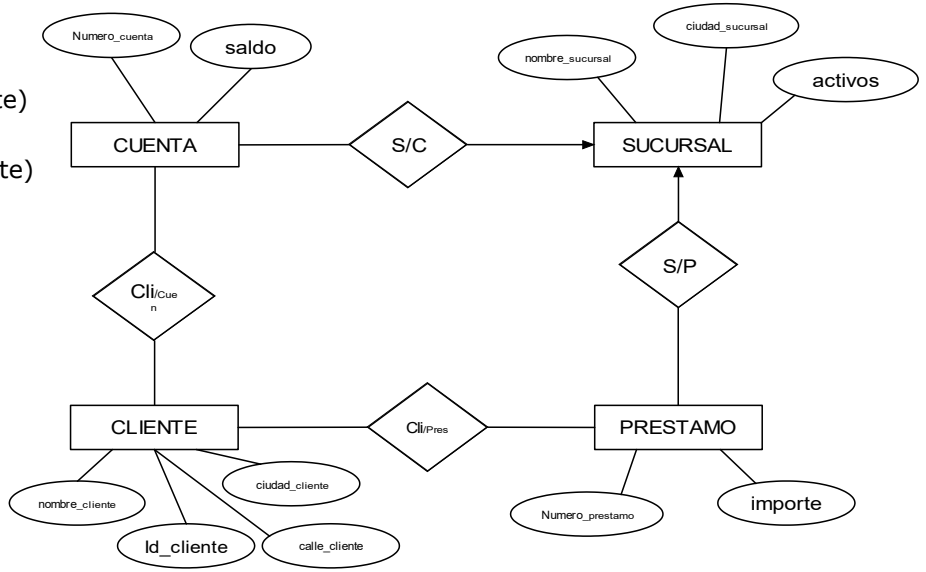
Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería

Cátedra: **Bases de datos I**

Lenguajes de Consulta

Esquema_sucursal=(nombre_sucursal, activo, ciudad_sucursal)
Esquema_cliente=(Id_cliente, nombre_cliente, calle, ciudad_cliente)
Esquema_cuenta=(nombre_sucursal, número_cuenta, saldo)
Esquema_préstamo=(nombre_sucursal,número_préstamo, importe)
Esquema_cli/cuen=(nombre_cliente, número_cuenta)
Esquema_Cli/Pres= (nombre_cliente, numero_prestamo)



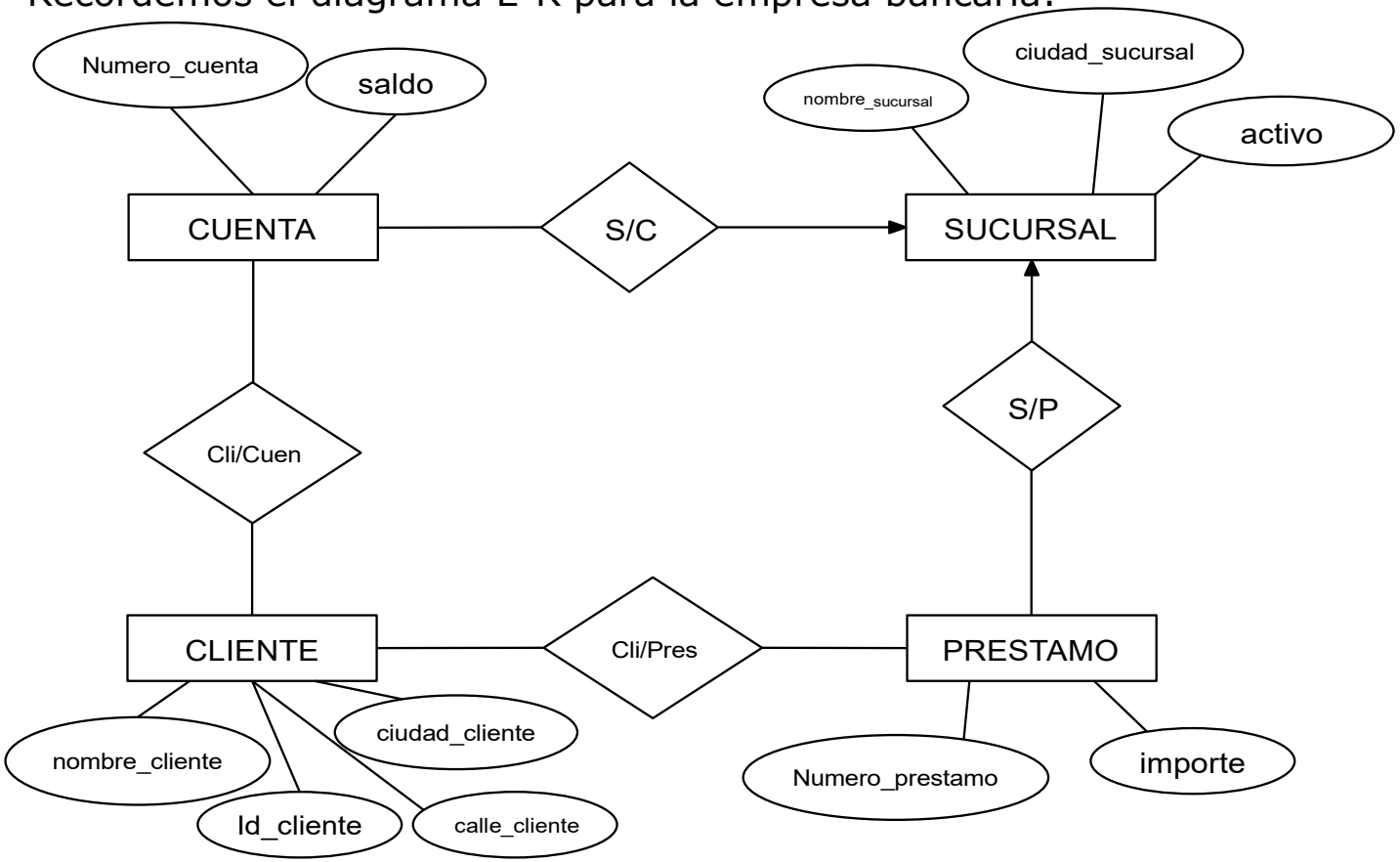
- Lenguajes de consulta

  - Es aquel en el que el usuario solicita información de la BD. Estos lenguajes son normalmente de más alto nivel que los lenguajes estándar de programación.
  - Pueden clasificarse en lenguajes **procedimentales** o **no procedimentales**.
  - ✓ En un lenguaje procedimental, el usuario da instrucciones para que se realice una secuencia de operaciones en la BD para calcular el resultado deseado.
  - ✓ En un lenguaje no procedimental, el usuario describe la información deseada sin dar un procedimiento específico para obtener esa información.
  - La mayor parte de los sistemas comerciales de BDR ofrecen un lenguaje de consulta que incluye elementos de los dos enfoques: procedimental y no procedimental.

  - A continuación veremos lenguajes puros:
  - ✓ El **álgebra relacional** → procedimental
  - ✓ El **cálculo relacional de tuplas** y el **calculo relacional de dominios** → no procedimentales

  - Estos lenguajes de consulta son rígidos y formales (a diferencia de los lenguajes comerciales) e ilustran las técnicas fundamentales para extraer datos de la BD.
  - Un lenguaje de manipulación de datos completo incluye no sólo un lenguaje de consulta, sino también un lenguaje para la modificación de la BD. Dichos lenguajes incluyen órdenes para insertar y borrar tuplas, así como para modificar partes de las tuplas existentes. Examinaremos la modificación de la BD después de completar nuestro tratamiento de los lenguajes de consulta "puros".

– Recordemos el diagrama E-R para la empresa bancaria:



– Los esquemas resultantes son:
  - ✓ Esquema_sucursal=(nombre_sucursal, activo, ciudad_sucursal)
  - ✓ Esquema_cliente=(id_cliente, nombre_cliente, calle_cliente, ciudad_cliente)
  - ✓ Esquema_cuenta=(número_cuenta, nombre_sucursal, saldo)
  - ✓ Esquema_préstamo=(número_préstamo, nombre_sucursal, importe)
  - ✓ Esquema_cli/cuen=(id_cliente, número_cuenta)
  - ✓ Esquema_cli/Pres= (id_cliente, numero_prestamo)

– Las relaciones quedan de esta manera:

CLIENTE

id_cliente	**nombre_cliente**	**calle_cliente**	**ciudad_cliente**
C-123	Lopez	San Martin	CRD
C-230	Sosa	Rivadavia	TRW
C-212	Herrera	Alem	MDY
C-235	Torres	Canada	CRD
C-189	Williams	Gales	TRW

SUCURSAL

nombre_sucursal	**ciudad_sucursal**	**activo**
Norte	CRD	1.800.000
Sur	TRW	420.000
Oeste	MDY	80.000
Centro	CRD	12.500.000
Roca	CRD	10.000.000

CUENTA

número_cuenta	nombre_sucursal	saldo
101	Norte	500
215	Sur	700
102	Oeste	400
305	Centro	350
201	Oeste	900
222	Norte	1800

CLI/CUE

id_cliente	**número_cuenta**
C-123	101
C-230	215
C-212	102
C-235	305
C-123	222

CLI/PRES

id_cliente	**número_préstamo**
C-212	17
C-230	23
C-189	15
C-235	14
C-230	93

- Álgebra Relacional

  - Es un lenguaje de consulta **procedimental**. Consta de un conjunto de operaciones que toman una o dos relaciones como entrada y producen una nueva relación como resultado.

  - Las operaciones fundamentales en el álgebra relacional son:

  ✓ Proyección: π
  ✓ Selección: σ
  ✓ Unión: $\cup$
  ✓ Diferencia: **-**
  ✓ Producto cartesiano: **x**

  - Además de las operaciones fundamentales existen otras operaciones, a saber:

  ✓ Renombramiento: ρ
  ✓ Intersección de conjuntos: $\cap$
  ✓ Join: $\bowtie$
  ✓ Join natural: $\bowtie$

  - Estas operaciones se definirán en términos de las operaciones fundamentales.

- Operaciones fundamentales
  - La operación **proyección**: Permite reducir el número de campos de una relación y además elimina las filas duplicadas. Se denota por la letra griega pi (π). Los atributos que se desea que aparezcan en el resultado se listan como subíndice de π. La relación de argumento se escribe entre paréntesis. P. e. en la relación cuenta, nos interesa ver solamente el nombre de la sucursal y el numero de cuenta: $\pi_{\text{nombre\_sucursal, numero\_cuenta}}(\text{cuenta})$
  - La relación resultado se muestra a continuación:

nombre_sucursal	número_cuenta
Norte	101
Sur	215
Oeste	102
Centro	305
Oeste	201
Norte	222

  - La operación **selección**: Selecciona tuplas que satisfacen una condición dada. Se utiliza la letra griega sigma minúscula (σ) para denotar selección. El predicado aparece como subíndice de σ. La relación del argumento se da entre paréntesis a continuación de σ.
  - Para seleccionar las tuplas de clientes que viven en CRD de la relación cliente hay que escribir: $\sigma_{\text{ciudad\_cliente=”CDR”}}$ (cliente)

nombre_cliente	calle	Ciudad_cliente
Lopez	San Martin	CRD
Torres	Canada	CRD

  - Así como usamos el operador = en el predicado, se pueden usar los operadores de comparación aritmética <, >, $\geq, \leq, \neq$.
  - Por ejemplo, de la relación cuenta se podría querer listar las cuentas con saldo > de $500. Además se pueden combinar varios predicados en uno mayor usando las conectivas: y ($\cap$) y o ($\cup$).

- **Composición de operaciones**: Es importante resaltar el hecho que el resultado de una operación relacional es una relación. Considere la consulta: encontrar los nombres de los clientes que viven en Madryn.

- La operación **unión**: Supongamos que queremos averiguar el id de los clientes del banco que tienen una cuenta, un préstamo o ambas cosas. La relación Cliente no contiene esa información dado que para ser cliente del banco se necesita tener una cuenta o un préstamo. Para esta consulta hacen falta las dos relaciones cli/cuen y cli/pres definidas anteriormente.
- Nombres de todos los clientes con préstamos en el banco: $\pi_{\text{Id\_cliente}}(\text{cli/pres})$
- Nombres de todos los clientes con cuenta en el banco: $\pi_{\text{Id\_cliente}}(\text{cli/cuen})$
- Para contestar la consulta inicial, debemos unir esos dos conjuntos, es decir, hacen falta todos los id's de clientes que aparecen en alguna de las dos relaciones o en ambas. La expresión es:

$$\pi_{\text{Id\_cliente}}(\text{cli/pres}) \cup \pi_{\text{Id\_cliente}}(\text{cli/cuen})$$

id_cliente
C-123
C-230
C-212
C-235
C-189

- Las relaciones son conjuntos → se eliminan los duplicados.

- Se debe asegurar que las uniones se realicen entre relaciones compatibles. No tendría sentido realizar la unión entre, p. e. relaciones de distinta cantidad de atributos, o entre nombres de clientes y conjunto de ciudades.
- Para que la operación unión $r \cup s$ sea válida, se deben cumplir dos condiciones:

1. Las relaciones r y s deben ser de la misma aridad.
2. Los dominios de los atributos de r y s deben ser iguales.

- La operación **diferencia**: Permite buscar las tuplas que están en una relación pero no en la otra, si las relaciones son compatibles. La expresión r – s da como resultado las tuplas que están en r pero no en s. P. e. se quieren buscar todos los clientes del banco que tienen abierta una cuenta pero no concedido un préstamo: $\pi_{Ide\_cliente}(cli/cuen) - \pi_{Id\_cliente}(cli/pres)$

id_cliente
C-123

- La operación **producto cartesiano**: esta operación permite combinar información de dos relaciones cualquiera. El producto cartesiano entre dos relaciones r y s es r x s. El producto cartesiano entre las relaciones cliente y cli/pres es el siguiente. (Observar en la tabla resultado como diferenciar entre el atributo nombre_cliente de la relación cliente y de la relación cli/pres)

Cli/pres.id_cliente	número_préstamo	Cliente.id_cliente	calle	ciudad
C-212	17	C-123	San Martin	CRD
C-212	17	C-230	Rivadavia	TRW
C-212	17	C-212	Alem	MDY
C-212	17	C-235	Canada	CRD
C-212	17	C-189	Gales	TRW
C-230	23	C-123	San Martin	CRD
C-230	23	C-230	Rivadavia	TRW
C-230	23	C-212	Alem	MDY
C-230	23	C-235	Canada	CRD
C-230	23	C-189	Gales	TRW
.....				

- Supongamos que queremos información sobre los clientes con préstamos en el banco que vivan en CDR, la consulta quedaría: $\sigma_{ciudad= \text{“CDR”}}(\text{ cliente x cli/pres})$
- Pero la columna cliente.Nombre_cliente contiene clientes que no tienen préstamos, entonces deberíamos poner la condición que Cli/pres.Nombre_cliente = Cliente.Nombre_cliente:

$$\sigma_{Cli/pres.Nombre\_cliente=Cliente.Nombre\_cliente}(\sigma_{ciudad=\text{“CDR”}}(\text{ cliente x cli/pres}))$$

- Otras Operaciones

  - Con las operaciones que vimos se puede expresar cualquier consulta en el álgebra relacional, pero hay consultas habituales en una BD que requerirían expresiones muy complicadas. Las siguientes operaciones tienen por objeto simplificar consultas habituales.

  - La operación **renombramiento**: a diferencia de las relaciones de la BD, los resultados de las expresiones del álgebra relacional no tienen un nombre que se pueda utilizar para referirse a ellas. La siguiente expresión permite poner un nombre a una consulta: $\rho_m(E)$ devuelve el resultado de la expresión E con el nombre m.
  - Puede usarse para cambiar el nombre a los atributos de la operación resultado:

    $\rho_{m\ (A, B\ \ldots\ldots\ N)}(E)$ devuelve el resultado de la expresión E con el nombre m y con los atributos con el nombre A, B, ...N.

  - La operación **intersección**: Si se desea averiguar todos los clientes que tienen un préstamo y una cuenta se utiliza esta operación:

$$\pi_{Id\_cliente}(cli/pres) \cap \pi_{Id\_cliente}(cli/cuen)$$

- La operación **join**: Es una operación binaria que permite combinar ciertas selecciones y un producto cartesiano en una sola operación. Esta operación forma un producto cartesiano y realiza una selección según un operador de comparación aritmética (=, <, etc.) entre dos atributos que aparecen en ambas relaciones.

- La operación **join natural**: Se escribe r ⋈ s, es una operación binaria que permite combinar selecciones y un producto cartesiano en una sola operación. Forma un producto cartesiano y realiza una selección forzando la igualdad de los atributos que aparecen en ambas relaciones, eliminando los duplicados.

- La consulta sobre los clientes con préstamos en el banco que vivan en CDR quedaría: $\sigma_{\text{ciudad= "CDR"}}$(cliente ⋈ cli/pres)

- Dado que las relaciones cliente y cli/pres tienen un campo con el mismo nombre (nombre_cliente), la operación join natural solo considera las tuplas que tienen el mismo valor en Id_cliente.

- **Álgebra relacional extendida - Operaciones**

- **Proyección generalizada**: amplía la operación proyección permitiendo que se usen funciones en la lista de proyección. $\pi_{F1,\ F2,\ldots Fn}(E)$ E: es una expresión del álgebra relacional o una relación

- P. e. La siguiente tabla se denomina información_préstamo, si se quiere averiguar el importe disponible para un futuro préstamo de un cliente: $\pi_{\text{id\_cliente, límite - saldo\_prestamo}}$(información_préstamo)

id_cliente	nombre_cliente	límite	saldo_prestamo
C-123	Lopez	1220000	900000
C-230	Sosa	50000	10000
C-212	Herrera	20000	5000
C-235	Torres	50000	50000
C-189	Williams	10000	5000

id_cliente	límite - saldo_prestamo
C-123	320000
C-230	40000
C-212	15000
C-235	0
C-189	5000

- **Funciones de agregación**: Toman un conjunto de valores de una columna de una tabla y devuelven un único valor como resultado. Ellas son:

  - ✓ **SUM**: devuelve la suma de un conjunto de valores
  - ✓ **AVG**: devuelve el promedio de un conjunto de valores
  - ✓ **COUNT**: devuelve la cantidad de elementos de un conjunto
  - ✓ **MAX**, **MIN**: devuelven el máximo / mínimo de un conjunto de valores

- P. e. La siguiente tabla (EMPLEADO) tiene información de los empleados de una empresa. Calcular el salario total que pagó la empresa: $SUM_{salario}(EMPLEADO)$

legajo	nombre_empleado	salario	departamento
123	Lopez	1220	Administrativo
212	Herrera	2000	Estadística
235	Torres	5000	Contaduría
230	Sosa	5000	Estadística
189	Williams	1000	Contaduría

suma_salario
14220

- **Agrupación**: Hay veces que no solo interesa calcular una función de agregación a todas las tuplas de una relación, sino a un grupo de tuplas. P. e. si se quiere averiguar el salario pagado por departamento: $\Gamma_{departamento}\ SUM_{salario}(EMPLEADO)$

- La relación o expresión (EMPLEADO) debe agruparse por departamento y luego sumar los salarios según departamento. Se lista departamento y la suma de salario. Primero agrupa de esta forma:

legajo	nombre_empleado	salario	departamento
123	Lopez	1220	Administrativo
230	Sosa	5000	Estadística
212	Herrera	2000	Estadística
235	Torres	5000	Contaduría
189	Williams	1000	Contaduría

- Luego encuentra la información:

suma_salario	departamento
1220	Administrativo
7000	Estadística
6000	Contaduría

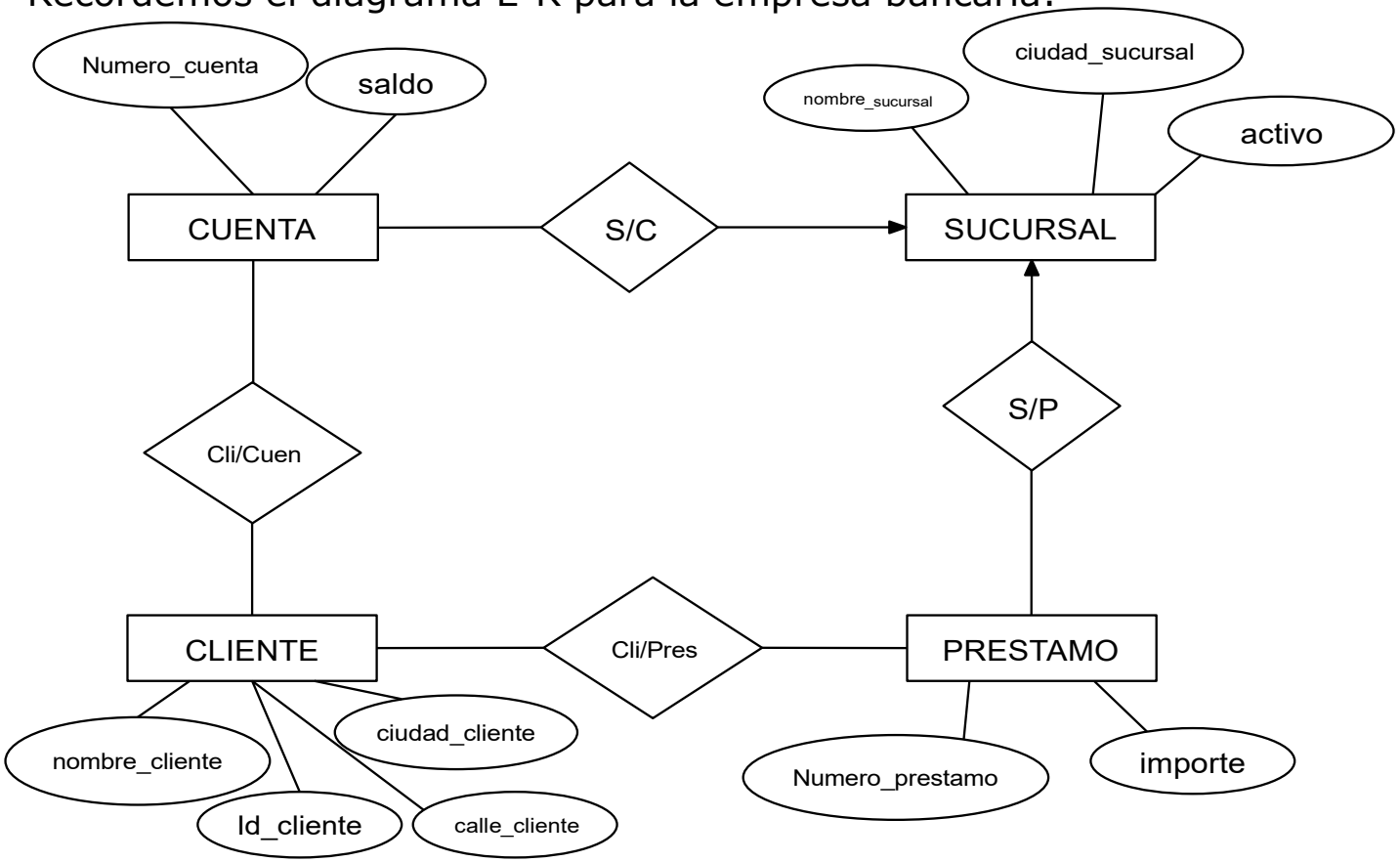
- Lenguajes de consulta

  - Es aquel en el que el usuario solicita información de la BD. Estos lenguajes son normalmente de más alto nivel que los lenguajes estándar de programación.
  - Pueden clasificarse en lenguajes **procedimentales** o **no procedimentales**.
  - ✓ En un lenguaje procedimental, el usuario da instrucciones para que se realice una secuencia de operaciones en la BD para calcular el resultado deseado.
  - ✓ En un lenguaje no procedimental, el usuario describe la información deseada sin dar un procedimiento específico para obtener esa información.
  - La mayor parte de los sistemas comerciales de BDR ofrecen un lenguaje de consulta que incluye elementos de los dos enfoques: procedimental y no procedimental.

  - A continuación veremos lenguajes puros:
  - ✓ El **álgebra relacional** → procedimental
  - ✓ El **cálculo relacional de tuplas** y el **calculo relacional de dominios** → no procedimentales

  - Estos lenguajes de consulta son rígidos y formales (a diferencia de los lenguajes comerciales) e ilustran las técnicas fundamentales para extraer datos de la BD.
  - Un lenguaje de manipulación de datos completo incluye no sólo un lenguaje de consulta, sino también un lenguaje para la modificación de la BD. Dichos lenguajes incluyen órdenes para insertar y borrar tuplas, así como para modificar partes de las tuplas existentes. Examinaremos la modificación de la BD después de completar nuestro tratamiento de los lenguajes de consulta "puros".

– Recordemos el diagrama E-R para la empresa bancaria:



– Los esquemas resultantes son:
  - ✓ Esquema_sucursal=(nombre_sucursal, activo, ciudad_sucursal)
  - ✓ Esquema_cliente=(id_cliente, nombre_cliente, calle_cliente, ciudad_cliente)
  - ✓ Esquema_cuenta=(número_cuenta, nombre_sucursal, saldo)
  - ✓ Esquema_préstamo=(número_préstamo, nombre_sucursal, importe)
  - ✓ Esquema_cli/cuen=(id_cliente, número_cuenta)
  - ✓ Esquema_cli/Pres= (id_cliente, numero_prestamo)

– Las relaciones quedan de esta manera:

CLIENTE

id_cliente	**nombre_cliente**	**calle_cliente**	**ciudad_cliente**
C-123	Lopez	San Martin	CRD
C-230	Sosa	Rivadavia	TRW
C-212	Herrera	Alem	MDY
C-235	Torres	Canada	CRD
C-189	Williams	Gales	TRW

SUCURSAL

nombre_sucursal	**ciudad_sucursal**	**activo**
Norte	CRD	1.800.000
Sur	TRW	420.000
Oeste	MDY	80.000
Centro	CRD	12.500.000
Roca	CRD	10.000.000

CUENTA

número_cuenta	nombre_sucursal	saldo
101	Norte	500
215	Sur	700
102	Oeste	400
305	Centro	350
201	Oeste	900
222	Norte	1800

CLI/CUE

id_cliente	**número_cuenta**
C-123	101
C-230	215
C-212	102
C-235	305
C-123	222

CLI/PRES

id_cliente	**número_préstamo**
C-212	17
C-230	23
C-189	15
C-235	14
C-230	93

- Cálculo Relacional de Tuplas (CRT)
  - Las consultas se expresan de la forma: { t / P (t) } y son el conjunto de todas las tuplas tales que el predicado P es cierto para t, donde t es una tupla variable → una variable que denota una tupla de alguna longitud fija (usamos t(i) para indicar es de aridad i), y P es una fórmula construida de átomos y de una colección de operadores. Los átomos de las fórmulas son de tres tipos:

1. $s \in r$, donde r es un nombre de relación y s es una tupla variable. Este átomo representa la afirmación de que s es una tupla de la relación R.
2. $s[x]\ \theta\ u[y]$, donde s y u son tuplas variables y θ es un operador de comparación aritmética (>, <, =, ≠). Es necesario que los atributos x e y, tengan dominios cuyos miembros puedan compararse mediante θ.
3. $s[x]\ \theta$ c, donde s es una variable tupla, θ es un operador de comparación y c es una constante en el dominio del atributo x.

  - Supongamos que queremos averiguar el nombre de las sucursales cuyos activos son mayores a $500.000
  - $\{ t^{(1)} / (\exists s^{(3)}) \cap \text{sucursal}(s) \cap s_{[3]} > \text{"500.000"} \cap t_{[1]} = s_{[1]} \}$
  - Averiguar el nombre y la calle en que viven aquellos clientes cuyos número de cuenta es menor a 100
  - $\{ t^{(2)} / (\exists c^{(4)}) \cap (\exists l^{(2)}) \cap \text{cliente}(c) \cap \text{cli/cuen}(l) \cap l_{[2]} < \text{"100"} \cap c_{[1]} = l_{[1]} \cap t_{[1]} = c_{[2]} \cap t_{[2]} = c_{[3]} \}$

  - Listar los clientes y número de cuentas con un saldo mayor a $599
  - Listar los datos de las sucursales y de las cuentas pertenecientes, cuyos activos sean menores a $500.000.

- Calculo Relacional De Dominios (CRD)
  - Está constituido con los mismos operadores que el CRT pero no hay tuplas sino variables dominio. Las expresiones son de la forma { (x, y, z.....) / P(x, y, z...) }. Donde x, y, z representan las variables de dominio, P representa una fórmula compuesta de átomos (igual que en el CRT). Los átomos del CRD tienen una de las siguientes formas:

1. $(x, y, z) \in r$, donde r es una relación con n atributos y x , y, z .son variables de dominio o constantes.
2. $x \theta y$, donde x e y son variables de dominio y θ es un operador de comparación aritmética (>, <, =, $\neq$). Es necesario que los atributos x e y, tengan dominios cuyos miembros puedan compararse mediante θ.
3. $x \theta c$, donde x es una variable de dominio, θ es un operador de comparación y c es una constante en el dominio del atributo x.

  - Supongamos que queremos averiguar el nombre de las sucursales cuyos activos son mayores a $500.000
  - $\{ x / (\exists a, b) \cap sucursal(x, a, b) \cap b >"500.000"\}$
  - Otro ejemplo: Averiguar el nombre y la calle en que viven aquellos clientes cuyas número de cuenta es menor a 100
  - $\{ x, y / (\exists a, b, c) \cap cliente(a, x, y, b) \cap cli/cuen(a, c) \cap (c <"100")\}$

  - Listar los clientes y número de cuentas con un saldo mayor a "599"
  - Listar los datos de las sucursales y de las cuentas pertenecientes, cuyos activos sean menores a $500.000.

- Introducción - Lenguajes Relacionales Comerciales: SQL y QbyE

  - La **integridad** de una BD significa la existencia de dos componentes importantes que son la **exactitud** (CORRECTNESS) y la **completitud** (COMPLETENESS) → La integridad de una BD garantiza que todos los datos son correctos (válidos).

  - Una BD puede estar sujeta a cualquier cantidad de restricciones de integridad, y el SGBD debe ser informado de todas estas restricciones, y debe hacerlas cumplir.
  - P. e. en una BD se pueden colocar restricciones para limitar el tipo de dato que puede ingresarse en una tabla.
  - Dichas restricciones pueden especificarse cuando la tabla se crea por primera vez a través de la instrucción CREATE TABLE, o luego de crear la tabla a través de la instrucción ALTER TABLE.

  - Los tipos comunes de restricciones incluyen las siguientes:
  - ✓ NOT NULL
  - ✓ UNIQUE
  - ✓ CHECK
  - ✓ Clave primaria
  - ✓ Clave externa

- **NOT NULL**: En forma predeterminada, una columna puede ser NULL. Si no desea permitir un valor NULL en una columna, querrá colocar una restricción en esta columna especificando que NULL no es un valor permitido.

```
CREATE TABLE CLIENTE
(id_cliente integer NOT NULL,
apellido varchar (30) NOT NULL,
nombre varchar(30));
```

- **UNIQUE**: Asegura que todos los valores en una columna sean distintos.

```
CREATE TABLE CLIENTE
(id_cliente integer Unique,
apellido varchar (30),
nombre varchar(30));
```

- La columna "id_cliente" no puede incluir valores duplicados, mientras dicha restricción no se aplica para las columnas "apellido" y "nombre".

- **CHECK**: Asegura que todos los valores en una columna cumplan ciertas condiciones.

```
CREATE TABLE CLIENTE
(id_cliente integer CHECK (id_cliente > 0),
apellido varchar (30),
nombre varchar(30));
```

- **CLAVE PRIMARIA**: Se utiliza para identificar en forma única cada tupla en la tabla. Puede consistir en uno o más campos en una tabla. Cuando se utilizan múltiples campos como clave primaria, se la denomina clave compuesta.
- Las claves primarias pueden especificarse cuando se crea la tabla (utilizando CREATE TABLE) o cambiando la estructura existente de la tabla (utilizando ALTER TABLE).

- MySQL

```
CREATE TABLE CLIENTE
(id_cliente integer ,
apellido varchar (30),
nombre varchar(30),
PRIMARY KEY (id_cliente));
```

- Oracle

```
CREATE TABLE CLIENTE
(id_cliente integer PRIMARY KEY,
apellido varchar (30),
nombre varchar(30));
```

- SQL Server

```
CREATE TABLE CLIENTE
(id_cliente integer PRIMARY KEY,
apellido varchar (30),
nombre varchar(30));
```

– A continuación se presentan ejemplos para la especificación de una clave primaria al modificar una tabla:

– MySQL, Oracle, SQL Server:

```
ALTER TABLE CLIENTE ADD PRIMARY KEY (id_cliente);
```

→ Antes de utilizar el comando ALTER TABLE para agregar una clave primaria es necesario asegurarse que el campo esté definido como 'NOT NULL' -- en otras palabras, NULL no puede aceptarse como valor para ese campo.

- **CLAVE EXTERNA (FORÁNEA)**: Es un campo (o campos) que señala la clave primaria de otra tabla. Su propósito es asegurar la integridad referencial.
- P. e. Tenemos dos tablas, una tabla CLIENTE y la tabla ORDEN que incluye los pedidos del cliente. La restricción aquí es que todos los pedidos deben asociarse con un cliente que ya se encuentra en la tabla cliente. En este caso, colocaremos una clave externa en la tabla ORDEN y la relacionaremos con la clave primaria de la tabla CLIENTE.

ORDEN

Nombre de columna	Característica
Id_Orden	Clave Primaria
fecha	
Id_cliente	Clave Externa
Monto	

CLIENTE

Nombre de columna	Característica
Id_cliente	Clave Primaria
Apellido	
Nombre	

- MySQL:

```
CREATE TABLE ORDENES
(id_orden integer,
fecha date,
id_cliente integer,
monto float,
Primary Key (id_orden),
Foreign Key (CLIENTE_id_cliente) references
  CLIENTE(id_cliente));
```

- Oracle y SQL Server:

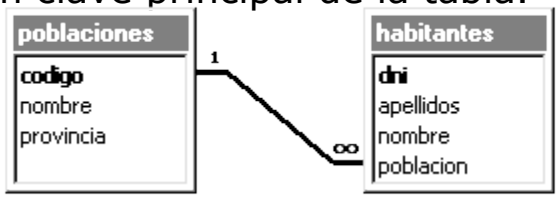
```
CREATE TABLE ORDENES
(id_orden integer primary key,
fecha date,
CLIENTE_id_cliente integer references CLIENTE(id_cliente),
monto float);
```

- Integridad referencial

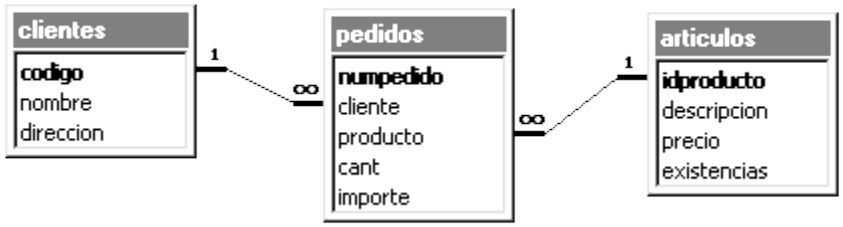
  - Es un sistema de reglas que utilizan la mayoría de las BDR para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo **errores de integridad**.
  - Primero recordemos los tipos de relaciones: entre dos tablas de cualquier BDR pueden haber dos tipos de relaciones: 1:1 y 1:M (las relaciones de M:M son tablas):

  - ✓ Relación 1:1: Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa.
  - ✓ Relación 1:M: Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y un registro de esta tabla puede tener más de un registro relacionado en la tabla muchos.
  - ✓ Relación M:M: Cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa.

  - 1:M: Ya vimos que una clave foránea es un campo de una tabla que contiene una referencia a un registro de otra tabla. Para el siguiente ejemplo en la tabla habitantes tenemos una columna población (ciudad) que contiene el código de la población en la que está empadronado el habitante, esta columna es clave foránea de la tabla habitantes, y en la tabla poblaciones tenemos una columna código de población clave principal de la tabla.



- M:M: En este caso las dos tablas no pueden estar relacionadas directamente, se tiene que añadir una tabla entre las dos que incluya los pares de valores relacionados entre sí. P. e.: tenemos dos tablas una con los datos de clientes y otra con los artículos que se venden en la empresa, un cliente podrá realizar un pedido con varios artículos, y un artículo podrá ser vendido a más de un cliente. Hace falta otra tabla (p. e. una tabla de pedidos) relacionada con clientes y con artículos. La tabla pedidos estará relacionada con cliente por una relación 1:M y también estará relacionada con artículos por un relación 1:M.



- Cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna o un valor nulo (ningún valor), o bien un valor que existe en la otra tabla, **un error sería asignar a un habitante una población que no está en la tabla de poblaciones**.

- Eso es lo que se denomina **integridad referencial** y consiste en que los datos que referencian otros (claves foráneas) deben ser correctos. La integridad referencial hace que el SGBD se asegure de que no haya en las claves foráneas valores que no estén en la tabla principal.

- La integridad referencial se activa en cuanto creamos una clave foránea y a partir de ese momento se comprueba cada vez que se modifiquen datos que puedan alterarla.

- ¿ Cuándo se pueden producir errores en los datos?

✓ Al insertar una nueva fila en la tabla secundaria y el valor de la clave foránea no existe en la tabla principal. P. e. insertamos un nuevo habitante y en la columna población escribimos un código de población que no está en la tabla de poblaciones (una población que no existe).

✓ Al modificar el valor de la clave principal de una tupla de una relación que está relacionada de 1:M con otra, modificamos el código de Comodoro Rivadavia, sustituimos el valor que tenía (1) por un nuevo valor (10), si Comodoro Rivadavia tenía habitantes asignados ¿qué pasa con esos habitantes? no pueden seguir teniendo el código de población 1 porque ya no existe. Hay dos alternativas, no dejar cambiar el código de Comodoro Rivadavia o bien cambiar el código de población de todos los habitantes de Comodoro Rivadavia y asignarles el código 10.

✓ Cuando modificamos el valor de la clave foránea, el nuevo valor debe existir en la tabla principal. P. e. cambiamos la población de un habitante, tenía asignada la población 1 (porque estaba empadronado en Comodoro Rivadavia) y ahora se le asigna la población 2 porque cambia de lugar de residencia. La población 2 debe existir en la tabla de poblaciones.

✓ Cuando queremos borrar una fila de la tabla uno y ese registro tiene hijos. P. e. queremos borrar la población 1 (Comodoro Rivadavia), si existen habitantes asignados a la población 1 no se pueden quedar con el valor 1 en la columna población, porque tendrían asignada una población que no existe. Tenemos dos alternativas, no dejar borrar la población 1 de la tabla de poblaciones, o bien borrarla y poner a valor nulo el campo población de todos sus 'hijos'.

- Asociada a la integridad referencial están los conceptos de **actualizar registros en cascada** y **eliminar en cascada**.

- Actualización y borrado en cascada
  - Son opciones que se definen cuando definimos la clave foránea y que le indican al sistema gestor qué hacer en los casos comentados en el punto anterior
  - ✓ **Actualizar registros en cascada**: Esta opción le indica al SGBD que cuando se cambie un valor del campo clave de la tabla principal, automáticamente cambiará el valor de la clave foránea de los registros relacionados en la tabla secundaria. P. e. si cambiamos en la tabla de poblaciones (la tabla principal) el valor 1 por el valor 10 en el campo código (la clave principal), automáticamente se actualizan todos los habitantes (en la tabla secundaria) que tienen el valor 1 en el campo población (en la clave ajena) dejando 10. Si no se tiene definida esta opción, no se puede cambiar los valores de la clave primaria de la tabla principal. En este caso, si intentamos cambiar el valor 1 del código de la tabla de poblaciones , no se produce el cambio y el sistema nos devuelve un mensaje que los registros no se han podido modificar por infracciones de clave.
  - ✓ **Eliminar registros en cascada**: Esta opción le indica al SGBD que cuando se elimina un registro de la tabla principal automáticamente se borran también los registros relacionados en la tabla secundaria. P. e. Si borramos la población Madryn en la tabla de poblaciones, automáticamente todos los habitantes de Madryn se borrarán de la tabla de habitantes. Si no se tiene definida esta opción, no se pueden borrar registros de la tabla principal si estos tienen registros relacionados en la tabla secundaria. En este caso, si intentamos borrar la población Madryn, no se produce el borrado y el sistema nos devuelve un error o mensaje que los registros no se han podido eliminar por infracciones de clave.

  - La integridad en el modelo relacional a evolucionado a través de los años. En un principio el énfasis estaba puesto en las claves. Pero pronto se vio que no eran suficientes, y se necesitaron restricciones de integridad generales.

- Clasificación de restricciones
  - Se clasifican en cuatro categorías:

  - ✓ **Restricciones de tipo (dominio)**: especifica los valores válidos para un tipo dado. Es una especificación de los valores que conforman el tipo en cuestión. Se la denomina también restricción de dominio, y nos dice que los dominios deben ser atómicos → deben ser una unidad mínima de información.
  - ✓ **Restricciones de atributo**: Especifica el valor válido para un atributo dado. Es sólo una declaración para que un atributo específico sea de un tipo particular. Por esta razón, una restricción de atributo puede ser eliminada únicamente mediante la eliminación del propio atributo. P. e. ciudad es char, cualquier intento de introducir un valor de un nombre que no sea de tipo char será rechazado.
  - ✓ **Restricciones de relación**: son las impuestas a una relación. También consideran los valores de los atributos de las claves primarias. Ningún componente de una clave primaria de una relación puede ser nulo. El valor nulo es un valor indefinido, y la clave primaria de una relación identifica de forma única y mínima a una sola tupla de la relación. Por ende, si algún componente de la clave primaria pudiese ser nulo, entonces dicha clave primaria no sería mínima, porque nos estaría diciendo que ese atributo no es necesario para identificar a una tupla dada. Por está razón, si existieran componentes nulos en una clave primaria, ésta no identificaría a ninguna tupla.
  - ✓ **Restricciones de bases de datos**: es aquella que relaciona dos o más relaciones de la base. P. e. ningún proveedor de Trelew puede suministrar el artículo A4. Estas restricciones no se pueden verificar en forma inmediata, sino que debe esperarse hasta que termine la transacción.

- Restricciones de estado vs. restricciones de transición

  - Hasta ahora hemos hablado de restricciones de estado, las cuales se ocupan de los estados correctos de una BD.

  - Sin embargo en ocasiones es necesario considerar restricciones de transición, es decir restricciones sobre transiciones válidas de un estado correcto a otro. P. e. una BD que tuviera la relación persona, donde uno de los atributos sea estado civil. Las siguientes transiciones no son válidas: soltero a viudo, soltero a divorciado, viudo a divorciado, divorciado a viudo.

- Vistas

  - Una vista es básicamente una expresión de algún lenguaje de consultas con un nombre. Por ejemplo:

```
create view buen_cliente as
(select nombre_cliente, dirección, ciudad
from cliente, compra
where valor_compra > "1000")
```

  - Al ejecutar esta instrucción, la expresión no es evaluada sino que es "recordada" por el SGBD, el cual la guarda en el catálogo bajo el nombre buen_cliente. Esta definición de vista permanece en la BD hasta que se ejecuta una orden `drop view buen_cliente`. Un usuario de la BD puede usar esa vista como si fuera una relación.

– ¿Para qué son las vistas?

- ✓ Proporcionan **seguridad para datos ocultos**, esto se refiere a los datos que no deben ser visibles a través de alguna vista. P. e. un cajero de un banco no puede ver el saldo de las cuentas de un cliente. Por lo tanto puede servir como un mecanismo de seguridad simple, pero efectivo.

- ✓ Las vistas permiten hacer **más simples las consultas complejas**.

- ✓ Permiten que **los datos sean vistos de distinta forma por diferentes usuarios** al mismo tiempo. Es decir, permiten a los usuarios concentrarse en la parte de la base de datos que les interesa e ignorar el resto.

- ✓ Las vistas pueden ofrecer la **independencia lógica de los datos**, esto es, si cambia la estructura lógica de la base de datos, los usuarios y los programas de usuario no deben cambiar. Suponga que cambia la estructura de la BD y una relación R se debe subdividir en dos más pequeñas R1 y R2, es importante señalar que la antigua R es la unión de las dos R1 y R2, entonces podemos definir una vista y la llamamos R. Toda aplicación que antes se refería a la relación R ahora puede referirse a la vista R.