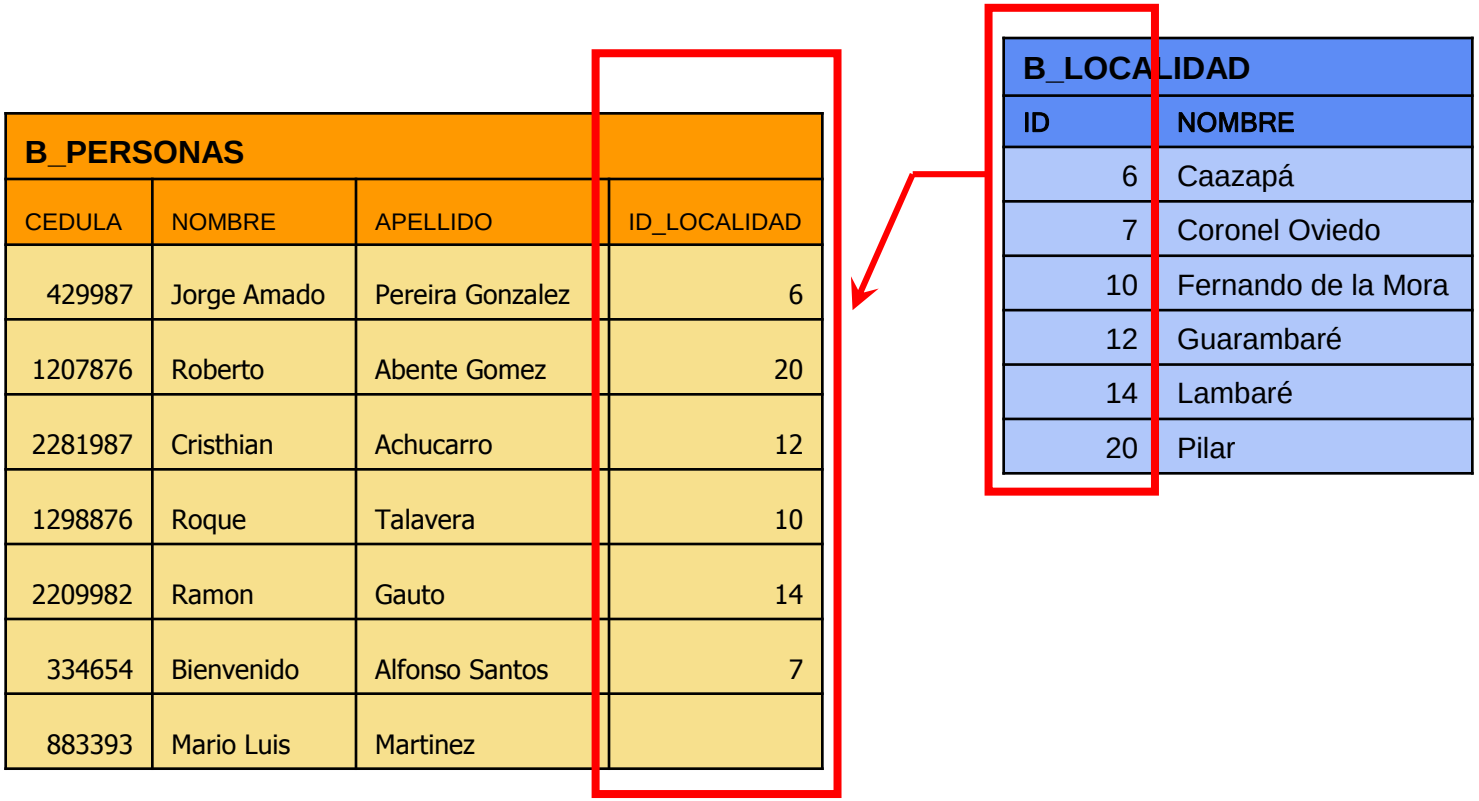
SELECCIÓN DE MÚLTIPLES TABLAS

Desplegando datos desde múltiples tablas

Que es un JOIN?

- Un JOIN (Reunión) se usa para consultar datos de más de una tabla
- Se unen filas usando valores comunes, típicamente claves primarias y foráneas o extranjeras
- Métodos de JOIN
  - Equijoin (Reunión Natural)
  - Non-equijoin (Reunión theta)
  - Outer join (Reunión Externa)
  - Self join
- A partir de la versión de ORACLE 9i se incorpora la sintaxis de SQL:1999

Relaciones entre las Tablas



B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7
883393	Mario Luis	Martinez	

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar

Tipos de Uniones

- Las uniones compatibles con el estándar SQL:1999 son:
  - Uniones cruzadas
  - Uniones naturales
  - Cláusula `USING`
  - Uniones externas completas (o de dos lados)
  - Condiciones de unión arbitrarias para uniones externas

Unión de Tablas mediante la Sintaxis SQL:1999

- Utilice una unión para consultar datos de más de una tabla:

```
SELECT      table1.column, table2.column
FROM        table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name =
table2.column_name)]|
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name =
table2.column_name)]|
[CROSS JOIN table2];
```

Unión de Tablas mediante la Sintaxis SQL:1999 (2)

- En la sintaxis:
  - **table1.column** muestra la tabla y la columna de las que se recuperan los datos
  - **NATURAL JOIN** une dos tablas basándose en el mismo nombre de columna
  - **JOIN table USING column_name** realiza una unión igualitaria basándose en el nombre de columna
  - **JOIN table ON table1.column_name** realiza una unión igualitaria basándose en la condición de la cláusula ON, = table2.column_name
  - **LEFT/RIGHT/FULL OUTER** se utiliza para realizar uniones externas
  - **CROSS JOIN** devuelve un producto cartesiano de las dos tablas

Creación de Uniones Naturales

- La cláusula NATURAL JOIN se basa en todas las columnas de las dos tablas que tienen el mismo nombre.
- Selecciona filas de las dos tablas que tienen valores iguales en todas las columnas correspondientes.
- Si las columnas que tienen los mismos nombres tienen tipos de datos diferentes, se devuelve un error.

Reunión Natural con sintaxis SQL:1999

- Un join Natural se basa en todas las columnas que tienen el mismo nombre en las dos tablas. Selecciona filas de las dos tablas que tienen valores iguales en las columnas relevantes.
- Al especificar las columnas que forman parte de un Natural Join, no debe utilizarse alias para las tablas ni para las columnas

```
SELECT tabla1.columna, tabla2.columna
  FROM   tabla1 NATURAL JOIN tabla2;
```

Reunión Natural con sintaxis SQL:1999. Ejemplo:

B_MAYOR				
ID_MAYOR (PK)	**CODIGO_CTA (FK)**	**ANIO**	**MES**	**ACUM_DEBITO**
26	5220000	2012	3	1200000
1	1110100	2012	2	3000000
22	5220000	2012	2	0

B_CUENTAS	
CODIGO_CTA (PK)	**NOMBRE_CTA**
5220000	LUZ
1110100	CAJA CHICA

```
SELECT id_mayor,codigo_cta, nombre_cta,anio,
mes, acum_debito
FROM b_mayor NATURAL JOIN b_cuentas;
```

ID_MAYOR	CODIGO_CTA	NOMBRE_CTA	ANIO	MES	ACUM_DEBITO
26	5220000	LUZ	2012	3	1200000
1	1110100	CAJA CHICA	2012	2	3000000
22	5220000	LUZ	2012	2	0

Creación de Uniones con la Cláusula USING

- Si hay varias columnas que tienen los mismos nombres pero los tipos de datos no se corresponden, la cláusula NATURAL JOIN se puede modificar mediante la cláusula **USING** para especificar las columnas que se deben utilizar para una unión igualitaria.
- Utilice la cláusula USING para asignar sólo una columna a la unión, cuando coincidan los nombres de más de una columna.
- No utilice un alias o un nombre de tabla en las columnas a las que se hace referencia.
- Las cláusulas NATURAL JOIN y USING se excluyen mutuamente.

Reunión Natural con la cláusula USING:

B_MAYOR				
ID_MAYOR (PK)	CODIGO_CTA (FK)	ANIO	MES	ACUM_DEBITO
26	5220000	2012	3	1200000
1	1110100	2012	2	3000000
22	5220000	2012	2	0

B_CUENTAS	
CODIGO_CTA (PK)	NOMBRE_CTA
5220000	LUZ
1110100	CAJA CHICA

```
SELECT id_mayor,codigo_cta, nombre_cta,anio,
mes, acum_debito
FROM b_mayor JOIN b_cuentas
USING (codigo_cta);
```

ID_MAYOR	CODIGO_CTA	NOMBRE_CTA	ANIO	MES	ACUM_DEBITO
26	5220000	LUZ	2012	3	1200000
1	1110100	CAJA CHICA	2012	2	3000000
22	5220000	LUZ	2012	2	0

Utilización de seudónimos (alias) en las reuniones.

- Cuando se realiza la reunión entre dos o más tablas, es fundamental que las columnas que tienen el mismo nombre en más de una tabla, tengan como prefijo el nombre de la tabla para evitar ambigüedad. (La excepción es la reunión utilizando NATURAL JOIN).
- Para evitar utilizar nombres largos, utilice seudónimos para las tablas
- El seudónimo solo es válido para esa declaración SELECT.
- Puede preceder las columnas con el nombre de la tabla, pero si usa seudónimos, para esa tabla, debe también referirse al seudónimo como prefijo de las columnas.

```
SELECT c.nombre, c.apellido FROM b_personas c;
```

En el ejemplo, la tabla **b_personas** tiene el alias **c**, el que luego se utiliza como prefijo de las columnas seleccionadas:

Equijoin

- Equijoin es aquel en el que la condición de reunión es un operador de igualdad
- Combina filas que tienen valores equivalentes para las columnas especificadas

```
SELECT tabla1.columna, tabla2.columna
   FROM   tabla1 INNER JOIN tabla2
ON tabla2.columna2 = tabla1.columna1;
```

Ejemplo de Equijoin con la sintaxis de SQL:1999 (sintaxis explícita):

B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar

```
SELECT  c.cedula, c.nombre, c.apellido, l.nombre
"Localidad"
FROM b_personas c INNER JOIN b_localidad l
ON      c.id_localidad = l.id
WHERE   c.ES_CLIENTE = 'S'
/
```

Ejemplo de Equijoin con la sintaxis de ORACLE (sintaxis implícita)

B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar

```
SELECT  c.cedula, c.nombre, c.apellido, l.nombre
"Localidad"
       FROM b_personas c, b_localidad l
       WHERE c.id_localidad = l.id
       AND   c.ES_CLIENTE = 'S';
```

Sintaxis de la Reunión con una condición simple (sintaxis implícita)

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1 = table2.column2;
```

- Se realiza el JOIN con la cláusula WHERE
- Utilizar prefijos para anteceder a los nombres de las columnas, cuando el mismo nombre de columna aparece en más de una tabla. El prefijo utilizado es el nombre de la tabla, o su prefijo

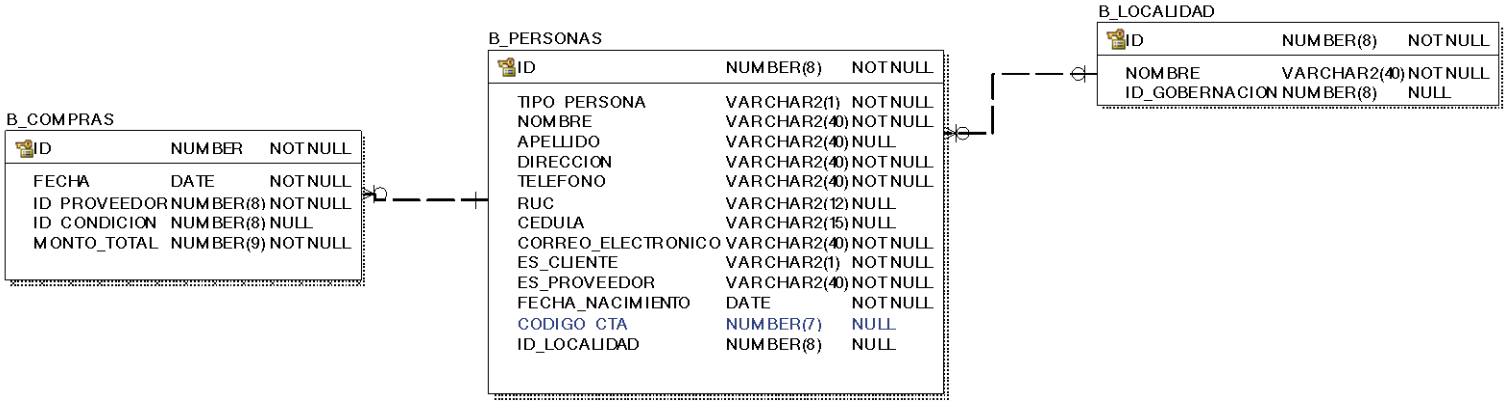
Sintaxis de la Reunión con una condición simple (sintaxis implícita)

- Tenga en cuenta:

Para reunir las tablas, se necesita un mínimo de condiciones de join igual al número de tablas menos 1.

Por ejemplo, para realizar operación de join de 4 tablas, se necesita 3 condiciones de join.

Ejemplo de reunión con 3 tablas:




```
SELECT c.fecha, c.id_proveedor, p.nombre,
l.nombre localidad
FROM   b_compras c JOIN b_personas p
ON     p.id = c.id_proveedor
JOIN   b_localidad l
ON     l.id = p.id_localidad;
```

Sintaxis Explícita

```
SELECT c.fecha, c.id_proveedor, p.nombre,
l.nombre localidad
FROM   b_compras c, b_personas p, b_localidad l
WHERE  p.id = c.id_proveedor
AND    l.id = p.id_localidad;
```

Sintaxis implícita (ORACLE)

Self Joins

B_EMPLEADOS (FUNCIONARIO)

APELLIDO	CEDULA_JEFE
Benitez	1607843
Caniza Livieres	1607843
Balmaceda	952160
Caballero	952160
Gonzalez	952160
Moreno	952160
Adorno	1098169
Aguilera	1098169

B_EMPLEADOS (JEFE)

CEDULA	APELLIDO
1607843	Brandenstein
952160	Caniza Livieres
1098169	Ferreira

Self Joins: Ejemplo

- Puede reunir filas de una tabla con otras filas de la misma tabla usando el SELF JOIN.
- Simule dos tablas en la cláusula FROM creando dos seudónimos para la tabla.

```
SELECT  funcionario.nombre || ' ' ||
funcionario.apellido||' trabaja para '||
jefe.apellido
FROM  b_empleados funcionario,b_empleados jefe
WHERE  jefe.cedula = funcionario.cedula_jefe;
```

```
SELECT funcionario.nombre || ' ' ||
funcionario.apellido ||' trabaja para '||
jefe.apellido
FROM  b_empleados funcionario JOIN b_empleados jefe
ON jefe.cedula = funcionario.cedula_jefe;
```

Non- Equijoins: Reuniones que no utilizan el operador =

- Se da este tipo de reunión cuando ninguna columna en una tabla corresponde directamente a una columna en la segunda tabla
- La condición JOIN contiene operadores diferentes al operador igual
- Para ejemplificar, suponga que existe una tabla B_NIVELSALARIO, que contenta los siguientes campos:
  - SALARIO_MINIMO NUMBER(8)
  - SALARIO_MAXIMO NUMBER(8)

Que determinen los niveles mínimo y máximo de la empresa. El ejemplo de una operación non-equijoin sería:

```
SQL> SELECT C.COD_CATEGORIA, C.NOMBRE_CAT, C.ASIGNACION
2  FROM B_CATEGORIAS_SALARIALES C, B_NIVELSALARIO N
3  WHERE C.ASIGNACION BETWEEN N.SALARIO_MINIMO AND N.SALARIO_MAXIMO;
```

OUTER JOIN vs INNER JOIN

- INNER JOIN: Cuando el JOIN de 2 o más tablas retorna únicamente las filas que tienen su correspondencia en la otra tabla
- OUTER JOIN: Cuando el JOIN de 2 o más tablas retorna, a más de las filas que tienen su correspondencia en la otra tabla:
  - Las filas que no tienen correspondencia en la tabla izquierda (LEFT)
  - Las filas que no tienen correspondencia en la tabla derecha (RIGHT)
  - Las filas que resultan del INNER, LEFT y RIGHT join (FULL OUTER)

Reunión Externa (Outer Join)

- La Reunión Externa permite ver filas que normalmente no aparecerían en una condición JOIN.
- El operador que indica Reunión Externa, con la sintaxis implícita, es el signo más (+)
- Ponga el signo (+) del lado del join donde exista una deficiencia de la información.
- El operador JOIN exterior sólo puede aparecer en un lado de la expresión.

```
SELECT    tabla1.columna, tabla2.columna
FROM      tabla1, tabla2
WHERE     tabla1.columna(+) = tabla2.columna;
```

Reunión Externa (Outer Join)

- Una condición que involucra un JOIN exterior no puede:
  - Usar el operador IN.
  - Ser unido a otra condición por el operador OR.
- El operador JOIN exterior sólo puede aparecer en un lado de la expresión.

Left Outer Join

B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7
883393	Mario Luis	Martinez	------

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar

Esta persona no tiene localidad

Ejemplo de LEFT OUTER JOIN

- Liste todas las personas, aún cuando no tengan localidad

```
SELECT  c.cedula, c.nombre, c.apellido, l.nombre
"Localidad"
FROM b_personas c, b_localidad l
WHERE l.id (+) = c.id_localidad;
```

```
SELECT  c.cedula, c.nombre, c.apellido, l.nombre
"Localidad"
FROM b_personas c LEFT OUTER JOIN b_localidad l
ON  l.id   = c.id_localidad;
```

Right Outer Join

B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7
883393	Mario Luis	Martinez	------

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar
21	San Lorenzo

Aún no se tienen clientes en San Lorenzo

Ejemplo de RIGHT OUTER JOIN

- Nos interesa listar todas las localidades, aún cuando no haya personas en dicha localidad

```
SELECT  c.cedula, c.nombre, c.apellido,
l.nombre "Localidad"
FROM b_personas c, b_localidad l
WHERE l.id = c.id_localidad (+);
```

```
SELECT  c.cedula, c.nombre, c.apellido, l.nombre
"Localidad"
FROM b_personas c RIGHT OUTER JOIN b_localidad l
ON  l.id   = c.id_localidad;
```

Full Outer Join

B_PERSONAS

CEDULA	NOMBRE	APELLIDO	ID_LOCALIDAD
429987	Jorge Amado	Pereira Gonzalez	6
1207876	Roberto	Abente Gomez	20
2281987	Cristhian	Achucarro	12
1298876	Roque	Talavera	10
2209982	Ramon	Gauto	14
334654	Bienvenido	Alfonso Santos	7
883393	Mario Luis	Martinez	------

B_LOCALIDAD

ID	NOMBRE
6	Caazapá
7	Coronel Oviedo
10	Fernando de la Mora
12	Guarambaré
14	Lambaré
20	Pilar
21	San Lorenzo

Aún no se tienen clientes en San Lorenzo

No se conoce la localidad de la persona

Ejemplo de FULL OUTER JOIN

- Nos interesa listar todas las localidades, aún cuando no haya personas en dicha localidad, y todas las personas, aún cuando se conozca la localidad a la que pertenecen

```
SELECT  c.cedula, c.nombre, c.apellido,
l.nombre "Localidad"
FROM b_personas c, b_localidad l
WHERE l.id (+)  = c.id_localidad (+);
```


```
SELECT  c.cedula, c.nombre, c.apellido,
l.nombre "Localidad"
FROM b_personas c FULL OUTER JOIN b_localidad l
ON  l.id   = c.id_localidad;
```

Producto Cartesiano

- Un producto Cartesiano se forma cuando:
  - Una condición join se omite
  - Una condición join es inválida
- En este caso, todos los registros en la primera tabla se unen a todas las filas en la segunda tabla
- Para evitar un producto Cartesiano, siempre debe incluir una condición válida en una cláusula WHERE
- Si se tiene el propósito de generar un producto cartesiano, es preferible utilizar **CROSS JOIN** en lugar de utilizar la sintaxis implícita omitiendo la cláusula WHERE.

Producto Cartesiano

Sintaxis implícita, omitiendo la cláusula WHERE:

```
SELECT c.fecha, c.id_proveedor, p.nombre
FROM   b_compras c,  b_personas p;
```

Sintaxis explícita, utilizando la cláusula CROSS JOIN:

```
SELECT c.fecha, c.id_proveedor, p.nombre
FROM   b_compras c CROSS JOIN  b_personas p;
```

OPERACIONES DE CONJUNTO

Unión

- La UNION permite combinar dos tablas que deben tener:
  - El mismo número de columnas
  - Las columnas correspondientes deben ser del mismo tipo de datos (no necesariamente deben tener el mismo nombre)
- UNION elimina filas duplicadas, en tanto que UNION ALL retorna todas las filas.
- Por defecto, el resultado es ordenado por la primera columna

Unión (2)

- Si se usa, el comando ORDER BY debe referirse a la posición, no al nombre del campo.

```
Select nombre, apellido from b_empleados
UNION
Select nombre, apellido from b_personas;
```

Intersección

- El comando INTERSECT tiene los mismos requerimientos de la UNION.
- Ej.: Ver todos los empleados que han realizado alguna venta

```
SELECT cedula FROM b_empleados
INTERSECT
SELECT cedula_vendedor FROM b_ventas;
```

Diferencia

- El comando MINUS reproduce la resta algebraica.
- Ej.: Ver todos los empleados que NO han realizado venta alguna

```
SELECT cedula FROM b_empleados
MINUS
SELECT cedula_vendedor FROM b_ventas;
```