Introduction to Databases

Lecture 4

Data Manipulation Language

Nga Nguyen

SQL Data Manipulation Language (DML)

- INSERT
- UPDATE
- DELETE
- SELECT

Table examples

```
CREATE TABLE employee (
   first_name VARCHAR2(30),
   last_name  VARCHAR2(30),
   recruited_date DATE);

CREATE TABLE student (
   first_name VARCHAR2(30),
   last_name  VARCHAR2(30),
   class CHAR(3) DEFAULT 'ADEO');
```

INSERT

- Simplified syntax :

```
INSERT INTO <table_name>
VALUES (<expr>[,<expr>...]);
```

- Example :

```
INSERT INTO employee
VALUES('DERAY', 'ODILE',
   to_date('01/09/1972','DD/MM/YYYY'));
```

INSERT

- Explicit syntax :

```
INSERT INTO <table_name> (<col1> [, ... ])
VALUES (<expr1> [, ... ]);
```

- Example :

```
INSERT INTO employee
   (first_name,last_name,recruited_date)
VALUES('DERAY','ODILE',
   to_date('01/09/1972','DD/MM/YYYY'));
```

INSERT : partial insertion

- Explicit completion :

```
INSERT INTO employee
VALUES('DERAY','ODILE',NULL);

CREATE TABLE student (...,
class CHAR(3) DEFAULT 'ADEO');

INSERT INTO student
VALUES('DUPONT','EMILE',DEFAULT);
```

INSERT : partial insertion

- Implicit completion :
  - With NULL value :

```
INSERT INTO employee(first_name,last_name)
VALUES('DERAY','ODILE');
```

  - With default value :

```
INSERT INTO student(first_name,last_name)
VALUES('DUPONT','EMILE');
```

  - **All default values :**

```
INSERT INTO my_table DEFAULT VALUES;
```

INSERT : constraints

- Be careful with constraints !
  - Primary keys and UNIQUE (no double)
  - Foreign keys (existence)
  - CHECK, NOT NULL

UPDATE

```
UPDATE <table_name>
SET <col>=<expr>[,<col>=<expr>…]
[WHERE <condition>];
```

UPDATE : example

```
UPDATE
   person
SET
   date_of_birth = DATE '1900-01-01'
WHERE
   date_of_birth IS NULL;
```

DELETE

DELETE FROM <tab> [WHERE <condition>];

DELETE : example

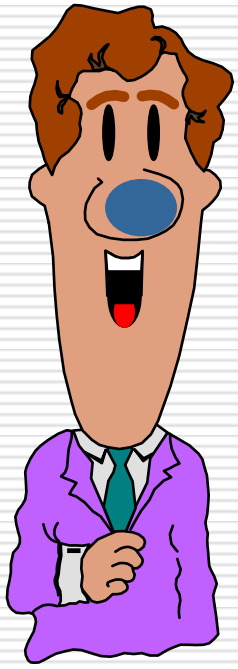
```
DELETE FROM person;

DELETE FROM person
WHERE date_of_birth IS NULL;
```

SELECT

```
SELECT { <col>[, <col>... ] | *}
FROM <table_name>
[WHERE <condition>];
```

Example

Wine	vintage	year	region
	CHENAS	1983	BEAUJOLAIS
	TOKAY	1980	ALSACE
	TAVEL	1986	RHONE
	CHABLIS	1986	BOURGOGNE
	ST-EMILION	1987	BORDELAIS



L'abus d'alcool est dangereux pour la santé, à consommer avec modération ☺

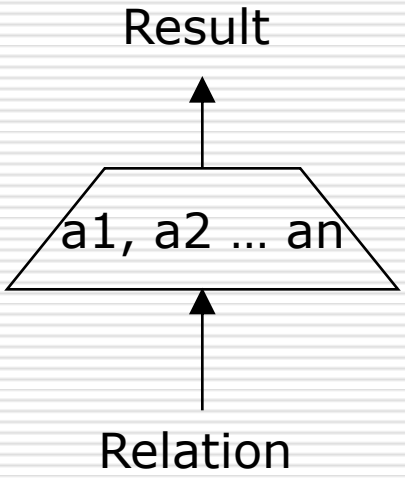
Projection with SELECT

```
SELECT vintage, region
FROM wine;
```

Uniqueness with DISTINCT

```
SELECT DISTINCT vintage,region
FROM wine;
```

Algebraic Tree : projection



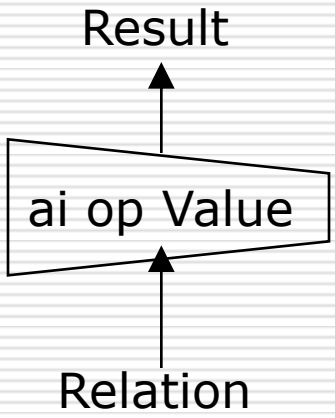
Restriction with WHERE

```
SELECT *
FROM my_table
WHERE <condition>;
```

- Example :

```
SELECT *
FROM wine
WHERE year > 1983;
```

Algebraic Tree : restriction



WHERE clause

- Comparing operators

  = , <> , <= , >= , < , >

- Logical operators

  AND, OR, NOT

- Example :

```
SELECT *
FROM my_table
WHERE c1 <> 12 AND c2 = 15;
```

NULL

- IS NULL or IS NOT NULL

- Example :

```
SELECT *
FROM my_table
WHERE c2 IS NULL;
```

LIKE

- String operation

- Meta-characters :

  _ : only one character

  % : zero or several characters

LIKE : example

- All persons whose name begins by ‘A’ :

```
SELECT *
FROM person
WHERE name LIKE 'A%' ;
```

- The first name of people whose name contains an ‘X’ at second position :

```
SELECT first_name
FROM person
WHERE name LIKE '_X%' ;
```

Other string operators

- Concatenation : || or CONCAT

```
SELECT first_name || ' ' || last_name
FROM …;

SELECT CONCAT(first_name,' ',last_name)
FROM …;
```

- Substring :

```
SUBSTR(str, pos, long)
```

Character conversion

- To upper case : UPPER(str)
- To lower case : LOWER(str)
- To number : TO_NUMBER(str)
- Example :

```
SELECT * FROM person
WHERE UPPER(name) = UPPER('Dupont');
```

Number text conversion

- Number => string :

  TO_CHAR(nb, mask)

- **The mask** defines the transformation rules :
  - 9 : number with insignificant 0
  - 0 : number
  - D : decimal separator (, or .)
  - G : group separator (. or ,)
  - C : ISO local money symbol

- Example : TO_CHAR(1555444.2, '999G999G999D99C')

  => 1.555.444,20EUR

Date conversion

- string => date : TO_DATE(str, format)
- date => string : TO_CHAR(date, format)
- Example : monday 17 november 2008
  - Year : YYYY, YY, YEAR
    - 2008, 08, TWO THOUSAND EIGHT
  - Month : MM, Mon, Month
    - 11, Nov, November
  - Week : WW (year), W (month)
    - 46, 3
  - Day : DDD (year), DD (month), D (week), DAY, DY
    - 322, 17, 1, Monday, MO

Date conversion

- Example : 14H53’45
  - Hours : HH, HH12, AM, PM
    - 14, 2, pm, pm
  - Minutes : MM
    - 53
  - Secondes : SS
    - 45
- Separators
  - - / . , ; : space
  - "a" : between quotes for the other characters
  - Example : 'HH"h"MM:SS' => 14h53:45

ORDER BY

- Syntax :

```
SELECT ...
FROM my_table
[WHERE ...]
ORDER BY COL1 [ASC|DESC]
        [, COL2 [ASC|DESC]*
```

ORDER BY

- Sort persons by year in decreasing order and then by name in alphabetical order :

```
SELECT *
FROM person
ORDER BY year DESC, name
```

Row functions

- Available functions
  - Mathematical functions : +, -, * , /
  - String, date
- Example :

```
SELECT name, quantity, unit_price, quantity*unit_price
FROM order_item;
```

```
SELECT name, quantity*unit_price AS total_price
FROM order_item;
```

Column functions

- AVG, SUM, MIN, MAX, COUNT, VARIANCE, STDDEV

SELECT function(an_attribute) FROM my_table;

Column functions

SELECT COUNT(*) FROM person;

person	id	name	firstname
	1	DERAY	Odile
	2	DUPONT	Emile
	3	DURAND	Norbert

R	count
	3

SELECT AVG(price) FROM product;

product	description	price
	chaise	25
	table	225
	lampe	20

R	avg
	90