In The Beginning There Was SQL

Session 1 - 9:00am-10:00am - room(s)201/203

James F. Koopmann

jkoopmann@pinehorse.com

www.pinehorse.com

In The Beginning There Was SQL

Objectives

- **Gain familiarity with SQL**
- **Introduce 'new' SQL**
- **Systematically explain SQL**
- **Learn how to write SQL**
- **Interactive presentation**

SQL got you tongue tied

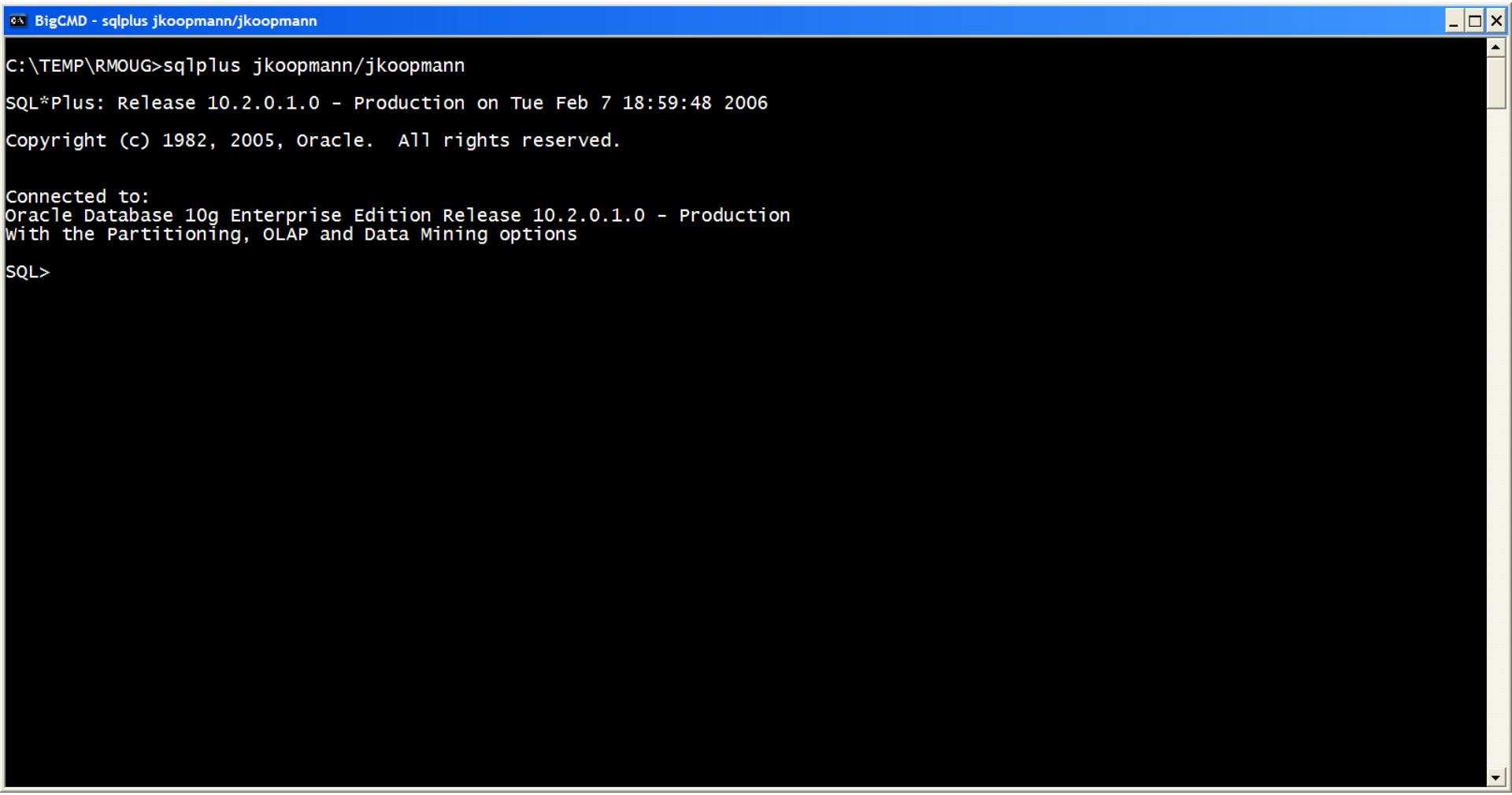
- SQL / SEQUEL
- E. F. Codd
- ANSI / ISO / IEC
- DML / DCL / DDL
- Performance
- SQL*Plus, JDeveloper, HTMLDB...

SQL got you tongue tied

- SQL / SEQUEL
- E. F. Codd
- ANSI / ISO / IEC
- DML / DCL / DDL
- Performance
- SQL*Plus, JDeveloper, HTMLDB...

Let's Play

Available for Queries



Oracle, SQL, and FIPS

- SQL Engines are not created equal
  - Oracle, DB2, SQL Server...
- FIPS
  - Federal Information Processing Standards
- Turning on Oracle's FIPS flagger
  - ***SQL > ALTER SESSION SET flagger=FULL;***

Oracle, SQL, and FIPS

Normal Operation

```
SQL  > CREATE TABLE t1 (
                          c1 CHAR(1),
                          n1 NUMBER);

Table created.

SQL  > INSERT INTO t1 VALUES ('1',1);

1 row created.

SQL  > COMMIT;

Commit complete.
```

```
SQL  > SELECT * FROM t1 WHERE c1 = '1';

C          N1
- ----------
1           1

SQL  > SELECT * FROM t1 WHERE n1 = '1';

C          N1
- ----------
1           1
```

Oracle, SQL, and FIPS

FIPS turned on

```
SQL  > ALTER SESSION SET flagger=full;
Session altered.
```

```
SQL  > SELECT * FROM t1 WHERE c1 = '1';
SELECT * FROM t1 WHERE c1 = '1'
                          *
ERROR at line 1:
ORA-00097: use of Oracle SQL feature not in SQL92 Full Level
ORA-06550: line 2, column 27:
PLS-01454: No operator may be used with values of data type CHAR

SQL  > SELECT * FROM t1 WHERE n1 = '1';
SELECT * FROM t1 WHERE n1 = '1'
                          *
ERROR at line 1:
ORA-00097: use of Oracle SQL feature not in SQL92 Full Level
ORA-06550: line 2, column 29:
PLS-01451: The data types of these <value expressions> must be comparable
```

```
SQL  > SELECT * FROM t1 WHERE c1 = (SELECT '1' from dual);
C         N1
- ----------
1          1

SQL  > SELECT * FROM t1 WHERE n1 = 1;
C         N1
- ----------
1          1
```

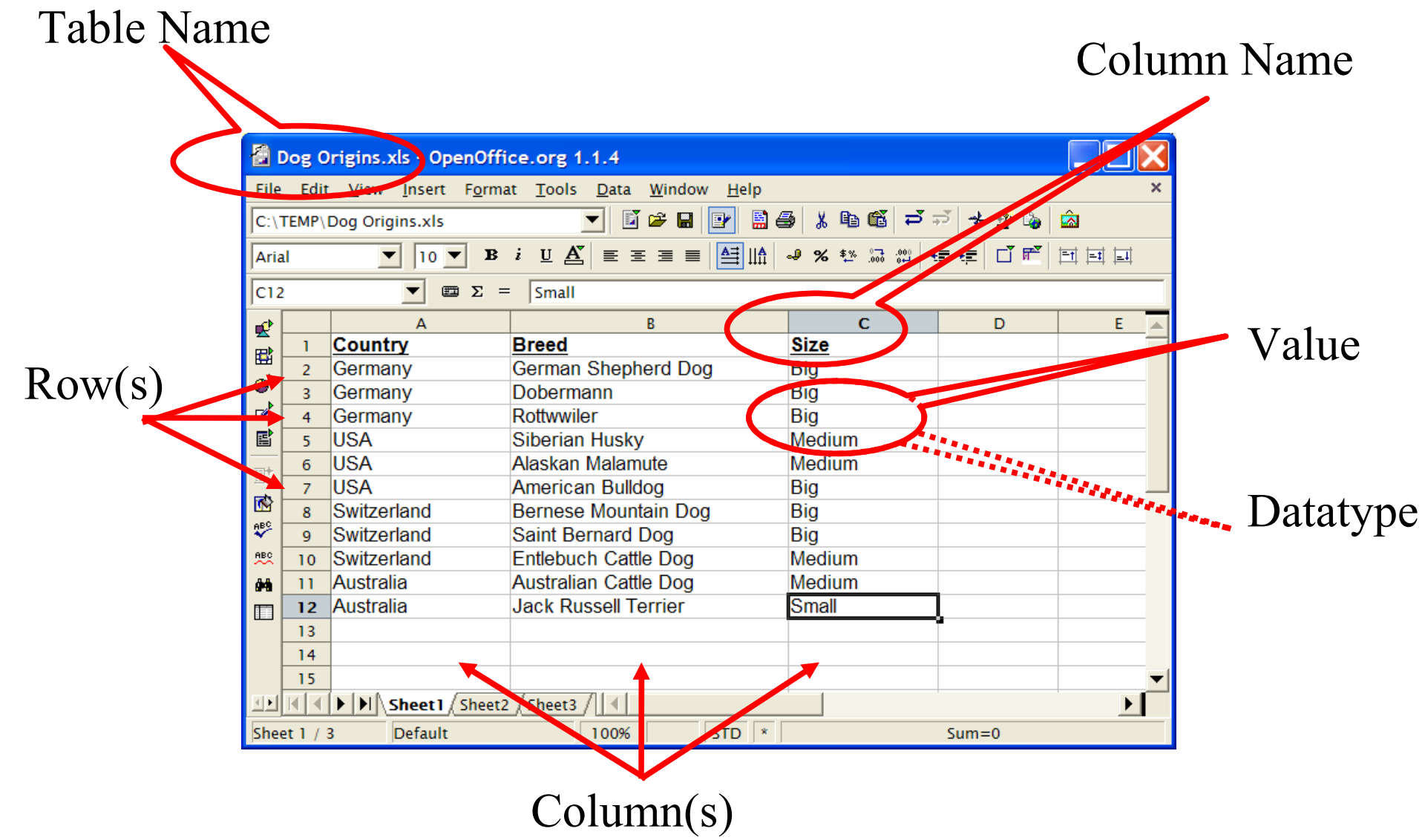
The Simple Select

Not so Simple

Datatypes	**User-Defined Types**	**Oracle-Supplied Types**	**Comparison Rules**
VARCHAR2	Object (abstract datatypes)	ANYTYPE	**Literal Handling**
NVARCHAR2	REF Datatypes	ANYDATA	**Formating**
NUMBER	VARRAYS	ANYDATASET	**NULLS**
LONG	Nested Tables	XML Types	**Object References**
DATE		Spatial Types	**Pseudocolumns**
BINARY_FLOAT		Media Types	**Operators**
BINARY_DOUBLE			**Functions**
TIMESTAMP			**Predicates**
TIMEZONE(s)			**Expresions**
INTERVAL(s)			**Conditions**
RAW			**Sub-queries**
ROWID			**Result sets**
CHAR			**SAVEPOINT(s)**
CLOB			**Constraints**
NCLOB			. . .
BLOB			
BFILE			

The Simple Select

The Table



The Simple Select

The Table

DOG_ORIGIN table

Country	Breed	Breed_Size
Germany	German Shepherd Dog	Big
Germany	Dobermann	Big
Germany	Rottweiler	Big
USA	Siberian Husky	Medium
USA	Alaskan Malamute	Medium
USA	American Bulldog	Big
Switzerland	Bernese Mountain Dog	Big
Switzerland	Saint Bernard Dog	Big
Switzerland	Entlebuch Cattle Dog	Medium
Australia	Australian Cattle Dog	Medium
Australia	Jack Russell Terrier	Small

DDL

```
CREATE TABLE DOG_ORIGIN (
             country      VARCHAR2(30),
             breed        VARCHAR2(30),
             breed_size   VARCHAR2(30));
```

Ownership

User who created the table or had a table created on their behalf.

Object Name (table)

DOG_ORIGIN

Column Names

COUNTRY
BREED
BREED_SIZE

Datatype(s)

VARCHAR – variable length character string

Notation

***owner*.DOG_ORIGIN.COUNTRY**
***owner*.DOG_ORIGIN.BREED**
owner.DOG_ORIGIN.BRED_SIZE

The Simple Select

Required parts of a SQL statement

SQL required part	Description
SELECT	Key word that signifies you want to query a table.
<select_list>	The list of COLUMNS, separated by commas if more than one COLUMN, from a table you would like to have the VALUES selected from.
FROM	Key word that denotes where the COLUMNS in the <select_list> will come from.
<table_reference>	The TABLE from where the COLUMNS reside in.

```
SELECT <select_list> FROM <table_reference>;

SELECT <table_reference>.<select_list> FROM <table_reference>;

SELECT <owner>.<table_reference>.<select_list> FROM <owner>.<table_reference>;

SELECT <alias>.<select_list> FROM <table_reference> alias;
```

The Simple Select

The simple select statement

```
SQL  > SELECT country, breed FROM dog_origin;

COUNTRY                        BREED
------------------------------ ------------------------------
Germany                        German Shepherd Dog
Germany                        Dobermann
Germany                        Rottweiler
USA                            Siberian Husky
USA                            Alaskan Malamute
USA                            American Bulldog
Switzerland                    Bernese Mountain Dog
Switzerland                    Saint Bernard Dog
Switzerland                    Entlebuch Cattle Dog
Australia                      Australian Cattle Dog
Australia                      Jack Russell Terrier
11 rows selected.
```

Additional methods

```
SELECT dog_origin.country, dog_origin.breed FROM dog_origin;

SELECT jkoopmann.dog_origin.country, jkoopmann.dog_origin.breed
  FROM jkoopmann.dog_origin;

SELECT do.country, do.breed FROM dog_origin do;
```

The Simple Select

Column expansion

```
SQL  > SELECT * FROM dog_origin;

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ --------------------
Germany                        German Shepherd Dog            Big
Germany                        Dobermann                      Big
Germany                        Rottweiler                     Big
USA                            Siberian Husky                 Medium
USA                            Alaskan Malamute               Medium
USA                            American Bulldog               Big
Switzerland                    Bernese Mountain Dog           Big
Switzerland                    Saint Bernard Dog              Big
Switzerland                    Entlebuch Cattle Dog           Medium
Australia                      Australian Cattle Dog          Medium
Australia                      Jack Russell Terrier           Small

11 rows selected.
```

The WHERE clause

Finding Data - WHERE

Think "Row Selection"

The WHERE clause

Finding Data - WHERE

DOG_ORIGIN table

Country	Breed	Breed_Size
Germany	German Shepherd Dog	Big
Germany	Dobermann	Big
Germany	Rottweiler	Big
USA	Siberian Husky	Medium
USA	Alaskan Malamute	Medium
USA	American Bulldog	Big
Switzerland	Bernese Mountain Dog	Big
Switzerland	Saint Bernard Dog	Big
Switzerland	Entlebuch Cattle Dog	Medium
Australia	Australian Cattle Dog	Medium
Australia	Jack Russell Terrier	Small

```
SQL  > SELECT country, breed, breed_size
         FROM dog_origin
        WHERE country = 'Germany';

COUNTRY      BREED                BREED_SIZE
------------ -------------------- ----------
Germany      German Shepherd Dog  Big
Germany      Dobermann            Big
Germany      Rottweiler           Big
```

WHERE Clause

```
country = 'Germany'     - Condition
country                 - Expression
          'Germany'     - Literal
        =               - Comparison Condition (logical operator)
```

The WHERE clause

Finding Data - AND

DOG_ORIGIN table

Country	Breed	Breed_Size
Germany	German Shepherd Dog	Big
Germany	Dobermann	Big
Germany	Rottweiler	Big
USA	Siberian Husky	Medium
USA	Alaskan Malamute	Medium
USA	American Bulldog	Big
Switzerland	Bernese Mountain Dog	Big
Switzerland	Saint Bernard Dog	Big
Switzerland	Entlebuch Cattle Dog	Medium
Australia	Australian Cattle Dog	Medium
Australia	Jack Russell Terrier	Small

```
SQL  > SELECT country, breed, breed_size
         FROM dog_origin
        WHERE country = 'USA'
          AND breed_size = 'Medium';

COUNTRY      BREED                BREED_SIZE
------------ -------------------- ----------
USA          Siberian Husky       Medium
USA          Alaskan Malamute     Medium
```

WHERE Clause

```
      country = 'Germany'      - Condition
      country                  - Expression
                'Germany'      - Literal
              =                - Comparison Condition (logical operator)
  AND                          - Logical AND Condition
```

The WHERE clause

Finding Data - OR

DOG_ORIGIN table

Country	Breed	Breed_Size
Germany	German Shepherd Dog	Big
Germany	Dobermann	Big
Germany	Rottweiler	Big
USA	Siberian Husky	Medium
USA	Alaskan Malamute	Medium
USA	American Bulldog	Big
Switzerland	Bernese Mountain Dog	Big
Switzerland	Saint Bernard Dog	Big
Switzerland	Entlebuch Cattle Dog	Medium
Australia	Australian Cattle Dog	Medium
Australia	Jack Russell Terrier	Small

```
SQL  > SELECT country, breed, breed_size
         FROM dog_origin
        WHERE breed_size = 'Big'
           OR breed_size = 'Small';

COUNTRY      BREED                 BREED_SIZE
-----------  --------------------  ----------
Germany      German Shepherd Dog   Big
Germany      Dobermann             Big
Germany      Rottweiler            Big
USA          American Bulldog      Big
Switzerland  Bernese Mountain Dog  Big
Switzerland  Saint Bernard Dog     Big
Australia    Jack Russell Terrier  Small
```

WHERE Clause

```
      country = 'Germany'      - Condition
      country                  - Expression
                'Germany'      - Literal
              =                - Comparison Condition (logical operator)
  AND                          - Logical OR Condition
```

The WHERE clause

Finding Data – AND & OR (Condition Precedence)

```
SQL  > SELECT country, breed, breed_size
         FROM dog_origin
        WHERE country = 'USA' AND breed_size = 'Big' OR breed_size = 'Small';

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ ----------
USA                            American Bulldog               Big
Australia                      Jack Russell Terrier           Small
2 rows selected.
```

```
SQL  > SELECT country, breed, breed_size
         FROM dog_origin
        WHERE breed_size = 'Big' OR breed_size = 'Small' AND country = 'USA';

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ ----------
Germany                        German Shepherd Dog            Big
Germany                        Dobermann                      Big
Germany                        Rottweiler                     Big
USA                            American Bulldog               Big
Switzerland                    Bernese Mountain Dog           Big
Switzerland                    Saint Bernard Dog              Big
6 rows selected.
```

The WHERE clause

Finding Data – AND & OR (Condition Precedence)

```
SQL  > SELECT country, breed, breed_size FROM dog_origin
         WHERE country = 'USA' AND breed_size = 'Big'
            OR country = 'USA' AND breed_size = 'Small';

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ ----------
USA                            American Bulldog               Big
```

```
SQL  > SELECT country, breed, breed_size FROM dog_origin
         WHERE country = 'USA'
           AND ( breed_size = 'Big' OR breed_size = 'Small' );

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ ----------
USA                            American Bulldog               Big
```

Joining Tables

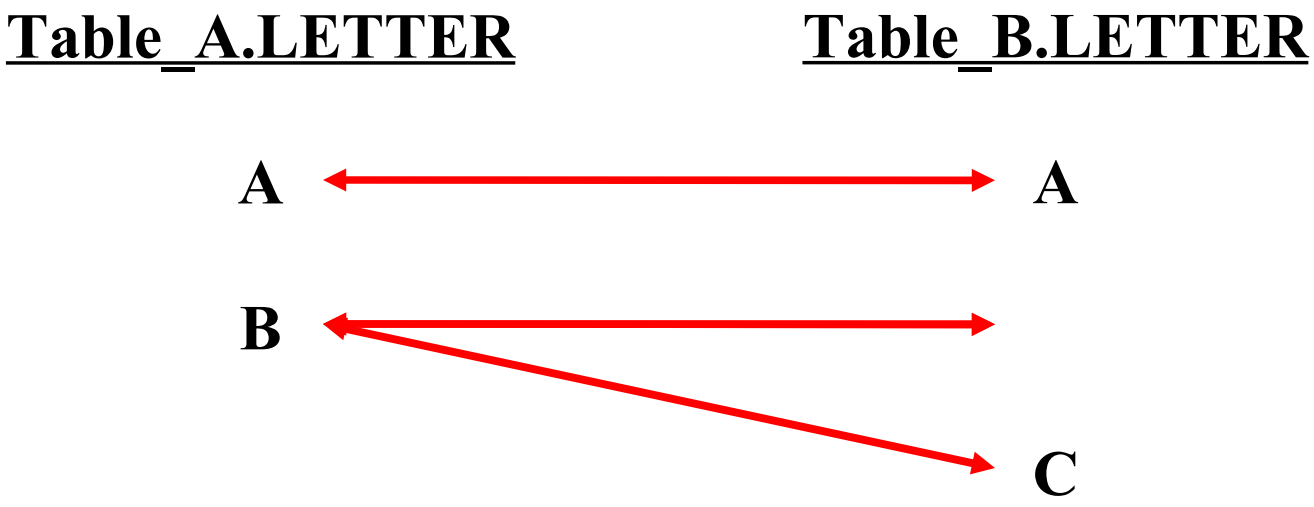
More than one table

- Joins are difficult
- Programming is easier
- Databases consist of more than one table
- Joins are nothing more than getering “like” information
- Involves two or more tables
- Joins allow us to limit / selectively choose similar information

Joining Tables

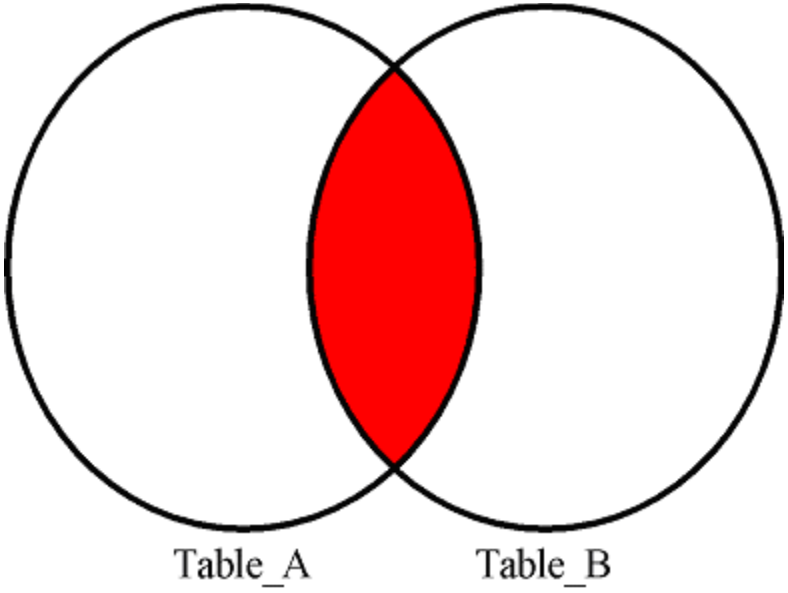
More than one table

Table_A.letter	Table_B.letter
A	A
B	
	C



Joining Tables

Equijoin or Inner Join – old method

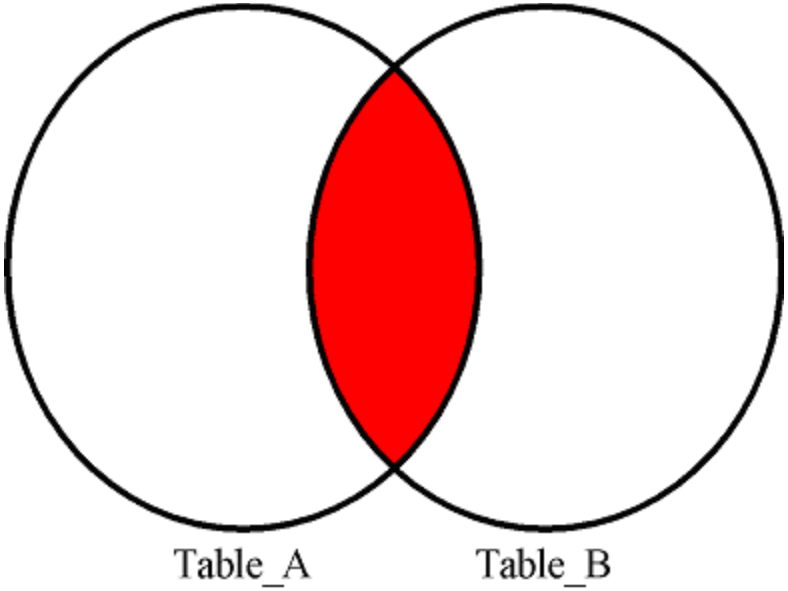


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL > SELECT Table_A.letter, Table_B.letter
        FROM Table_A, Table_B
       WHERE Table_A.letter = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
```

Joining Tables

Equijoin or Inner Join – ANSI syntax

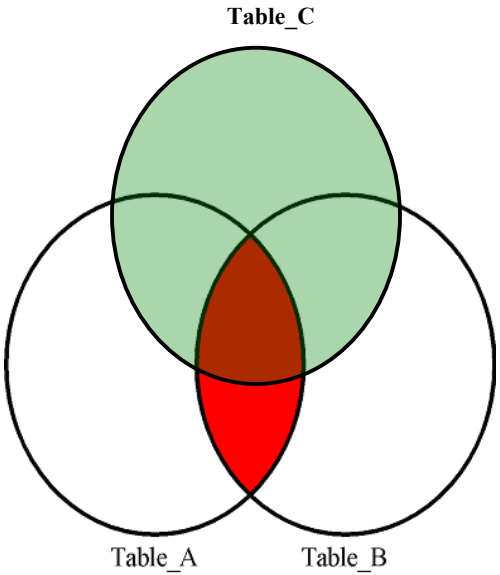


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL > SELECT Table_A.letter, Table_B.letter
        FROM Table_A INNER JOIN Table_B
          ON Table_A.letter = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
```

Joining Tables

Equijoin or Inner Join – Multi-Table Join

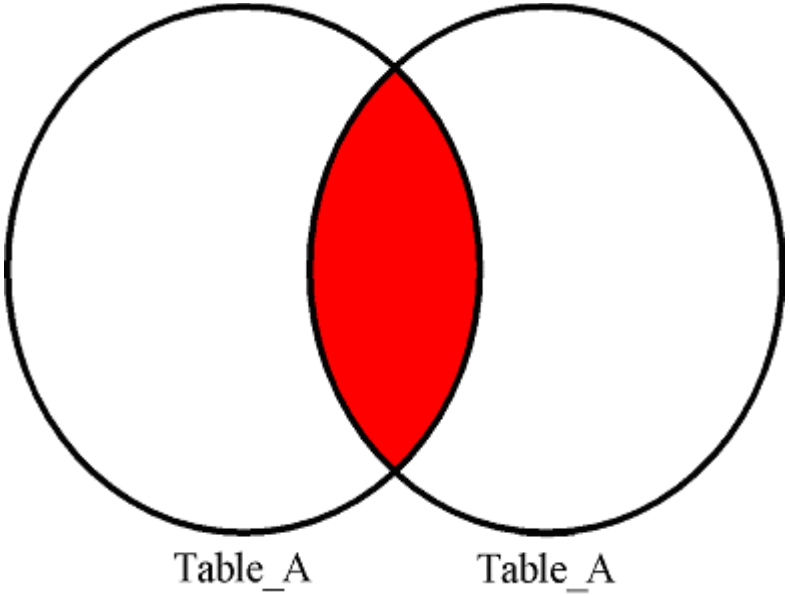


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL > SELECT Table_A.letter, Table_C.letter, Table_B.letter
        FROM Table_A
       INNER JOIN Table_C ON Table_A.letter = Table_C.letter
       INNER JOIN Table_B ON Table_A.letter = Table_B.letter

LETTER     LETTER     LETTER
---------- ---------- ----------
A          A          A
```

Joining Tables

Self Join – old method

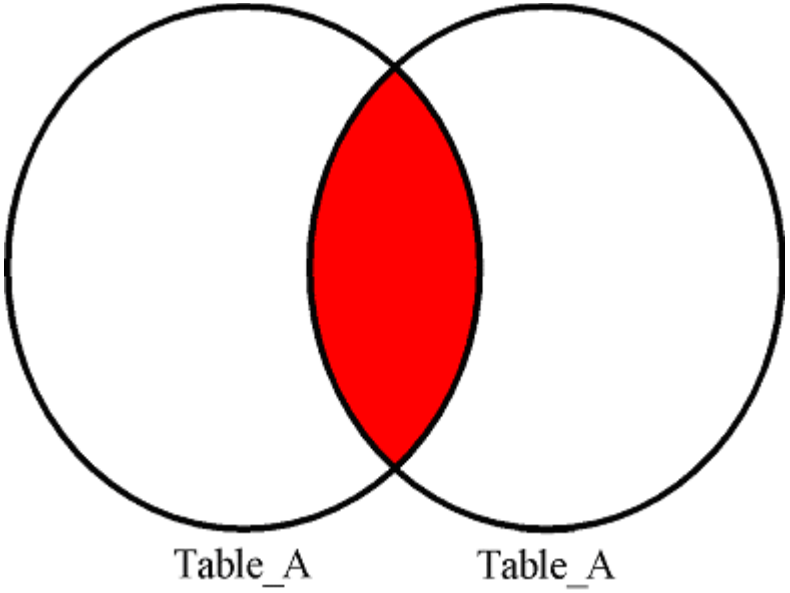


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT A1.letter, A2.letter
         FROM Table_A A1, Table_A A2
        WHERE A1.letter = A2.letter;

LETTER     LETTER
---------- ----------
A          A
B          B
```

Joining Tables

Self Join – ANSI syntax

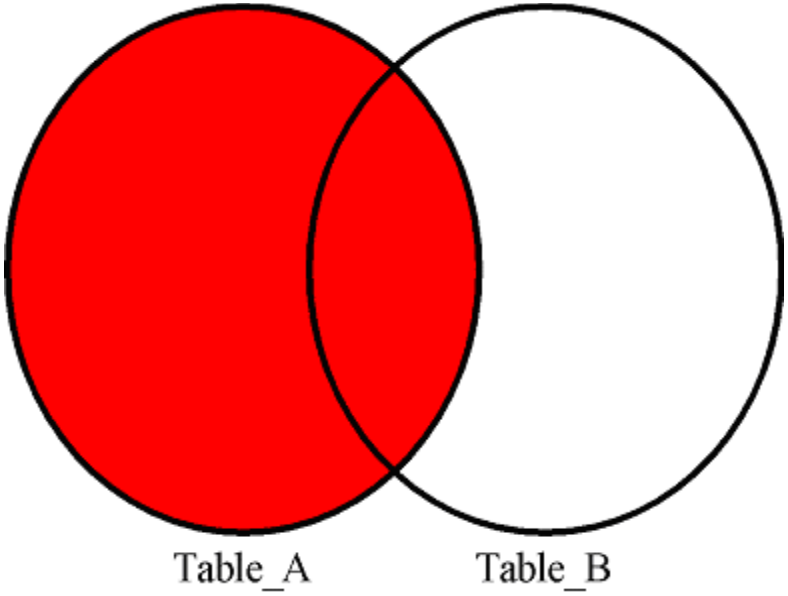


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT A1.letter, A2.letter
         FROM Table_A A1 INNER JOIN Table_A A2
           ON A1.letter = A2.letter;

LETTER     LETTER
---------- ----------
A          A
B          B
```

Joining Tables

Left Outer Join – old method

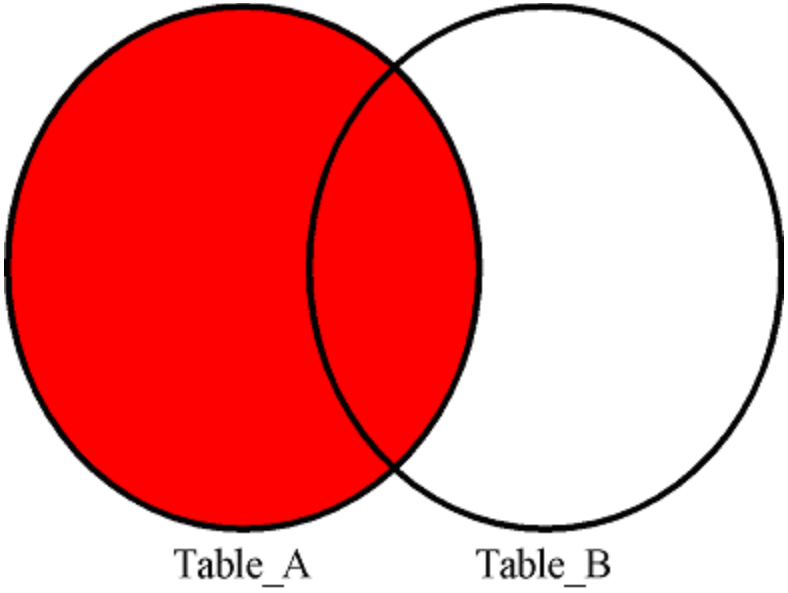


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A, Table_B
        WHERE Table_A.letter = Table_B.letter(+);

LETTER     LETTER
---------- ----------
A          A
B
```

Joining Tables

Left Outer Join – ANSI syntax

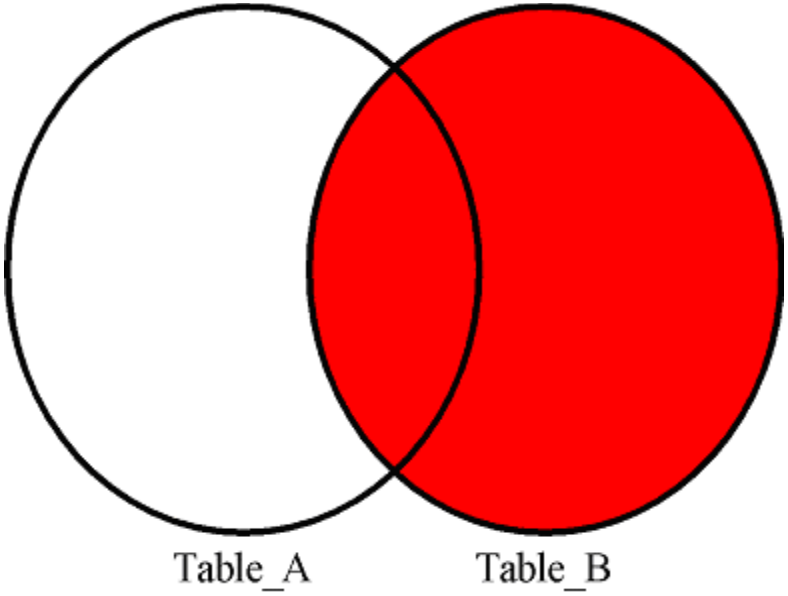


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A LEFT OUTER JOIN Table_B
           ON Table_A.letter = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
B
```

Joining Tables

Right Outer Join – old method

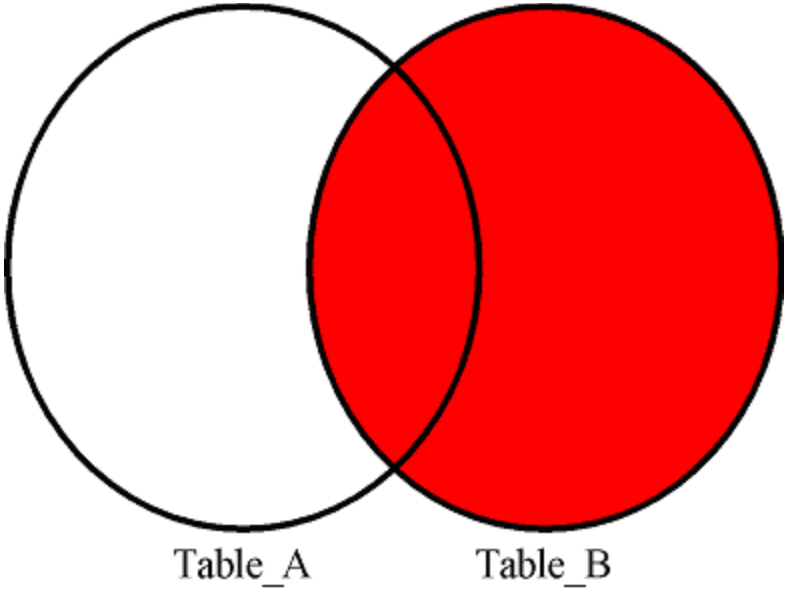


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A, Table_B
        WHERE Table_A.letter(+) = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
           C
```

Joining Tables

Right Outer Join – ANSI syntax

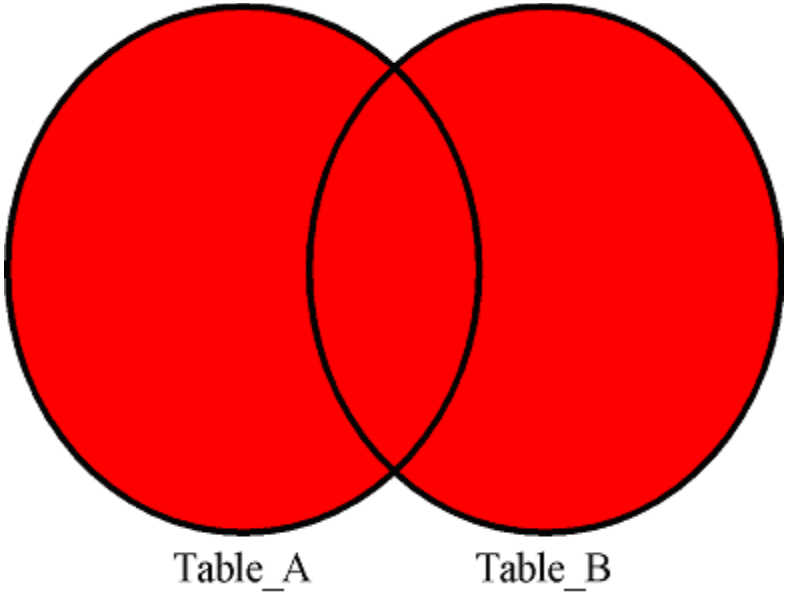


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A RIGHT OUTER JOIN Table_B
           ON Table_A.letter = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
           C
```

Joining Tables

Full Outer Join – old method

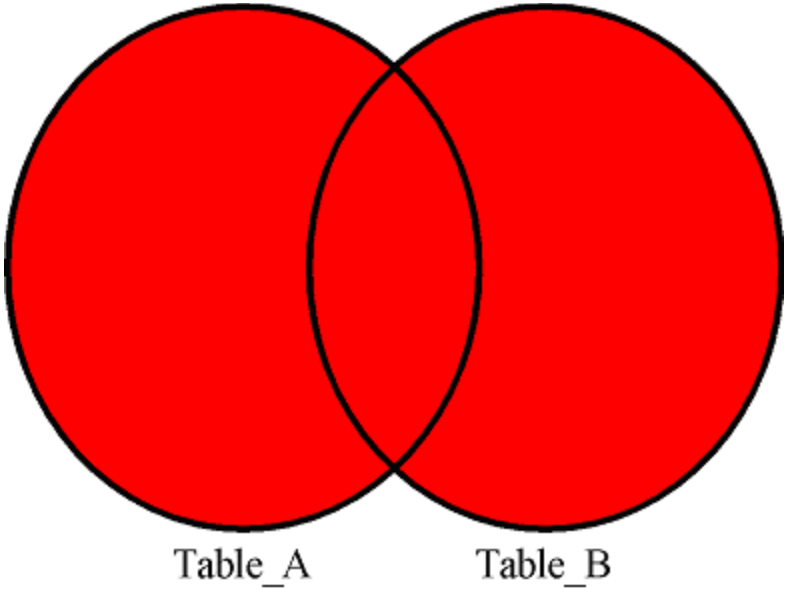


Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A, Table_B
        WHERE Table_A.letter = Table_B.letter(+)
        UNION
       SELECT Table_A.letter, Table_B.letter
         FROM Table_A, Table_B
        WHERE Table_A.letter(+) = Table_B.letter;
LETTER     LETTER
---------- ----------
A          A
B
           C
```

Joining Tables

Full Outer Join – ANSI syntax



Table_A.letter	Table_B.letter
A	A
B	
	C

```
SQL  > SELECT Table_A.letter, Table_B.letter
         FROM Table_A FULL OUTER JOIN Table_B
           ON Table_A.letter = Table_B.letter;

LETTER     LETTER
---------- ----------
A          A
B
           C
```

Joining Tables

Cartesian Product

Table_A.letter	Table_B.letter
A	A
B	
	C

Old Method

```
SQL > SELECT Table_A.letter, Table_B.letter
        FROM Table_A, Table_B;

LETTER     LETTER
---------- ----------
A          A
A          C
B          A
B          C
```

ANSI Syntax

```
SQL > SELECT Table_A.letter, Table_B.letter
        FROM Table_A CROSS JOIN Table_B;

LETTER     LETTER
---------- ----------
A          A
A          C
B          A
B          C
```

Counting with SQL

Country	Breed	Breed_Size
Germany	German Shepherd Dog	Big
Germany	Dobermann	Big
Germany	Rottweiler	Big
USA	Siberian Husky	Medium
USA	Alaskan Malamute	Medium
USA	American Bulldog	Big
Switzerland	Bernese Mountain Dog	Big
Switzerland	Saint Bernard Dog	Big
Switzerland	Entlebuch Cattle Dog	Medium
Australia	Australian Cattle Dog	Medium
Australia	Jack Russell Terrier	Small

Common counting dilemmas

- How many do I have?
- Do I have at least one?
- What's the greatest?
- What's the least?
- Give me the top 5 rows.

Counting with SQL

How many do I have

How many 'Big' dogs

```
SQL  > SELECT count(*)
         FROM dog_origin
        WHERE breed_size = 'Big';

  COUNT(*)
----------
         6
```

How many 'Big' dogs in the USA

```
SQL  > SELECT count(*)
         FROM dog_origin
        WHERE breed_size = 'Big'
          AND country = 'USA';

  COUNT(*)
----------
         1
```

Counting with SQL

How many do I have

How many 'Big' dogs in each country

```
SQL  > SELECT country, count(*)
         FROM dog_origin
        WHERE breed_size = 'Big'
        GROUP BY country;

COUNTRY                          COUNT(*)
------------------------------ ----------
Germany                                 3
USA                                     1
Switzerland                             2
```

How many 'Big' dogs in countries that have more than one Big dog

```
SQL  > SELECT country, count(*)
         FROM dog_origin
        WHERE breed_size = 'Big'
        GROUP BY country
       HAVING count(*) > 1;

COUNTRY                          COUNT(*)
------------------------------ ----------
Germany                                 3
Switzerland                             2
```

Counting with SQL

Who has the most – clunky syntax

What country has the most big dogs

```
SQL  > SELECT a.country, count(*)
         FROM dog_origin a
        WHERE a.breed_size = 'Big'
        GROUP BY a.country
       HAVING count(*) = (SELECT MAX(count(*))
                            FROM dog_origin b
                           WHERE b.breed_size = 'Big'
                           GROUP BY b.country);

COUNTRY                         COUNT(*)
------------------------------ ----------
Germany                                 3
```

Counting with SQL

RANK & DENSE_RANK

```
SQL> create table Table_Rank (The_Rank NUMBER);
SQL> insert into Table_Rank values (1);
SQL> insert into Table_Rank values (2);
SQL> insert into Table_Rank values (2);
SQL> insert into Table_Rank values (3);

SQL> SELECT RANK(2) WITHIN GROUP(ORDER BY The_Rank) Ranking FROM Table_Rank;

RANKING
-------
      2
```

```
SELECT The_Rank,
       RANK() OVER(ORDER BY The_Rank) Ranking
  FROM Table_Rank;

THE_RANK    RANKING
-------- ----------
       1          1
       2          2
       2          2
       3          4
```

```
SELECT The_Rank,
 DENSE_RANK() OVER(ORDER BY The_Rank) Ranking
  FROM Table_Rank;

THE_RANK    RANKING
-------- ----------
       1          1
       2          2
       2          2
       3          3
```

Counting with SQL

Who has the most – using RANK

What country has the most big dogs

```
SQL  > SELECT country, count_dogs
         FROM (SELECT country, COUNT(*) count_dogs,
                      RANK() OVER (ORDER BY COUNT(*) DESC) rank
                 FROM dog_origin WHERE breed_size = 'Big'
                GROUP BY country )
        WHERE rank = 1;
```

```
COUNTRY                        COUNT_DOGS
------------------------------ ----------
Germany                                 3
```

Counting with SQL

Row-counts as columns

I want to show the number of dogs as columns

```
SQL  > SELECT country,
               SUM(DECODE(breed_size,'Big'   ,1,0)) Big,
               SUM(DECODE(breed_size,'Medium',1,0)) Medium,
               SUM(DECODE(breed_size,'Small' ,1,0)) Small
         FROM dog_origin
        GROUP BY country;
```

COUNTRY	BIG	MEDIUM	SMALL
Germany	3	0	0
USA	1	2	0
Australia	0	1	1
Switzerland	2	1	0

Famous Pseudocolumns

Definition

A pseudocolumn

is a column

that looks like a column

but really isn't a column.

Famous Pseudocolumns

Some Famous Pseudocolumns

Hierarchical Pseudocolumns

- CONNECT_BY_ISCYCLE
- CONNECT_BY_ISLEAF
- LEVEL

Sequence Pseudocolumns

- CURVAL, NEXTVAL
- ORA_ROWSCN
- ROWID
- ROWNUM
- ROW_NUMBER

Famous Pseudocolumns

ROWID

```
SQL  > SELECT ROWID, dog_origin.*
         FROM dog_origin;
```

ROWID	COUNTRY	BREED	BREED_SIZE
AAASO2AAEAAAAE1AAL	Germany	German Shepherd Dog	Big
AAASO2AAEAAAAE1AAM	Germany	Dobermann	Big
AAASO2AAEAAAAE1AAN	Germany	Rottweiler	Big
AAASO2AAEAAAAE1AAO	USA	Siberian Husky	Medium
AAASO2AAEAAAAE1AAP	USA	Alaskan Malamute	Medium
AAASO2AAEAAAAE1AAQ	USA	American Bulldog	Big
AAASO2AAEAAAAE1AAR	Switzerland	Bernese Mountain Dog	Big
AAASO2AAEAAAAE1AAS	Switzerland	Saint Bernard Dog	Big
AAASO2AAEAAAAE1AAT	Switzerland	Entlebuch Cattle Dog	Medium
AAASO2AAEAAAAE1AAU	Australia	Australian Cattle Dog	Medium
AAASO2AAEAAAAE1AAV	Australia	Jack Russell Terrier	Small

Famous Pseudocolumns

ROWID

```
SQL  > DELETE FROM dog_origin a
        WHERE rowid <> ( SELECT MAX(rowid) FROM dog_origin b
                          WHERE a.country    = b.country
                            AND a.breed      = b.breed
                            AND a.breed_size = b.breed_size);

0 rows deleted.
```

Famous Pseudocolumns

ROWNUM

```
SQL  > SELECT rownum, country, breed, breed_size
         FROM dog_origin;
```

ROWNUM	COUNTRY	BREED	BREED_SIZE
1	Germany	German Shepherd Dog	Big
2	Germany	Dobermann	Big
3	Germany	Rottweiler	Big
4	USA	Siberian Husky	Medium
5	USA	Alaskan Malamute	Medium
6	USA	American Bulldog	Big
7	Switzerland	Bernese Mountain Dog	Big
8	Switzerland	Saint Bernard Dog	Big
9	Switzerland	Entlebuch Cattle Dog	Medium
10	Australia	Australian Cattle Dog	Medium
11	Australia	Jack Russell Terrier	Small

Famous Pseudocolumns

ROWNUM

```
SQL  > SELECT rownum, country, breed, breed_size
         FROM dog_origin
        ORDER BY breed;
```

ROWNUM	COUNTRY	BREED	BREED_SIZE
5	USA	Alaskan Malamute	Medium
6	USA	American Bulldog	Big
10	Australia	Australian Cattle Dog	Medium
7	Switzerland	Bernese Mountain Dog	Big
2	Germany	Dobermann	Big
9	Switzerland	Entlebuch Cattle Dog	Medium
1	Germany	German Shepherd Dog	Big
11	Australia	Jack Russell Terrier	Small
3	Germany	Rottweiler	Big
8	Switzerland	Saint Bernard Dog	Big
4	USA	Siberian Husky	Medium

Famous Pseudocolumns

ROWNUM

```
SQL  > SELECT rownum, country, breed, breed_size
         FROM (SELECT country, breed, breed_size
                 FROM dog_origin ORDER BY breed);
```

ROWNUM	COUNTRY	BREED	BREED_SIZE
1	USA	Alaskan Malamute	Medium
2	USA	American Bulldog	Big
3	Australia	Australian Cattle Dog	Medium
4	Switzerland	Bernese Mountain Dog	Big
5	Germany	Dobermann	Big
6	Switzerland	Entlebuch Cattle Dog	Medium
7	Germany	German Shepherd Dog	Big
8	Australia	Jack Russell Terrier	Small
9	Germany	Rottweiler	Big
10	Switzerland	Saint Bernard Dog	Big
11	USA	Siberian Husky	Medium

Famous Pseudocolumns

ROWNUM

```
SQL  > SELECT rownum, country, breed, breed_size
         FROM (SELECT country, breed, breed_size
                 FROM dog_origin ORDER BY breed)
        WHERE rownum < 4;
```

```
    ROWNUM COUNTRY                        BREED                          BREED_SIZE
---------- ------------------------------ ------------------------------ -----------
         1 USA                            Alaskan Malamute               Medium
         2 USA                            American Bulldog               Big
         3 Australia                      Australian Cattle Dog          Medium
```

Famous Pseudocolumns

ROWNUM - CAUTION!

```
SQL  > SELECT * FROM dog_origin WHERE rownum = 1;

COUNTRY                        BREED                          BREED_SIZE
------------------------------ ------------------------------ -----------
Germany                        German Shepherd Dog            Big
```

```
SQL  > SELECT * FROM dog_origin WHERE rownum = 2;

no rows selected
```

```
SQL  > SELECT * FROM dog_origin WHERE rownum > 1;

no rows selected
```

Famous Pseudocolumns

ROW_NUMBER function

```
SQL  > SELECT rownum, country, breed, breed_size,
              row_number()OVER (ORDER BY breed) as row_number
         FROM dog_origin;
```

ROWNUM	COUNTRY	BREED	BREED_SIZE	ROW_NUMBER
5	USA	Alaskan Malamute	Medium	1
6	USA	American Bulldog	Big	2
10	Australia	Australian Cattle Dog	Medium	3
7	Switzerland	Bernese Mountain Dog	Big	4
2	Germany	Dobermann	Big	5
9	Switzerland	Entlebuch Cattle Dog	Medium	6
1	Germany	German Shepherd Dog	Big	7
11	Australia	Jack Russell Terrier	Small	8
3	Germany	Rottweiler	Big	9
8	Switzerland	Saint Bernard Dog	Big	10
4	USA	Siberian Husky	Medium	11

Famous Pseudocolumns

ROWNUM, ROW_NUMBER, & Rank Oh MY!

```
SQL  > SELECT The_Rank,
              ROWNUM,
              RANK()        OVER(ORDER BY The_Rank) Rank,
              DENSE_RANK()  OVER(ORDER BY The_Rank) Dense_Rank,
              ROW_NUMBER()  OVER(ORDER BY the_rank) Row_Number
         FROM Table_Rank
```

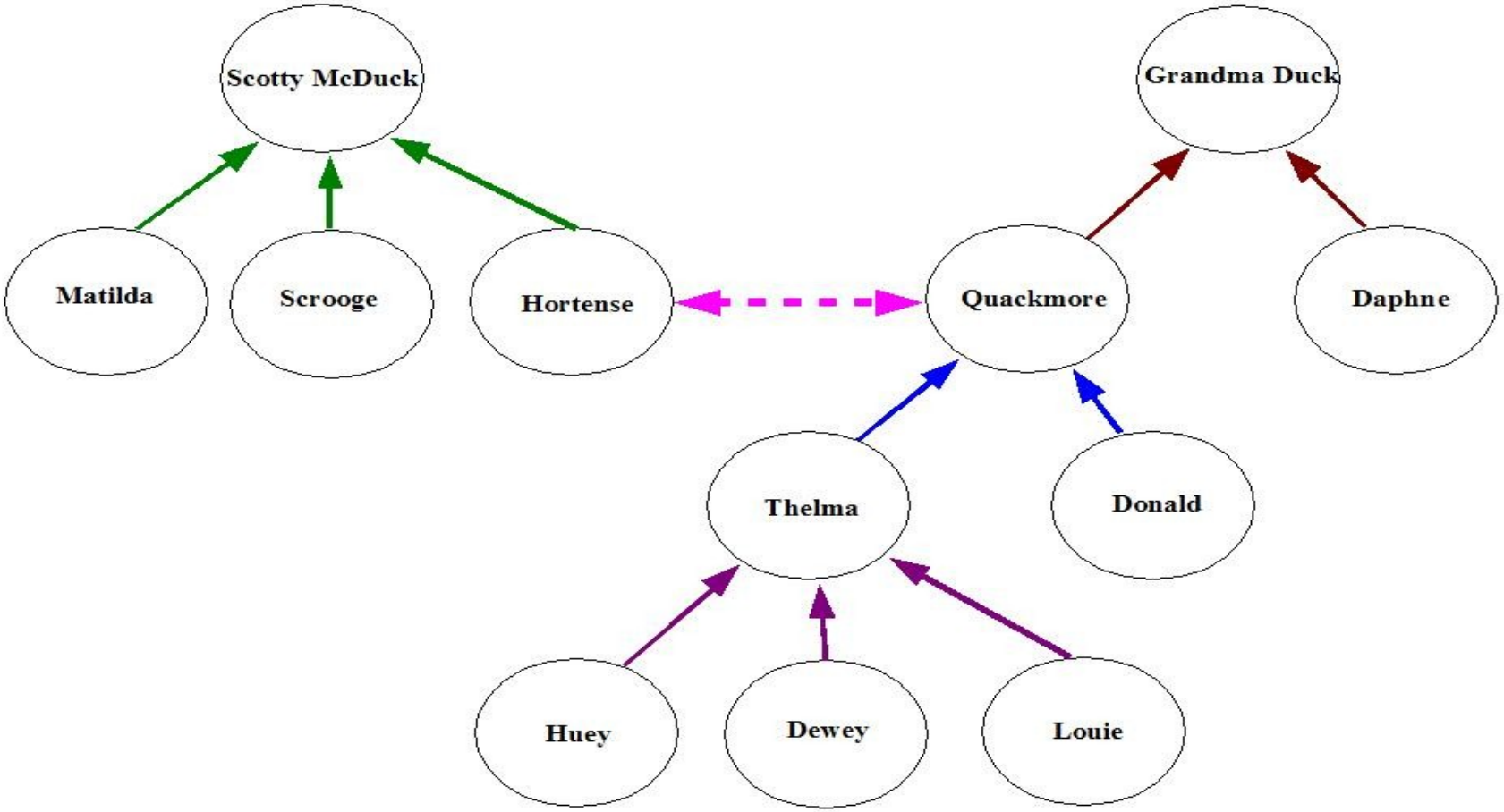
THE_RANK	ROWNUM	RANK	DENSE_RANK	ROW_NUMBER
1	1	1	1	1
2	2	2	2	2
2	3	2	2	3
3	4	4	3	4

```
SQL  > SELECT The_Rank,
              ROWNUM,
              RANK()        OVER(ORDER BY The_Rank desc) Rank,
              DENSE_RANK()  OVER(ORDER BY The_Rank desc) Dense_Rank,
              ROW_NUMBER()  OVER(ORDER BY the_rank desc) Row_Number
         FROM Table_Rank
```

THE_RANK	ROWNUM	RANK	DENSE_RANK	ROW_NUMBER
3	4	1	1	1
2	2	2	2	2
2	3	2	2	3
1	1	4	3	4

Hierarchical Queries

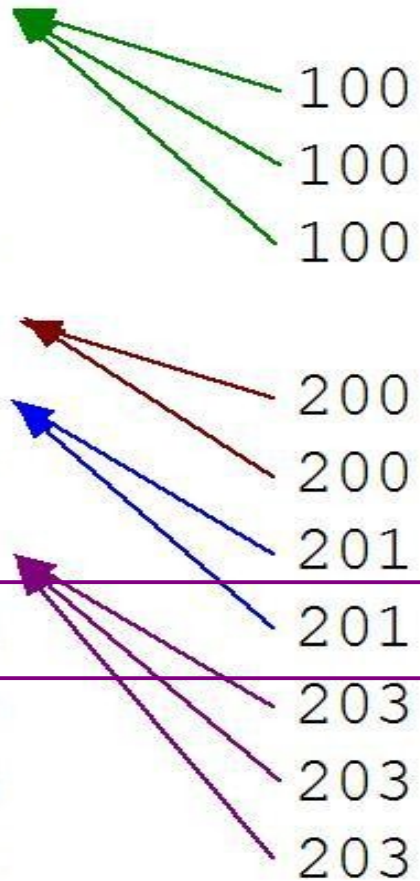
Disney Duckdom family tree



Hierarchical Queries

Disney Duckdom family tree

First_Name	Last_Name	Family_ID	Parent_ID
Scotty	McDuck	100	
Matilda	McDuck	101	100
Scrooge	McDuck	102	100
Hortense	McDuck	103	100
Grandma	Duck	200	
Quackmore	Duck	201	200
Daphne	Duck	202	200
Thelma	Duck	203	201
Donald	Duck	204	201
Huey	Duck	205	203
Dewey	Duck	206	203
Louie	Duck	207	203



Hierarchical Queries

Disney Duckdom family tree

```
SQL  > SELECT *
         FROM duck_tree
        ORDER BY family_id;
```

```
FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Scotty     McDuck            100
Matilda    McDuck            101        100
Scrooge    McDuck            102        100
Hortense   McDuck            103        100
Grandma    Duck              200
Quackmore  Duck              201        200
Daphne     Duck              202        200
Thelma     Duck              203        201
Donald     Duck              204        201
Huey       Duck              205        203
Dewey      Duck              206        203
Louie      Duck              207        203
```

Hierarchical Queries

Find the Children of Quackmore

```
SQL  > SELECT b.*
         FROM duck_tree a, duck_tree b
        WHERE a.first_name = 'Quackmore'
          AND a.family_id = b.parent_id
        ORDER BY b.family_id;
```

```
FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Thelma     Duck              203        201
Donald     Duck              204        201
```

Hierarchical Queries

Find the Grandchildren of Quackmore

```
SQL  > SELECT c.*
         FROM (SELECT b.*
                 FROM duck_tree a, duck_tree b
                WHERE a.first_name = 'Quackmore'
                  AND a.family_id = b.parent_id) b,
              duck_tree c
        WHERE b.family_id = c.parent_id
        ORDER BY c.family_id;

FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Huey       Duck              205        203
Dewey      Duck              206        203
Louie      Duck              207        203
```

Hierarchical Queries

Disney Duckdom family tree – CONNECT BY PRIOR

```
SELECT first_name,
       last_name,
       family_id,
       parent_id
  FROM duck_tree
CONNECT BY PRIOR
    family_id = parent_id;
```

```
FIRST_NAME LAST_NAME  FAMILY_ID  PARENT_ID      LEVEL
---------- ---------- ---------- ---------- ----------
Matilda    McDuck            101        100          1
Scrooge    McDuck            102        100          1
Hortense   McDuck            103        100          1
Quackmore  Duck              201        200          1
Thelma     Duck              203        201          2
Huey       Duck              205        203          3
Dewey      Duck              206        203          3
Louie      Duck              207        203          3
Donald     Duck              204        201          2
Daphne     Duck              202        200          1
Thelma     Duck              203        201          1
Huey       Duck              205        203          2
Dewey      Duck              206        203          2
Louie      Duck              207        203          2
Donald     Duck              204        201          1
Huey       Duck              205        203          1
Dewey      Duck              206        203          1
Louie      Duck              207        203          1
Scotty     McDuck            100                     1
Matilda    McDuck            101        100          2
Scrooge    McDuck            102        100          2
Hortense   McDuck            103        100          2
Grandma    Duck              200                     1
Quackmore  Duck              201        200          2
Thelma     Duck              203        201          3
Huey       Duck              205        203          4
Dewey      Duck              206        203          4
Louie      Duck              207        203          4
                                                     ...
```

Hierarchical Queries

Disney Duckdom family tree – START WITH

```
SELECT first_name,
       last_name,
       family_id,
       parent_id
  FROM duck_tree
CONNECT BY PRIOR
    family_id = parent_id
 START WITH
    first_name = 'Quackmore';
```

FIRST_NAME	LAST_NAME	FAMILY_ID	PARENT_ID	LEVEL
Matilda	McDuck	101	100	1
Scrooge	McDuck	102	100	1
Hortense	McDuck	103	100	1
Quackmore	Duck	201	200	1
Thelma	Duck	203	201	2
Huey	Duck	205	203	3
Dewey	Duck	206	203	3
Louie	Duck	207	203	3
Donald	Duck	204	201	2
Daphne	Duck	202	200	1
Thelma	Duck	203	201	1
Huey	Duck	205	203	2
Dewey	Duck	206	203	2
Louie	Duck	207	203	2
Donald	Duck	204	201	1
Huey	Duck	205	203	1
Dewey	Duck	206	203	1
Louie	Duck	207	203	1
Scotty	McDuck	100		1
Matilda	McDuck	101	100	2
Scrooge	McDuck	102	100	2
Hortense	McDuck	103	100	2
Grandma	Duck	200		1
Quackmore	Duck	201	200	2
Thelma	Duck	203	201	3
Huey	Duck	205	203	4
Dewey	Duck	206	203	4
Louie	Duck	207	203	4

...

Hierarchical Queries

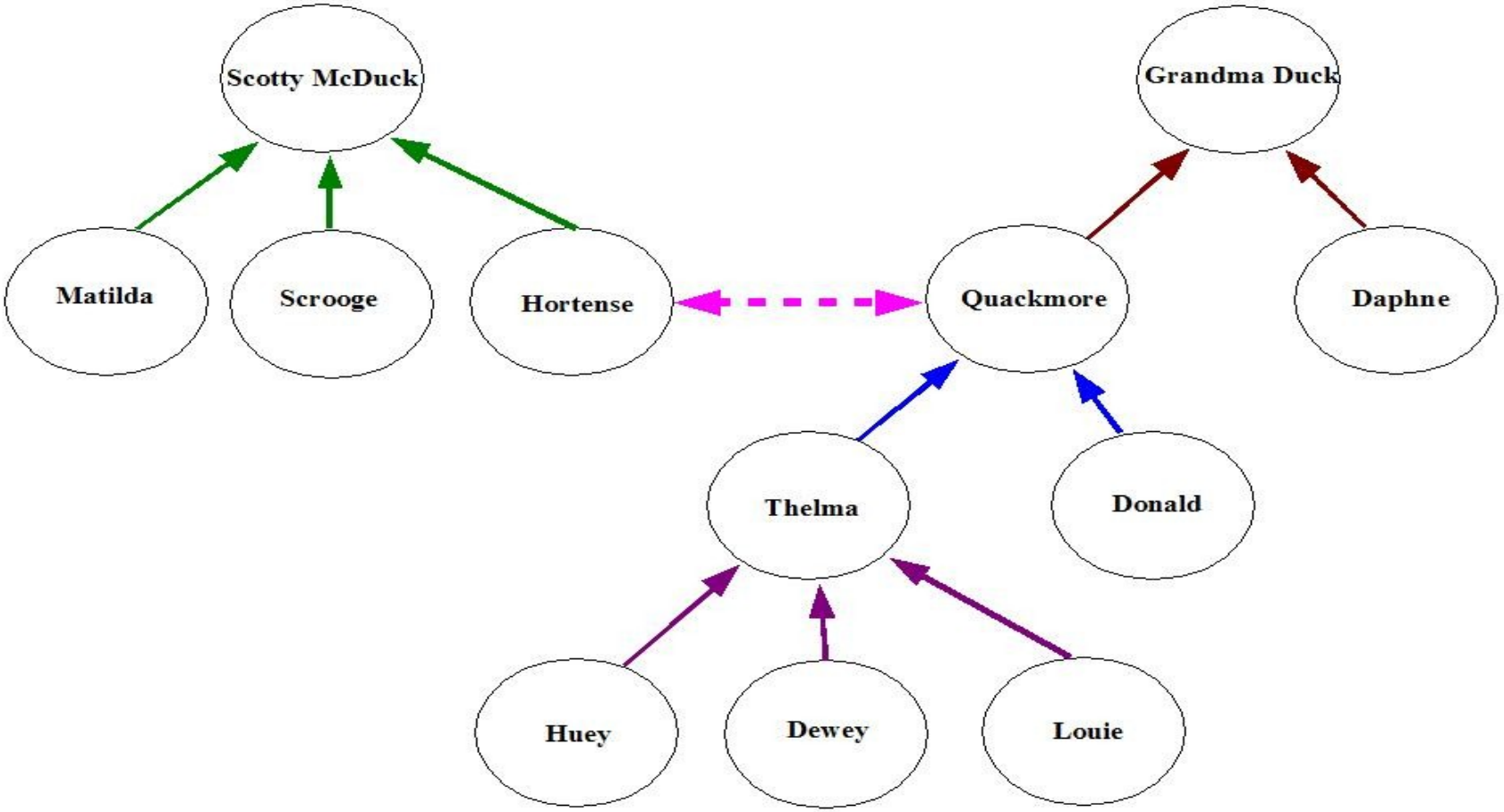
Find the Grandchildren of Quackmore

```
SQL  > SELECT *
         FROM duck_tree
       CONNECT BY PRIOR family_id = parent_id
         START WITH first_name = 'Quackmore';

FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Quackmore  Duck              201        200
Thelma     Duck              203        201
Huey       Duck              205        203
Dewey      Duck              206        203
Louie      Duck              207        203
Donald     Duck              204        201
```

```
SQL  > SELECT *
         FROM duck_tree
        WHERE level = 3
       CONNECT BY PRIOR family_id = parent_id
         START WITH first_name = 'Quackmore';

FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Huey       Duck              205        203
Dewey      Duck              206        203
Louie      Duck              207        203
```

Hierarchical Queries

Disney Duckdom family tree



Hierarchical Queries

Find the Parents & Grandparents for Donald Duck

```
SQL  > SELECT *
         FROM duck_tree
      CONNECT BY family_id = PRIOR parent_id
        START WITH first_name = 'Donald';
```

```
FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
Donald     Duck              204        201
Quackmore  Duck              201        200
Grandma    Duck              200
```

Hierarchical Queries

Indentation

```
SQL  > SELECT LPAD(' ', 2*level-1)||first_name first_name,
              last_name, family_id, parent_id
         FROM duck_tree
       CONNECT BY PRIOR family_id = parent_id
         START WITH first_name = 'Quackmore';
```

```
FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID
---------- ---------- ---------- ----------
 Quackmore Duck              201        200
   Thelma  Duck              203        201
     Huey  Duck              205        203
     Dewey Duck              206        203
     Louie Duck              207        203
   Donald  Duck              204        201
```

Hierarchical Queries

Path

```
SQL  > SELECT first_name, last_name, family_id, parent_id,
              SYS_CONNECT_BY_PATH(first_name, '/') "Path"
         FROM duck_tree
       CONNECT BY PRIOR family_id = parent_id
         START WITH first_name = 'Quackmore';
```

```
FIRST_NAME LAST_NAME   FAMILY_ID  PARENT_ID Path
---------- ---------- ---------- ---------- -------------------------
Quackmore  Duck              201        200 /Quackmore
Thelma     Duck              203        201 /Quackmore/Thelma
Huey       Duck              205        203 /Quackmore/Thelma/Huey
Dewey      Duck              206        203 /Quackmore/Thelma/Dewey
Louie      Duck              207        203 /Quackmore/Thelma/Louie
Donald     Duck              204        201 /Quackmore/Donald
```

Grouping – Rollup operation

Finding aggregates at different levels

DOG_ORIGIN table

Country	Breed	Breed_Size	Population
Germany	German Shepherd Dog	Big	12000
Germany	Dobermann	Big	8000
Germany	Rottweiler	Big	9000
USA	Siberian Husky	Medium	5000
USA	Alaskan Malamute	Medium	3000
USA	American Bulldog	Big	8000
Switzerland	Bernese Mountain Dog	Big	2000
Switzerland	Saint Bernard Dog	Big	2000
Switzerland	Entlebuch Cattle Dog	Medium	2000
Australia	Australian Cattle Dog	Medium	6000
Australia	Jack Russell Terrier	Small	7000

- Same table, new column
- Counting Rows / Summing Data
- How can we group data
- What are totals
- What are subtotals
- What types of questions can we ask
- What types of answers will we get
- Do we need an application

Grouping – Rollup operation

Finding a Grand Total and Subtotal

Total dogs in the world

```
SQL > SELECT SUM(population)
        FROM dog_origin;


SUM(POPULATION)
---------------
          64000 <- grand total
```

Total dogs in America

```
SQL > SELECT SUM(population)
        FROM dog_origin
       WHERE country = 'USA';

SUM(POPULATION)
---------------
          16000 <- WHERE subtotal (group by country)
```

Grouping – Rollup operation

GROUP BY – finding aggregates across full table

Total dogs in each country?

```
SQL  > SELECT country,SUM(population)
         FROM dog_origin
        GROUP BY country;
```

```
COUNTRY          SUM(POPULATION)
---------------- ---------------
Australia                 13000 <- group by country
Germany                   29000 <- group by country
Switzerland                6000 <- group by country
USA                       16000 <- group by country
```

Grouping – Rollup operation

GROUP BY – ROLLUP

- ROLLUP is an extension of the GROUP BY clause
- ROLLUP operation works on a set of columns
- Aggregates a summary row for each group of columns supplied
- A grand total for all rows is presented

Syntax

```
SELECT blah, blah, blah FROM a_table
GROUP BY ROLLUP ([columns of interest])
```

Groupings for 4 Columns Become:

```
GROUP BY ROLLUP  (c1,c2,c3,c4)   - group by (c1)
                                 - group by (c1,c2)
                                 - group by (c1,c2,c3)
                                 - group by (c1,c2,c3,c4)
                                 - grand total (t1)
```

Grouping – Rollup operation

Using ROLLUP to 'replicate' GROUP BY

How many dogs are in each country?

```
SQL  > SELECT country,SUM(population)
         FROM dog_origin
        GROUP BY ROLLUP(country);
                        (c1)
```

```
COUNTRY         SUM(POPULATION)
--------------- ---------------
Australia                13000 <- rollup by country (c1)
Germany                  29000 <- rollup by country (c1)
Switzerland               6000 <- rollup by country (c1)
USA                      16000 <- rollup by country (c1)
                         64000 <- grand total given (t1)
```

Grouping – Rollup operation

ROLLUP on multiple columns

```
SQL  > SELECT country,breed_size,SUM(population)
         FROM dog_origin
        GROUP BY ROLLUP(country,breed_size);
                                (c1,c2)
```

```
COUNTRY         BREED_SIZE SUM(POPULATION)
--------------- ---------- ---------------
Australia       Medium               6000 <- rollup by country & breed_size (c1,c2)
Australia       Small                7000 <- rollup by country & breed_size (c1,c2)
Australia                           13000 <- rollup by country              (c1)
Germany         Big                 29000 <- rollup by country & breed_size (c1,c2)
Germany                             29000 <- rollup by country              (c1)
Switzerland     Big                  4000 <- rollup by country & breed_size (c1,c2)
Switzerland     Medium               2000 <- rollup by country & breed_size (c1,c2)
Switzerland                          6000 <- rollup by country              (c1)
USA             Big                  8000 <- rollup by country & breed_size (c1,c2)
USA             Medium               8000 <- rollup by country & breed_size (c1,c2)
USA                                 16000 <- rollup by country              (c1)
                                    64000 <- grand total line               (t1)
```

Grouping – Rollup operation

Partial ROLLUP

```
SQL  > SELECT country,breed_size,SUM(population)
         FROM dog_origin
        GROUP BY country, ROLLUP(breed_size);
```

```
COUNTRY         BREED_SIZE SUM(POPULATION)
--------------- ---------- ---------------
Australia       Medium               6000
Australia       Small                7000
Australia                           13000
Germany         Big                 29000
Germany                             29000
Switzerland     Big                  4000
Switzerland     Medium               2000
Switzerland                          6000
USA             Big                  8000
USA             Medium               8000
USA                                 16000
                                            <- grand total line gone
```

Grouping – Rollup operation

Partial ROLLUP '=' GROUP BY

```
SQL  > SELECT country,breed_size,SUM(population)
         FROM dog_origin
        GROUP BY country,breed_size;
```

```
COUNTRY         BREED_SIZE SUM(POPULATION)
--------------- ---------- ---------------
Australia       Medium                6000
Australia       Small                 7000
                                            <- rollup by country gone
Germany         Big                  29000
                                            <- rollup by country gone
Switzerland     Big                   4000
Switzerland     Medium                2000
                                            <- rollup by country gone
USA             Big                   8000
USA             Medium                8000
                                            <- rollup by country gone
                                            <- grand total line gone
```

Grouping – Rollup operation

Partial ROLLUP

- ROLLUP is an extension of the GROUP BY clause
- ROLLUP operation works on a set of columns
- Aggregates a summary row for each group of columns supplied
- A grand total for all rows is presented
- Partial rollups are available by moving columns between the GROUP BY list and the ROLLUP list.

Syntax

```
SELECT blah, blah, blah FROM a_table
GROUP BY (column[s]) ROLLUP (column[s])
```

Possible Groupings for 4 Columns Become:

```
group by rollup  (c1,c2,c3,c4) - full rollup option
group by  c1,rollup (c2,c3,c4) - removes grand total line (t1)
group by  c1,c2,rollup (c3,c4) - removes (t1) and group by (c1)
group by  c1,c2,c3,rollup (c4) - removes (t1) and group by (c1) & (c1,c2)
group by (c1,c2,c3,c4)         - removes (t1) and group by (c1),(c1,c2),&(c1,c2,c3)
```

A Primer on Globalization

Session 3 - 1:30pm-2:00pm - room(s)107/109/111/113

James F. Koopmann

jkoopmann@pinehorse.com

www.pinehorse.com