Structured Query Language

- Originally developed in the System-R project of IBM (1974)
- Industry standard for relational databases (SQL92 is an ANSI/ISO standard)

SQL - Data Definition Language

- Create table
  - Integrity Constraints (next lecture)
- Delete Table
- Alter Table

SQL DDL, creation (simple)

```
CREATE TABLE relation-name
  (attribute-name domain
  [, attribute-name domain ]*)
```

Example:
```
CREATE TABLE branch
  (name varchar(10),
   city varchar(20),
   director varchar(20),
   assets number)
```

Integrity Constraints – Primary Key

- **Primary Key** is a set of attributes which identifies uniquely a tuple (i.e., a row in a table)
  - E.g., your NRIC or your email address
  - The combination (name, city) in the branch table
- You cannot have two tuples with the same Primary Key in a table
  - E.g., there cannot be two persons with the same NRIC. There cannot be two branches with the same name in the same city
SQL DDL, creation example

CREATE TABLE branch
( name varchar(10),
city varchar(20),
director varchar(20),
assets number,
PRIMARY KEY (name, city) )

SQL DDL, Reference to other table

CREATE TABLE workfor
( branch_name varchar(10),
city varchar(20),
employee varchar(20)
REFERENCES staff(name) )

Example:

name must be the primary key of staff

SQL DDL, deletion

DROP TABLE relation_name

Example: DROP TABLE branch

SQL DDL, alteration

Example:

ALTER TABLE branch ADD zip INTEGER

ALTER TABLE branch DROP zip

SQL – Data Manipulation Language

- Insert
- Delete
- Update
- Queries
  - Simple Selections
  - Advanced queries (Aggregations, etc.)
  - Nested Queries
  - Views

SQL DML, insertion (values)

Example:

INSERT INTO branch (name, director, city, assets)
VALUES ('Clementi', 'Ng Wee Hiong', 'Singapore', 3000000)
SQL DML, insertion (query)

**Example:** INSERT INTO `johor_director` (we assume such a table was created)
- SELECT `director`
- FROM `branch`
- WHERE `city` = 'Johor Barhu'

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Ng Wei Hong</td>
<td>3000000</td>
</tr>
<tr>
<td>F_branch</td>
<td>Johor Barhu</td>
<td>John</td>
<td>1500000</td>
</tr>
<tr>
<td>S_branch</td>
<td>Johor Barhu</td>
<td>George</td>
<td>1200000</td>
</tr>
</tbody>
</table>

SQL DML, deletion

**Example:** DELETE FROM `branch` [WHERE `qualification`]
- WHERE `city` = 'Jakarta' and assets < 1000000

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monas</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
<td>4000000</td>
</tr>
</tbody>
</table>

SQL DML, update

**Example:** UPDATE `branch`
- SET `assets` = `assets` * 1.5
- WHERE `city` = 'Kuala Lumpur'

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Kuala Lumpur</td>
<td>Yu Fei</td>
<td>1500000</td>
</tr>
</tbody>
</table>

SQL DML, simple query

**Example:**
- SELECT `*`
- FROM `work_for`

<table>
<thead>
<tr>
<th>branch_name</th>
<th>city</th>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Kuala Lumpur</td>
<td>Yu Fei</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Ng Wei Hong</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Peter Ho</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Jean Do</td>
</tr>
<tr>
<td>Monas</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
</tr>
<tr>
<td>Monas</td>
<td>Jakarta</td>
<td>Reza Santi</td>
</tr>
</tbody>
</table>

SQL DML, update example

**Example:** UPDATE `branch`
- SET `assets` = `assets` * 1.5
- WHERE `city` = 'Kuala Lumpur'

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Kuala Lumpur</td>
<td>Yu Fei</td>
<td>1500000</td>
</tr>
</tbody>
</table>

**Example:** UPDATE `relation_name`
- SET `att` = `expr`
- [WHERE `qualification`]

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Ng Wei Hong</td>
<td>3000000</td>
</tr>
<tr>
<td>F_branch</td>
<td>Johor Barhu</td>
<td>John</td>
<td>1500000</td>
</tr>
</tbody>
</table>

**Example:**
- SELECT `DISTINCT` `target-list`
- FROM `relation-list`
- [WHERE `qualification`]

**Example:**
- SELECT `*`
- FROM `work_for`

**Example:**
- SELECT `*`
- FROM `relation_name`
- [WHERE `qualification`]

<table>
<thead>
<tr>
<th>branch_name</th>
<th>city</th>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Kuala Lumpur</td>
<td>Yu Fei</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Ng Wei Hong</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Peter Ho</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Jean Do</td>
</tr>
<tr>
<td>Monas</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
</tr>
<tr>
<td>Monas</td>
<td>Jakarta</td>
<td>Reza Santi</td>
</tr>
</tbody>
</table>
SQL DML, simple query example

```sql
SELECT name, city FROM branch
```

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch_one</td>
<td>Jakarta</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
</tr>
<tr>
<td>P_branch</td>
<td>Johor Barhu</td>
</tr>
<tr>
<td>KL_branch</td>
<td>Kuala Lumpur</td>
</tr>
<tr>
<td>Monas</td>
<td>Jakarta</td>
</tr>
<tr>
<td>S_branch</td>
<td>Johor Barhu</td>
</tr>
</tbody>
</table>

SQL DML, simple query example

```sql
SELECT DISTINCT employee, director FROM branch, workfor WHERE name = branch_name AND branch.city = workfor.city
```

<table>
<thead>
<tr>
<th>employee</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agus Arianto</td>
<td>Agus Arianto</td>
</tr>
<tr>
<td>Jean Do</td>
<td>Ng Wee Hiong</td>
</tr>
<tr>
<td>Ng Wee Hiong</td>
<td>Ng Wee Hiong</td>
</tr>
<tr>
<td>Peter Ho</td>
<td>Ng Wee Hiong</td>
</tr>
<tr>
<td>Reza Santi</td>
<td>Agus Arianto</td>
</tr>
</tbody>
</table>

However… Bag Semantics

```sql
SELECT DISTINCT branch_name FROM work_for
```

<table>
<thead>
<tr>
<th>branch_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
</tr>
<tr>
<td>Monas</td>
</tr>
</tbody>
</table>

```sql
SELECT DISTINCT branch_name FROM work_for
```

<table>
<thead>
<tr>
<th>branch_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
</tr>
<tr>
<td>Monas</td>
</tr>
</tbody>
</table>
However… List Semantics

- SELECT name, city
  FROM branch
  ORDER BY name ASC, city DESC

SQL Advanced

Arithmetic in SQL

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>director</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minas</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
<td>4000000</td>
</tr>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>Ng Wee Hiong</td>
<td>3000000</td>
</tr>
</tbody>
</table>

SELECT name, city, assets * 1.7 as assets_USD
FROM branch

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>assets_USD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clementi</td>
<td>Singapore</td>
<td>5100000</td>
</tr>
<tr>
<td>Minas</td>
<td>Jakarta</td>
<td>6800000</td>
</tr>
</tbody>
</table>

Arithmetic in SQL

- SELECT name, city
  FROM branch
  WHERE 1700000 > assets * 1.7

- There are numerous other built-in functions for the various data types available

Aggregate Queries

- SELECT COUNT(DISTINCT *)
  FROM branch

<table>
<thead>
<tr>
<th>Expr1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Aggregate Queries

- SELECT COUNT(DISTINCT city)
  FROM branch

<table>
<thead>
<tr>
<th>Expr1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Aggregate Queries

- SELECT COUNT(ALL city) FROM branch

```
<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Manager</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>Singapore</td>
<td>Ng Wei Ding</td>
<td>2500000</td>
</tr>
<tr>
<td>Agus</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
<td>3000000</td>
</tr>
<tr>
<td>George</td>
<td>Jakarta</td>
<td>George</td>
<td>1000000</td>
</tr>
<tr>
<td>John</td>
<td>Johor Barhu</td>
<td>John</td>
<td>1500000</td>
</tr>
<tr>
<td>Ng Wee Hiong</td>
<td>Singapore</td>
<td>Clementi</td>
<td>3000000</td>
</tr>
</tbody>
</table>
```

- SELECT AVG(assets) FROM branch

```
<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Manager</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>Singapore</td>
<td>Ng Wei Ding</td>
<td>2500000</td>
</tr>
<tr>
<td>Agus</td>
<td>Jakarta</td>
<td>Agus Arianto</td>
<td>3000000</td>
</tr>
<tr>
<td>George</td>
<td>Jakarta</td>
<td>George</td>
<td>1000000</td>
</tr>
<tr>
<td>John</td>
<td>Johor Barhu</td>
<td>John</td>
<td>1500000</td>
</tr>
<tr>
<td>Ng Wee Hiong</td>
<td>Singapore</td>
<td>Clementi</td>
<td>3000000</td>
</tr>
</tbody>
</table>
```

- SELECT city, AVG(assets) FROM branch GROUP BY city

```
<table>
<thead>
<tr>
<th>City</th>
<th>Expr1001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jakarta</td>
<td>2500000</td>
</tr>
<tr>
<td>Singapore</td>
<td>2000000</td>
</tr>
</tbody>
</table>
```

Aggregate Queries

- SELECT city, name, AVG(assets) FROM branch GROUP BY city, name

This particular query is uninteresting. Why?

- SELECT city, name, assets FROM branch

It is the same as the above, since (city, name) is a primary key!

Aggregate Queries

- References in the SELECT clause can only be made to aggregates and to attributes in the GROUP BY clause

SELECT manager, MIN(assets) FROM branch GROUP BY city

is incorrect

- The correct one is:

SELECT manager, MIN(assets) FROM branch GROUP BY city, name, manager
Aggregate Queries

- SELECT city
  FROM branch
  GROUP BY city
  WHERE AVG(assets) > 2000000
  IS incorrect!

Aggregate Queries

- SELECT city
  FROM branch
  GROUP BY city
  HAVING AVG(assets) > 2000000

Nested Queries

- SELECT employee
  FROM workfor
  WHERE branch_name IN (SELECT name FROM branch WHERE city = 'Singapore')

Nested Queries

- SELECT employee
  FROM workfor
  WHERE branch_name NOT IN (SELECT name FROM branch WHERE city = 'Singapore')

Nested Queries

- SELECT city
  FROM branch b1
  GROUP BY city
  HAVING AVG(b1.assets) > (SELECT AVG(b2.assets) FROM branch b2)

Nested Queries (Variable Scope)

- A reference to attribute can only be used within the SELECT and WHERE clauses where it is defined or within recursively nested queries.
Nested Queries (Variable Scope)

```
SELECT city, AVG(b1.assets), AVG(b2.assets)
FROM branch b1
GROUP BY city
HAVING AVG(b1.assets) > (SELECT AVG(b2.assets) FROM branch b2)
```

is incorrect

Nested Query

```
SELECT branch.name, branch.city
FROM branch
WHERE EXISTS (SELECT workfor.city
               FROM workfor
               WHERE workfor.employee = 1234
               AND workfor.city = branch.city)
```

Nested query sometimes increase the readability of queries

Nested queries increase the expressive power of SQL

Views

```
CREATE VIEW name [schema]
AS sql_query
```

CREATE VIEW branch_singapore
AS
```
SELECT *
FROM branch
WHERE city = 'Singapore'
```

SELECT * FROM branch_singapore

```
CREATE VIEW branch_singapore
AS
SELECT *
FROM branch
WHERE city = 'Singapore'
```

SELECT * FROM branch_singapore

```
CREATE VIEW branch_singapore
AS
SELECT *
FROM branch
WHERE city = 'Singapore'
```

SELECT * FROM branch_singapore

```
CREATE VIEW branch_singapore
AS
SELECT *
FROM branch
WHERE city = 'Singapore'
```

SELECT * FROM branch_singapore
**Views**

Update: add, delete, or modify

**View**

**Branch_Singapore**

**Table**

**Branch**

---

**Views**

- Can one update the views?

**View**

**Branch_Singapore**

**Table**

**Branch**

---

**Views**

- Can one update the views?

**CREATE VIEW branch_singapore**

**AS**

SELECT *

FROM branch

WHERE city = 'Singapore'

---

**Views**

- Can one update the views?

**CREATE VIEW branchname_singapore**

**AS**

SELECT name

FROM branch

WHERE city = 'Singapore'
CREATE VIEW assets_singapore AS
SELECT SUM(assets) as total
FROM branch
WHERE city = ‘Singapore’

Logical Data Independence is achieved by means of views
Views can be pre-compiled
However views may fool the optimizer