SQL Basics

Introduction to
Standard Query Language

SQL – What Is It?

- Structured Query Language
- Common Language For Variety of Databases
- ANSI Standard BUT....
- Two Types of SQL
  - DML – Data Manipulation Language (SELECT)
  - DDL – Data Definition Language (CREATE TABLE)

Where To Use

- SQL*Plus
- TOAD
- SQL Navigator
- ODBC Supported Connections
  - Excel
  - Access
  - Lotus 1-2-3
- Heart of PL/SQL

Pros & Cons of SQL

- Pros:
  - Very flexible
  - Universal (Oracle, Access, Paradox, etc)
  - Relatively Few Commands to Learn
- Cons:
  - Requires Detailed Knowledge of the Structure of the Database
  - Can Provide Misleading Results

Basic SQL Components

- SELECT *schema.table.*column
- FROM table *alias*
- *WHERE [conditions]*
- *ORDER BY [columns]*
- *;*
  - Defines the end of an SQL statement
  - Some programs require it, some do not (TOAD Does Not)
  - Needed only if multiple SQL statements run in a script

Optional Elements

SELECT Statement

- SELECT Statement Defines WHAT is to be returned (separated by commas)
  - Database Columns (From Tables or Views)
  - Constant Text Values
  - Formulas
  - Pre-defined Functions
  - Group Functions (COUNT, SUM, MAX, MIN, AVG)
- “*” Mean All Columns From All Tables In the FROM Statement
- Example: SELECT state_code, state_name

FROM Statement

- Defines the Table(s) or View(s) Used by the SELECT or WHERE Statements
- You MUST Have a FROM statement
- Multiple Tables/Views are separated by Commas

Examples

- SELECT state_name, state_abbr
  FROM states
- SELECT *
  FROM agencies
- SELECT arithmetic_mean – minimum_value
  FROM annual_summaries

WHERE Clause

- Optional
- Defines what records are to be included in the query
- Uses Conditional Operators
  - =, >, >=, <, <=, != (<>)
  - BETWEEN x AND y
  - IN (*list*)
  - LIKE '%string' ("%" is a wild-card)
  - IS NULL
  - NOT {BETWEEN / IN / LIKE / NULL}
- Multiple Conditions Linked with AND & OR Statements
- Strings Contained Within SINGLE QUOTES!!

AND & OR

- Multiple WHERE conditions are Linked by AND / OR Statements
- “AND” Means All Conditions are TRUE for the Record
- “OR” Means at least 1 of the Conditions is TRUE
- You May Group Statements with ( )
- BE CAREFUL MIXING “AND” & “OR” Conditions

Examples with WHERE

- SELECT *
  FROM annual_summaries
  WHERE sd_duration_code = ‘1’
- SELECT state_name
  FROM states
  WHERE state_population > 15000000

More Examples

- SELECT state_name, state_population
  FROM states
  WHERE state_name LIKE '%NORTH%'
- SELECT *
  FROM annual_summaries
  WHERE sd_duration_code IN ('1', 'W', 'X')
  AND annual_summary_year = 2000

Be Careful!

- SELECT mo_mo_id, sd_duration_code
  FROM annual_summaries
  WHERE annual_summary_year = 2003
  AND values_gt_pri_std > 0
  OR values_gt_sec_std > 0
- SELECT mo_mo_id, sd_duration_code
  FROM annual_summaries
  WHERE annual_summary_year = 2003
  AND (values_gt_pri_std > 0
  OR values_gt_sec_std > 0)

ORDER BY Statement

- Defines How the Records are to be Sorted
- Must be in the SELECT statement to be ORDER BY
- Default is to order in ASC (Ascending) order
- Can Sort in Reverse (Descending) Order with “DESC” After the Column Name

ORDER BY Example

- SELECT *
  FROM agencies
  ORDER BY agency_desc
- SELECT cc_cn_stt_state_code, site_id
  FROM sites
  WHERE lut_land_use_type = ‘MOBILE’
  ORDER BY cc_cn_stt_state_code DESC

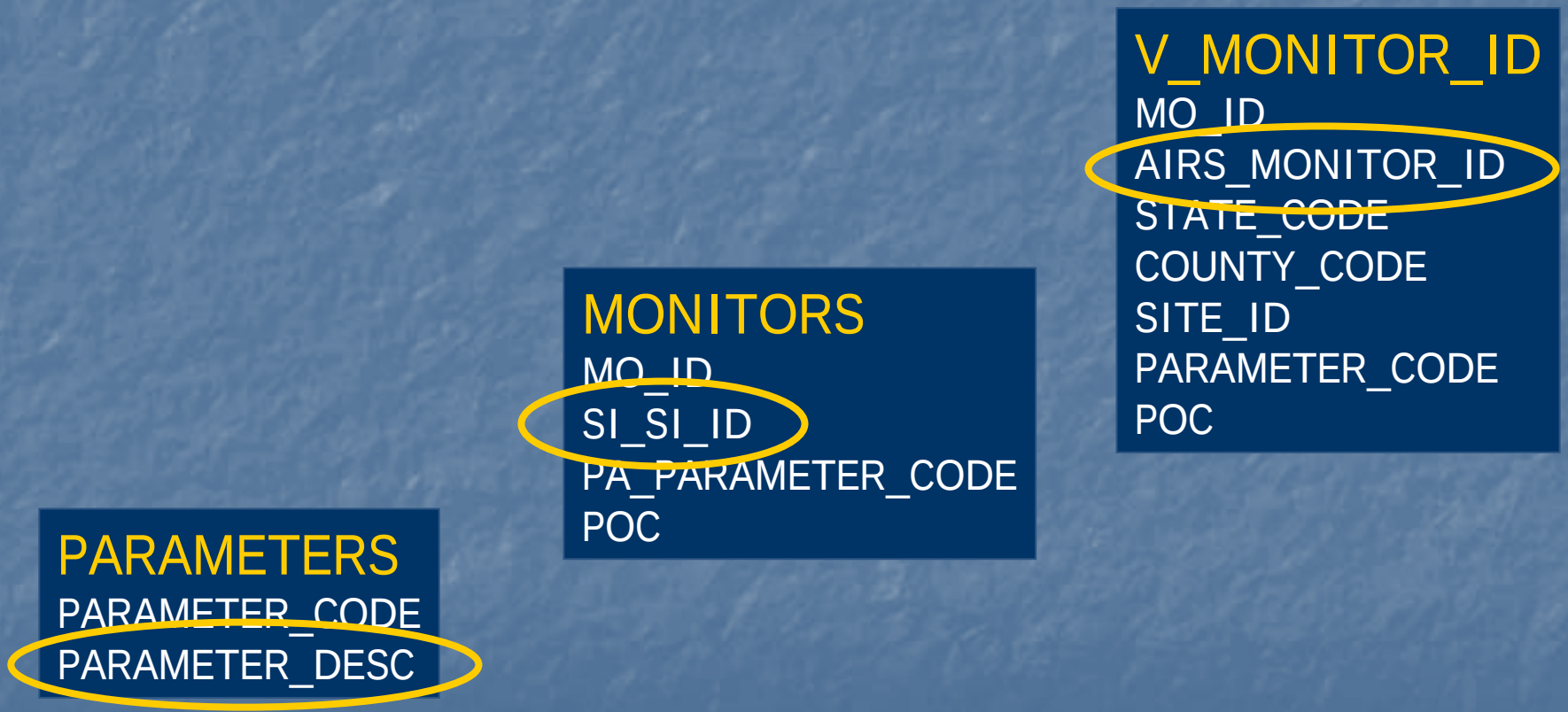
Group Functions

- Performs Common Mathematical Operations on a Group of Records
- Must define what Constitutes a Group by Using the GROUP BY Clause
- All non-Group elements in the SELECT Statement Must be in the GROUP BY Clause (Additional Columns are Optional)

Group By Example

- SELECT si_si_id, COUNT(mo_id)
  FROM monitors
  GROUP BY si_si_id
- SELECT AVG(max_sample_value)
  FROM summary_maximums
  WHERE max_level <= 3
  AND max_ind = ‘REG’
  GROUP BY ans_ans_id

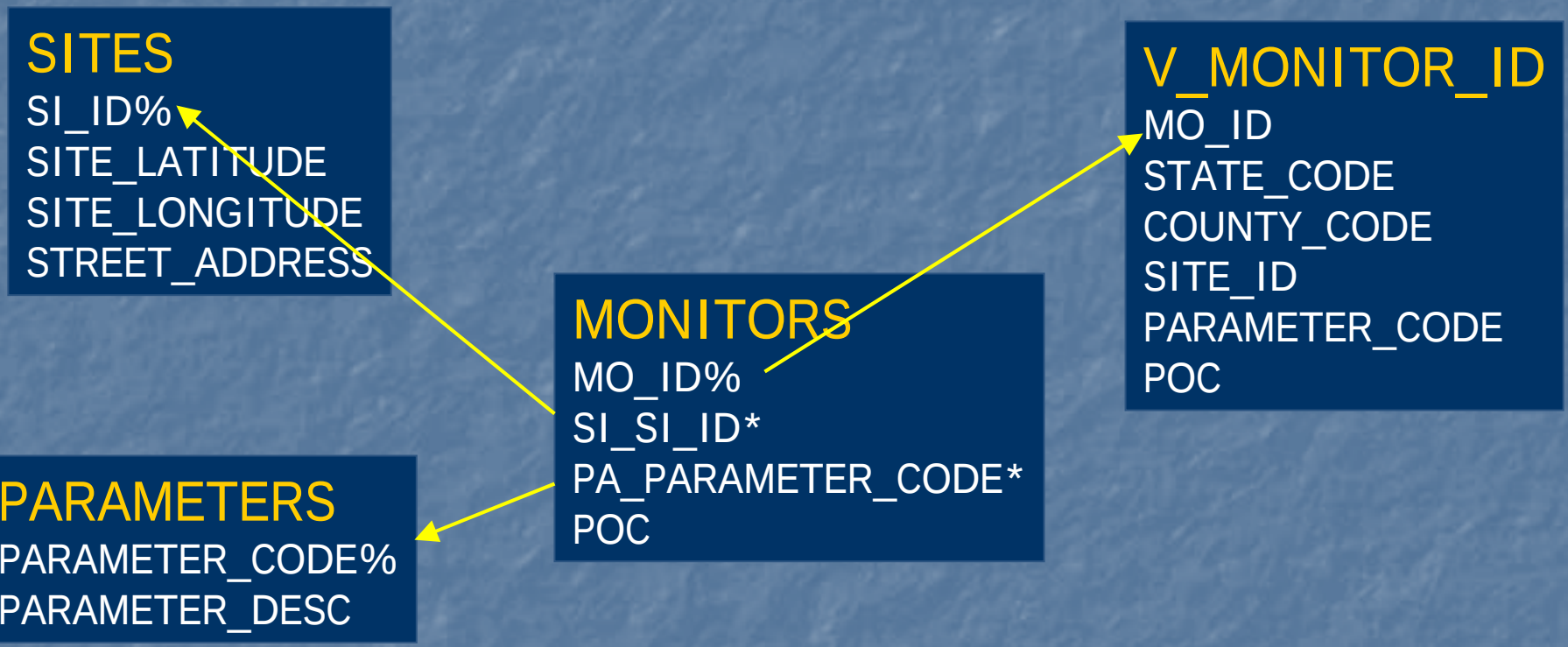
OK, I understand How to Get Data From 1 Table... What about Multiple Tables?



Primary & Foreign Keys

- Primary Keys
  - 1 or More Columns Used to Uniquely Identify a record.
  - All Columns Defined as PK's MUST be populated
- Foreign Keys
  - Value on a table that references a Primary Key from a different table

Primary & Foreign Keys



* = Foreign Key

% = Primary Key

Joining Tables

PARAMETERS	
Parameter_Code	Parameter_Desc
44201	Ozone
42101	CO
42401	SO2
81102	PM10

MONITORS			
MO_ID	SI_SI_ID	PA_PARAMETER_CODE	POC
1	1	44201	1
2	1	42101	1
3	1	42101	2
4	2	81102	1
5	2	44201	1
6	3	42401	1

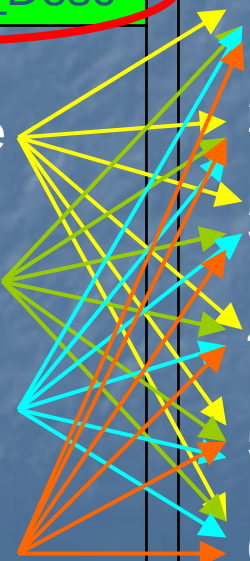
Default behavior is to show every possible combination between the two tables

Cartesian Join / Simple Join

SELECT mo_id, poc, parameter_desc
FROM monitors, parameters

PARAMETERS

Parameter_Code	Parameter_Desc
44201	Ozone
42101	CO
42401	SO2
81102	PM10

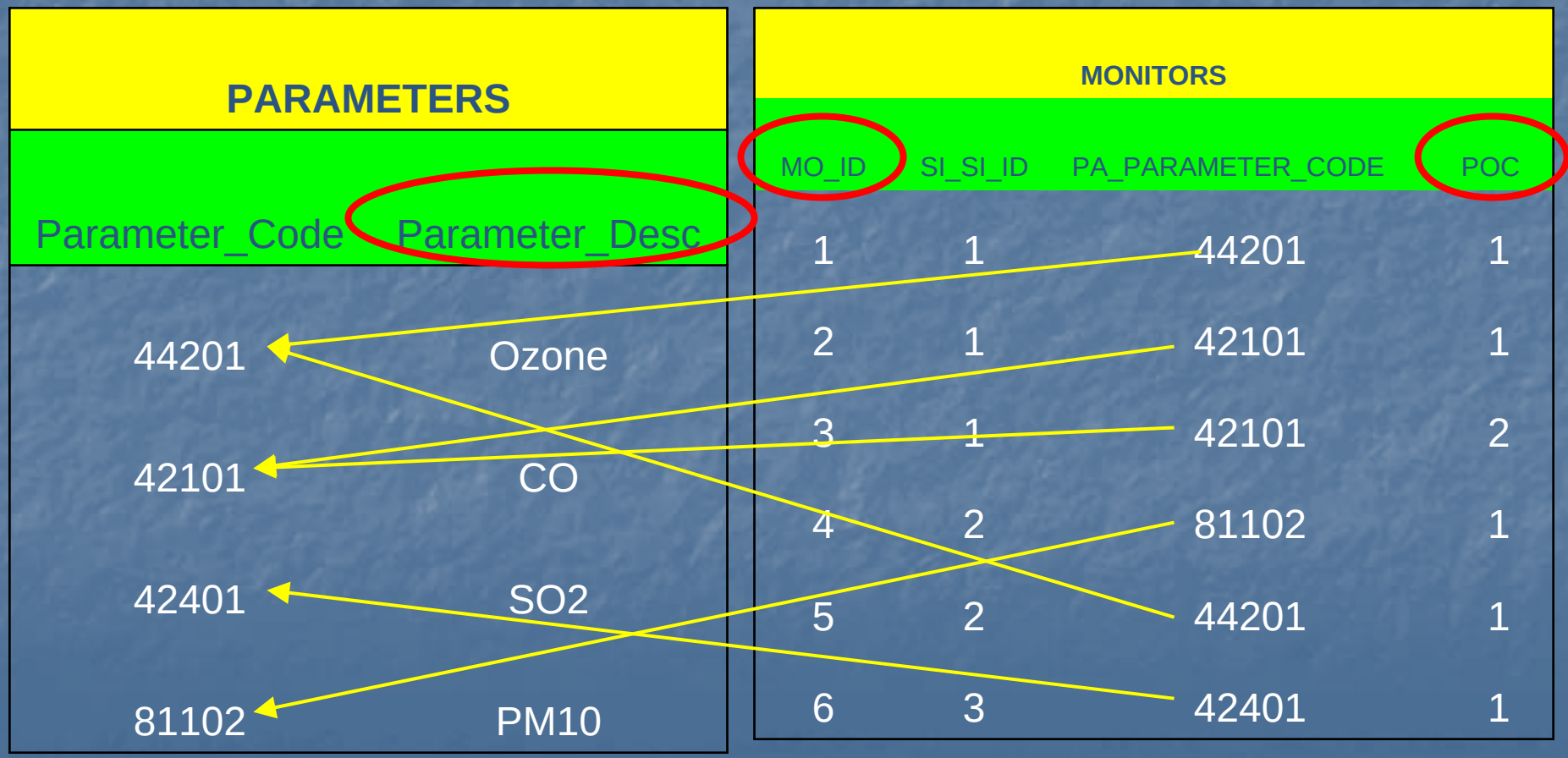


MONITORS

MO_ID	SI_SI_ID	PA_PARAMETER_CODE	POC
1	1	44201	1
2	1	42101	1
3	1	42101	2
4	2	81102	1
5	2	44201	1
6	3	42401	1

Joining Tables

```
SELECT mo_id, poc, parameter_desc
FROM monitors, parameters
WHERE pa_parameter_code = parameter_code
```



Joining Tables

- Joins Between Tables are Usually Based on Primary / Foreign Keys
- Make Sure Joins Between All Tables in the FROM Clause Exist
- List Joins Between Tables Before Other Selection Elements

Aliases

- “Shorthand” for Table or Column References
- SELECT Aliases Appear as Column Headers in the Output
- Aliases Cannot be Keywords

Previous SQL With Aliases

```
SELECT mo.mo_id, mo.poc, pa.parameter_desc parameter
FROM monitors mo, parameters pa
WHERE mo.pa_parameter_code = pa.parameter_code
```

Why Use an Alias?

- Saves Typing
- Good Internal Documentation
- Better Headers
- If the same column name exists on multiple tables, SQL needs a way to know which element you are referencing (MO_MO_ID for example)

Recap

- Basic Structural Elements
  - SELECT
  - FROM
  - WHERE
  - ORDER BY
  - GROUP BY
- Selecting From Multiple Tables
  - Join Multiple Tables via Primary & Foreign Keys
  - Aliases