CS2102 Database Systems

Slides adapted from Prof. Chan Chee Yong

LECTURE 03

SQL #1

PREVIOUS LECTURE

Create table

Create table syntax

```
CREATE TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type
      [ column_constraints [ ... ] ]
    | table_constraints }
  [, ...]
] );
```

Constraint checking keywords

◦ **Primary key:** PRIMARY KEY PRIMARY KEY (attr, …)
◦ **Foreign Key:** REFERENCES
  FOREIGN KEY … REFERENCES (…)
  ◦ ON [DELETE | UPDATE] action
◦ **Not-null:** NOT NULL CHECK (attr IS NOT NULL)
◦ **Unique:** UNIQUE UNIQUE (attr, …)
◦ **General:** CHECK (constraint)
◦ **Default:** DEFAULT value

Table modification

Insert into syntax

```
INSERT INTO table_name
[ ( column_name [, ...] ) ]
VALUES ( { expression | DEFAULT } [, ...] );
```

Delete from syntax

```
DELETE FROM table_name
[ WHERE condition ];
```

Update syntax

```
UPDATE table_name
SET column_name = { expression | DEFAULT }
[ WHERE condition ];
```

Table modification

Example

```
CREATE TABLE Students (
  studentID     integer PRIMARY KEY,
  name          varchar(100),
  dept          varchar(20) DEFAULT ‘CS’
);
```

Students

studentID	*name*	*dept*

Table modification

Example

```
CREATE TABLE Students (
  studentID     integer PRIMARY KEY,
  name          varchar(100),
  dept          varchar(20) DEFAULT ‘CS’
);
INSERT INTO Students VALUES (12345, ‘Alice’, ‘Eng’);
```

Students

studentID	*name*	*dept*
12345	Alice	Eng

Table modification

Example

```
CREATE TABLE Students (
  studentID     integer PRIMARY KEY,
  name          varchar(100),
  dept          varchar(20) DEFAULT ‘CS’
);
INSERT INTO Students VALUES (12345, ‘Alice’, ‘Eng’);
INSERT INTO Students (studentID) VALUES (23456);
```

Students

studentID	*name*	*dept*
12345	Alice	Eng
23456	null	CS

Table modification

Example

```
CREATE TABLE Students (
  studentID     integer PRIMARY KEY,
  name          varchar(100),
  dept          varchar(20) DEFAULT ‘CS’
);
INSERT INTO Students VALUES (12345, ‘Alice’, ‘Eng’);
INSERT INTO Students (studentID) VALUES (23456);
INSERT INTO Students (studentID) VALUES (12345);
```

Students

studentID	***name***	***dept***
12345	Alice	Eng
23456	null	CS

Table modification

Example

```
CREATE TABLE Students (
  studentID     integer PRIMARY KEY,
  name          varchar(100),
  dept          varchar(20) DEFAULT ‘CS’
);
INSERT INTO Students VALUES (12345, ‘Alice’, ‘Eng’);
INSERT INTO Students (studentID) VALUES (23456);
```

~~INSERT INTO Students (studentID) VALUES (12345);~~

Students

studentID	*name*	*dept*
12345	Alice	Eng
23456	null	CS

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
```

Students

studentID	*name*	*dept*
12345	Alice	Eng
23456	null	CS
34567	Bob	Eng

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
INSERT INTO Students (dept, name, studentID)
VALUES (‘Maths’, ‘Carol’, 45678);
```

Students

<u>studentID</u>	***name***	***dept***
12345	Alice	Eng
23456	null	CS
34567	Bob	Eng
45678	Carol	Maths

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
INSERT INTO Students (dept, name, studentID)
VALUES (‘Maths’, ‘Carol’, 45678);
DELETE FROM Students WHERE dept=‘Eng’;
```

Students

studentID	*name*	*dept*
~~12345~~	~~Alice~~	~~Eng~~
23456	null	CS
~~34567~~	~~Bob~~	~~Eng~~
45678	Carol	Maths

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);

INSERT INTO Students (dept, name, studentID)

VALUES (‘Maths’, ‘Carol’, 45678);

DELETE FROM Students WHERE dept=‘Eng’;
```

Students

studentID	*name*	*dept*
23456	null	CS
45678	Carol	Maths

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
INSERT INTO Students (dept, name, studentID)
VALUES (‘Maths’, ‘Carol’, 45678);
DELETE FROM Students WHERE dept=‘Eng’;
DELETE FROM Students; -- remove all
```

Students

studentID	*name*	*dept*
~~23456~~	~~null~~	~~CS~~
~~45678~~	~~Carol~~	~~Maths~~

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
INSERT INTO Students (dept, name, studentID)
VALUES (‘Maths’, ‘Carol’, 45678);
DELETE FROM Students WHERE dept=‘Eng’;
DELETE FROM Students; -- remove all
```

Students

studentID	*name*	*dept*

Table modification

Example

```
INSERT INTO Students VALUES (34567, ‘Bob’, ‘Eng’);
INSERT INTO Students (dept, name, studentID)
VALUES (‘Maths’, ‘Carol’, 45678);
DELETE FROM Students WHERE dept=‘Eng’;
DELETE FROM Students; -- remove all
INSERT INTO Students VALUES (12345, ‘Alice’, ‘Eng’);
INSERT INTO Students (studentID) VALUES (23456);
```

Students

studentID	*name*	*dept*
12345	Alice	Eng
23456	null	CS

Table modification

Example

```
UPDATE Students SET name = ‘Bob’ WHERE dept = ‘CS’;
```

Students

studentID	***name***	***dept***
12345	Alice	Eng
23456	Bob	CS

Table modification

Example

```
UPDATE Students SET name = ‘Bob’ WHERE dept = ‘CS’;

UPDATE Students SET studentID = studentID + 1;

-- update all

-- add 1 to studentID
```

Students

studentID	*name*	*dept*
12346	Alice	Eng
23457	Bob	CS

Schema modification

Alter table

- Add/remove/modify columns
    - `ALTER TABLE Students ALTER COLUMN dept DROP DEFAULT;`
    - `ALTER TABLE Students DROP COLUMN dept;`
    - `ALTER TABLE Students ADD COLUMN faculty varchar(20);`
    - etc
- Add/remove constraints
- etc

- Structured Query Language
  - Null values
  - Create/drop table
  - Table modification
  - Constraint checking
- Queries
  - Simple queries

Queries

Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT [ DISTINCT ]  select_list  -- select clause
FROM                 from_list    -- from clause
[ WHERE              condition ]  -- where clause
```

Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT [ DISTINCT ]  select_list -- select clause
FROM                 from_list   -- from clause
[ WHERE              condition ] -- where clause
```

`select_list`	specifies columns to be included in output
`from_list`	specifies list of relations
`condition`	specifies conditions on relations

Simple queries

Basic syntax

◦ Basic form of SQL query consists of *three clauses*

```
SELECT [ DISTINCT ]  select_list  -- select clause
FROM                 from_list    -- from clause
[ WHERE              condition ]  -- where clause
```

◦ `select_list`	specifies columns to be included in output
◦ `from_list`	specifies list of relations
◦ `condition`	specifies conditions on relations

❖ **Output:** relation generated from `from_list` containing attributes based on `select_list` that satisfies `condition`

- ➢ Output relation could contain duplicate record if `DISTINCT` is not used in the `SELECT` clause

Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT   DISTINCT     $a_1, a_2, \dots, a_m$   -- select clause
FROM                  $r_1, r_2, \dots, r_n$   -- from clause
  WHERE               $c$                      -- where clause
```

- **Relational algebra form**
  - $\pi_{a_1,a_2,\dots,a_m}\left(\sigma_c(r_1 \times r_2 \times \cdots \times r_n)\right)$

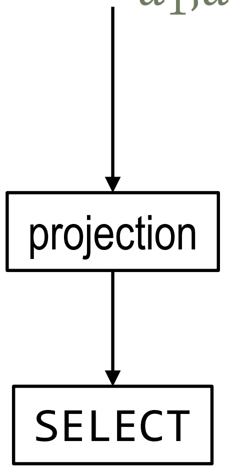
Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT   DISTINCT     $a_1, a_2, …, a_m$   -- select clause
FROM                  $r_1, r_2, …, r_n$   -- from clause
  WHERE               $c$                  -- where clause
```

- **Relational algebra form**
  - $\pi_{a_1,a_2,…,a_m}(\sigma_c(r_1 \times r_2 \times \cdots \times r_n))$



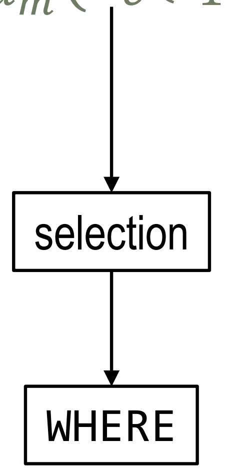
Simple queries

Basic syntax

- Basic form of SQL query consists of <u>*three clauses*</u>

SELECT DISTINCT $a_1, a_2, \ldots, a_m$ -- select clause
FROM $r_1, r_2, \ldots, r_n$ -- from clause
WHERE c -- where clause

- **Relational algebra form**
  - $\pi_{a_1,a_2,\ldots,a_m}\left(\sigma_c(r_1 \times r_2 \times \cdots \times r_n)\right)$



Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

SELECT DISTINCT $a_1, a_2, \ldots, a_m$ -- select clause

FROM $r_1, r_2, \ldots, r_n$ -- from clause

WHERE c -- where clause

- **Relational algebra form**
  - $\pi_{a_1,a_2,\ldots,a_m}\left(\sigma_c\left(\underbrace{r_1 \times r_2 \times \cdots \times r_n}\right)\right)$

$$\begin{tikzcd} r_1 \times r_2 \times \cdots \times r_n \arrow[d] \\ \boxed{\text{cross product}} \arrow[d] \\ \boxed{\text{FROM}} \end{tikzcd}$$

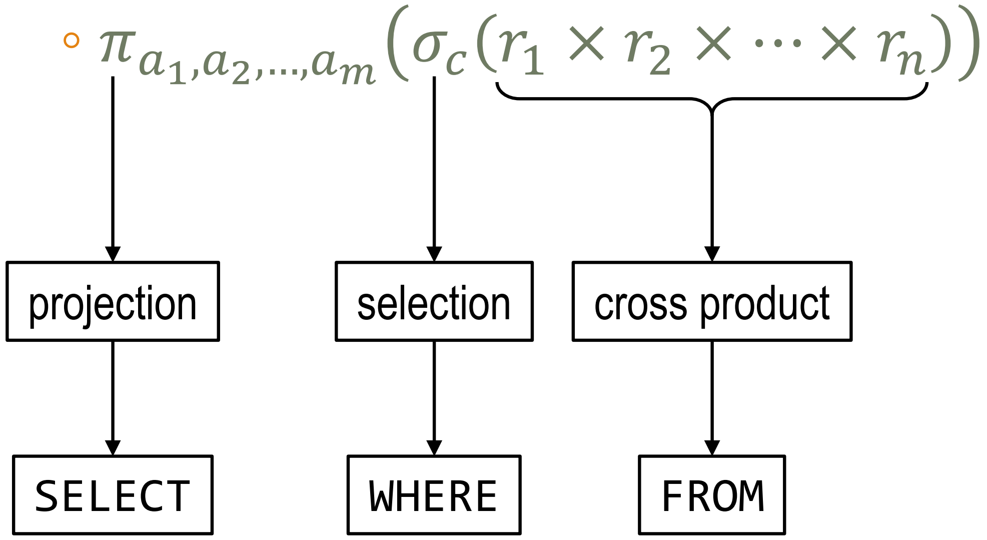
Simple queries

Basic syntax

- Basic form of SQL query consists of ***three clauses***

```
SELECT    DISTINCT      a_1, a_2, …, a_m    -- select clause
FROM                    r_1, r_2, …, r_n    -- from clause
  WHERE                 c                   -- where clause
```

- **Relational algebra form**
  - $\pi_{a_1,a_2,\dots,a_m}(\sigma_c(r_1 \times r_2 \times \cdots \times r_n))$



$\pi_{a_1,a_2,\dots,a_m}(\sigma_c(\underbrace{r_1 \times r_2 \times \cdots \times r_n}))$
projection
selection
cross product
SELECT
WHERE
FROM

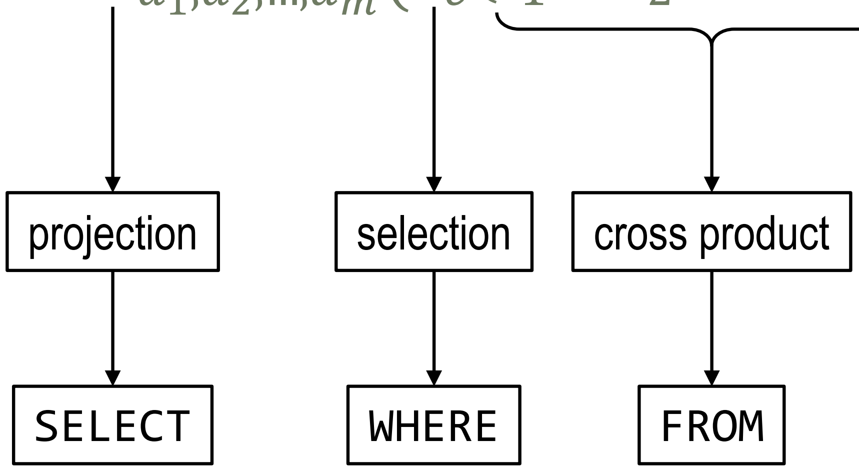
Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT    DISTINCT      a1, a2, ..., am    -- select clause
FROM                    r1, r2, ..., rn    -- from clause
  WHERE                 c                  -- where clause
```

- **Relational algebra form**
  - $\pi_{a_1,a_2,\ldots,a_m}(\sigma_c(r_1 \times r_2 \times \cdots \times r_n))$



What about renaming? ρ

Simple queries

Basic syntax

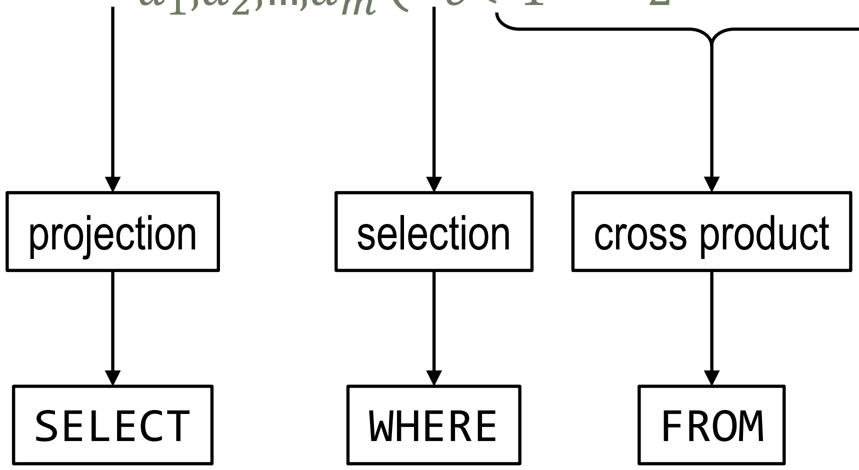
- Basic form of SQL query consists of *three clauses*

`SELECT DISTINCT` $\boldsymbol{a_1, a_2, \ldots, a_m}$ `-- select clause`

`FROM` $\boldsymbol{r_1, r_2, \ldots, r_n}$ `-- from clause`

`WHERE` $\boldsymbol{c}$ `-- where clause`

- **Relational algebra form**
  - $\pi_{a_1, a_2, \ldots, a_m}(\sigma_c(\underbrace{r_1 \times r_2 \times \cdots \times r_n}))$



What about renaming? ρ

$(\boldsymbol{a_i}$ AS $\boldsymbol{b_i})$

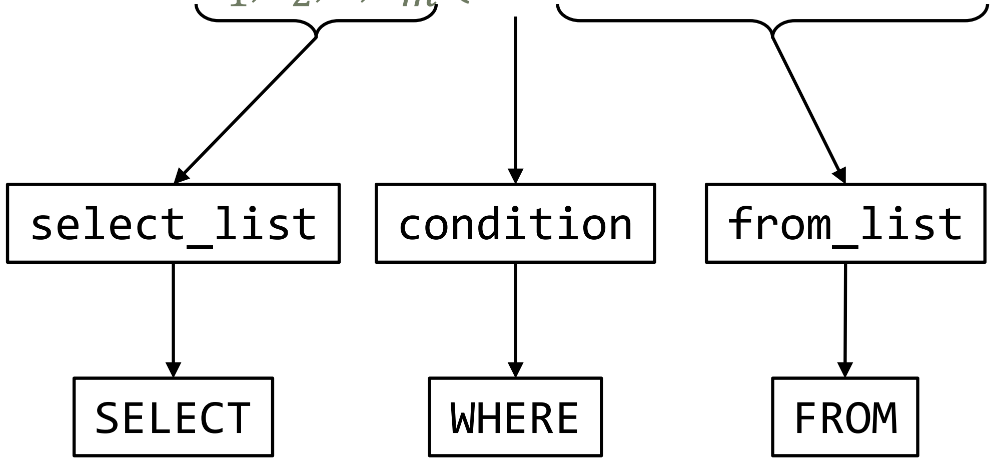
Simple queries

Basic syntax

- Basic form of SQL query consists of *three clauses*

```
SELECT    DISTINCT    $a_1, a_2, \dots, a_m$    -- select clause
FROM                  $r_1, r_2, \dots, r_n$    -- from clause
  WHERE               $c$                       -- where clause
```

- **Relational algebra form**
  - $\pi_{a_1,a_2,\dots,a_m}(\sigma_c(r_1 \times r_2 \times \cdots \times r_n))$



Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?**
  - **From which relation?**
  - **What condition?**

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem

Find what?	`rname, pizza, price`
From which relation?	`Sells`
What condition?	`price < 20`

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?** `rname, pizza, price`
  - **From which relation?** `Sells`
  - **What condition?** `price < 20`
- Construct query

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?** `rname, pizza, price`
  - **From which relation?** `Sells`
  - **What condition?** `price < 20`
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20;
```

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?** `rname, pizza, price`
  - **From which relation?** `Sells`
  - **What condition?** `price < 20`
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20;
```

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?** `rname, pizza, price`
  - **From which relation?** `Sells`
  - **What condition?** `price < 20`
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20;
```

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Deconstruct problem
  - **Find what?** `rname, pizza, price`
  - **From which relation?** `Sells`
  - **What condition?** `price < 20`
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20;
```

Sells

rname	***pizza***	***price***
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma’s Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Output

rname	***pizza***	***price***
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Pizza King	Diavola	17

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20;
```

Sells

rname	***pizza***	*price*
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma’s Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Output

rname	***pizza***	*price*
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Pizza King	Diavola	17

Simple queries

Example

- Find the names of restaurants, the pizzas that they sell, and their prices, where the price is under $20
- Construct query

```
SELECT rname, pizza, price
FROM   Sells
WHERE  price < 20
```

- Alternative query

```
SELECT *   -- all columns
FROM   Sells
WHERE  price < 20;
```

Output

rname	pizza	price
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Pizza King	Diavola	17

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma’s Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Simple queries

Example

- Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under $20 or the pizza is “Marinara”, and (2) the pizza is not “Diavola”
- Construct query

Simple queries

Example

- Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under $20 or the pizza is “Marinara”, and (2) the pizza is not “Diavola”
- Construct query

```
SELECT rname, pizza, price -- or simply *
FROM   Sells
WHERE  (price < 20 OR pizza = ‘Marinara’)
  AND  pizza <> ‘Diavola’;
```

Simple queries

Example

- Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under $20 or the pizza is “Marinara”, and (2) the pizza is not “Diavola”
- Construct query

```
SELECT rname, pizza, price -- or simply *
FROM   Sells
WHERE  (price < 20 OR pizza = ‘Marinara’)
  AND  pizza <> ‘Diavola’;
```

Simple queries

Example

- Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under $20 or the pizza is “Marinara”, and (2) the pizza is not “Diavola”
- Construct query

```
SELECT rname, pizza, price -- or simply *
FROM   Sells
WHERE  (price < 20 OR pizza = ‘Marinara’)
  AND  pizza <> ‘Diavola’;
```

Sells

rname	*pizza*	*price*
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma’s Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Output

rname	*pizza*	*price*
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Mamma’s Place	Marinara	22

Simple queries

Removing duplicates

r

a	*b*	*c*
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

Simple queries

Removing duplicates

- `q1: SELECT a, c FROM r;`

r

a	*b*	*c*
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

Simple queries

Removing duplicates

- `q1: SELECT a, c FROM r;`

r

a	*b*	*c*
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

q1

a	*c*
10	2
10	2
20	Null
20	Null
30	2
30	9

Simple queries

Removing duplicates

- `q1: SELECT a, c FROM r;`
- `q2: SELECT DISTINCT a, c FROM r;`

r

a	*b*	*c*
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

q1

a	*c*
10	2
10	2
20	Null
20	Null
30	2
30	9

Simple queries

Removing duplicates

- `q1: SELECT a, c FROM r;`
- `q2: SELECT DISTINCT a, c FROM r;`

r

a	*b*	*c*
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

q1

a	*c*
10	2
10	2
20	Null
20	Null
30	2
30	9

q2

a	*c*
10	2
20	Null
30	2
30	9

Simple queries

Removing duplicates

- `q1: SELECT a, c FROM r;`
- `q2: SELECT DISTINCT a, c FROM r;`

r

a	b	c
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

q1

a	c
10	2
10	2
20	Null
20	Null
30	2
30	9

q2

a	c
10	2
20	Null
30	2
30	9

- Two tuples $(a_1, a_2, \dots, a_n)$ and $(b_1, b_2, \dots, b_n)$ are considered to be distinct if the following evaluates to `TRUE`:
  - (a_1 IS DISTINCT FROM b_1) or
    (a_2 IS DISTINCT FROM b_2) or
    ... or
    (a_n IS DISTINCT FROM b_n)

Simple queries

Removing duplicates

- q1: `SELECT a, c FROM r;`
- q2: `SELECT DISTINCT a, c FROM r;`

r

a	b	c
10	1	2
10	7	2
20	3	Null
20	9	Null
30	3	2
30	5	9

q1

a	c
10	2
10	2
20	Null
20	Null
30	2
30	9

q2

a	c
10	2
20	Null
30	2
30	9

- Two tuples $(a_1, a_2, \dots, a_n)$ and $(b_1, b_2, \dots, b_n)$ are considered to be distinct if the following evaluates to TRUE:
  - $(a_1$ IS DISTINCT FROM $b_1)$ or
    $(a_2$ IS DISTINCT FROM $b_2)$ or
    ... or
    $(a_n$ IS DISTINCT FROM $b_n)$
  - In other words, if *any one of the attributes have different values*

Simple queries

Renaming column

- `q: SELECT item, price * qty AS cost FROM Orders;`

Orders

item	*price*	*qty*
A	2.50	100
B	4.00	100
C	7.50	100

Simple queries

Renaming column

- q: `SELECT item, price * qty AS cost FROM Orders;`

Orders

item	*price*	*qty*
A	2.50	100
B	4.00	100
C	7.50	100

q

item	*Cost*
A	250.00
B	400.00
C	750.00

Simple queries

Renaming column

- `q: SELECT item, price * qty AS cost FROM Orders;`

Orders

item	*price*	*qty*
A	2.50	100
B	4.00	100
C	7.50	100

q

item	*Cost*
A	250.00
B	400.00
C	750.00

Renaming values

```
q: SELECT ‘Price of ’ || pizza || ‘ is ’ ||
          round(price / 1.3) || ‘ USD’ AS menu
   FROM Sells WHERE rname = ‘Corleone Corner’;
```

Sells

rname	*pizza*	*price*
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Pizza King	Hawaiian	21

Simple queries

Renaming column

- q: SELECT item, price * qty AS cost FROM Orders;

Orders

item	*price*	*qty*
A	2.50	100
B	4.00	100
C	7.50	100

q

item	*Cost*
A	250.00
B	400.00
C	750.00

Renaming values

```
q: SELECT ‘Price of ’ || pizza || ‘ is ’ ||
          round(price / 1.3) || ‘ USD’ AS menu
   FROM Sells WHERE rname = ‘Corleone Corner’;
```

Sells

rname	***pizza***	*price*
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Pizza King	Hawaiian	21

q

menu
Price of Diavola is 18 USD
Price of Hawaiian is 19 USD
Price of Margherita is 15 USD

Summary

- SQL is the standard query language for relational DBMS
  - Table creation `CREATE TABLE` **`table_name`**
  - Table removal `DROP TABLE` **`table_name`**
  - Modification `ALTER TABLE` **`table_name`**
    - Insert `INSERT INTO` **`table_name`** `VALUES ( .. )`
    - Delete `DELETE FROM` **`table_name`** `WHERE ..`
    - Update `UPDATE` **`table_name`** `SET .. WHERE ..`
  - Queries

    `SELECT DISTINCT` $a_1, a_2, \ldots, a_m$

    `FROM` $r_1, r_2, \ldots, r_n$

    `WHERE` c

    - $\pi_{a_1,a_2,\ldots,a_m}\left(\sigma_c(r_1 \times r_2 \times \cdots \times r_n)\right)$