**SQL-Part 2**

1. **Things to Remember about SQL**
2. **SQL keywords are case insensitive**

* **We try to CAPITALISE them to make them clear**
* **Improve readability of your statements**

1. **Table names are Operating System Sensitive**

* **If case sensitivity exists in the operating system, then the table names are case sensitive! (i.e. Linux, Unix)**
* **Account != ACCOUNT**

1. **Field names are case insensitive**

* **ACCOUNTID == AccountID == AcCoUnTID**

1. **You can do maths in SQL…**

* **SELECT 1\*1+1/1 1;**

1. **You can create your own columns that are not in the table**

* **SELECT '123459999' as MyID**

1. **Note On SELECT**
2. **The select statement’s job is just to return rows of data, it doesn’t care about the order of these rows unless you specify the ORDER BY clause**

**select语句的任务只是返回数据行，它不关心这些行的顺序，除非指定order BY子句**

1. **So what order do rows come out in if you don’t specify the ORDER BY clause?**

* **Any order**
* **Possibly the order the records were created in**
* **It is undefined**
* **Because SQL may optimise the query which may change the order of results…因为SQL可能会优化查询，这可能会改变结果的顺序…**

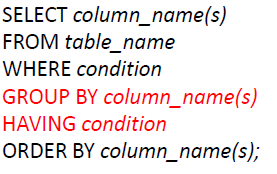
1. **So make sure you get into the habit of using the ORDER BY clause if you need a particular order**

**因此，如果需要特定的顺序，请确保养成使用ORDER BY子句的习惯**

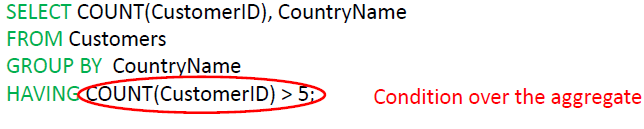
* **If you don’t need order, don’t use it it’s going to slow down the execution**

1. **HAVING Clause: Revisited**
2. **The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.**

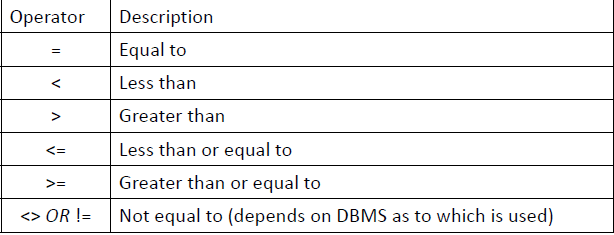
**在SQL中添加HAVING子句是因为WHERE关键字不能用于聚合函数。**

* ****

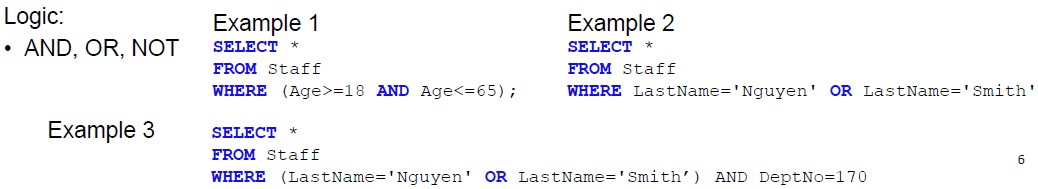
1. **Example**

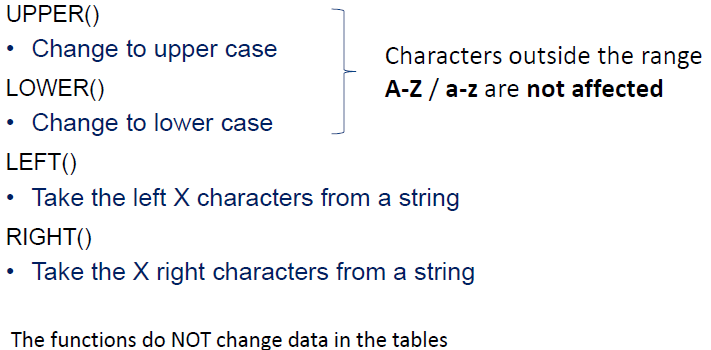
* **List the number of customers of each country, but ONLY include countries with more than 5 customers**
* ****

1. **Comparison and Logic Operators**
2. **Comparison:**

* ****

1. **Logic:**

* ****

1. **Some Useful String Functions**
2. ****
3. **Set Operations**
4. **UNION**

* **combine the results of two queries (or tables) into a single result set**

**将两个查询(或表)的结果合并到一个结果集中**

* **The number and data types of the columns selected by each component query must be the same, but the column lengths can be different**

**每个组件查询选择的列的数目和数据类型必须相同，但列的长度可以不同**

1. **INTERSECT**

* **Shows only rows that are common in the queries (or the tables)**

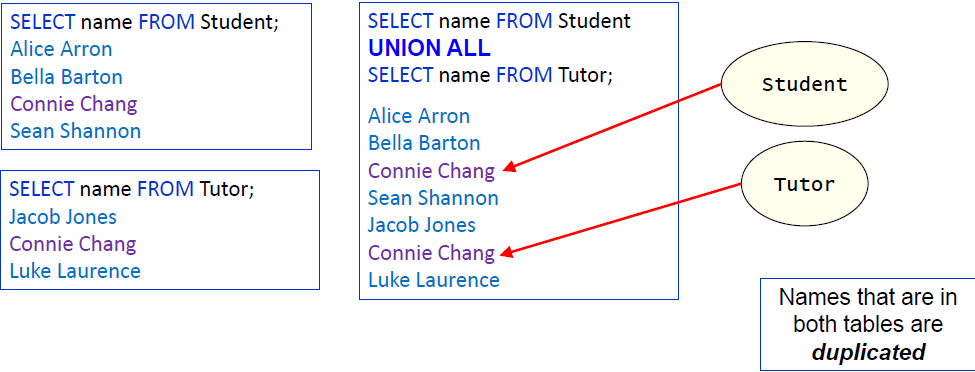
**只显示查询(或表)中常见的行**

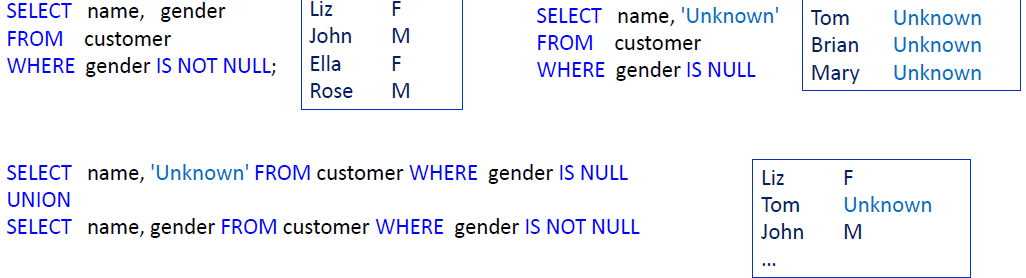
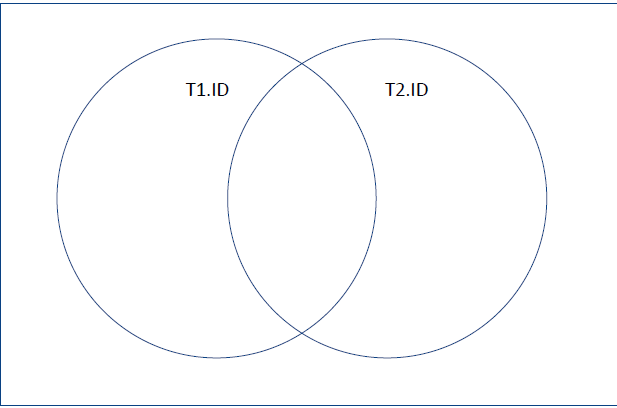
1. **[UNION/INTERSECT] ALL**

* **If you want to have duplicate rows in the result set you need to use the ALL keyword.. UNION ALL etc.**

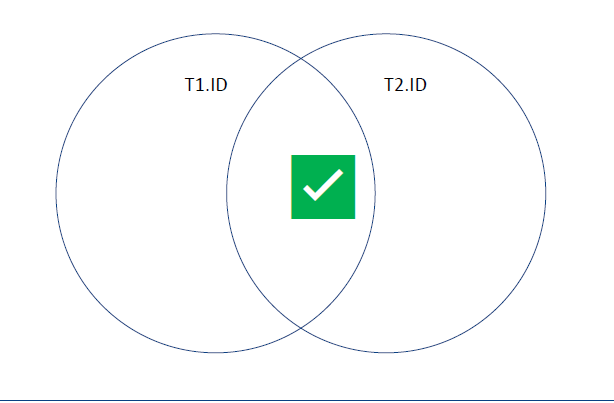
**如果你想在结果集中有重复的行，你需要使用ALL关键字。UNION ALL等等。**

1. **In MySQL only UNION and UNION ALL are supported**

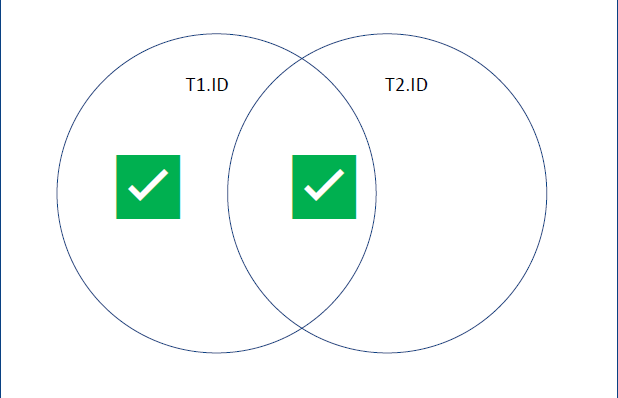
* **UNION ALL Example**
* **A Union ALL operator causes ALL rows to be added. Duplicates may occur.** **Union ALL操作符导致添加ALL行。可能发生重复。**
* ****
* **UNION Example**
* 

1. **SELECT with Literals选择与文字**
2. **When dealing with non matching columns between the tables, you may need to utilize literal values在处理表之间的非匹配列时，可能需要使用文字值**
3. **A literal value is a value 'hardcoded' into the query; the value is not generated from the table.** **文字值是一个“硬编码”到查询中的值;该值不是由表生成的。**
4. ****
5. **JOINS depicted as Venn Diagrams连接描述为维恩图**
6. ****
7. **T1 INNER JOIN T2 ON T1.ID = T2.ID**

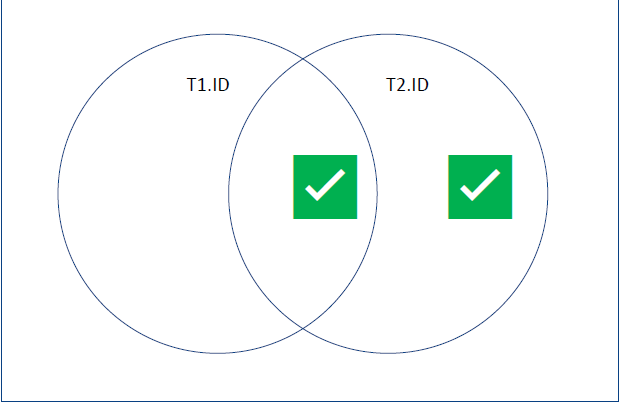
**T1 NATURAL JOIN T2**

****

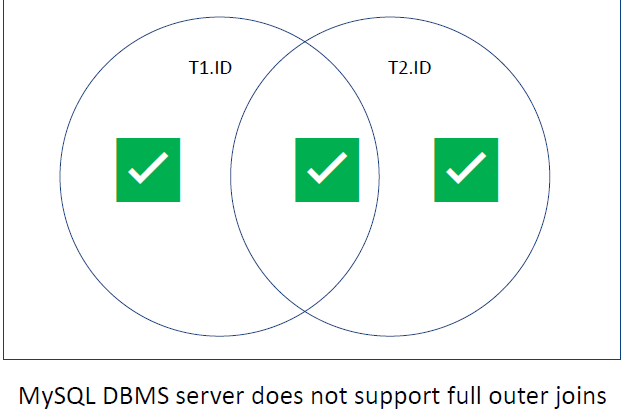
1. **T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID**

****

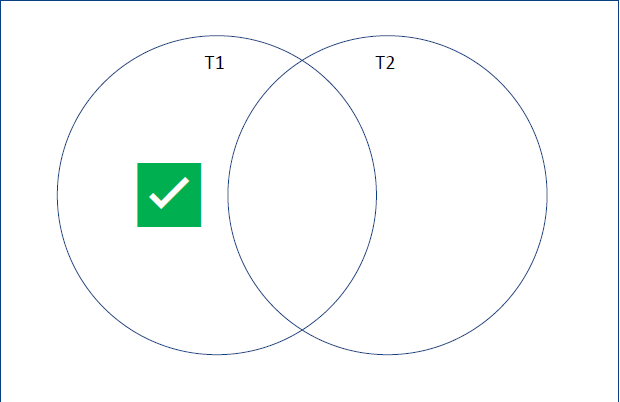
1. **T1 RIGHT OUTER JOIN T2 ON T1.ID = T2.ID**

****

1. **T1 FULL OUTER JOIN T2 ON T1.ID = T2.ID**

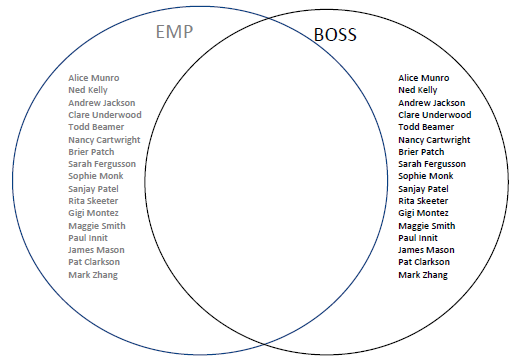
****

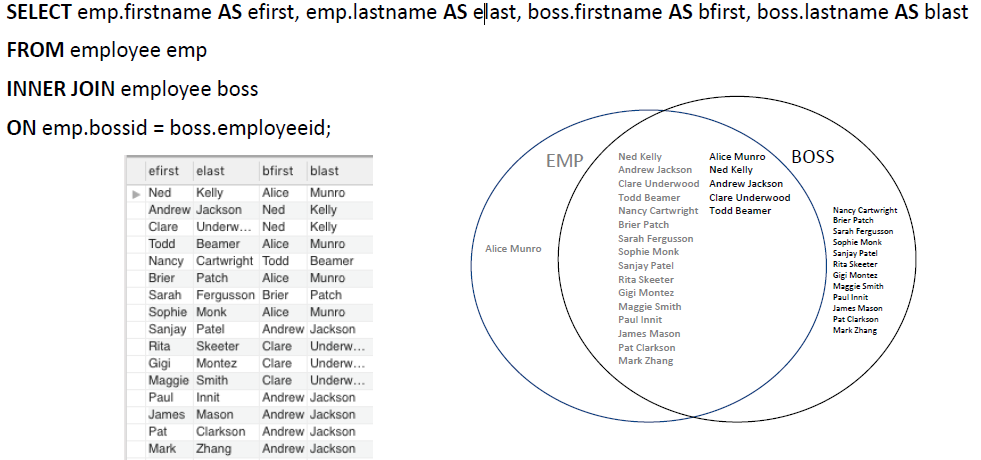
1. **T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID WHERE T2.ID IS NULL**

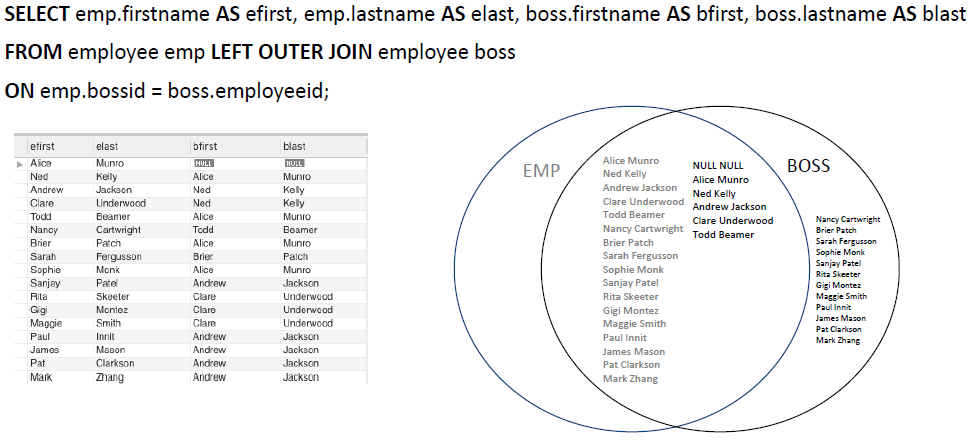
****

1. **INNER Join-Labs demo内部Join-Labs演示**
2. **Unary join of the Employee table aliased as EMP, BOSS**

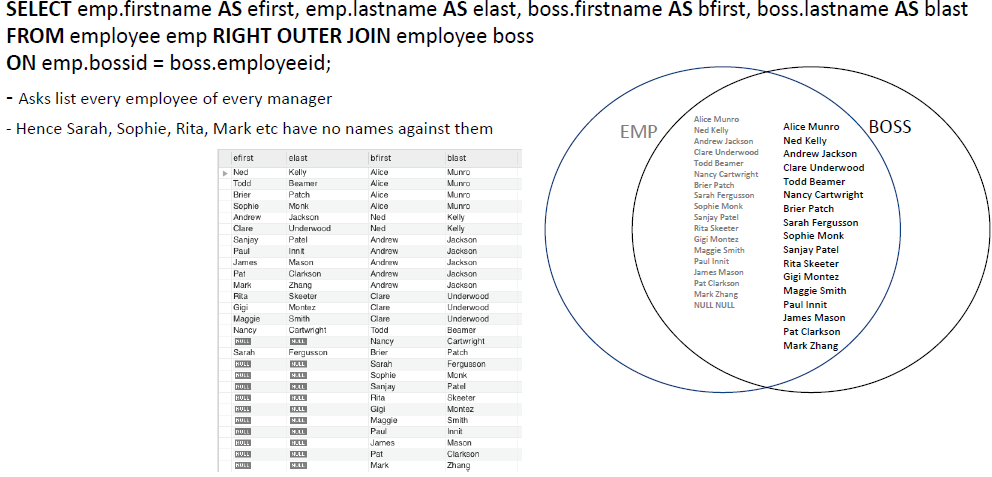
**别名为EMP, BOSS的Employee表的一元联接**

* ****

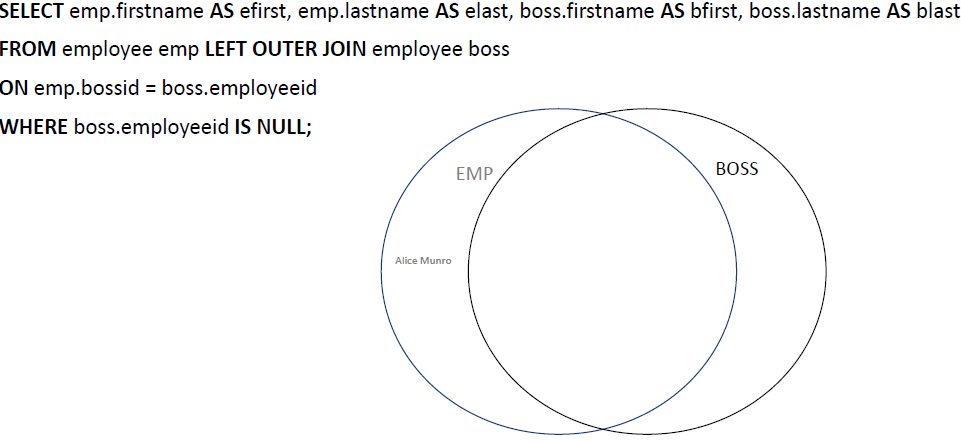
1. ****
2. **LEFT OUTER JOIN-Labs demo**

* ****

1. **RIGHT OUTER JOIN-Labs demo**

* ****

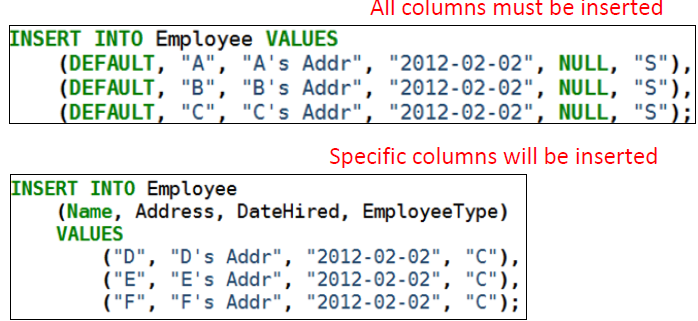
1. **Inner Join-Labs demo**

* ****

1. **More on INSERT**
2. **Inserting records from a table:**

* 

1. **Note: table must already exist**
2. **Multiple record inserts:**

* ****

1. **The UPDATE Statement**
2. **Changes existing data in tables**

* **Order of statements is important**
* **Specifying a WHERE clause is important**
* **Unless you want it to operate on the whole table**
* 
* **Example: Increase all salaries greater than $100000 by 10% and all other salaries by 5%**
* 
* **A better solution in this case is to use the CASE command**
* 
* **If salary is lower than 100000 increase it by 5%, otherwise increase it by 10%**

1. **DELETE, REPLACE**
2. **REPLACE**

* **REPLACE works identically as INSERT**
* **Except if an old row in a table has a key value the same as the new row then it is overwritten…**

**除非表中的旧行具有与新行相同的键值，那么它将被覆盖…**

1. **DELETE**

* **The DANGEROUS command deletes ALL records**
* ****
* **The better (safer) version (unless you are really, really sure)**

**更好(更安全)的版本(除非你非常非常确定)**

* ****

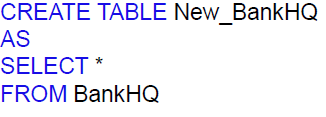
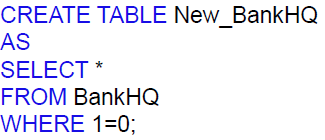
1. **Be aware of the foreign key constraints**

* **ON DELETE CASCADE or ON DELETE RESTRICT (lab practice)**

1. **ALTER (ADD/DROP), RENAME, TRUNCATE, CTAS**
2. **More DDL Commands**

* **ALTER**
* **Allows us to add or remove attributes (columns) from a relation (table)** **允许我们从关系(表)中添加或删除属性(列)**
* **ALTER TABLE TableName ADD AttributeName AttributeType**
* **ALTER TABLE TableName DROP AttributeName**
* **RENAME**
* **Allows the renaming of tables (relations)**
* **RENAME TABLE CurrentTableName TO NewTableName**
* **TRUNCATE截断**
* **Same as DELETE \* FROM table;**
* **Faster but cannot ROLL BACK a TRUNCATE command**
* **Have to get data back from backup…**
* **DROP**
* **Potentially DANGEROUS**
* **Kills a relation removes the data, removes the relation**
* **There is NO UNDO COMMAND! (have to restore from backup)**

**没有撤销命令!(必须从备份中恢复)**

* **DROP TABLE TableName**
* **CTAS (CREATE TABLE as SELECT)**
* ****
* **To create table structure with no rows**
* ****