

데이터베이스 프로그래밍

SELECT 문

학습목표

- SELECT의 projection 기능을 이용해 원하는 컬럼만 조회할 수 있다.
- WHERE 절을 사용한 selection 기능을 이용해 특정 행만 조회할 수 있다.
- ORDER BY 절을 이용하여 데이터를 원하는 순서대로 정렬해서 조회할 수 있다.

SELECT 문의 기능

- PROJECTION

- 테이블에서의 질의 결과로 반환될 컬럼을 사용자가 필요한 만큼 선택할 수 있다.

- SELECTION

- 테이블에서의 질의 결과로 반환될 행을 선택할 수 있으며 다양한 조건을 사용하여 행을 선택적으로 제한할 수 있다.

- JOIN

- 서로 다른 테이블 간에 연결을 생성하여 각 테이블에 저장된 데이터를 함께 가져올 수 있다.

SELECT 구문

- 필요한 자료를 DB에 검색(요청)
- 튜플(tuple) 연산
 - 각각의 (행)레코드에 대해서 1줄마다 처리
- **SELECT** [**DISTINCT**] { * | column명 | 표현식 [별칭] } 5th
FROM 테이블명 1st
[WHERE 검색조건] 2nd
[GROUP BY column | 표현식] 3rd
[HAVING 검색조건] 4th
[ORDER BY {속성} [ASC | DESC]] ; 6th

표현식
기본 연산, 함수 사용 가능

별칭
컬럼 이름 변경하여 표시

기본 SELECT

- 여러 라인에 걸쳐서 작성이 가능
- 각 절은 되도록 다른 행에 작성하여 읽기 쉽고, 편집하기 쉽도록 작성
- TAB과 들여쓰기를 사용하여 좀 더 읽기 쉬운 SQL로 작성
- 키워드는 여러 행에 나누어 쓰거나 약어로 쓸 수 없다.
- 여러 컬럼을 검색할 때 쉼표로 컬럼 구분
- 출력 결과에 표시할 순서대로 컬럼 지정
- SELECT 결과 열 머리글(Heading)은 기본적으로 대문자로 표시

employees table에서 사원번호, 이름, 성, 급여, 담당업무를 조회

- SELECT employee_id, first_name, last_name, salary, job_id FROM employees;

Run SQL Command Line

```
SQL> SELECT employee_id, first_name, last_name, salary, job_id FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOB_ID
100	Steven	King	24000	AD_PRES
101	Neena	Kochhar	17000	AD_VP
102	Lex	De Haan	17000	AD_VP
103	Alexander	Hunold	9000	IT_PROG
104	Bruce	Ernst	6000	IT_PROG
105	David	Austin	4800	IT_PROG
106	Valli	Pataballa	4800	IT_PROG
107	Diana	Lorentz	4200	IT_PROG
108	Nancy	Greenberg	12008	FI_MGR
109	Daniel	Faviet	9000	FI_ACCOUNT
110	John	Chen	8200	FI_ACCOUNT

Run SQL Command Line

196	Alana	Walsh	3100	SH_CLERK
197	Kevin	Feeney	3000	SH_CLERK
198	Donald	OConnell	2600	SH_CLERK

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOB_ID
199	Douglas	Grant	2600	SH_CLERK
200	Jennifer	Whalen	4400	AD_ASST
201	Michael	Hartstein	13000	MK_MAN
202	Pat	Fay	6000	MK_REP
203	Susan	Mavris	6500	HR_REP
204	Hermann	Baer	10000	PR_REP
205	Shelley	Higgins	12008	AC_MGR
206	William	Gietz	8300	AC_ACCOUNT

107 rows selected.

SQL>

산술 표현식

- 산술 연산자의 종류 : +, -, *, /
- 여러 개의 산술 연산자를 사용할 때 우선순위 : *, / > +, -
- 같은 우선순위의 연산자는 좌측 > 우측
- ()를 사용하면 괄호가 첫 번째 우선순위
- SQL 문에서는 FROM 절을 제외한 모든 절에서 산술 연산자 사용 가능
- 산술 연산자 앞뒤의 공백 무시

연산자 우선 순위 확인을 위해 두 문장의 실행 결과 비교

- `SELECT first_name, last_name, salary, 2*salary+100 FROM employees;`
- `SELECT first_name, last_name, salary, 2*(salary+100) FROM employees;`

Run SQL Command Line

```
SQL> SELECT first_name, last_name, salary, 2*salary+100 FROM employees;
```

FIRST_NAME	LAST_NAME	SALARY	2*SALARY+100
Steven	King	24000	48100
Neena	Kochhar	17000	34100
Lex	De Haan	17000	34100
Alexander	Hunold	9000	18100
Bruce	Ernst	6000	12100
David	Austin	4800	9700
Valli	Pataballa	4800	9700
Diana	Lorentz	4200	8500
Nancy	Greenberg		
Daniel	Faviet		
John	Chen		

Run SQL Command Line

```
SQL> SELECT first_name, last_name, salary, 2*(salary+100) FROM employees;
```

FIRST_NAME	LAST_NAME	SALARY	2*(SALARY+100)
Ismael	Sciarra		
Jose Manuel	Urman		
Luis	Popp		
Den	Raphaely		
Alexander	Khoo		

Run SQL Command Line

```
SQL> SELECT first_name, last_name, salary, 2*(salary+100) FROM employees;
```

FIRST_NAME	LAST_NAME	SALARY	2*(SALARY+100)
Steven	King	24000	48200
Neena	Kochhar	17000	34200
Lex	De Haan	17000	34200
Alexander	Hunold	9000	18200
Bruce	Ernst	6000	12200
David	Austin	4800	9800
Valli	Pataballa	4800	9800
Diana	Lorentz	4200	8600
Nancy	Greenberg	12008	24216
Daniel	Faviet	9000	18200
John	Chen	8200	16600

Run SQL Command Line

```
SQL> SELECT first_name, last_name, salary, 2*(salary+100) FROM employees;
```

FIRST_NAME	LAST_NAME	SALARY	2*(SALARY+100)
Ismael	Sciarra	7700	15600
Jose Manuel	Urman	7800	15800
Luis	Popp	6900	14000
Den	Raphaely	11000	22200
Alexander	Khoo	3100	6400

NULL (")

- indicate that a data value does **not exist in the database.**
- SQL null is a **state**, not a value.
- 한 행의 특정 컬럼에 데이터 값이 지정되지 않을 때의 값
- 알 수 없는 값으로 0 또는 공백과는 다름
- 0은 숫자이고 공백은 하나의 문자
- **널 값을 포함하는 연산의 경우 결과 값도 널**
 - 숫자를 0으로 나누면 오류가 발생하지만 널로 나누면 결과는 널
- 길이는 0

employees table에 데이터 삽입

```
INSERT INTO employees VALUES(207, 'JOHN', 'AUSTIN', 'AUSTINMAIL', '515.123.9999',  
'02/06/07', 'SH_CLERK', NULL, NULL, 124, 50);
```

Run SQL Command Line

515.123.4444	03/09/17 AD_ASST	4400		101	10
201 Michael	Hartstein		MHARTSTE		
515.123.5555	04/02/17 MK_MAN	13000		100	20
202 Pat	Fay		PFAY		
603.123.6666	05/08/17 MK_REP	6000		201	20
203 Susan	Mavris		SMAVRIS		
515.123.7777	02/06/07 HR_REP	6500		101	40
204 Hermann	Baer		HBAER		
515.123.8888	02/06/07 PR_REP	10000		101	70
205 Shelley	Higgins		SHIGGINS		
515.123.8080	02/06/07 AC_MGR	12008		101	110
206 William	Gietz		WGIEZT		
515.123.8181	02/06/07 AC_ACCOUNT	8300		205	110

107 rows selected.

SQL> _

Run SQL Command Line

515.123.4444	03/09/17 AD_ASST	4400		101	10
201 Michael	Hartstein		MHARTSTE		
515.123.5555	04/02/17 MK_MAN	13000		100	20
202 Pat	Fay		PFAY		
603.123.6666	05/08/17 MK_REP	6000		201	20
203 Susan	Mavris		SMAVRIS		
515.123.7777	02/06/07 HR_REP	6500		101	40
204 Hermann	Baer		HBAER		
515.123.8888	02/06/07 PR_REP	10000		101	70
205 Shelley	Higgins		SHIGGINS		
515.123.8080	02/06/07 AC_MGR	12008		101	110
206 William	Gietz		WGIEZT		
515.123.8181	02/06/07 AC_ACCOUNT	8300		205	110
207 JOHN	AUSTIN		AUSTINMAIL		
515.123.9999	02/06/07 SH_CLERK		124	50	

108 rows selected.

NULL 값 연산의 결과는 NULL

SELECT employee_id, first_name, salary, salary+300 from employees;

Run SQL Command Line

196 Alana	3100	3400
197 Kevin	3000	3300
198 Donald	2600	2900

EMPLOYEE_ID	FIRST_NAME	SALARY	SALARY+300
199	Douglas	2600	2900
200	Jennifer	4400	4700
201	Michael	13000	13300
202	Pat	6000	6300
203	Susan	6500	6800
204	Hermann	10000	10300
205	Shelley	12008	12308
206	William	8300	8600
207	JOHN		

108 rows selected.

SELECT employee_id, first_name, salary, NVL(salary,0)+300 from employees;

198 Donald	2600	2900
------------	------	------

EMPLOYEE_ID	FIRST_NAME	SALARY	NVL(SALARY,0)+300
199	Douglas	2600	2900
200	Jennifer	4400	4700
201	Michael	13000	13300
202	Pat	6000	6300
203	Susan	6500	6800
204	Hermann	10000	10300
205	Shelley	12008	12308
206	William	8300	8600
207	JOHN		300

108 rows selected.

NVL

- NVL(string1, replace_with)
- string1 : 해당 attr
- replace_with : NULL일 경우의 대체값

NVL

```
SQL> SELECT empno, ename, comm, NVL(comm, 0) FROM emp;
```

EMPNO	ENAME	COMM	NVL(COMM,0)
7369	SMITH		0
7499	ALLEN	300	300
7521	WARD	500	500
7566	JONES		0
7654	MARTIN	1400	1400
7698	BLAKE		0
7782	CLARK		0
7839	KING		0
7844	TURNER	0	0
7900	JAMES		0
7902	FORD		0
7934	MILLER		0

12 rows selected.



컬럼 별칭

- 질의의 결과를 표시할 때 일반적으로 선택한 컬럼의 이름을 HEADING으로 사용
- 속성에 별칭 부여 (ALIAS)
 - 속성의 이름을 변환
 - 주로 연산이 수행된 컬럼에 사용
 - 별칭에 공백 또는 특수문자(#, \$ 등)가 있거나 대소문자를 구분할 경우 " 가 필수
 - 속성 별칭 / 속성 "별칭" / 속성 as 별칭 / 속성 as "별칭"
 - 기본적으로 별칭 머리글은 대문자로 표시

리터럴(literal)

어학사전

영어사전

[literal](#) 미국·영국 [ˈlɪtərəl]  영국식  ★ [다른 뜻\(1건\)](#) | [예문보기](#)

1. (어구의 뜻이) 문자 그대로의 2. (번역이) 직역의 3. 상상력이 부족한

스페인어사전

[literal](#)  [예문보기](#)

[형용사] 1. 글자[문자]대로의 2. (번역이나 ... 3. (설명, 정보, 기술 등) 정확한, 엄밀한, 있는 그대로의

어학사전 literal에 대한 검색결과

독일어(3건), 포르투갈어(1건), 루마니아어(1건), 체코어(1건), 폴란드어(1건)

[어학사전 더보기](#) 

지식백과

[상수](#) [[literal](#)] [컴퓨터인터넷IT용어대사전](#) | [기술/공학](#) > [컴퓨터/통신/IT](#)

리터럴 상수(literal constant)를 줄여서 리터럴이라 한다. 문자열의 형태를 취하면서 직접 이것이 문자나 숫자를 표시하는 것이며, 변수와는 달리 이 문자열에 대한 값의 대입은 행할 수 없다. 예를... [더보기](#)

[리터럴](#) [[literal](#)] [전자용어사전](#) | [용어해설](#) > [기술/공학](#)

프로그래밍 언어에서 문자열 그 자체가 값을 나타내는 것. 예를 들면 X = '90'에서 문자열 90은 90이라는 값을 나타내는 리터럴이다.

Literal 문자열

- 열 이름이나 열 별칭이 아니면서 SELECT 목록에 포함된 문자, 숫자 또는 날짜 등의 상수
- 질의 결과에 포함되어 SELECT 목록의 열과 동일하게 취급됨
- 날짜 및 문자 리터럴은 반드시 작은 따옴표('')로 묶어야 하지만 숫자 리터럴은 작은 따옴표로 묶지 않는다.

각 문자열은 각 행(row)이 반환될 때마다 한
번씩 출력

연결(Concatenation) 연산자

- 컬럼을 다른 컬럼, 산술 계산식 또는 상수값에 연결하여 문자식 생성 가능
- 연산자의 좌우에 있는 컬럼이 결합되어 단일 컬럼으로 출력
- 속성의 이름과 값을 매끄럽게
- 기본적으로 || 기호 사용
- 문자열 literal을 사용하는 경우 'literal' 기호 사용
- 속성의 데이터값을 사용하는 경우 속성명만 사용
- 문자열에 null 값을 결합할 경우 결과는 문자열

중복 행 제거(DISTINCT)

- **DISTINCT 키워드를 사용하면**
 - 해당 컬럼에 같은 값이 존재하는 경우 결과를 출력할 때 중복되는 값은 하나만 출력
- **DISTINCT 키워드를 사용하지 않으면**
 - default가 ALL(모든 값 출력)
- **SELECT DISTINCT로 사용**

employees 테이블에서 업무의 종류를 조회

Pseudo Column

- **ROWNUM**

- column과 비슷한 성격의 Pseudo column으로써 SQL 처리 결과 집합의 각 행에 대해 임시로 부여하는 일련번호
- 테이블이나 집합에서 원하는 만큼의 행을 가져오고 싶을 때 WHERE 절에서 행의 개수를 제한하는 목적으로 사용
- 검색된 행의 일련번호
- ORDER BY에 의한 정렬 이전에 부여

- **ROWID**

- 테이블 내의 특정한 행을 유일하게 구별해 주는 값
- 데이터 타입은 ROWID

employees 테이블에서 Pseudo Column 조회

```
SELECT ROWID, ROWNUM, first_name, salary FROM employees;
```

file#
object
block#
block 내 위치

ROWID	ROWNUM	FIRST_NAME	SALARY
AAAEABAAEAAAADNABY	89	Kelly	3800
AAAEABAAEAAAADNABZ	90	Jennifer	3600
AAAEABAAEAAAADNABa	91	Timothy	2900
AAAEABAAEAAAADNABb	92	Randall	2500
AAAEABAAEAAAADNABc	93	Sarah	4000
AAAEABAAEAAAADNABd	94	Britney	3900
AAAEABAAEAAAADNABe	95	Samuel	3200
AAAEABAAEAAAADNABf	96	Vance	2800
AAAEABAAEAAAADNABg	97	Alana	3100
AAAEABAAEAAAADNABh	98	Kevin	3000
AAAEABAAEAAAADNABi	99	Donald	2600

ROWID	ROWNUM	FIRST_NAME	SALARY
AAAEABAAEAAAADNABj	100	Douglas	2600
AAAEABAAEAAAADNABk	101	Jennifer	4400
AAAEABAAEAAAADNABl	102	Michael	13000
AAAEABAAEAAAADNABm	103	Pat	6000
AAAEABAAEAAAADNABn	104	Susan	6500
AAAEABAAEAAAADNABo	105	Hermann	10000
AAAEABAAEAAAADNABp	106	Shelley	12008
AAAEABAAEAAAADNABq	107	William	8300
AAAEABAAEAAAADNABr	108	JOHN	

108 rows selected.

SQL>

행의 제한(WHERE)

- 레코드 필터링
 - WHERE 절을 사용하여 질의에서 반환되는 행 제한 가능
- WHERE 절은 만족해야 할 조건을 기술하며, FROM 절 바로 다음에 사용
- 조건을 만족하는 행이 반환됨
- 문자형 타입과 날짜 타입을 가진 컬럼을 특정 값과 비교하기 위해서는 작은 따옴표('')로 묶어서 비교 처리
- 숫자 타입의 값은 인용부호 없이 사용
- WHERE 절에서는 별칭을 사용할 수 없다.

SELECT 처리 순서

(SELECT 결과 WHERE 절의 순서 비교)

```
Run SQL Command Line
SQL> SELECT  employee_id 사원번호, salary FROM employees WHERE employee_id > 200;

사원번호      SALARY
-----
201           13000
202            6000
203            6500
204           10000
205           12008
206            8300
207
208            2400
209            3400
210

10 rows selected.

SQL> _
```

```
Run SQL Command Line
SQL> SELECT  employee_id 사원번호, salary FROM employees WHERE 사원번호 > 200;
SELECT  employee_id 사원번호, salary FROM employees WHERE 사원번호 > 200
                                           *
ERROR at line 1:
ORA-00904: "사원번호": invalid identifier

SQL> _
```


WHERE 절 사용

- **SELECT [DISTINCT] { * | column명 | 표현식 [별칭] }**
FROM 테이블명
[WHERE 검색조건];
- 조건식은 컬럼, 비교연산자, 비교대상(상수, 열 이름, 값 목록)으로 구성
- WHERE 절에 명시할 수 있는 조건
 - 테이블에 있는 데이터들을 걸러내는 필터 역할을 하는 일반 조건
 - 조인 시 테이블들을 연결하는 조인 조건
- 아래 항목들이 모여 표현식(expression)을 만들게 되고 표현식이 모여 조건절 구성
 - 산술연산자(+, -, *, /)
 - 비교연산자(=, >, <, <> 등)
 - 컬럼, 숫자나 문자상수
 - LIKE, IN, BETWEEN
 - IS NULL, IS NOT NULL;
 - 함수
 - 논리연산자(AND, OR, NOT)
 - ANY, SOME, ALL

비교 연산자

- 표현식을 다른 값이나 표현식과 비교하는 조건문에 사용

- =, >, >=, <, <=, <>, !=, ^=

- 예) WHERE salary >= 1500

WHERE first_name = 'JOHN'

WHERE salary > commission

WHERE salary * 12 < 27000

비교 연산자

연산자	설 명
단순 비교연산자	· 같다(=), 다르다(<>, !=, ^=), 부등호(>, <, >=, <=)
BETWEEN a AND b	· a와 b사이의 데이터 출력(a, b값 포함) · expr >= a AND expr<= b와 동일
IN (list)	· list의 여러 값 중 어느 하나와 일치하는 데이터 출력 · ANY, SOME과 같은 연산
LIKE	· 문자 형태로 일치하는 데이터 출력(%, _사용) · 특정 패턴에 속하는 값을 조회하고자 하는 경우 사용 - wildcard를 이용한 문자열 부분 매칭 - % : 0개/1개 이상의 문자와 대응(zero or many) - _ : 한 개의 문자와 대응(only one) - ESCAPE 뒤의 문자열로 시작하는 문자는 wildcard가 아닌 것으로 해석
IS NULL	· NULL값을 가진 데이터 출력
NOT BETWEEN a AND b	· a와b사이에 있지 않은 데이터 출력(a, b값 포함하지 않음)
NOT IN (list)	· list의 값과 일치하지 않는 데이터 출력
NOT LIKE	· 문자 형태와 일치하지 않는 데이터 출력
IS NOT NULL	· NULL값을 갖지 않는 데이터 출력

employees 테이블에 데이터 삽입

```
Run SQL Command Line

SQL> INSERT INTO employees VALUES(208, 'Andy', 'Chaplin', 'ChaplinMAIL', '515.135.9876', '12/03/27', 'FI_ACCOUNT', 2400, NULL, 100, 30);

1 row created.

SQL> INSERT INTO employees VALUES(209, 'M%ary', 'Queen', 'QueenMAIL', '123.456.7890', '11/11/14', 'FI_MGR', 3400, NULL, 114, 20);

1 row created.

SQL> _
```

```
Run SQL Command Line

      101      515.123.8888      02/06/07 PR_REP      10000
205 Shelley      70
      101      515.123.8080      Higgins      SHIGGINS
206 William      110      02/06/07 AC_MGR      12008
      205      515.123.8181      Gietz      WGIETZ
207 JOHN      110      02/06/07 AC_ACCOUNT      8300
      124      515.123.9999      AUSTIN      AUSTINMAIL
208 Andy      50      02/06/07 SH_CLERK
      100      515.135.9876      Chaplin      ChaplinMAIL
209 M%ary      30      12/03/27 FI_ACCOUNT      2400
      114      123.456.7890      Queen      QueenMAIL
      20      3400

110 rows selected.

SQL>
```

따옴표 사용 정리

- **AS** 사용시 큰따옴표 “ ”
- **||** 사용시 작은따옴표 ‘ ’
- **LIKE** 사용시 작은따옴표 ‘ ’

논리 연산자

- WHERE 절에 여러 개의 조건을 지정할 때 사용
- 두 조건의 결과를 결합하여 하나의 결과를 생성하거나 단일 조건의 결과를 부정하기도 함
- 조건 전체가 참인 경우에만 행이 반환
- 우선순위는 NOT, AND, OR의 순
- ()는 모든 우선 순위 규칙보다 우선

연산자	설 명
AND	양쪽 조건이 모두 true이면 true 반환
OR	양쪽 조건 중 하나만 true이면 true 반환
NOT	뒤따르는 조건이 false 인 경우 true 반환

NOT 연산자

- 뒤따르는 조건의 결과가 false이면 해당 ROW가 선택

연산자	설 명
NOT column명 = ~	~와 같지 않은 데이터 출력
NOT column명 > ~	~보다 크지 않은(작거나 같은) 데이터 출력
NOT column명 < ~	~보다 작지 않은(크거나 같은) 데이터 출력
~ NOT BETWEEN a AND b	a와 b사이에 값이 있지 않은 데이터 출력(a, b값 포함하지 않음)
~ NOT IN (list)	list의 값과 일치하지 않는 데이터 출력
~ NOT LIKE 해당문자형태	해당문자형태와 일치하지 않는 데이터 출력
~ IS NOT NULL	NULL값을 갖지 않는 데이터 출력

연산자 우선순위

실행순서	연 산 자
1	()
2	산술연산자
3	연결연산자()
4	비교연산자
5	IS [NOT] NULL, [NOT] LIKE, [NOT] IN
6	[NOT] BETWEEN .. AND ..
7	NOT
8	AND
9	OR

- 연산자들의 우선순위를 염두에 두지 않고 작성한다면 원하는 자료를 찾지 못하거나, 틀린 자료를 사용할 수 있다.
- 실수하기 쉬운 비교 연산자와 논리 연산자의 경우 괄호를 사용하여 우선순위를 표시하기를 권고

데이터 정렬

- ORDER BY 절은 조회된 데이터들을 다양한 목적에 맞게 특정 column을 기준으로 정렬하여 출력하는데 사용
- ORDER BY 절에 column 명 대신에 SELECT 절에서 사용한 ALIAS 명이나 column 순서를 나타내는 정수도 사용 가능
- 별도로 정렬 방식을 지정하지 않으면 기본적으로 오름차순이 적용
- 오름차순 정렬 시 숫자 값은 작은 값부터, 날짜 값은 과거 값부터, 문자 값은 영문자 순으로 표시
- 내림차순(DISCENDING) 정렬 수행 시 DESC 키워드 사용해야 함
- ORDER BY 절은 항상 SELECT 문의 마지막에 사용
- null은 오름차순에서는 마지막에, 내림차순에서는 제일 처음에 표시됨
- null을 NULLS FIRST(ASC시)나 NULLS LAST(DISC시) 키워드를 사용하여 순서 변경 가능
- 여러 열을 기준으로 정렬 시 각 열마다 오름차순, 내림차순을 개별적으로 지정 가능
- DISTINCT 키워드를 사용하지 않았다면, SELECT 절에 없는 컬럼 기준으로 정렬 가능
- DISTINCT 키워드를 사용했다면, ORDER BY 절의 컬럼은 SELECT 절에서 사용한 컬럼만 지정 가능