Set Theory

\[(1, 7, 2) = \{1, 2, 7\} = \{2, 1, 2, 7\}\]  
set (also: multiset, bags)

\[1 \in \{1, 7, 2\}\]  
element

\[3 \notin \{1, 7, 2\}\]  

\[(2, 7) \subseteq \{1, 7, 2\}\]  
subset

\[(7, 3) \not\subseteq \{1, 7, 2\}\]  

Set Theory

- $\overline{A} = \{x: x \notin A\}$ complement
- $A \cup B = \{x: x \in A \text{ or } x \in B\}$ union
- $A \cap B = \{x: x \in A \text{ and } x \in B\}$ intersection
- $A - B = \{x: x \in A \text{ and } x \notin B\}$ difference

$\mathcal{P}(A) = \{B: B \text{ is subset of } A\}$ powerset

$A \times B = \{(a,b): a \in A \text{ and } b \in B\}$ Cartesian Product

Inner Joins in SQL

Equijoin

SELECT S.*, SG.*
FROM student AS S, studentgroup AS SG
WHERE S.SID = SG.PresidentID;

Explicit joins

SELECT S.*, SG.*
FROM (student AS S JOIN studentgroup AS SG
ON S.SID = SG.PresidentID);

join condition

University
- List students and courses they are enrolled in.

Outer Joins in SQL

SELECT S.*, SG.*
FROM (student AS S LEFT OUTER JOIN studentgroup AS SG
ON S.SID = SG.PresidentID);

SELECT S.*, SG.*
FROM (student AS S RIGHT OUTER JOIN studentgroup AS SG
ON S.SID = SG.PresidentID);

SELECT S.*, SG.*
FROM (student AS S FULL OUTER JOIN studentgroup AS SG
ON S.SID = SG.PresidentID);

In Oracle
- can drop OUTER
- alternative notation using (+)
Join Examples

University
• List all students who are enrolled in courses.
• List all students and, if they are enrolled in a course, which courses they are enrolled in.
• List all students and what courses they are enrolled in; list students if they are not enrolled in any course and list courses even if there are no enrollments.
• List all students who are not enrolled in a course.
• List student groups without presidents.
• List students who are not president.

Set Operations

UNION
INTERSECT
EXCEPT (MINUS)

Oracle

Intersection and Difference not supported in some systems (Access, SQLServer). Workaround later.

Set Operations Examples

University
• List student members of DeFrag and HerCTI.
• List students that are members of both DeFrag and HerCTI.
• We only allow gaming students to join DeFrag; list students that violate this rule.
• We require that all gaming students are members of DeFrag; list students that violate this rule.
• List students that are not enrolled in any courses.
• List students that are not presidents of any group.
Duplicates with Set Operations

Duplicates are eliminated if we use set operations like

- **UNION** (union)
- **INTERSECT** (intersection)
- **EXCEPT** (set difference)

Adding the keyword **ALL** retains duplicated:

- **UNION ALL**
- **INTERSECT ALL**
- **EXCEPT ALL**

Only **UNION ALL** is supported in Oracle.

---

Set Operations Using Joins

Example:

All employees who are instructors or staff.

```sql
SELECT E.*
FROM employee AS E, instructor AS I, staff AS S
WHERE E.ID = I.ID or E.ID = S.ID
```

---

Set Operations Using Joins

Example:

All employees who are instructors or staff.

```sql
SELECT E.*
FROM employee AS E, instructor AS I, staff AS S
WHERE E.ID = I.ID or E.ID = S.ID
```

**Is \( E \cap (I \cup S) \) the same as**

\( E \times I \times S \) restricted to tuples

where \( E.ID = I.ID \) or \( E.ID = S.ID \) ?
The IN Operator

Conditions can contain IN for "element of"

SELECT LastName, FirstName
FROM student
WHERE started IN (2010, 2013, 2014);

SELECT LastName, FirstName
FROM student
WHERE started NOT IN (2010, 2013, 2014);

SELECT Department, CourseName
FROM Course
WHERE Department IN ('CSC', 'IT', 'IS');

The IN Operator

SELECT *
FROM enrolled
WHERE (Quarter, Year) in (('Fall', 2012), ('Winter', 2012));

This will not work in many systems (e.g. Access).
Can redo as OR of ANDS.

SELECT *
FROM enrolled
WHERE ((Quarter = 'Fall' and Year = 2012) or (Quarter = 'Winter' and Year = 2012));
Nesting Queries with IN

```
SELECT LastName, FirstName, SID
FROM student
WHERE SID IN
    (SELECT PresidentID
     FROM studentgroup)
```

In Examples

- List all students enrolled in a computer science course.
- List all students who are members of HerCTI.
- List undergraduate computer science students.
- Presidents who are members of their groups.

Excursion:
Using IN for set operations
Set Intersection

Example: Presidents that were enrolled in 2013.

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid IN (SELECT presidentID
FROM studentgroup)
AND sid IN (SELECT studentID
FROM enrolled
WHERE year = 2013);
```

- Students who enrolled in both 2012 and 2013
- Courses which ran in both 2012 and 2013.

Set Complement

Example: Students who did not enroll in 2013.

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid NOT IN (SELECT studentID
FROM enrolled
WHERE year = 2013);
```

- Courses not offered in 2013
- Students who are not presidents

Set Complement

Students who are not presidents

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid NOT IN (SELECT PresidentID
FROM studentgroup);
```

Why is there a problem? How to solve it?
Set Complement

Students who are not presidents

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid NOT IN (SELECT PresidentID
                   FROM studentgroup);
```

Why is there a problem? How to solve it?

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid NOT IN (SELECT PresidentID
                   FROM studentgroup
                   WHERE presidentID is not null);
```

Set Difference

Example: Students who are presidents but not members of any group.

```sql
SELECT LastName, FirstName, SID
FROM student
WHERE sid IN (SELECT presidentID
               FROM studentgroup)
   AND sid NOT IN (SELECT studentID
                   FROM memberof);
```

• CS students who are enrolled in a course, but no CS course.

Set UNION and OR

```sql
(Selectable)
```

```sql
(Selectable)
```

```
```
Set Operations Examples

• List students who have a mentor who is a president of a student group.
• List courses that exist both as graduate and undergraduate courses.
• List members of HerCTI that are not enrolled in courses.
• Courses not offered in 2013 (i.e. no record of anybody being enrolled).

END of EXCURSION

The ALL and ANY Operators

\[
= \text{ALL} \quad <> \quad \text{ALL} \\
< \text{ALL} \quad \leq \quad \text{ALL} \\
> \text{ALL} \quad \geq \quad \text{ALL} \\
= \text{ANY} \quad <> \quad \text{ANY} \\
< \text{ANY} \quad \leq \quad \text{ANY} \\
> \text{ANY} \quad \geq \quad \text{ANY}
\]

SELECT LastName, FirstName, SID
FROM student
WHERE started \geq \text{ALL} \quad \text{(SELECT started FROM student)};
Nesting Queries with ALL

• List the oldest studentgroup.
• List students belonging to the first student cohort.
• List courses that have a unique number.
• For all departments list the highest course number used by that department.

Naming Scope for nested queries

Correlated Nested Queries

• Presidents who are not members of their groups.
• List classes for which there is another class with the same name and a higher course number
• List students that started at the university before some group they belong to was founded

Existence

Tests that a set is nonempty

```
SELECT LastName, FirstName, sid
FROM student
WHERE EXISTS (SELECT *
    FROM enrolled
    WHERE sid = studentID);
```

```
SELECT LastName, FirstName, sid
FROM student
WHERE NOT EXISTS (SELECT *
    FROM enrolled
    WHERE sid = studentID);
```
**Unique Existence**

Tests that a set contains one element

```sql
SELECT LastName, FirstName, sid
FROM student
WHERE UNIQUE (SELECT *
    FROM enrolled
    WHERE sid = studentID);
```

Not supported by Oracle, Access or SQLServer

**Examples**

- List students who have taken IT, but no CSC courses.
- List students who have taken classes in CSC, IT and GAM.
- List student groups that have both graduate and undergraduate members.
- List courses in which nobody enrolled in 2013.
- List courses in which no student from Chicago ever enrolled.

**CONTAINS**

- List students who are members of all student groups.
- List students who have taken courses in all departments.
- List students who have enrolled in courses every year that courses were offered.
- List students who have enrolled in courses every year since they started (harder)