Introduction to SQL

Data Infrastructure IRAP Training
6/27/2016
Agenda

- UCDW Database Models
- Integrity Constraints
- Training Database
- SQL Defined
  - Types of SQL Languages
- SQL Basics
- Simple SELECT
  - SELECT with Aliases
  - SELECT with Conditions/Rules
  - SELECT with Comparison Operators – Not equal to, Equal to, Less than, Greater than, Less than or equal to, Greater than or equal to
  - SELECT with Compound Conditions – AND, OR, IN, NOT, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE, EXISTS, NOT EXISTS Operators
  - SELECT with Group By, Order By and Having Clauses
  - SELECT with Concatenated Fields
- Joins
  - Inner Join
  - Left Join
  - Right Join
  - Full Join
UCDW Database Models

Relational and Dimensional Models
UCDW Database Models

- Database Models
  - Data Model
  - Data Structure
  - Integrity Constraints

- UCDW Database Models
  - Relational Data Modeling
    - Relational Structure
    - UCDW Base Layer
  - Dimensional Data Modeling
    - Dimensional Structure
    - UCDW BI (Business Intelligence) Layer
Relational vs. Dimensional Models

- Data is stored in **relational** database system
- **Several tables** and chains of relationships between them
- **Volatile**
- Data is **normalized**
- **Detailed** level of transactional data

- Data is stored in **multi-dimensional** databases
- **Few facts** are connected to dimension tables
- **Non-volatile**
- Data is **de-normalized**
- **Summarized** transactional data (**aggregates** and **measures**) used in business decisions
Relational Databases

- Collection of **tables** and **relationships**
- Tables contain **rows** and **columns** or **attributes**
- Includes **integrity constraints**
- Includes **domains** – set of possible values for a given attribute

**Relational Data Structure**
Integrity Constraints

Constraints in UCDW
Ensures That:
- Data **conforms** to guidelines specified by the Database Architect
- Data is **consistent** and **correct**
- Queries are **optimized**
- **Performance** is adequate

**Constraint Types**
- Unique
- Primary Key
- Foreign Key
- Check
- Not Null
Unique Constraint

- Used to **enforce uniqueness** of a column or a combination of columns that is **not** the primary key
- Ensures that **all values in a column are different**
- Uniquely identifies each record in a table
- **Does not repeat**
- Multiple unique key constraints can be applied per table
- Unique constraints **allows NULL** values
- Example – SSN – unique constraint enforced in **STUDENT_D** dimension table
Primary Key Constraint

- **Uniquely identifies** a record/row in a table
- Ensures **all values in a column are different**
- Automatically has a unique constraint
- **Does not repeat**
- Only one per table
- Could be **natural** or **surrogate**
- Could be **composite** (made up of more than one column/attribute)
- Primary key constraints do **not allow** NULL values
- Example – **AWRD_KEY** – primary key constraint on the **AWARD_D** dimension table
Foreign Key Constraint

- Used to link two tables
- Refers to the primary key in another table
- Table containing a foreign key is called the child table
- Table containing the primary key is called the parent table
- Prevents actions that will violate relationship between tables
- Ensures that only valid data is inserted in child table
- Example – ACAD_SUB_T_KEY, STUD_KEY, CRSE_KEY and CRSE_ENRL_STAT_KEY are all foreign keys in the COURSE_ENROLLMENT_F fact table.
Check Constraint

- Used to **limit the values** that can be placed in a column
- Allowable values are **defined from a logical expression**
- Defined on a single column means only certain values are allowed
- Defined on a table **means values in certain columns must be based on values in other columns in the row**
- Example – **STUD_IPEDS_GNDR_CD** within **STUDENT_D** can only have values (F, M)

**CHECK Constraint Example**

- This CHECK constraint ensures that a value entered for end_date is later than start_date.

```sql
CREATE TABLE copy_job_history
(employee_id NUMBER(6,0),
 start_date DATE,
 end_date DATE,
 job_id VARCHAR2(10),
 department_id NUMBER(4,0),
 CONSTRAINT cjhist_emp_id_at_date_pk
 PRIMARY KEY(employee_id, start_date),
 CONSTRAINT cjhist_end ck CHECK (end_date > start_date));
```

- As this CHECK CONSTRAINT is referencing two columns in the table, it MUST be defined at table level.
Not Null Constraint

- Requires that **every row** has a value for the **NOT NULL** column
- **Enforces** a field to always contain a value
- **Example** – `STUD_FST_NAM` and `STUD_LST_NAME` cannot be NULL in the `STUDENT_D` dimension table

Table EMP

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>JOB</th>
<th>MGR</th>
<th>HIREDATE</th>
<th>SAL</th>
<th>COMM</th>
<th>DEPTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7329</td>
<td>SMITH</td>
<td>CEO</td>
<td>7529</td>
<td>17-DEC-85</td>
<td>9,000.00</td>
<td>100.00</td>
<td>20</td>
</tr>
<tr>
<td>7599</td>
<td>ALLEN</td>
<td>VP_SALES</td>
<td>7529</td>
<td>20-FEB-85</td>
<td>7,500.00</td>
<td>100.00</td>
<td>30</td>
</tr>
<tr>
<td>7521</td>
<td>WARD</td>
<td>MANAGER</td>
<td>7459</td>
<td>22-FEB-85</td>
<td>5,500.00</td>
<td>200.00</td>
<td>30</td>
</tr>
<tr>
<td>7566</td>
<td>JONES</td>
<td>SALESMAN</td>
<td>7521</td>
<td>02-APR-90</td>
<td>2,975.00</td>
<td>400.00</td>
<td>30</td>
</tr>
</tbody>
</table>

NOT NULL Constraint (no row may contain a null value for this column)

Absence of NOT NULL Constraint (any row can contain null for this column)
Training Database

Subset of UCDW Dimensions and Facts for Training
Training Database (Enrollment)

ACADEMIC_SUBTERM_D

STUDENTLEVEL_D

ENROLLMENTHEAD_COUNT_M

STUDENT_D

ENROLLMENTSTATUS_D

ACAD_SUB_T_KEY

STUD_KEY

STUD_LVL_KEY

ENRL_STAT_KEY

ACAD_SUB_T_KEY

STUD_KEY

STUD_LVL_KEY

ENRL_STAT_KEY

STUD_LVL_KEY

ENRL_STAT_KEY
Training Database (Degree)
Training Database: Financial Aid

STUDENT_D
- STUD_KEY
- CAMPUS_LOCATION_D
  - CAMPUS_LOC_KEY
FUND_D
- FD_KEY
AWARD_D
- AWRD_KEY
STUDENT_EARNINGS_F
- STUD_KEY
- AWARD_KEY
- CAMPUS_LOC_KEY
- FD_KEY
- ACCT_KEY
- FILE_CYCLE_KEY
ACCOUNT_D
- ACCT_KEY
FILE_CYCLE_D
- FILE_CYCLE_KEY
**SQL Languages**

- **Data Definition Language (DDL)**
  - Standard for commands that define the different structures in a database
  - Includes `CREATE`, `ALTER` and `DROP` commands
  - Used by Data Architects and Database Administrators

- **Data Manipulation Language (DML)**
  - Standards for commands that manipulate data in a database
  - Includes `SELECT`, `INSERT`, `UPDATE`, and `DELETE`
  - Used by IT and Business Users to manipulate and extract data
SQL Basics
Basic SQL Statements
Basics of SQL

- **SQL** – Structured Query Language
- **Create** – create a data structure *
- **Select** – read one or more rows from a table
- **Insert** – add one or more rows to a table *
- **Delete** – remove one or more rows from a table *
- **Update** – change the value of one or more fields in a row or within a table *
- **Drop** – remove a data structure *
Basics of a Simple SELECT

You are asking for the location, student ID, first name, last name, date of birth, gender and current active flag of students.

You want the data from the STUDENT_D dimension table.

```
FROM STUD_BI.STUDENT_D
```

To get all columns from a table:
```
SELECT *
FROM STUD_BI.STUDENT_D
```
SELECT STUD_LOC_CMP_CD as Campus_Location,
STUD_ID as Student_Identification_Number,
STUD_FST_NAM as First_Name,
STUD_LST_NAM as Last_Name,
STUD_DT_OF_BTH as Date_of_Birth,
STUD_GNDR_CD as Gender_Code,
STUD_GNDR_DESC as Gender_Description,
STUD_CUR_ACTV_FL as Current_Active_Flag
FROM STUD_BI.STUDENT_D
You are asking for the location, student ID, first name, last name, date of birth, gender and current active flag of students.

You want the data from the STUDENT_D dimension table.

You have a condition – the current active flag must be set to ‘Y’.


**FROM** STUD_BI.STUDENT_D

**WHERE** STUD_CUR_ACTV_FL = ‘Y’

The **WHERE** clause evaluates to true or false.

Comparison operators include:
- <> or !!= Not Equal To
- = Equal to
- < Less than
- > Greater than
- <= or !> Less than or equal to (or not greater than)
- >= or !< Greater than or equal to (or not less than)

The AND operator joins two or more conditions. Returned rows must meet all conditions.
You are asking for the location, student ID, first name, last name, date of birth, gender and current active flag of students.

You want the data from the STUDENT_D dimension table.

You have three conditions - (1) the current active flag must be set to ‘Y’, (2) the location must be ‘01 – Berkeley’ and (3) whether student is domestic or foreign.

```
SELECT STUD_LOC_CMP_CD,
STUD_ID,
STUD_FST_NAM,
STUD_LST_NAM,
STUD_DT_OF_BTH,
STUD_GNDR_CD,
STUD_GNDR_DESC,
STUD_CUR_ACTV_FL
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_DMSTC_FGN_CZ_STAT_CD = 'F'
```
Logical Operators

AND, OR, NOT, IN, BETWEEN, LIKE, EXISTS
SELECT with **AND & OR** Operators

**SELECT**
- STUD_LOC_CMP_CD,
- STUD_ID,
- STUD_FST_NAM,
- STUD_LST_NAM,
- STUD_DT_OF_BTH,
- STUD_GNDR_CD,
- STUD_GNDR_DESC,
- STUD_CUR_ACTV_FL

**FROM**
- STUD_BI.STUDENT_D

**WHERE**
- STUD_CUR_ACTV_FL = 'Y'
- STUD_LOC_CMP_CD IN ('01', '04')
- STUD_DMSTC_FGN_CZ_STAT_CD = 'F'
- ((STUD_GNDR_CD = 'M' **OR** STUD_IPEDS_GNDR_CD = 'M')
- **OR** (STUD_GNDR_IDNTY_CD = 'M' **OR** STUD_GNDR_AT_BTH_CD = 'M'))

**AND mandates that all specified conditions must be met!**

**OR mandates that at least one condition must be met!**

The **OR** operator joins two or more conditions but returns a row when ANY of the conditions are met.
SELECT with **NOT** Operator

```sql
SELECT STUD_LOC_CMP_CD,
STUD_ID,
STUD_FST_NAM,
STUD_LST_NAM,
STUD_DT_OF_BTH,
STUD_GNDR_CD,
STUD_GNDR_DESC,
STUD_CUR_ACTV_FL
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD **NOT IN** ('01', '04', '06', '02', '07', '08')
AND STUD_DMSTC_FGN_CZ_STAT_CD = 'F'
AND ((STUD_GNDR_CD = 'M' **OR** STUD_IPEDS_GNDR_CD = 'M')
**OR** (STUD_GNDR_IDNTY_CD = 'M' **OR** STUD_GNDR_AT_BTH_CD = 'M'))
```

Sometimes it's easier to specify what you don't want by using the **NOT** operator.
You are asking for the location, student ID, first name, last name, date of birth, gender, citizenship status and current active flag of students.

You want the data from the STUDENT_D dimension table.

You have three conditions – (1) the current active flag must be set to ‘Y’, (2) the location must be ‘01 – Berkeley’ and (3) the citizenship status code has to one of a predefined set of values.

```
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_CZ_STAT_CD IN ('US', 'PR', 'RF', 'AM', 'AP', 'AS', 'DA', '')
```
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_DT_OF_BTH BETWEEN '01-08-1971' AND '01-08-1991'

You are asking for the location, student ID, first name, last name, date of birth, gender, citizenship status and current active flag of students

You want the data from the STUDENT_D dimension table

You have three conditions – (1) the current active flag must be set to ‘Y’, (2) the location must be ‘01 – Berkeley’ and (3) the date of birth is not between January 8th 1971 and January 8th 1991
You are asking for the location, student ID, first name, last name, date of birth, gender, citizenship status and current active flag of students.

You want the data from the STUDENT_D dimension table.

You have three conditions – (1) the current active flag must be set to ‘Y’, (2) the location must be ‘01 - Berkeley’ and (3) the date of birth is between January 8th 1971 and January 8th 1991.

FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_DT_OF_BTH NOT BETWEEN '01-08-1971' AND '01-08-1991'
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_LST_NAM LIKE ('DELM%')
You are asking for the location, student ID, first name, last name, date of birth, gender, citizenship status and current active flag of students.

You want the data from the STUDENT_D dimension table.

You have three conditions – (1) the current active flag must be set to ‘Y’, (2) the location must be ‘01 – Berkeley’ and (3) the last name does not start with the characters ‘DELM’.

```
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD_LOC_CMP_CD = '01'
AND STUD_LST_NAM NOT LIKE ('DELM%')
```
SELECT with **EXISTS** Operator

```
SELECT CMP_LOC_LOC1_CD,
      CMP_LOC_LOC1_SHRT_DESC,
      CMP_LOC_LOC1_LNG_DESC,
      CMP_LOC_LOC1_MXD_CASE_LNG_DESC,
      CMP_LOC_LOC1_ABRV_DESC
FROM  STUD_BI.CAMPUS_LOCATION_D
WHERE EXISTS ( SELECT STUD_LOC_CMP_CD
                FROM   STUD_BI.STUDENT_D
                WHERE STUD_CUR_ACTV_FL = 'Y'
                      AND STUD_LOC_CMP_CD IN ('01', '03', '05', '07', '09')
                      AND STUD_BI.CAMPUS_LOCATION_D.CMP_LOC_LOC1_CD
                           = STUD_BI.STUDENT_D.STUD_LOC_CMP_CD)
ORDER BY CMP_LOC_LOC1_CD
```
SELECT with **NOT EXISTS** Operator

```
SELECT CMP_LOC_LOC1_CD, 
    CMP_LOC_LOC1_SHRT_DESC, 
    CMP_LOC_LOC1_LNG_DESC, 
    CMP_LOC_LOC1_MXD_CASE_LNG_DESC, 
    CMP_LOC_LOC1_ABRV_DESC
FROM STUD_BI.CAMPUS_LOCATION_D
WHERE NOT EXISTS (SELECT STUD_LOC_CMP_CD, 
    FROM STUD_BI.STUDENT_D 
    WHERE STUD_CUR_ACTV_FL = 'Y' 
    AND STUD_LOC_CMP_CD IN ('01', 
    '03', '05', '07', '09')
AND STUD_BI.CAMPUS_LOCATION_D.CMP_LOC_LOC1_CD 
    = STUD_BI.STUDENT_D.STUD_LOC_CMP_CD)
ORDER BY CMP_LOC_LOC1_CD
```
You are asking for the location, gender, citizenship status (domestic or foreign) and a count of students.

You want the data from the STUDENT_D dimension table.

You have one condition – the current active flag must be set to ‘Y’.

Because you have a group/aggregate function (COUNT), you must include a GROUP BY clause to group the result-set.

```
SELECT STUD_LOC_CMP_CD, STUD_GNDR_DESC, STUD_DMSTC_FGN_CZ_STAT_CD, COUNT(DISTINCT STUD_ID) as Student_Count
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
GROUP BY STUD_LOC_CMP_CD, STUD_GNDR_DESC, STUD_DMSTC_FGN_CZ_STAT_CD
```
SELECT STUD_LOC_CMP_CD, STUD_GNDR_DESC, STUD_DMSTC_FGN_CZ_STAT_CD, COUNT (DISTINCT STUD_ID) as Student_Count
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
GROUP BY STUD_LOC_CMP_CD, STUD_GNDR_DESC, STUD_DMSTC_FGN_CZ_STAT_CD
ORDER BY STUD_LOC_CMP_CD DESC

You are asking for the location, gender, citizenship status (domestic or foreign) and a count of students.

You want the data from the STUDENT_D dimension table.

You have one condition – the current active flag must be set to ‘Y’.

You have a group by clause because of the aggregate function COUNT.

You have an order by clause to sort the results using campus location in descending order.
You are asking for a count of students by campus location. The DISTINCT keyword eliminates duplicates.

You want the data from the STUDENT_D dimension table.

You have one condition - the current active flag must be set to ‘Y’.

You have a group by clause because of the aggregate function COUNT.

Because you have an aggregate function, you need a HAVING clause for your condition.

You have an ORDER BY clause to sort the results using campus location in ascending order.
SELECT STUD_LOC_CMP_CD || STUD_ID as Student_Identification_Number,
STUD_LST_NAM || ',' || ' ' || STUD_FST_NAM as Student_Name,
STUD_DT_OF_BTH,
STUD_GNDR_CD,
STUD_GNDR_DESC,
STUD_CZ_STAT_CD,
STUD_CUR_ACTV_FL
FROM STUD_BI.STUDENT_D
WHERE STUD_CUR_ACTV_FL = 'Y'
AND STUD.LOC_CMP_CD = '01'
AND STUD_DT_OF_BTH BETWEEN '01-08-1971' AND '01-08-1991'

It may be necessary to concatenate campus location with student ID to join different content areas. For reporting purposes, you may also want to concatenate the last name with the first name to combine the fields into one.
Simple Joins

Joining Dimensions and Fact Tables
Joins

- **Left Join**
- **Right Join**
- **Inner Join**
- **Full Join**
Inner Joins

- An inner join returns all records at the intersection of table A and table B
Next Step: Intermediate SQL

- Joins
- Sub-Queries
- Views
- Inline View
- UNION, UNION ALL, INTERSECT, MINUS
- WITH
- CASE Statements
- DECODE Function
- Aggregate Functions
- Date Functions
- Character Functions