 값 기준으로 조건 넣는 법
-

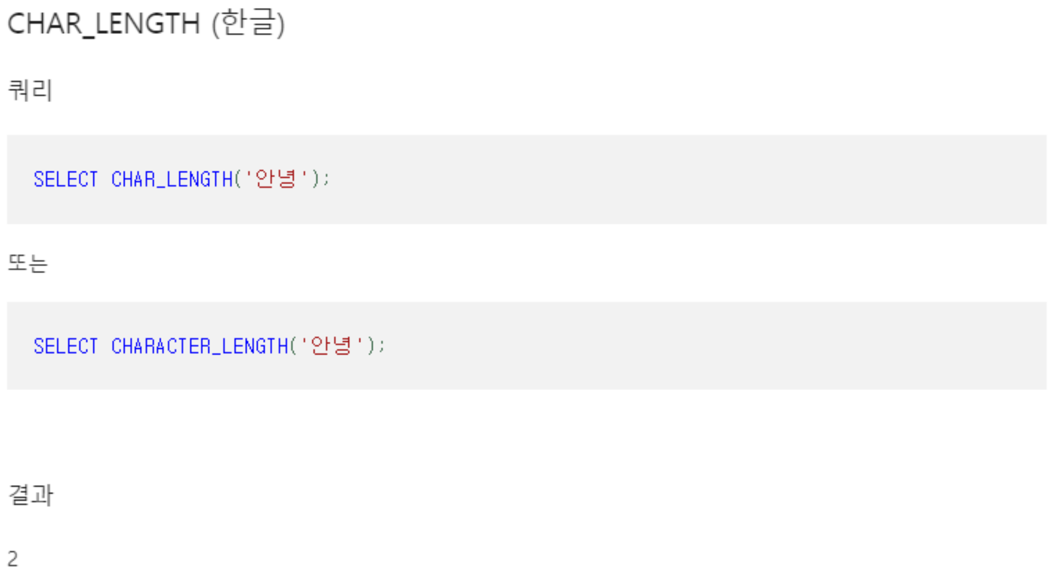
Built-in Functions

AVG, MAX, MIN, STD, SUM

COUNT

COUNT_DISTINCT

CHARACTER_LENGTH(str) or CHAR_LENGTH() -> 문자 길이 추출함수



CHAR_LENGTH (한글)

쿼리

```
SELECT CHAR_LENGTH('안녕');
```

또는

```
SELECT CHARACTER_LENGTH('안녕');
```

결과

2

CONCAT(str1, str2, ...) -> 문자열 합치기 함수

기본 사용

쿼리

```
SELECT CONCAT('안녕하세요.', '감사해요.', '잘있어요.', '다시만나요.') AS hello;
```

결과

hello
안녕하세요.감사해요.잘있어요.다시만나요.

예제 테이블 : hero_collection

idx	type	name
1	1	안중근
2	1	윤봉길
3	2	김유신
4	2	이순신
5	3	이성계
6	3	왕건
7	4	반갑수

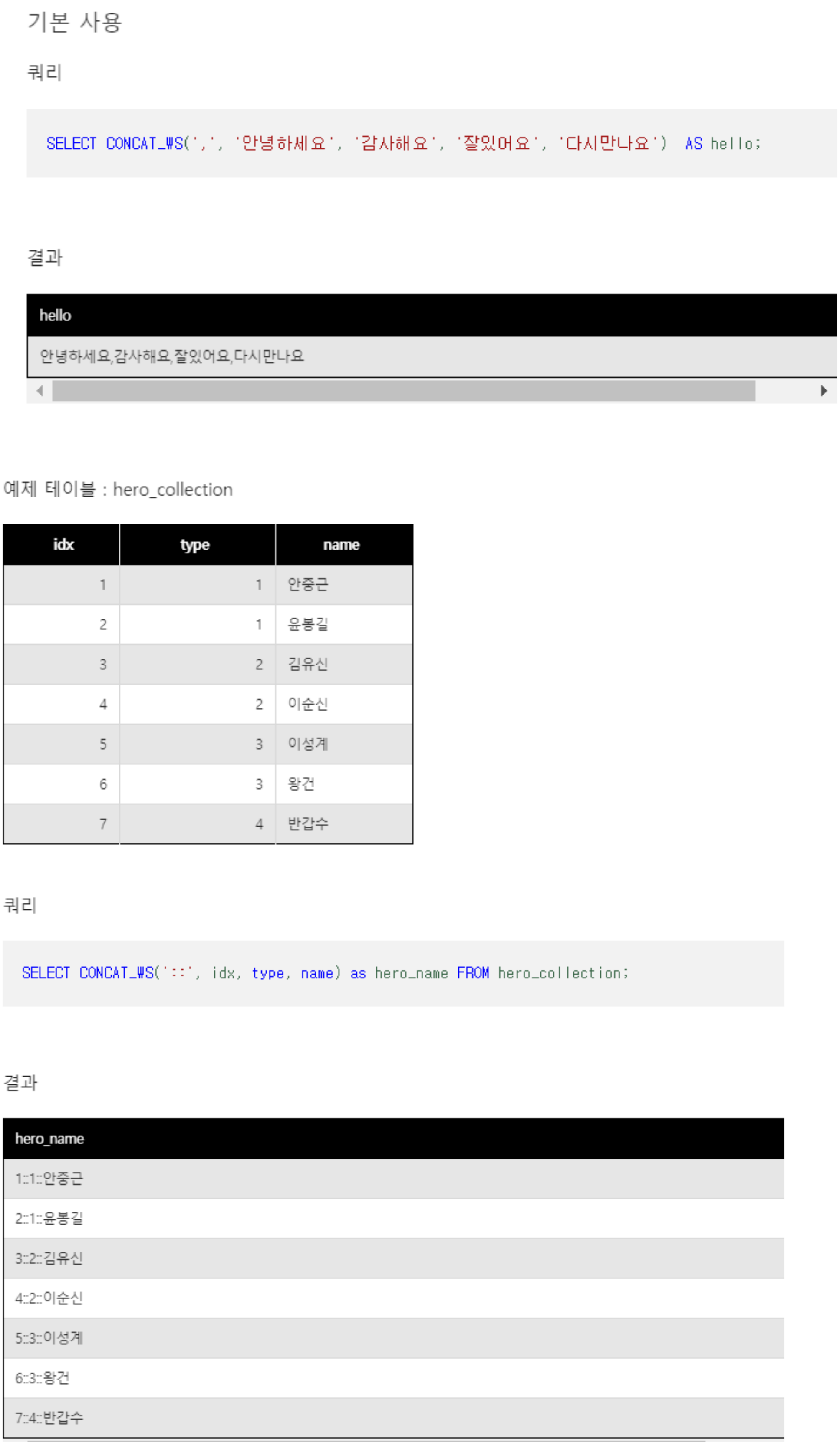
쿼리

```
SELECT CONCAT(type, '::', name) as hero_name FROM hero_collection;
```

결과

hero_name
1::안중근
1::윤봉길
2::김유신
2::이순신
3::이성계
3::왕건
4::반갑수

CONCAT_WS(separator, str1, str2, ...) -> 문자열 구분자 넣어서 합치기



기본 사용

쿼리

```
SELECT CONCAT_WS(',', '안녕하세요', '감사해요', '잘있어요', '다시만나요') AS hello;
```

결과

hello
안녕하세요,감사해요,잘있어요,다시만나요

예제 테이블 : hero_collection

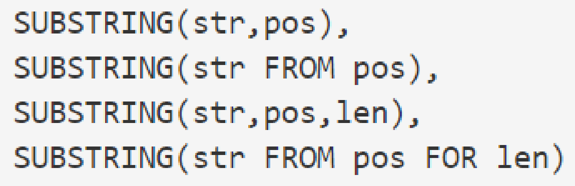
idx	type	name
1	1	안중근
2	1	윤봉길
3	2	김유신
4	2	이순신
5	3	이성계
6	3	왕건
7	4	반갑수

쿼리

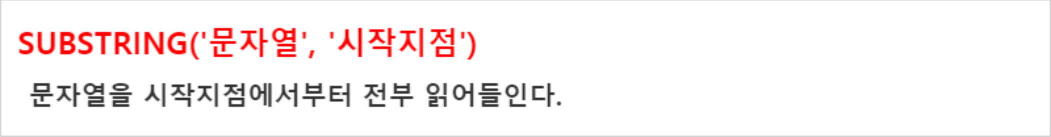
```
SELECT CONCAT_WS('::', idx, type, name) as hero_name FROM hero_collection;
```

결과

hero_name
1::1::안중근
2::1::윤봉길
3::2::김유신
4::2::이순신
5::3::이성계
6::3::왕건
7::4::반갑수

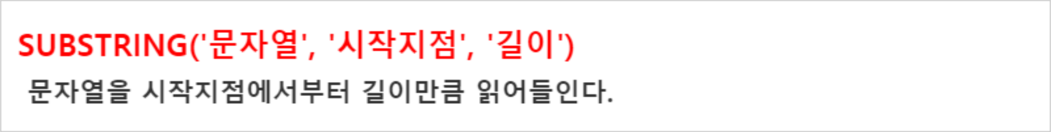


```
SUBSTRING(str,pos),
SUBSTRING(str FROM pos),
SUBSTRING(str,pos,len),
SUBSTRING(str FROM pos FOR len)
```



SUBSTRING('문자열', '시작지점')

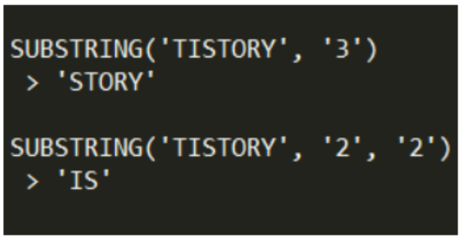
문자열을 시작지점에서부터 전부 읽어들인다.



SUBSTRING('문자열', '시작지점', '길이')

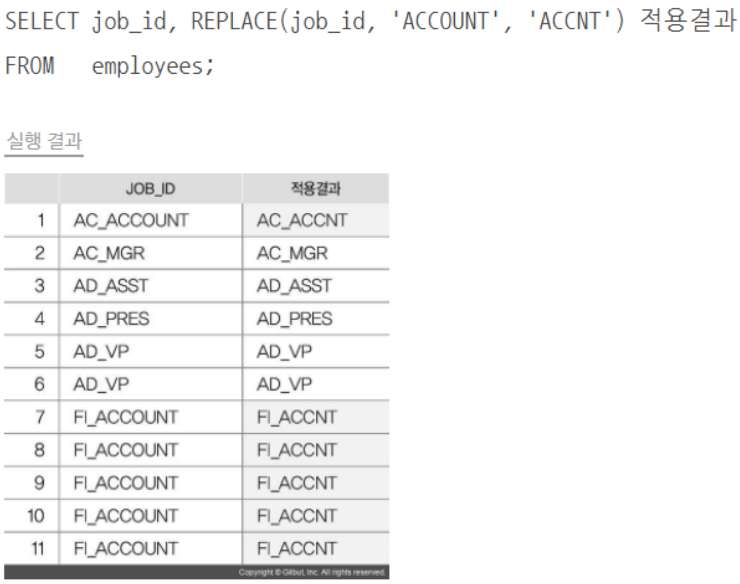
문자열을 시작지점에서부터 길이만큼 읽어들인다.

위와 같이 두가지 방법으로 사용할 수 있다.



```
SUBSTRING('TISTORY', '3')
 > 'STORY'

SUBSTRING('TISTORY', '2', '2')
 > 'IS'
```

REPLACE(str, from_str, to_str) -> 문자열 교체하기



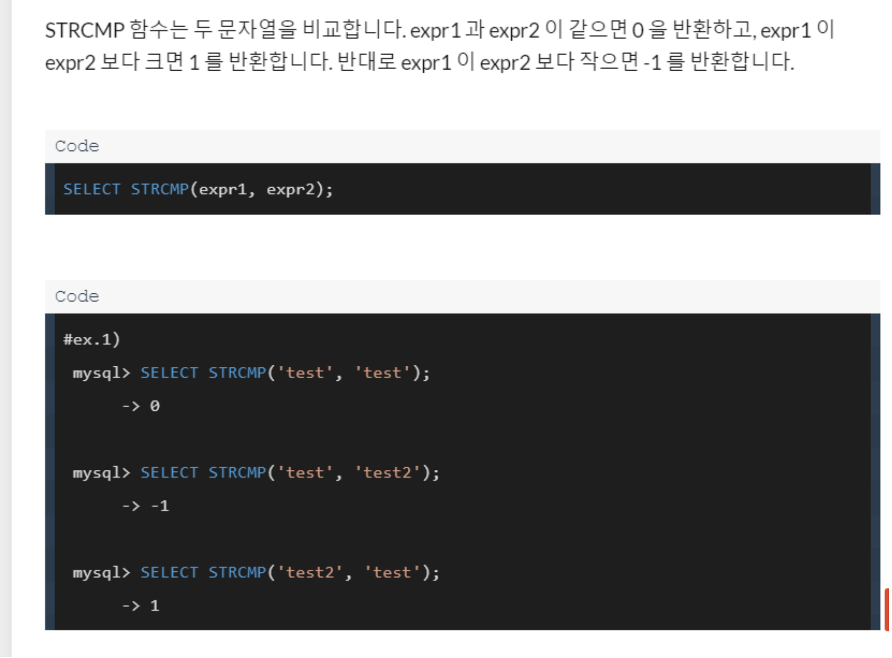
```
SELECT job_id, REPLACE(job_id, 'ACCOUNT', 'ACCNT') 적용결과
FROM   employees;
```

실행 결과

	JOB_ID	적용결과
1	AC_ACCOUNT	AC_ACCNT
2	AC_MGR	AC_MGR
3	AD_ASST	AD_ASST
4	AD_PRES	AD_PRES
5	AD_VP	AD_VP
6	AD_VP	AD_VP
7	FI_ACCOUNT	FI_ACCNT
8	FI_ACCOUNT	FI_ACCNT
9	FI_ACCOUNT	FI_ACCNT
10	FI_ACCOUNT	FI_ACCNT
11	FI_ACCOUNT	FI_ACCNT

Copyright © Gilbut, Inc. All rights reserved.

STRCMP(expr1, expr2) -> 문자열 비교하기 **(1,0,-1)**



STRCMP 함수는 두 문자열을 비교합니다. expr1 과 expr2 이 같으면 0 을 반환하고, expr1 이 expr2 보다 크면 1 를 반환합니다. 반대로 expr1 이 expr2 보다 작으면 -1 를 반환합니다.

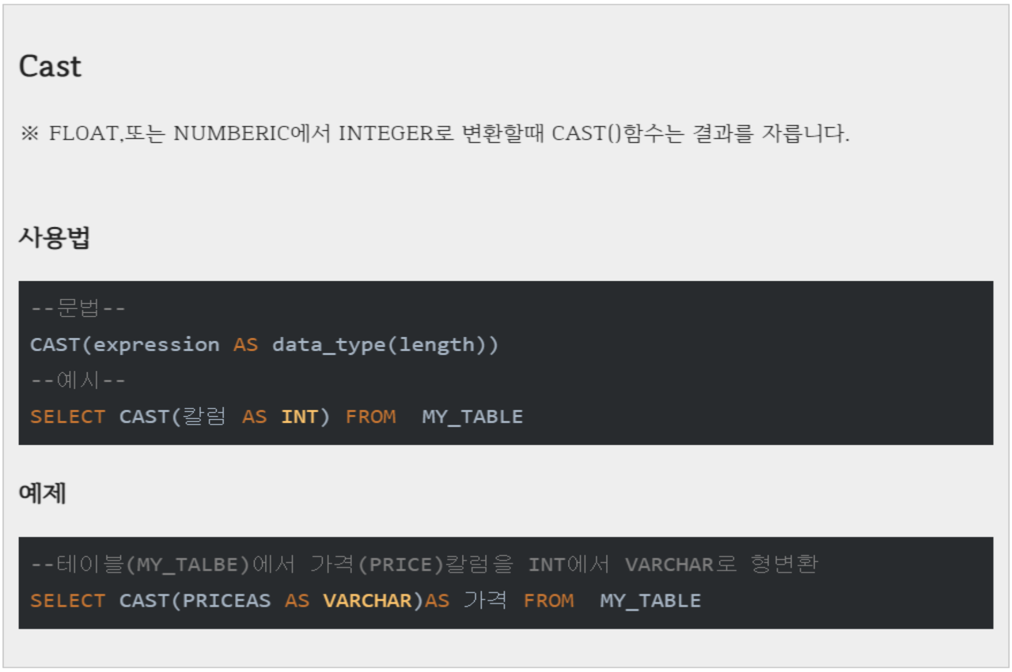
Code

```
SELECT STRCMP(expr1, expr2);
```

Code

```
#ex.1)
 mysql> SELECT STRCMP('test', 'test');
      -> 0

 mysql> SELECT STRCMP('test', 'test2');
      -> -1

 mysql> SELECT STRCMP('test2', 'test');
      -> 1
```

CAST(expr AS type) -> 형 바꾸기



Cast

※ FLOAT.또는 NUMBERIC에서 INTEGER로 변환할때 CAST()함수는 결과를 자릅니다.

사용법

```
--문법--
CAST(expression AS data_type(length))
--예시--
SELECT CAST(칼럼 AS INT) FROM  MY_TABLE
```

예제

```
--테이블(MY_TALBE)에서 가격(PRICE)칼럼을 INT에서 VARCHAR로 형변환
SELECT CAST(PRICEAS AS VARCHAR)AS 가격 FROM  MY_TABLE
```

CURDATE & CURTIME

- **SELECT CURDATE();**

  결과 : **YYYY-MM-DD || YYYYMMDD(**시간 반환**X)**

- **SELECT CURTIME();**

  HH:MM:SS || HHMMSS

UNIX_TIMESTAMP

- 현재시간을 unix time으로 구하기 : SELECT UNIX_TIMESTAMP()
- date를 유닉스 시간으로 바꾸기 : SELECT UNIX_TIMESTAMP('2009-05-15 20:11:22')

FROM_UNIXTIME(unix_timestamp)

- 유닉스 시간을 보기쉬운 포멧으로 나타내줌

  SELECT FROM_UNIXTIME(13191184471);   -> 2011-10-20 22:47:48

YEAR

MONTH

DAYOFMONTH   DAYOFWEEK

HOUR

MINUTE

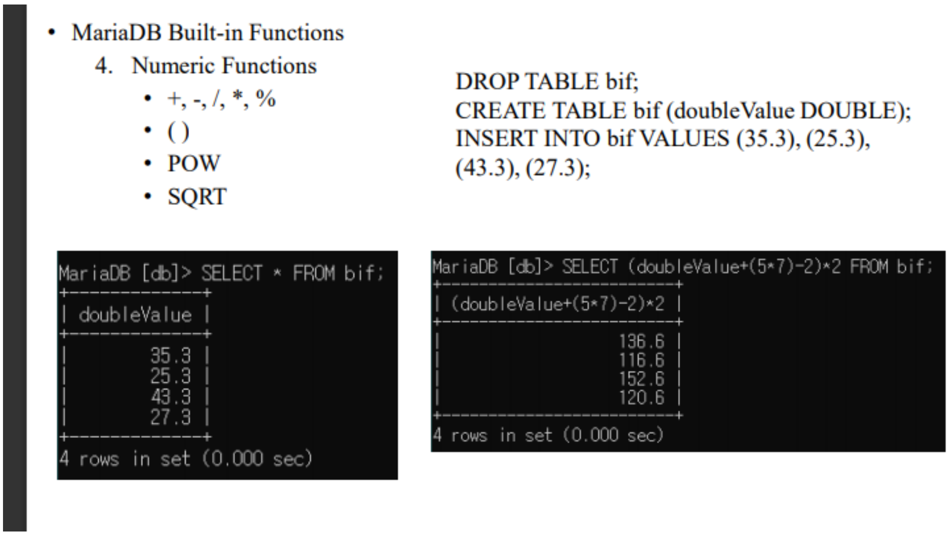
SECOND

STR_TO_DATE(str,format)

DATE_FORMAT(date, format[, locale])

VARIANCE

+, -, /, *, %, ( ) -> select할때 계산할 수 있는 함수



- MariaDB Built-in Functions
  4. Numeric Functions
     - +, -, /, *, %
     - ()
     - POW
     - SQRT

DROP TABLE bif;
CREATE TABLE bif (doubleValue DOUBLE);
INSERT INTO bif VALUES (35.3), (25.3),
(43.3), (27.3);

```
MariaDB [db]> SELECT * FROM bif;
+-------------+
| doubleValue |
+-------------+
|        35.3 |
|        25.3 |
|        43.3 |
|        27.3 |
+-------------+
4 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT (doubleValue+(5*7)-2)*2 FROM bif;
+-------------------------+
| (doubleValue+(5*7)-2)*2 |
+-------------------------+
|                   136.6 |
|                   116.6 |
|                   152.6 |
|                   120.6 |
+-------------------------+
4 rows in set (0.000 sec)
```

POW -> 지수곱

SELECT POW(2, 4);

// 결과는 16

SELECT POW(2, 3);

// 결과는 8

SQRT -> 루트

- SELECT SQRT(9), -> 결과는 3

FLOOR -> 소수점 버림

- `SELECT FLOOR(135.375); -- 135`

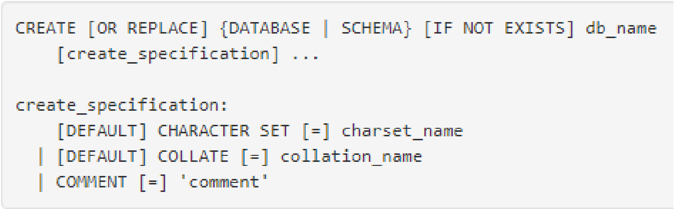
CEILING -> 소수점 반올림

- **SELECT CEIL(135.375); -- 136**
- **SELECT CEILING(135.375); -- 136**

RAND

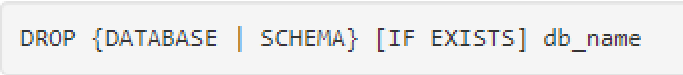
- **Rand() -> 0~1**에서 값을 랜덤으로 리턴한다 **(**예**: 0.43325987654098)**

CREATE DATABASE



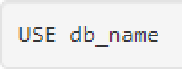
```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```

DROP DATABASE



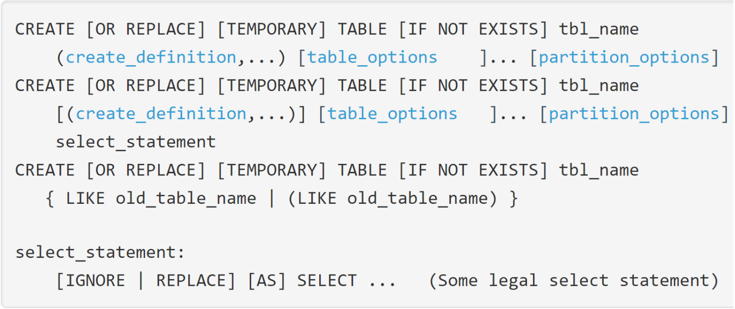
```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

USE



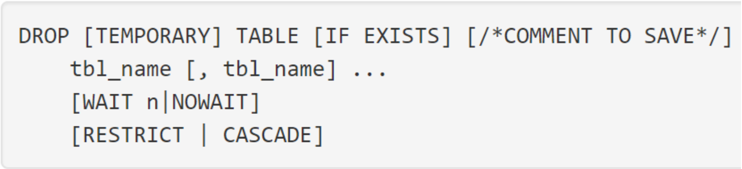
```
USE db_name
```

CREATE TABLE



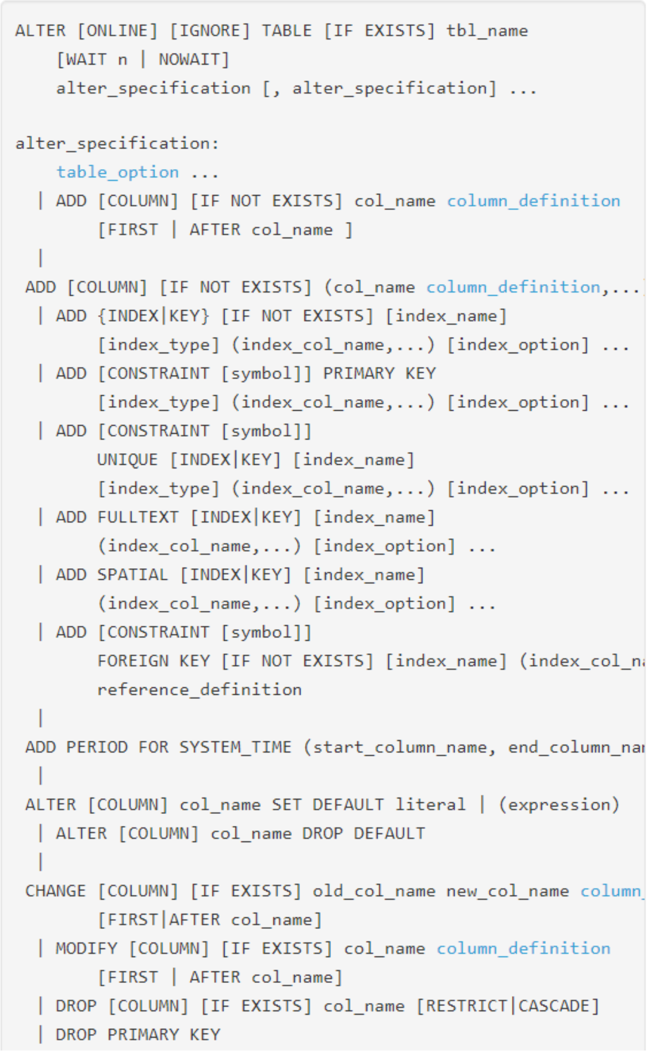
```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options   ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

DROP TABLE

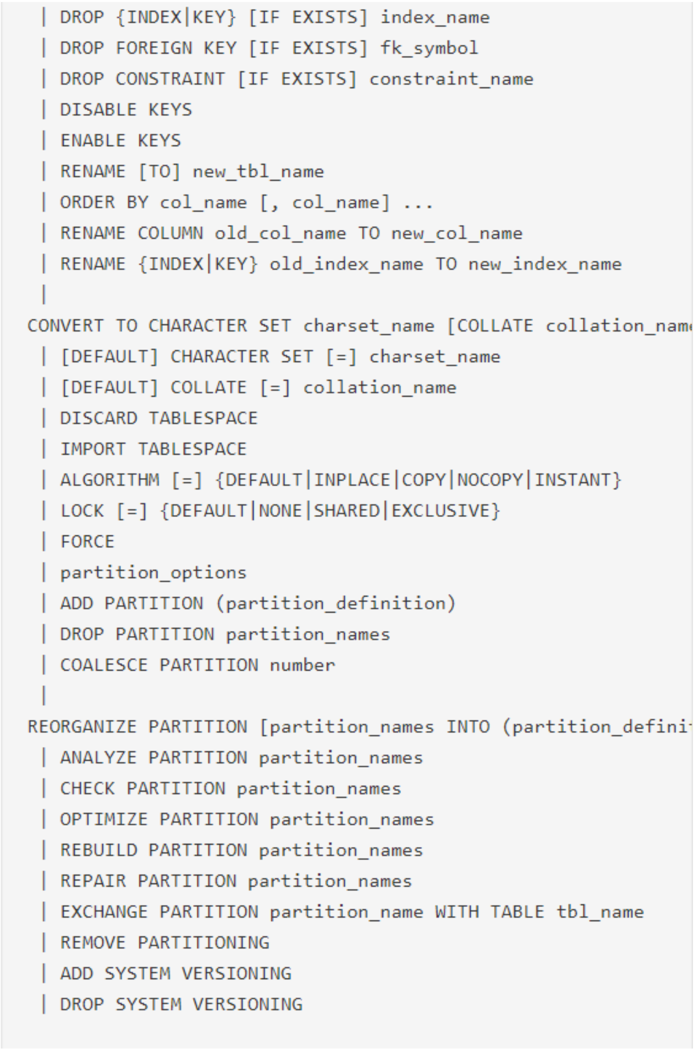


```
DROP [TEMPORARY] TABLE [IF EXISTS] [/*COMMENT TO SAVE*/]
    tbl_name [, tbl_name] ...
    [WAIT n|NOWAIT]
    [RESTRICT | CASCADE]
```

ALTER TABLE

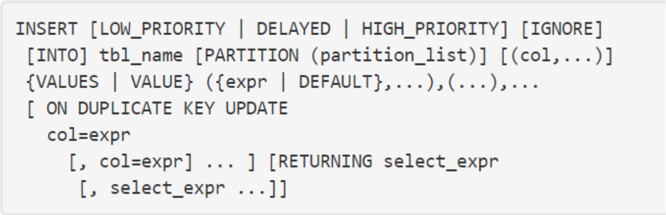


```
ALTER [ONLINE] [IGNORE] TABLE [IF EXISTS] tbl_name
    [WAIT n | NOWAIT]
    alter_specification [, alter_specification] ...

alter_specification:
    table_option ...
  | ADD [COLUMN] [IF NOT EXISTS] col_name column_definition
        [FIRST | AFTER col_name ]
  |
 ADD [COLUMN] [IF NOT EXISTS] (col_name column_definition,...
  | ADD {INDEX|KEY} [IF NOT EXISTS] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [IF NOT EXISTS] [index_name] (index_col_n
        reference_definition
  |
 ADD PERIOD FOR SYSTEM_TIME (start_column_name, end_column_na
  |
 ALTER [COLUMN] col_name SET DEFAULT literal | (expression)
  | ALTER [COLUMN] col_name DROP DEFAULT
  |
 CHANGE [COLUMN] [IF EXISTS] old_col_name new_col_name column_
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] [IF EXISTS] col_name column_definition
        [FIRST | AFTER col_name]
  | DROP [COLUMN] [IF EXISTS] col_name [RESTRICT|CASCADE]
  | DROP PRIMARY KEY
```



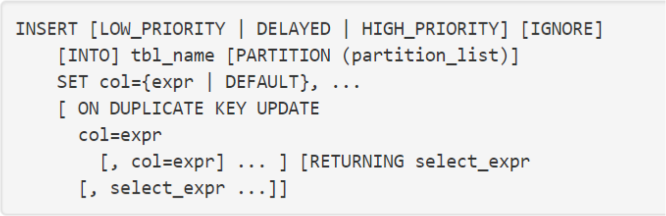
```
  | DROP {INDEX|KEY} [IF EXISTS] index_name
  | DROP FOREIGN KEY [IF EXISTS] fk_symbol
  | DROP CONSTRAINT [IF EXISTS] constraint_name
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name [, col_name] ...
  | RENAME COLUMN old_col_name TO new_col_name
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  |
CONVERT TO CHARACTER SET charset_name [COLLATE collation_nam
  | [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | FORCE
  | partition_options
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | COALESCE PARTITION number
  |
REORGANIZE PARTITION [partition_names INTO (partition_defini
  | ANALYZE PARTITION partition_names
  | CHECK PARTITION partition_names
  | OPTIMIZE PARTITION partition_names
  | REBUILD PARTITION partition_names
  | REPAIR PARTITION partition_names
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name
  | REMOVE PARTITIONING
  | ADD SYSTEM VERSIONING
  | DROP SYSTEM VERSIONING
```

INSERT INTO



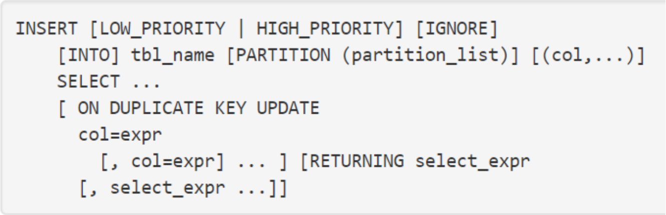
```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:



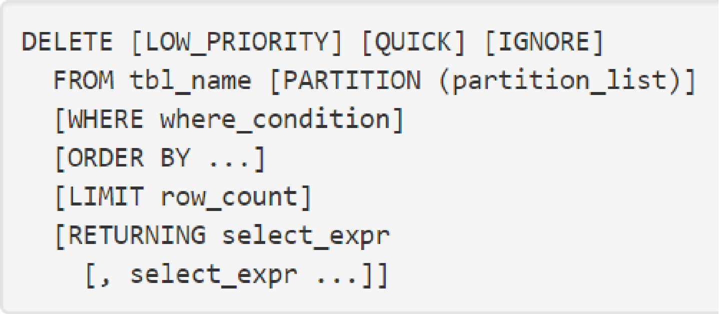
```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:



```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

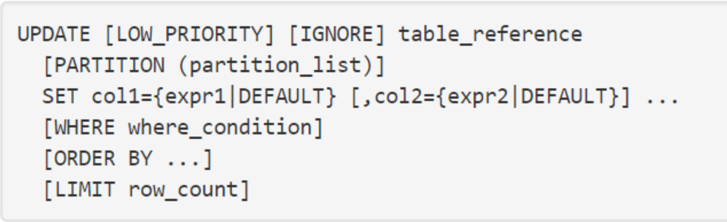
DELETE FROM



```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name [PARTITION (partition_list)]
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
    [, select_expr ...]]
```

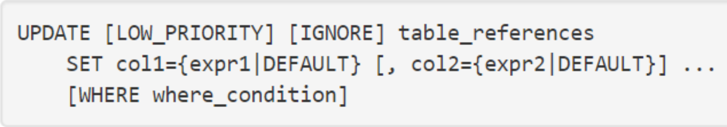
UPDATE TABLE

Single-table syntax:



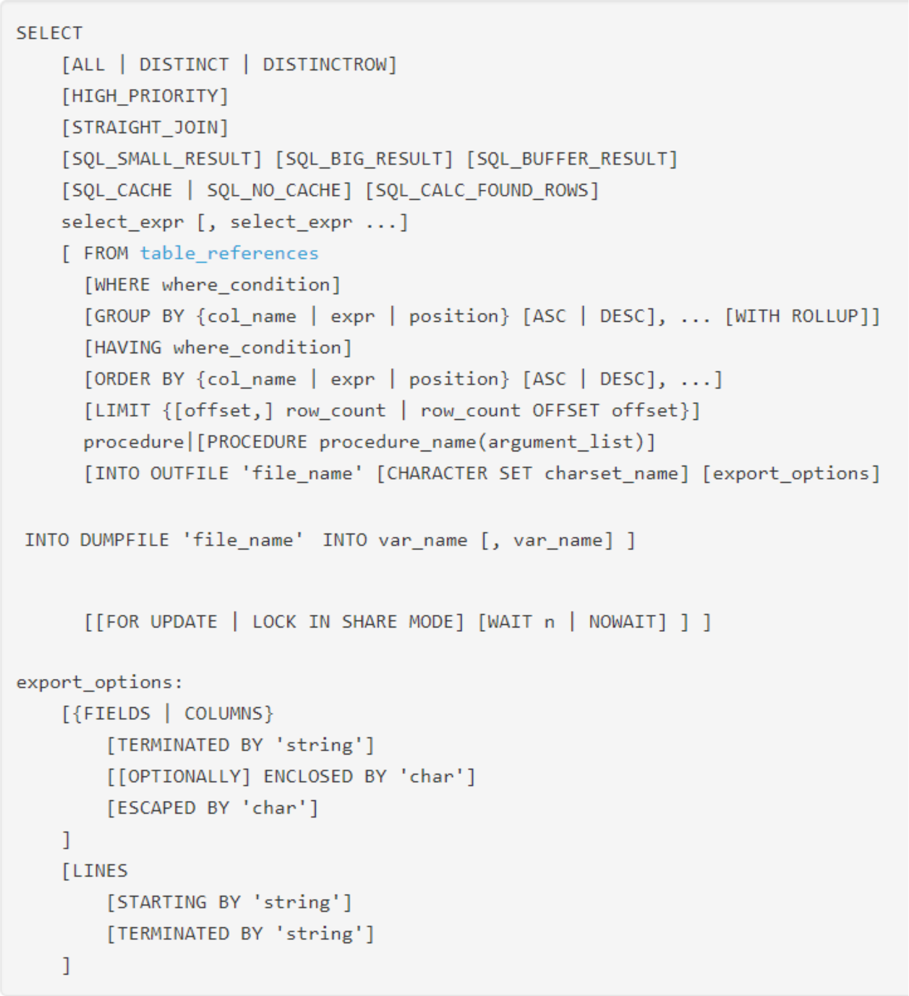
```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  [PARTITION (partition_list)]
  SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
```

Multiple-table syntax:



```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col1={expr1|DEFAULT} [, col2={expr2|DEFAULT}] ...
    [WHERE where_condition]
```

SELECT



```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```