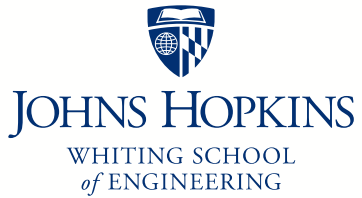
Johns Hopkins Engineering

Principles of Database Systems

Module 9 / Lecture 1

Structured Query Language (SQL)

The Relational DB Language I



Structured Query Language History

- The relational database model was originally developed by Dr. Codd. The idea was published in 1970 to become one of the greatest research papers in computer history.

- Dr. E(dger). F. "Ted" Codd is one of the great names in computing. He is often called the Father of Relational Databases.

Structured Query Language History (cont.)

- Colleagues at IBM and Berkeley experimented with his ideas and found that they simplified queries that were complex or too tied to the internal structures in other database paradigms.

- The Structured Query Language (SQL) language was originally developed by IBM in a prototype RDBMS in the mid-1970. SQL has also been implemented in IBM’s DB2.

Structured Query Language History (cont.)

- The original commercial SQL language was introduced by Oracle Cooperation in 1979.
- The first standard SQL1 (or SQL 86) was introduced by ANSI in 1986. The second standard SQL2 (or SQL 92) was introduced by ISO in 1992. The third standard SQL3 with object-oriented and other new features was introduced by ISO in 1999.
- SQL 2003 includes XML support, sequences, columns with auto-generated values, and MERGE (upsert). SQL 2006 includes more XML features and XQuery.
- SQL 2008 includes user-defined types and routines, reference type, collection types, triggers, BLOBs, and CLOBs, TRUNCATE, INSTEAD OF triggers and others.
- SQL 2011 adds temporal data and FETCH clause.
- SQL 2016 adds JSON, polymorphic table functions, and row pattern matching.

Benefits for Using SQL

- SQL processes sets of records rather than just one at a time.
- SQL does not require you to specify the access method to the data. This feature makes it easier for you to concentrate on obtaining the desired data.
- It acquires the desired results without specifying an access method to the data.
- RDBMS uses an optimizer for determining the fastest means of accessing the specified data.

Benefits for Using SQL (cont.)

- Can be used by all users. SQL provides easy-to-learn commands that are both consistent and applicable for all users.

- Performs commands for a variety of tasks:
  - Data definition
  - Data manipulation
  - Data retrieval

- Is supported by all major relational databases. All programs written in standard SQL are portable.

Programming Paradigms

- Structured programming
- Procedural programming
- Modular programming
- Data abstraction
- Object-oriented programming

Programming Paradigms (cont.)

- SQL is designed for a specific, limited purpose – DDL, DML, and DQL (querying data) in a relational database.
- SQL is a case insensitive language. A string constant (e.g. 'Smith') is case sensitive.
- Different RDBMS's may use different syntax and support various subsets of SQL standards.
- Standard syntax ensures code portability. Vendor’s specific features/syntax are not portable.

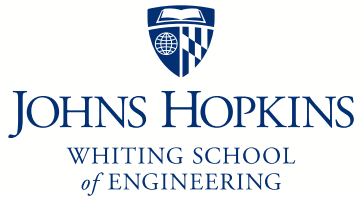
Johns Hopkins Engineering

Principles of Database Systems

Module 9 / Lecture 2

Structured Query Language (SQL)

The Relational DB Language I



SQL Data Definition Language (DDL)

- DDL commands (e.g., CREATE, ALTER, DROP) are automatically committed while DML commands may need an explicit commit statement.
- The order for creating tables may be important because the DBMS verifies the referential integrity constraints.

SQL Data Definition Language (cont.)

- Oracle is used for a SQL demo. A sample database with two tables “emp” and “dept”. The database is from an Oracle default instance ORCL.

EMP

empno	ename	title	mgr	hire_date	salary	comm	deptno
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	2/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	2/22/1981	1250	500	30
7566	JONES	MANAGER	7839	4/2/1981	2975		20
7654	MARTIN	SALESMAN	7698	9/28/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	5/1/1981	2850		30
7782	CLARK	MANAGER	7839	6/9/1981	2450		10
7788	SCOTT	ANALYST	7566	4/19/1987	3000		20
7839	KING	PRESIDENT		11/17/1981	5000		10
7844	TURNER	SALESMAN	7698	9/8/1981	1500	0	30
7876	ADAMS	CLERK	7788	5/23/1987	1100		20
7900	JAMES	CLERK	7698	12/3/1981	950		30
7902	FORD	ANALYST	7566	12/3/1981	3000		20
7934	MILLER	CLERK	7782	1/23/1982	1300		10

DEPT

deptno	dname	loc
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL Data Definition Language (cont.)

- To create multiple tables and views and other database objects (not an instance of an object, or a row of a table) in *a single transaction* use CREATE SCHEMA.

  Example:

  ```
  CREATE SCHEMA db_proj AUTHORIZATION jdoe
  ```

- Database objects such as tables, views, schemas are associated with a database USER. Therefore, a user (e.g., jdoe) has to be created before creating the schema.

SQL Data Definition Language (cont.)

- RDBMS uses the catalog in which descriptions of data items are stored and the catalog is accessible to users.

```
SQL> SELECT * FROM CAT;
TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
DEPT                           TABLE
EMP                            TABLE
```

- The “DESCRIBE” command returns the definitions of tables and views.

```
SQL> DESCRIBE dept;
NAME      Null?    TYPE
--------- -------- -------------
DEPTNO    NOT NULL NUMBER(2)
DNAME              VARCHAR2(20)
LOC                VARCHAR2(30)
```

SQL – CREATE TABLE

- Purpose: To create a *table*, the basic structure to hold user data, specifying the following information:
  - Column definitions
  - Integrity constraints (different approaches)
  - Others – storage characteristics, tablespace, cluster, degree of parallelism used to create the table and the default degree of parallelism for queries on the table (Oracle features)

- SQL 92 Summary document for reference

SQL – CREATE TABLE (cont.)

- CREATE TABLE Syntax

CREATE TABLE TableName
(<**Column-Definition**>* [, <**Table-Constraint**>*])

<**Column-Definition**>: ColumnName DataType
[DEFAULT { DefaultValue | USER | NULL }]
[[CONSTRAINT ConstraintName] NOT NULL]
[[CONSTRAINT ConstraintName] UNIQUE]
[[CONSTRAINT ConstraintName] PRIMARY KEY]
[[CONSTRAINT ConstraintName] FOREIGN KEY REFERENCES TableName [(ColumnName)] [ON DELETE <Action-Specification>]
[ON UPDATE <Action-Specification>]]

The asterisk * after a syntax element indicates that a comma-separated list can be used.
Names enclosed in angle brackets <> denote definitions defined later in the syntax.
Square brackets [] enclose optional elements.
Curly brackets {} enclose choice elements.
The parentheses ( ) denote themselves.
Double hyphens -- denote comments that are not part of the syntax.

SQL – CREATE TABLE (cont.)

- CREATE TABLE Syntax

  CREATE TABLE TableName
  (<**Column-Definition**>* [, <**Table-Constraint**>*])

  <**Table-Constraint**>: [CONSTRAINT ConstraintName]
  { <Primary-Key-Constraint> |
  <Foreign-Key-Constraint> |
  <Uniqueness-Constraint> |
  <Primary-Key-Constraint>: PRIMARY KEY (ColumnName*)
  <Foreign-Key-Constraint> FOREIGN KEY (ColumnName*)
  REFERENCES TableName (ColumnName*)
  [ON DELETE <Action-Specification>]
  [ON UPDATE <Action-Specification>]
  <Uniqueness-Constraint>: UNIQUE (ColumnName*)
  <Action-Specification>: { CASCADE | SET NULL | SET DEFAULT | NO ACTION }

SQL – CREATE TABLE (cont.)

A sample database with two tables "**emp**" and "**dept**" in an Oracle default instance ORCL. Table with *column constraints* to define the "**emp**" table owned by "SCOTT":

```
CREATE TABLE scott.emp
(empno NUMBER        CONSTRAINT pk_emp PRIMARY KEY,
 ename VARCHAR2(10) CONSTRAINT nn_ename NOT NULL
                    CONSTRAINT upper_ename
                    CHECK (ename = UPPER(ename)),
 job VARCHAR2(9),
 mgr NUMBER         CONSTRAINT fk_mgr REFERENCES scott.emp(empno),
 hiredate DATE      DEFAULT SYSDATE,
 sal NUMBER(10,2)   CONSTRAINT ck_sal CHECK (sal > 500),
 comm NUMBER(9,0)   DEFAULT NULL,
 deptno NUMBER(2)   CONSTRAINT nn_deptno NOT NULL
                    CONSTRAINT fk_deptno REFERENCES scott.dept(deptno))
```

SQL – CREATE TABLE (cont.)

Table with table constraints to define the “emp_1” table owned by SCOTT:

```
CREATE TABLE scott.emp_1
(empno NUMBER        NOT NULL,
 ename VARCHAR2(10)      NOT NULL,
 job VARCHAR2(9),
 mgr NUMBER,
 hiredate DATE       DEFAULT SYSDATE,
 sal NUMBER(10,2)    CONSTRAINT ck_sal_1 CHECK (sal > 500),
 comm NUMBER(9,0)    DEFAULT NULL,
 deptno NUMBER(2)    NOT NULL,
  CONSTRAINT pk_emp PRIMARY KEY(empno),
  CONSTRAINT fk_deptno_1 FOREIGN KEY(deptno) REFERENCES scott.dept(deptno),
  CONSTRAINT fk_mgr FOREIGN KEY(mgr) REFERENCES scott.emp(empno) )
```

SQL – ALTER TABLE

- ALTER TABLE Syntax

```
ALTER TABLE TableName
    { ADD { <Column-Definition> | , <Table-Constraint> } |
      ALTER ColumnName { SET DEFAULT DefaultValue | DROP DEFAULT } |
      DROP ColumnName { CASCADE | RESTRICT } |
      DROP CONSTRAINT ConstraintName { CASCADE | RESTRICT } }
```

SQL – ALTER TABLE (cont.)

- Purpose: To alter the definition of a table in one of the following ways:
  - Add or delete a column
  - Add an integrity constraint
  - Redefine a column (datatype, size, default value)
  - Enable, disable, or drop an integrity constraint or trigger
  - Modify storage characteristics or other parameters, explicitly allocate an extent, explicitly de-allocate the unused space of a table, allow or disallow writing to a table, modify the degree of parallelism for a table (Oracle features)

SQL – ALTER TABLE (cont.)

Using CREATE TABLE to create "**emp**" table without specifying PK and FKs, and using ALTER TABLE to add PK and FKs:

```
CREATE TABLE emp
(empno NUMBER NOT NULL,
 ename VARCHAR2(10) NOT NULL,
 job VARCHAR2(9),
 mgr NUMBER,
 hiredate DATE       DEFAULT SYSDATE,
 sal NUMBER(10,2)    CONSTRAINT ck_sal CHECK (sal > 500),
 comm NUMBER(9,0)    DEFAULT NULL,
 deptno NUMBER(2) NOT NULL);
```

```
ALTER TABLE scott.emp
ADD (CONSTRAINT pk_emp PRIMARY KEY(empno),
 CONSTRAINT fk_deptno FOREIGN KEY(deptno) REFERENCES scott.dept(deptno),
 CONSTRAINT fk_mgr FOREIGN KEY(mgr) REFERENCES scott.emp(empno) )
```

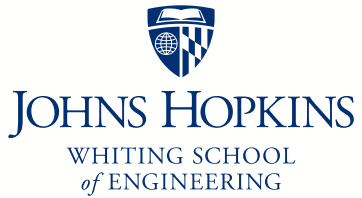
Johns Hopkins Engineering

Principles of Database Systems

Module 9 / Lecture 3

Structured Query Language (SQL)

The Relational DB Language I



Types of Constraints in SQL

- Types of constraints are C (CHECK constraint for a domain and NOT NULL constraint for required data), P (PRIMARY KEY), R (FOREIGN KEY), and U (UNIQUE KEY).
  - Example: Use Oracle to confirm constraints on a table with the USER_CONSTRAINTS data dictionary table:

```
SQL> DESCRIBE user_constraints
    Name                                      Null?    Type
    ----------------------------------------- -------- ----
    OWNER                                     NOT NULL VARCHAR2(30)
    CONSTRAINT_NAME                           NOT NULL VARCHAR2(30)
    CONSTRAINT_TYPE                                    VARCHAR2(1)
    TABLE_NAME                                NOT NULL VARCHAR2(30)
```

Types of Constraints in SQL (cont.)

- Example: Using SQL to view table's constraints associated with EMP3

```
SQL> SELECT owner, constraint_name, constraint_type, table_name
  2  FROM user_constraints
  3  WHERE table_name = 'EMP3';

OWNER    CONSTRAINT_NAME      C TABLE_NAME
-------- -------------------- - ----------
SCOTT    SYS_C00997           C EMP3
SCOTT    SYS_C00998           C EMP3
SCOTT    SYS_C00999           C EMP3
SCOTT    CK_SAL_3             C EMP3
SCOTT    PK_EMP3              P EMP3
SCOTT    FK_DEPTNO3           R EMP3
SCOTT    FK_MGR3              R EMP3

7 rows selected.
```

Domain in SQL

- In a column constraint, the CHECK clause can reference a domain constraint or can be defined explicitly.

```
CREATE DOMAIN DomainName AS DataType
[ DEFAULT defaultValue]
[ CHECK (SearchCondition) ];
```

Example:

```
sex CHAR CHECK (sex in ('M', 'F')),

CREATE DOMAIN SexType AS CHAR
DEFAULT 'M'
CHECK ( VALUE IN ('M', 'F') );
```

DROP and TRUNCATE in SQL

- Removing a table, schema, and other database objects:

  DROP TABLE TableName { CASCADE | RESTRICT }

  DROP SCHEMA SchemaName { CASCADE | RESTRICT }

- Common database objects – SCHEMA, TABLE, INDEX, VIEW, DOMAIN. *Use DROP carefully!*

- Quickly removing all records in a table, and the data may not be able to recover. RDBMS implements differently.

  TRUNCATE TABLE TableName

Data Type in SQL

- ISO Data Type

ISO Data Type	Declarations
Boolean	BOOLEEAN
Character	CHAR
Bit	BIT
Exact Numeric	NUMERIC, DECIMAL, INTEGER
Approximate Numeric	FLOAT, REAL, DOUBLE PRCISION
DateTime	DATE, TIME, TIMESTAMP
Large objects	BLOB (Binary Large Object)
Interval	INTERVAL

Check your RDBMS for the supported data types

Data Type in SQL (cont.)

- Data type and length are the most fundamental integrity constraints applied to data in a database.
- Common practice for data types:
  - Boolean – True or False
  - Exact Numeric – SMALLINT, INTEGER, or DECIMAL
  - Approximate Numeric – FLOAT or REAL
  - Character – CHAR or VARCHAR
    - Present different characteristics
  - Date and time – DATE, TIME, or TIMESTAMP
    - Allow arithmetic calculation and build-in functions
    - Is “10302007” good or bad?
  - Large Objects – BLOB, CLOB, DBCLOB, GRAPHIC, VARGRAPHIC (DBMS dependent)

Query in SQL

- SELECT Syntax:

  SELECT [DISTINCT] <Column-Specification>*
  FROM <Table-Specification>
  [**WHERE** <Row-Condition>]
  [**GROUP BY** ColumnName*]
  [**HAVING** <Group-Condition>]

  - The SELECT-clause lists the attributes or functions to be retrieved
  - The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries

Query in SQL (cont.)

- SELECT Syntax:
  - The WHERE clause specifies the conditions for selection and join of records from the tables specified in the FROM clause
  - GROUP BY specifies grouping attributes
  - HAVING specifies a condition for selection of groups
  - ORDER BY specifies an order for displaying the result of a query

Query in SQL (cont.)

- Comparison Operators
  - = Equal to
  - <> Not equal to (ISO standard; != may work for some RDBMS)
  - > Greater than
  - >= Greater than or equal to
  - < Less than
  - <= Less than or equal to

Query in SQL (cont.)

- Logical Operator Precedence
  - The logical operators and arithmetic operators in SQL are handled according to precedence rules.
  - These operators can affect the SQL evaluation of an expression in subtle and unexpected ways in your results.

Query in SQL (cont.)

- Logical Operator Precedence (cont.)
  - The precedence hierarchy is:
    Parentheses (highest)
    Multiplication, **Division**
    Subtraction/Addition
    NOT
    AND
    OR (lowest)
  - The parentheses operators can be used to ensure the correct evaluation sequence for the SQL statements.

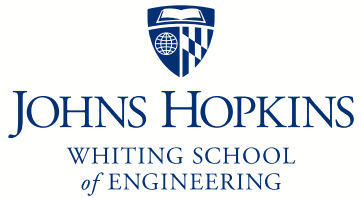
Johns Hopkins Engineering

Principles of Database Systems

Module 9 / Lecture 4

Structured Query Language (SQL)

The Relational DB Language I



SQL Query Examples

- List all employees and departments.

```
SQL> SELECT * FROM emp;
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
--------- ---------- --------- --------- --------- --------- --------- ---------
     7369 SMITH      CLERK          7902 17-DEC-80       800                    20
     7499 ALLEN      SALESMAN       7698 20-FEB-81      1600        300         30
     7521 WARD       SALESMAN       7698 22-FEB-81      1250        500         30
     7566 JONES      MANAGER        7839 02-APR-81      2975                    20
     7654 MARTIN     SALESMAN       7698 28-SEP-81      1250       1400         30
     7698 BLAKE      MANAGER        7839 01-MAY-81      2850                    30
     7782 CLARK      MANAGER        7839 09-JUN-81      2450                    10
     7788 SCOTT      ANALYST        7566 19-APR-87      3000                    20
     7839 KING       PRESIDENT           17-NOV-81      5000                    10
     7844 TURNER     SALESMAN       7698 08-SEP-81      1500          0         30
     7876 ADAMS      CLERK          7788 23-MAY-87      1100                    20
     7900 JAMES      CLERK          7698 03-DEC-81       950                    30
     7902 FORD       ANALYST        7566 03-DEC-81      3000                    20
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10
14 rows selected.
```

```
SQL> SELECT * FROM dept;
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
4 rows selected.
```

SQL Query Examples (cont.)

- List all jobs for all the employees

```
SQL> SELECT job FROM emp;
JOB
---------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALYST
PRESIDENT
SALESMAN
CLERK
CLERK
ANALYST
CLERK
14 rows selected.
```

- List unique jobs for all the employees

```
SQL> SELECT DISTINCT job FROM emp;
JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
5 rows selected.
```

SQL Query Examples (cont.)

- List all employees from department 10

```
SQL> SELECT *
  2  FROM emp
  3  WHERE deptno = 10;
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
--------- ---------- --------- --------- --------- --------- --------- ---------
     7782 CLARK      MANAGER        7839 09-JUN-81      2450                    10
     7839 KING       PRESIDENT           17-NOV-81      5000                    10
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10
3 rows selected.
```

- List the name, salary and job of all employees in department 20 whose salary is more than $2,000

```
SQL> SELECT ename, sal, job
  2  FROM emp
  3  WHERE deptno = 20
  4  AND sal > 2000;
 ENAME             SAL JOB
---------- --------- ---------
JONES            2975 MANAGER
SCOTT            3000 ANALYST
FORD             3000 ANALYST
 3 rows selected.
```

SQL Query Examples (cont.)

- List the name, job and salary of all employees whose job is manager or president

```
SQL> SELECT ename, job, sal
  2  FROM emp
  3  WHERE job = 'MANAGER' OR job = 'PRESIDENT';
 ENAME      JOB              SAL
---------- --------- ---------
JONES      MANAGER         2975
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
KING       PRESIDENT       5000
 4 rows selected.
```

- List all employees whose job is manager or is a clerk in department 10

```
SQL> SELECT empno, ename, job, deptno
  2  FROM emp
  3  WHERE job = 'MANAGER' OR job = 'CLERK' AND deptno = 10;
     EMPNO ENAME      JOB           DEPTNO
--------- ---------- --------- ---------
      7566 JONES      MANAGER           20
      7698 BLAKE      MANAGER           30
      7782 CLARK      MANAGER           10
      7934 MILLER     CLERK             10
 4 rows selected.
```

SQL Query Examples (cont.)

- List the employees who work in department 10 and whose job is either manager or clerk

```
SQL> SELECT empno, ename, job, deptno
  2  FROM emp
  3  WHERE (job = 'MANAGER' OR job = 'CLERK') AND deptno = 10;
    EMPNO ENAME      JOB           DEPTNO
--------- ---------- --------- ---------
     7782 CLARK      MANAGER           10
     7934 MILLER     CLERK             10
2 rows selected.
```

- List all employees whose salary is between $1,200 and $1,400 inclusive

```
SQL> SELECT ename, job, sal
  2  FROM emp
  3  WHERE sal BETWEEN 1200 AND 1400;
ENAME      JOB              SAL
---------- --------- ---------
WARD       SALESMAN        1250
MARTIN     SALESMAN        1250
MILLER     CLERK           1300
3 rows selected.
```

Pattern Matching in SQL

- “LIKE” for pattern matching to specify a search condition in WHERE clause
  - **%** for any sequence of characters as a wild card character
  - _ for any single character
  - Not good for full-text search in long text attributes or documents

Pattern Matching in SQL (cont.)

- List employees whose name begins with an 'M'

```
SQL> SELECT ename, job, deptno
  2  FROM emp
  3  WHERE ename LIKE 'M%';
ENAME      JOB           DEPTNO
---------- --------- ---------
MARTIN     SALESMAN          30
MILLER     CLERK             10
2 rows selected.
```

- List employees whose name has an 'R' as the third letter

```
SQL> SELECT ename, job, deptno
  2  FROM emp
  3  WHERE ename LIKE '__R%';
     --(2 CONSECUTIVE UNDERSCORES)
  ENAME      JOB           DEPTNO
---------- --------- ---------
WARD       SALESMAN          30
MARTIN     SALESMAN          30
TURNER     SALESMAN          30
FORD       ANALYST           20
4 rows selected.
```

Arithmetic Expressions in SQL

- SQL command can contain arithmetic expressions made up of column names and constants connected by arithmetic operators (+, -, *, /)

- List employee name salary, commission, salary plus commission of employees whose commission is more than a quarter of their salary

```
SQL> SELECT ename, sal, comm, sal + comm
  2  FROM emp
  3  WHERE comm > 0.25 * sal;
ENAME            SAL      COMM  SAL+COMM
---------- --------- --------- ---------
WARD            1250       500      1750
MARTIN          1250      1400      2650
2 rows selected.
```

NULL Value Comparison in SQL

- NULL means “no value specified”, not a zero or a field with spaces
- NULL value comparison
  - Compare a variable with NULL in WHERE clause using IS NULL or IS NOT NULL
  - Don’t use “= NULL”
  - Use ISNULL(), NVL(), IFNULL() functions for properly handling NULL values
  - Check your RDBMS for the supported null-related functions
  - Deal with NULL values with care

NULL Value Comparison Query Examples

- List the employees in department 30 who do not receive a commission

```
SQL> SELECT ename, job
  2  FROM emp
  3  WHERE deptno = 30
  4  AND comm IS NULL;
ENAME      JOB
---------- ---------
BLAKE      MANAGER
JAMES      CLERK
2 rows selected.
```

```
SQL> SELECT ename, job
  2  FROM emp
  3  WHERE deptno = 30
  4  AND comm = NULL;

no rows selected.
```

```
SQL> SELECT * FROM emp;
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
--------- ---------- --------- --------- --------- --------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80       800                    20
     7499 ALLEN      SALESMAN       7698 20-FEB-81      1600        300         30
     7521 WARD       SALESMAN       7698 22-FEB-81      1250        500         30
     7566 JONES      MANAGER        7839 02-APR-81      2975                    20
     7654 MARTIN     SALESMAN       7698 28-SEP-81      1250       1400         30
     7698 BLAKE      MANAGER        7839 01-MAY-81      2850                    30   <-
     7782 CLARK      MANAGER        7839 09-JUN-81      2450                    10
     7788 SCOTT      ANALYST        7566 19-APR-87      3000                    20
     7839 KING       PRESIDENT           17-NOV-81      5000                    10
     7844 TURNER     SALESMAN       7698 08-SEP-81      1500          0         30   <-
     7876 ADAMS      CLERK          7788 23-MAY-87      1100                    20
     7900 JAMES      CLERK          7698 03-DEC-81       950                    30   <-
     7902 FORD       ANALYST        7566 03-DEC-81      3000                    20
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10
14 rows selected.
```

Arithmetic Expressions Query Examples

- List name job, salary commission, and sum of salary and commission of employees in department 30

```
SQL> SELECT ename, job, sal, comm, sal + comm
  2  FROM emp
  3  WHERE deptno = 30;

ENAME      JOB              SAL      COMM  SAL+COMM
---------- --------- --------- --------- ---------
ALLEN      SALESMAN        1600       300      1900
WARD       SALESMAN        1250       500      1750
MARTIN     SALESMAN        1250      1400      2650
BLAKE      MANAGER         2850                      <--
TURNER     SALESMAN        1500         0      1500
JAMES      CLERK            950                      <--
6 rows selected.
```

```
SQL> SELECT ename, job, sal, comm, sal + NVL(comm,0)
  2  FROM emp
  3  WHERE deptno = 30;

ENAME      JOB              SAL      COMM SAL+NVL(COMM,0)
---------- --------- --------- --------- ---------------
ALLEN      SALESMAN        1600       300            1900
WARD       SALESMAN        1250       500            1750
MARTIN     SALESMAN        1250      1400            2650
BLAKE      MANAGER         2850                      2850   <--
TURNER     SALESMAN        1500         0            1500
JAMES      CLERK            950                       950   <--
6 rows selected.
```

ORDER BY Clause in SQL

- ORDER BY Clause
  - Syntax:
    ORDER BY expression [**ASC** | DESC], …
  - Expression
    - Column
    - Expression based on columns
    - Column alias
    - Column position in the SELECT statement (not recommended)

ORDER BY Clause in SQL (cont.)

- ORDER BY Clause
  - Sort can be ascending (ASC, the default) or descending (DESC)
  - Sort can apply to multiple columns
  - By default, nulls sort high

ORDER BY Query Examples

- List job and name of employees in department 30, order by job in ascending, and then by name in descending order

```
SQL> SELECT job, ename
  2  FROM emp
  3  WHERE deptno = 30
  4  ORDER BY job, ename DESC;

SQL> SELECT job, ename
  2  FROM emp
  3  WHERE deptno = 30
  4  ORDER BY job, 2 DESC;

JOB       ENAME
--------- ----------
CLERK     JAMES
MANAGER   BLAKE
SALESMAN  WARD
SALESMAN  TURNER
SALESMAN  MARTIN
SALESMAN  ALLEN
6 rows selected.
```

Aggregate Functions in SQL

- Aggregate functions for a set or sets of records
  - AVG – Computes the average value
  - SUM – Computes the total value
  - MIN – Finds the minimum value
  - MAX – Finds the maximum value
  - COUNT – Counts a number of occurrences in a set

Aggregate Functions Query Examples

- Calculate the average salary for clerks

```
SQL> SELECT AVG(sal)
  2  FROM emp
  3  WHERE job = 'CLERK';

 AVG(SAL)
---------
   1037.5
```

- List the job, and the number of employees, and yearly salary of jobs where more than 2 employees are employed

```
SQL> SELECT JOB, COUNT(*), AVG(sal)*12
  2  FROM emp
  3  GROUP BY job
  4  HAVING COUNT(*) > 2;

JOB         COUNT(*) AVG(SAL)*12
--------- --------- -----------
CLERK             4       12450
MANAGER           3       33100
SALESMAN          4       16800
```

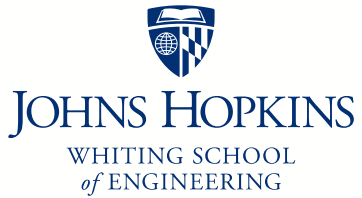
Johns Hopkins Engineering

Principles of Database Systems

Module 9 / Lecture 5

Structured Query Language (SQL)

The Relational DB Language I



UNION in SQL

- UNION (one of set operations) using SQL:
  - Union combines results of two queries. Union is a set of elements that is in one set, another set, or both sets.
  - We perform union queries when information comes from divergent sources.
  - UNION has no repeated rows.
  - UNION ALL has repeated rows.

INTERSECT in SQL

- Intersection (INTERSECT) Using SQL:
  - Intersection: what two query results have in common. Intersection of two sets represents all elements that are members of both sets.
  - Intersection is based on exact row matches.
  - It can be simulated with “EXISTS”.

EXCEPT in SQL

- EXCEPT (DIFFERENCE or MINUS) Using SQL:
  - What is in the first query result is not in the second query.
  - Elements are in the original set and not in the second set.
  - The results can be simulated with “NOT EXISTS”.

SQL SET Operator Query Examples

- Two tables INSTRUCTOR and STUDENT are union compatible. List all instructors (see Figure 6.4)

```
SQL> SELECT fname, lname FROM instructor;
FNAME                LNAME
-------------------- --------------------
John                 Smith
Ricardo              Browne
Susan                Yao
Francis              Johnson
Ramesh               Shah
```

- List all students

```
SQL> SELECT fname, lname FROM student;
FNAME                LNAME
-------------------- --------------------
Susan                Yao
Ramesh               Shah
Johnny               Kohler
Barbara              Jones
Amy                  Ford
Jimmy                Wang
Ernest               Gilbert
```

UNION and UNION ALL Query Examples

- List all names with instructors and students with UNION and UNION ALL

```
SQL> SELECT fname, lname FROM student
  2  UNION
  3  SELECT fname, lname FROM instructor;
FNAME                LNAME
-------------------- --------------------
Amy                  Ford
Barbara              Jones
Ernest               Gilbert
Francis              Johnson
Jimmy                Wang
John                 Smith
Johnny               Kohler
Ramesh               Shah
Ricardo              Browne
Susan                Yao
10 rows selected.
```

```
SQL> SELECT fname, lname FROM student
  2  UNION ALL
  3  SELECT fname, lname FROM instructor;
FNAME                LNAME
-------------------- --------------------
Amy                  Ford
Barbara              Jones
Ernest               Gilbert
Francis              Johnson
Jimmy                Wang
John                 Smith
Johnny               Kohler
Ramesh               Shah
Ricardo              Browne
Susan                Yao
Ramesh               Shah
Susan                Yao
12 rows selected.
```

INTERSECT (MINUS) Query Examples

- List all names that are both students and instructors

```
SQL> SELECT fname, lname FROM student
  2  INTERSECT
  3  SELECT fname, lname FROM instructor;

FNAME                LNAME
-------------------- --------------------
Ramesh               Shah
Susan                Yao
```

Note: EXCEPT is ISO standard; MINUS is Oracle implementation.

- List all names that are students who are not instructors

```
SQL> SELECT fname, lname FROM STUDENT
  2  MINUS
  3  SELECT fname, lname FROM INSTRUCTOR;

FNAME                LNAME
-------------------- --------------------
Amy                  Ford
Barbara              Jones
Ernest               Gilbert
Jimmy                Wang
Johnny               Kohler
```

INTERSECT (MINUS) Query Examples (cont.)

- List all names that are instructors who are not students

```
SQL> SELECT fname, lname FROM instructor
  2  MINUS
  3  SELECT fname, lname FROM student;

FNAME                LNAME
-------------------- --------------------
Francis              Johnson
John                 Smith
Ricardo              Browne
```

Note: EXCEPT is ISO standard; MINUS is Oracle implementation.

Column Prefix in SQL

- Column Prefix:
  - Used to identify the table to which a column belongs

    Syntax: table.column

    Example: emp.ssn; dept.dept_name
  - Required when two tables have an identical column name (e.g., PK and FK) and one of the columns is referenced in one SQL statement

    Example:

```
SQL> SELECT emp.name, dept.deptno, dept.name
  2  FROM emp, dept;
```

Alias for Table and Column in SQL

- Alias:
  - Used to rename an object within the SQL
  - Two types:
    - Column alias – rename long or cryptic column names
    - Table alias – rename long table names
  - Each column or table may be followed by an alias
  - Syntax:

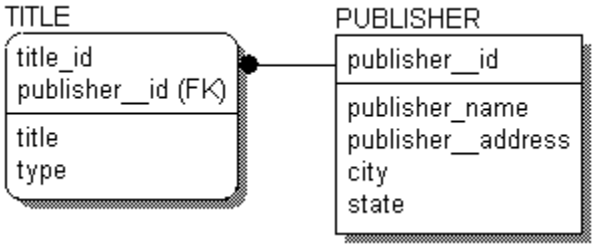
    SELECT col_name1 **[AS]** col_alias1, col_name1 **[AS]** col_alias1 …
    FROM table1 table_alias1, table1 table_alias1…;

Subquery in SQL

- **Subquery:**
  - a SELECT statement embedded within another SELECT statement
  - **Noncorrelated subquery**: evaluate from the inside out. The outer query takes an action based on the results of the inner query.

Example: Retrieve publishers who publish the ‘education’ type titles

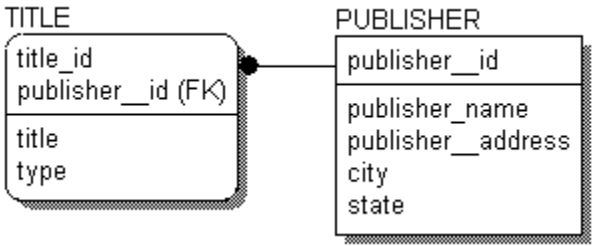
```
SELECT publisher_name
FROM publisher
WHERE publisher_id IN (SELECT UNIQUE publisher_id
                       FROM title
                       WHERE type = ‘education’);
```



Subquery in SQL (cont.)

- **Subquery (cont.):**
  - **Correlated subquery**: The outer query provides the values for the inner subquery to use in its evaluation.

    Example: Retrieve publishers who publish the ‘education’ type titles




```
SELECT publisher_name
FROM publisher p
WHERE EXISTS (SELECT *
              FROM title
              WHERE title.publisher_id = p.publisher_id AND
                    type = ‘education’);
```

  - EXISTS for existence is the connector for most correlated subqueries.

Subquery in SQL (cont.)

- **Subquery (cont.):**
  - Three main types of subquery connections:
    - May return a single value
      - Use a single comparison operator (<, <=, <>, >, >=)
    - May return zero or any number of items
      - Use IN, NOT IN, or with a comparison operator with ANY or ALL
      - E.g., > ANY; > ALL
    - May test existence or nonexistence
      - EXSITS, NOT EXISTS

Subquery Examples

- **Subquery (cont.):**
  - Subquery returns only one value.
  - List all employees whose job is the same as JONES

```
SQL> SELECT ename, job
  2  FROM emp
  3  WHERE job =
  4                (SELECT job
  5                 FROM emp
  6                 WHERE ename = 'JONES');

ENAME      JOB
---------- ---------
JONES      MANAGER
BLAKE      MANAGER
CLARK      MANAGER

3 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - Subquery returns a set of values.
  - List employees who earn more money than any single employee in department 30;

```
SQL> SELECT DISTINCT sal, job, ename, deptno
  2  FROM emp
  3  WHERE sal > ANY
  4          (SELECT sal
  5           FROM emp
  6           WHERE deptno = 30)
  7  ORDER BY sal DESC;
       SAL JOB       ENAME          DEPTNO
---------- --------- ---------- ---------
      5000 PRESIDENT KING               10
      3000 ANALYST   FORD               20
      3000 ANALYST   SCOTT              20
      2975 MANAGER   JONES              20
      2850 MANAGER   BLAKE              30
      2450 MANAGER   CLARK              10
      1600 SALESMAN  ALLEN              30
      1500 SALESMAN  TURNER             30
      1300 CLERK     MILLER             10
      1250 SALESMAN  MARTIN             30
      1250 SALESMAN  WARD               30
      1100 CLERK     ADAMS              20
12 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - Subquery returns more than one column.
  - List the employees whose job and salary are identical to that of FORD;

```
SQL> SELECT ename, job, sal
  2  FROM emp
  3  WHERE (job, sal) =
  4        (SELECT job, sal
  5         FROM emp
  6         WHERE ename = 'FORD');

ENAME      JOB              SAL
---------- --------- ---------
SCOTT      ANALYST         3000
FORD       ANALYST         3000

2 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - Compound query has multiple subqueries.
  - List the name, job, department number and salary of employees whose job is the same as JONES' or whose salary is at least as much as FORD's

```
SQL> SELECT ename, job, deptno, sal
  2  FROM emp
  3  WHERE job =
  4        (SELECT job
  5         FROM emp
  6         WHERE ename = 'JONES')
  7       OR sal > =
  8       (SELECT sal
  9        FROM emp
 10        WHERE ename = 'FORD')
 11  ORDER BY job, sal;
ENAME      JOB          DEPTNO       SAL
---------- --------- --------- ---------
SCOTT      ANALYST          20      3000
FORD       ANALYST          20      3000
CLARK      MANAGER          10      2450
BLAKE      MANAGER          30      2850
JONES      MANAGER          20      2975
KING       PRESIDENT        10      5000
6 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - List select name and job of employee in department 10 whose job is the same as any employee in department sales

```
SQL> SELECT ename, job
  2  FROM emp
  3  WHERE deptno = 10
  4     AND job IN
  5        (SELECT job
  6         FROM emp
  7         WHERE deptno IN
  8               (SELECT deptno
  9                FROM dept
 10                WHERE dname = 'SALES'));

ENAME      JOB
---------- ---------
MILLER     CLERK
CLARK      MANAGER

2 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - List the employees who work in CHIGAGO and who have the same job as ALLEN

```
SQL> SELECT ename, loc, sal, job
  2  FROM emp, dept
  3  WHERE loc = 'CHICAGO'
  4    AND emp.deptno = dept.deptno
  5    AND job IN
  6        (SELECT job FROM emp
  7         WHERE ename = 'ALLEN')
  8  ORDER BY ename;

ENAME      LOC                 SAL JOB
---------- ------------- --------- --------
-
ALLEN      CHICAGO            1600 SALESMAN
MARTIN     CHICAGO            1250 SALESMAN
TURNER     CHICAGO            1500 SALESMAN
WARD       CHICAGO            1250 SALESMAN

4 rows selected.
```

Subquery Examples (cont.)

- **Subquery (cont.):**
  - Correlated query
  - List the department number employee name and salary of all employees whose salary is more than the average salary of the department they work in

```
SQL> SELECT deptno, ename, sal
  2  FROM emp X
  3  WHERE sal >
  4        (SELECT AVG(sal)
  5         FROM emp
  6         WHERE X.deptno = deptno)
  7  ORDER BY deptno;
    DEPTNO ENAME             SAL
---------- ---------- ---------
        10 KING             5000
        20 JONES            2975
        20 SCOTT            3000
        20 FORD             3000
        30 ALLEN            1600
        30 BLAKE            2850
6 rows selected.
```