Interact with Data

http://goo.gl/hqEp1Z

CS5200 DBMS
Bruce Chhay

Interact with Data

L1: Relational Algebra

Relational Algebra

- Recap: theory for modeling data in a relational db.
  - Design module covered normalization, reducing redundancies and inconsistencies.
  - Implementation module covered the data definition language (DDL) portion of SQL.
- Basis of query language to interact with the data.
  - Data manipulation language (DML) of SQL, specifically declarative queries via SELECT statements.
  - Basic operations for interacting with data.

Relational Algebra

- Recap: theory for modeling data in a relational db.
  - Design module covered normalization, reducing redundancies and inconsistencies.
  - Implementation module covered the data definition language (DDL) portion of SQL.
- Basis of query language to interact with the data.
  - Basic operations for interacting with data.
  - Data manipulation language (DML) of SQL, specifically declarative queries via SELECT statements.

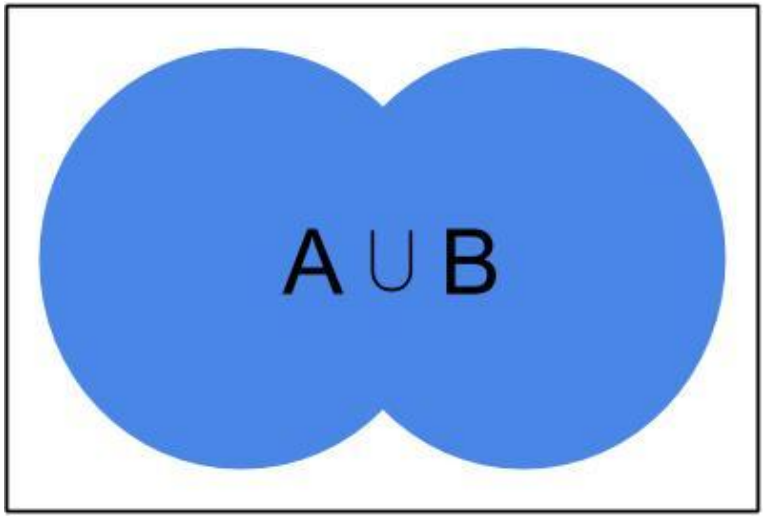
Basic Operations

- Set operators (combining tables).
- Selection (keeping specific rows).
- Projection (keeping specific columns).

Set operators are binary, requiring 2 sets. Selection and projection are unary, requiring 1 set.

Set Operations: Union

- Union
  - A, B need to be compatible, IE same attributes.
  - Union of sets A and B: “A ∪ B” (blue).



Set Operations: Union

- Union
  - A, B need to be compatible, IE same attributes.
  - Union of sets A and B: “A ∪ B”.

Relation A: CatPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets

Relation B: DogPosts

PostId	Title
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

Set Operations: Union

- Union
  - A, B need to be compatible, IE same attributes.
  - Union of sets A and B: “A ∪ B”.

Relation A: CatPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets

Relation B: DogPosts

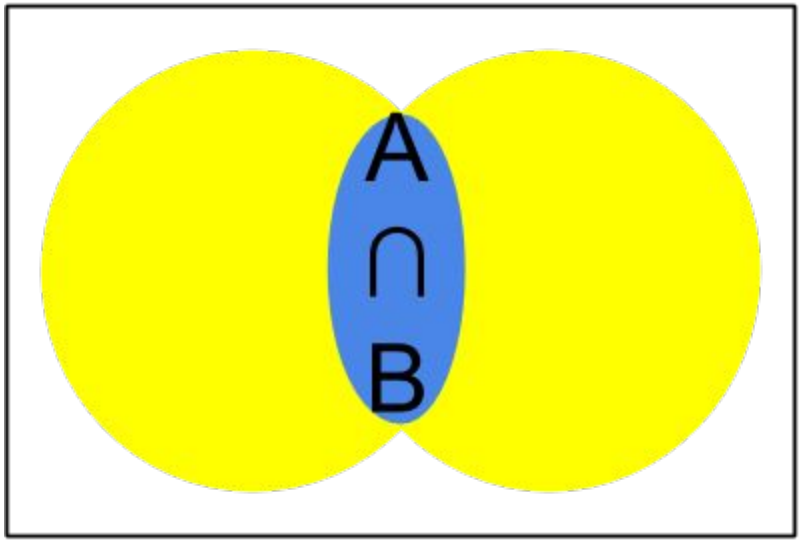
PostId	Title
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

A ∪ B

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

Set Operations: Intersection

- Intersection
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Intersection of Sets A and B: "A ∩ B" (blue).



Set Operations: Intersection

- Intersection
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Intersection of Sets A and B: "A ∩ B".

Relation A: BlogPosts Relation B: BlogComments

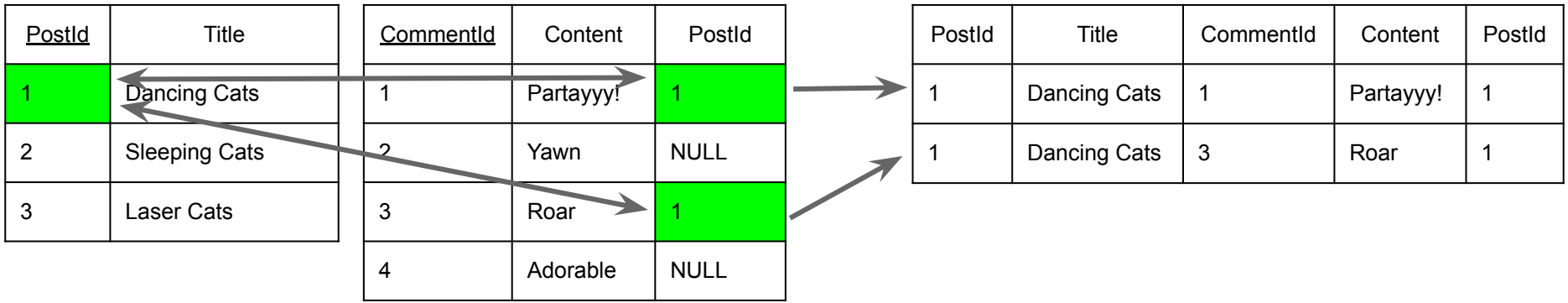
PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

Set Operations: Intersection

- Intersection
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Intersection of Sets A and B: "A ∩ B".

Relation A: BlogPosts Relation B: BlogComments A ∩ B



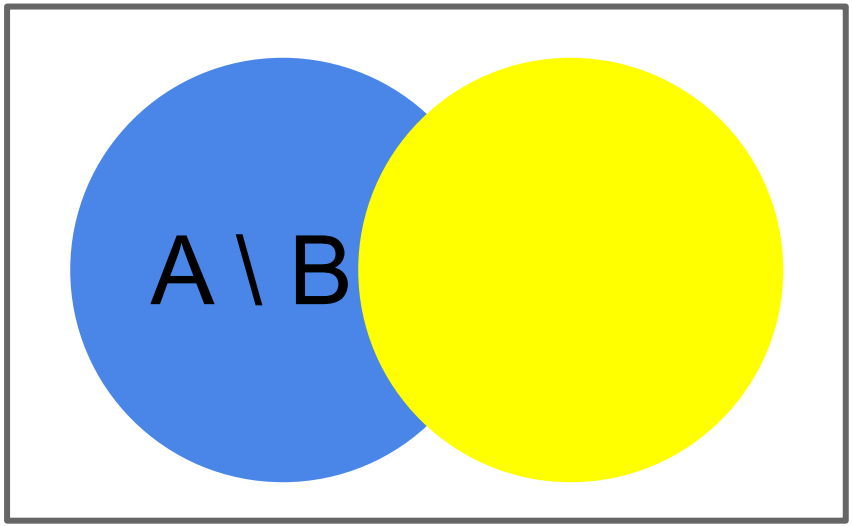
PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

Set Operations: Difference

- Difference
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Set difference of A and B: "A \ B" (blue).



Sometimes A-B is used for set difference.

Set Operations: Difference

- Difference
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Set difference of A and B: "A \ B".

Relation A: BlogPosts Relation B: BlogComments

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

Set Operations: Difference

- Difference
  - A, B need to be compatible, IE PK-FK, Heath's Theorem.
  - Set difference of A and B: "A \ B".

Relation A: BlogPosts Relation B: BlogComments A \ B

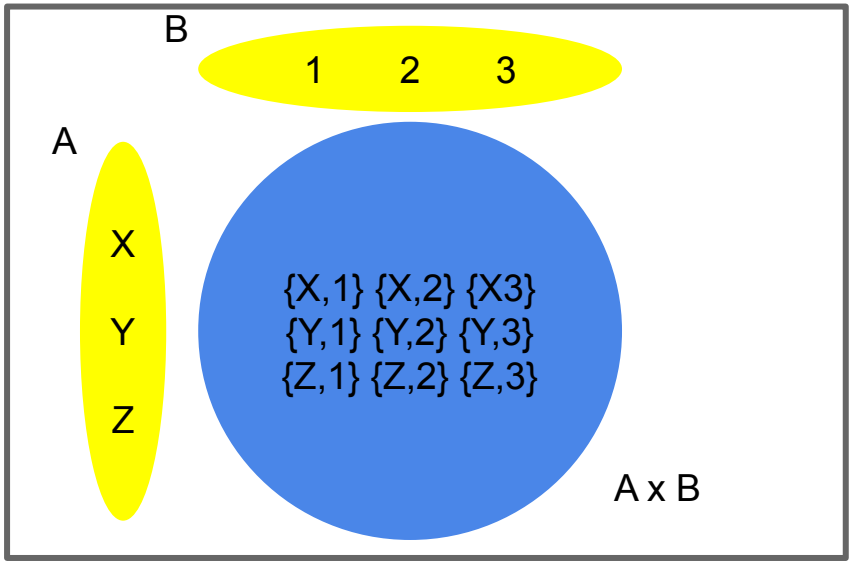
PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

PostId	Title	CommentId	Content	PostId
2	Sleeping Cats	NULL	NULL	NULL
3	Laser Cats	NULL	NULL	NULL

Set Operations: Cartesian Product

- Cartesian Product
  - A, B are not compatible.
  - Each record in A is combined with each record in B.
  - Size (cardinality) of |A x B| = |A| * |B| (blue).



Set Operations: Cartesian Product

- Cartesian Product
  - A, B are not compatible.
  - Each record in A is combined with each record in B.
  - Size (cardinality) of |A x B| = |A| * |B|.

Relation A

AId	A1	A2
A1	M	N
A2	Y	Z

Relation B

BId	B1	B2	B3
B1	200	400	600
B2	500	1000	1500
B3	700	1400	2100

Set Operations: Cartesian Product

- Cartesian Product
  - A, B are not compatible.
  - Each record in A is combined with each record in B.
  - Size (cardinality) of |A x B| = |A| * |B|.

Relation A

AId	A1	A2
A1	M	N
A2	Y	Z

Relation B

BId	B1	B2	B3
B1	200	400	600
B2	500	1000	1500
B3	700	1400	2100

A X B

AId	A1	A2	BId	B1	B2	B3
A1	M	N	B1	200	400	600
A1	M	N	B2	500	1000	1500
A1	M	N	B3	700	1400	2100
A2	Y	Z	B1	200	400	600
A2	Y	Z	B2	500	1000	1500
A2	Y	Z	B3	700	1400	2100

Set Operations: Cartesian Product

- Cartesian Product
  - A, B are not compatible.
  - Each record in A is combined with each record in B.
  - Size (cardinality) of |A x B| = |A| * |B|.

Relation A

AId	A1	A2
A1	M	N
A2	Y	Z

Relation B

BId	B1	B2	B3
B1	200	400	600
B2	500	1000	1500
B3	700	1400	2100

A X B

AId	A1	A2	BId	B1	B2	B3
A1	M	N	B1	200	400	600
A1	M	N	B2	500	1000	1500
A1	M	N	B3	700	1400	2100
A2	Y	Z	B1	200	400	600
A2	Y	Z	B2	500	1000	1500
A2	Y	Z	B3	700	1400	2100

Set Operations: Cartesian Product

- Cartesian Product
  - A, B are not compatible.
  - Each record in A is combined with each record in B.
  - Size (cardinality) of |A x B| = |A| * |B|.

Relation A

AId	A1	A2
A1	M	N
A2	Y	Z

Relation B

BId	B1	B2	B3
B1	200	400	600
B2	500	1000	1500
B3	700	1400	2100

A X B

AId	A1	A2	BId	B1	B2	B3
A1	M	N	B1	200	400	600
A1	M	N	B2	500	1000	1500
A1	M	N	B3	700	1400	2100
A2	Y	Z	B1	200	400	600
A2	Y	Z	B2	500	1000	1500
A2	Y	Z	B3	700	1400	2100

Selection

- Restriction of tuples (to keep specific rows).
- Selection: $\sigma_{a\theta b}(R)$ where a,b are attributes in relation R (or a constant) and θ is a conditional operator (<, <=, =, <>, >=, >).
- Sequences of aθb can be chained together through logical operators (and, or, negation).

Selection

- Examples:
  - $\sigma_{\text{FirstName==Jae}}(\text{BlogUsers})$
  - $\sigma_{\text{DoB>1990-02-05}}(\text{BlogUsers})$
  - $\sigma_{\text{FirstName==Jae AND DoB>1990-02-05}}(\text{BlogUsers})$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Selection

- Examples:
  - $\sigma_{\text{FirstName==Jae}}(\text{BlogUsers})$
  - $\sigma_{\text{DoB>1990-02-05}}(\text{BlogUsers})$
  - $\sigma_{\text{FirstName==Jae AND DoB>1990-02-05}}(\text{BlogUsers})$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Selection

- Examples:
  - $\sigma_{\text{FirstName==Jae}}(\text{BlogUsers})$
  - $\sigma_{\text{DoB>1990-02-05}}(\text{BlogUsers})$
  - $\sigma_{\text{FirstName==Jae AND DoB>1990-02-05}}(\text{BlogUsers})$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Selection

- Examples:
  - $\sigma_{\text{FirstName==Jae}}(\text{BlogUsers})$
  - $\sigma_{\text{DoB>1990-02-05}}(\text{BlogUsers})$
  - $\sigma_{\text{FirstName==Jae AND DoB>1990-02-05}}(\text{BlogUsers})$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Projection

- Restriction of attributes (to keep specific columns).
- Projection: $\pi_{a1,..,an}(R)$ where a1,…,an represent a set of attributes in relation R.

Projection

- Examples:
  - $\pi_{FirstName}(BlogUsers)$
  - $\pi_{FirstName,LastName}(BlogUsers)$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Projection

- Examples:
  - $\pi_{FirstName}(BlogUsers)$
  - $\pi_{FirstName,LastName}(BlogUsers)$

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Rename

- Rename of attributes.
- Rename: $\rho_{a/b}(R)$ where b is an attribute in relation R that is renamed to a.

Rename

- Examples:
  - $\rho_{Login/UserName}(BlogUser)$
  - $\rho_{Login/UserName,DateOfBirth/DoB}(BlogUser)$

Relation: BlogUsers

Login	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Rename

- Examples:
  - $\rho_{Login/UserName}(BlogUser)$
  - $\rho_{Login/UserName,DateOfBirth/DoB}(BlogUser)$

Relation: BlogUsers

Login	FirstName	LastName	DateOfBirth
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

Interact with Data

L2: Structured Query Language (SQL)

Structured Query Language (SQL)

- SELECT statements are declarative, and is a significant part of the data manipulation language of SQL.
- SELECT statements are queries to retrieve data.
- Basic clauses:
  - SELECT clause allows projection, $\pi_{a1,..,an,}$ and rename, $\rho_{a/b}$ (which columns should be in the result set).
  - FROM clause allows set operations (which table or combination of tables should be in the result set).
  - WHERE clause allows selection, $\sigma_{a\theta b}$ (which rows should be in the result set).

http://dev.mysql.com/doc/refman/5.7/en/select.html

SELECT Clause

- Syntax: SELECT select_expr
- select_expr can:
  - Be a projection: list of column names (or constants).
  - Be a rename: select_expr AS alias_name.
  - Include operators and functions:
    - Examples: logical operators (AND, OR), arithmetic operators (+, -), control flow functions (CASE, IF), data type specific (string functions, numeric functions, date time functions).

http://dev.mysql.com/doc/refman/5.7/en/functions.html

SELECT Clause

```
SELECT FirstName
SELECT FirstName AS First, LastName AS Last, 1 AS Const
SELECT MONTH(DoB) AS BirthMonth
SELECT CONCAT(FirstName, ‘ ’, LastName) AS FullName
SELECT IF(MONTH(DoB) > 6 AND MONTH(DoB) < 9,
                ‘Summer’, ‘NotSummer’) AS BirthSeason
SELECT tbl_name1.a, tbl_name2.b
SELECT *
```

SELECT Clause (Projection)

- Examples:
  - $\pi_{\text{FirstName}}$(BlogUsers) → SELECT FirstName
  - $\pi_{\text{FirstName,LastName}}$(BlogUsers) → SELECT FirstName, LastName

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

SELECT Clause (Projection)

- Examples:
  - $\pi_{FirstName}(BlogUsers) \rightarrow$ SELECT FirstName
  - $\pi_{FirstName,LastName}(BlogUsers) \rightarrow$ SELECT FirstName, LastName

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

SELECT Clause (Rename)

- Examples:
  - $\rho_{Login/UserName}(BlogUser) \rightarrow$ SELECT UserName AS Login
  - $\rho_{Login/UserName,DateOfBirth/DoB}(BlogUser) \rightarrow$ SELECT UserName AS Login, DoB AS DateOfBirth

Relation: BlogUsers

Login	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

SELECT Clause (Rename)

- Examples:
  - $\rho_{Login/UserName}(BlogUser) \rightarrow$ SELECT UserName AS Login
  - $\rho_{Login/UserName,DateOfBirth/DoB}(BlogUser) \rightarrow$

    SELECT UserName AS Login, DoB AS DateOfBirth

Relation: BlogUsers

Login	FirstName	LastName	DateOfBirth
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

WHERE Clause

- Syntax: WHERE where_condition
- where_condition can:
  - Include operators and function (similar to SELECT clause) that is a selection.
  - Evaluate to true/false.
    - NULL value evaluates to NULL and is filtered out (equivalent false). However, there is an operator for explicit checks:
      column IS [NOT] NULL.

WHERE Clause

WHERE FirstName = ‘Jae’

WHERE CONCAT(FirstName, ‘ ’, LastName) = ‘Jae Yoon’

WHERE FirstName = ‘Jae’ OR FirstName IS NULL

WHERE MONTH(DoB) > 6 AND MONTH(DoB) < 9

WHERE Clause (Selection)

- Examples:
  - $\sigma_{FirstName==Jae}(BlogUsers)$ → WHERE Firstname = ‘Jae’
  - $\sigma_{DoB>1990\text{-}02\text{-}05}(BlogUsers)$ → WHERE DoB > ‘1990-02-05’
  - $\sigma_{FirstName==Jae\ AND\ DoB>1990\text{-}02\text{-}05}(BlogUsers)$ →

    WHERE FirstName = ‘Jae’ AND DoB > ‘1990-02-05’

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

WHERE Clause (Selection)

- Examples:
  - $\sigma_{FirstName==Jae}(BlogUsers)$ → WHERE Firstname = ‘Jae’
  - $\sigma_{DoB>1990\text{-}02\text{-}05}(BlogUsers)$ → WHERE DoB > ‘1990-02-05’
  - $\sigma_{FirstName==Jae\ AND\ DoB>1990\text{-}02\text{-}05}(BlogUsers)$ →

    WHERE FirstName = ‘Jae’ AND DoB > ‘1990-02-05’

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

WHERE Clause (Selection)

- Examples:
  - $\sigma_{FirstName==Jae}(BlogUsers)$ → WHERE Firstname = ‘Jae’
  - $\sigma_{DoB>1990\text{-}02\text{-}05}(BlogUsers)$ → WHERE DoB > ‘1990-02-05’
  - $\sigma_{FirstName==Jae\ AND\ DoB>1990\text{-}02\text{-}05}(BlogUsers)$ →

    WHERE FirstName = ‘Jae’ AND DoB > ‘1990-02-05’

Relation: BlogUsers

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	O	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FROM Clause

- Syntax: FROM table_references
- table_references can be:
  - A table name, with an alias.
  - A subquery (nested SELECT statement).
  - A JOIN expression between two table_references.
    - Examples: INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN.
  - Multiple JOIN expressions are evaluated from left-to-right.

http://dev.mysql.com/doc/refman/5.7/en/join.html

UNION

- Union
  - Syntax: [SELECT statement] UNION [DISTINCT|ALL] [SELECT statement]
  - UNION DISTINCT removes duplicates (default behavior)
    UNION ALL returns all matching rows
  - Note: all set operations are FROM clause JOIN expressions except UNION.

```
SELECT PostId, Title FROM CatPosts
UNION DISTINCT
SELECT PostId, Title FROM DogPosts;
```

Relation A: CatPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets

Relation B: DogPosts

PostId	Title
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

A ∪ B

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

UNION

- Union
  - Syntax: [SELECT statement] UNION [DISTINCT|ALL] [SELECT statement]
  - UNION DISTINCT removes duplicates (default behavior)
    UNION ALL returns all matching rows
  - Note: all set operations are FROM clause JOIN expressions except UNION.

```
SELECT PostId, Title FROM CatPosts
UNION ALL
SELECT PostId, Title FROM DogPosts;
```

Relation A: CatPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets

Relation B: DogPosts

PostId	Title
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

A ∪ B

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Fun Pets
3	Fun Pets
4	Singing Dogs
5	Leaping Dogs

FROM Clause (Intersection)

- Intersection
  - Syntax: FROM tbl_A INNER JOIN tbl_B ON tbl_A.pk = tbl_B.fk

```
FROM BlogPosts INNER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

FROM Clause (Intersection)

SELECT Statement (fully-qualified column names):

```
SELECT BlogPosts.PostId, BlogPosts.Title,
    BlogComments.CommentId, BlogComments.Content, BlogComments.PostId
FROM BlogPosts INNER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId;
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

FROM Clause (Intersection)

SELECT Statement (implied column names, through uniqueness):

```
SELECT BlogPosts.PostId,Title,
    CommentId, BlogComments.Content, BlogComments.PostId
FROM BlogPosts INNER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId;
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

FROM Clause (Intersection)

SELECT Statement (all columns):

```
SELECT *
FROM BlogPosts INNER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId;
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

FROM Clause (Difference)

- LEFT OUTER JOIN: difference of left_tbl \ right_tbl plus intersection.
  - Syntax: FROM tbl_A LEFT OUTER JOIN tbl_B ON tbl_A.pk = tbl_B.fk
- RIGHT OUTER JOIN: difference of right_tbl \ left_tbl plus intersection. Can be re-written as LEFT OUTER JOIN.

```
FROM BlogPosts LEFT OUTER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A \ B + A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1
2	Sleeping Cats	NULL	NULL	NULL
3	Laser Cats	NULL	NULL	NULL

FROM Clause (Difference)

- Difference: LEFT OUTER JOIN without intersection:
  - FROM tbl_A LEFT OUTER JOIN tbl_B ON tbl_A.pk = tbl_B.fk
    WHERE tbl_B.pk IS NULL

```
FROM BlogPosts LEFT OUTER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId
WHERE BlogComments.CommentId IS NULL
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A \ B

PostId	Title	CommentId	Content	PostId
2	Sleeping Cats	NULL	NULL	NULL
3	Laser Cats	NULL	NULL	NULL

FROM Clause (Difference)

- Anti-pattern: what if selection “WHERE tbl_B.fk IS NOT NULL” is applied?
  → Intersection (INNER JOIN).

```
FROM BlogPosts LEFT OUTER JOIN BlogComments
    ON BlogPosts.PostId = BlogComments.PostId
WHERE BlogComments.CommentId IS NOT NULL
```

Relation A: BlogPosts

PostId	Title
1	Dancing Cats
2	Sleeping Cats
3	Laser Cats

Relation B: BlogComments

CommentId	Content	PostId
1	Partayyy!	1
2	Yawn	NULL
3	Roar	1
4	Adorable	NULL

A ∩ B

PostId	Title	CommentId	Content	PostId
1	Dancing Cats	1	Partayyy!	1
1	Dancing Cats	3	Roar	1

FROM Clause (Cartesian Product)

- Cartesian Product
  - Syntax: FROM tbl_A CROSS JOIN tbl_B

FROM A CROSS JOIN B

Relation A

AId	A1	A2
A1	M	N
A2	Y	Z

Relation B

BId	B1	B2	B3
B1	200	400	600
B2	500	1000	1500
B3	700	1400	2100

A X B

AId	A1	A2	BId	B1	B2	B3
A1	M	N	B1	200	400	600
A1	M	N	B2	500	1000	1500
A1	M	N	B3	700	1400	2100
A2	Y	Z	B1	200	400	600
A2	Y	Z	B2	500	1000	1500
A2	Y	Z	B3	700	1400	2100

Query Evaluation

1. Determine the table reference through the FROM clause. Perform join operations if specified. This yields an intermediate result set.
2. Filter out rows according to the WHERE clause conditions. This yields an intermediate result set.
3. Filter (and transform) columns according to the SELECT clause. Return this resultset.

Interact with Data

L3: Exercises

Exercise

- BlogApplication
  - Create tables: http://goo.gl/86a11H
  - Insert data: http://goo.gl/m4Y7rh
  - Run previous the INNER JOIN and LEFT OUTER JOIN examples (data will be a little different). Add a SELECT clause as necessary.
- Example of SELECT statements: http://goo.gl/MbyNR2

Exercise

- Baby names
  - Create tables and insert data
    - http://www.ssa.gov/oact/babynames/limits.html, http://www.ssa.gov/oact/babynames/names.zip (national)
  - How many different baby names started with ‘Jae’ in 2015?
  - How many different baby boy names started with ‘Jae’ in 2015?
  - How many different baby girl names started with ‘Lia’ in 2018?
  - Which names have three or more e’s in 2018?
  - Interesting facts?
  - How do we analyze data across multiple years?
- Example of solution: http://goo.gl/mBRJeC