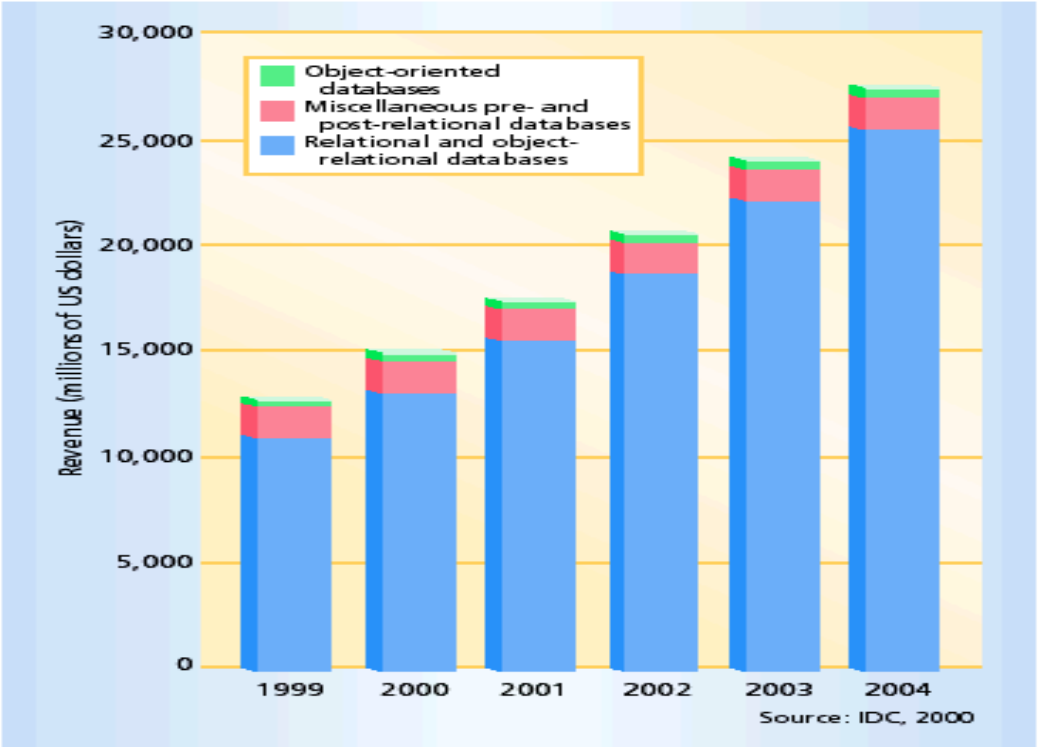
BD Orientadas a Objetos

- debilidades en Modelo Relacional en áreas como por ejemplo CAD/CAM o CASE ...
  - mejorar desempeño (mucho desarmar y armar la información en modelo relacional)
  - elevar el nivel de abstracción (uno debería agregar una factura y no una tupla de una tabla y 10 de otra
- se puede usar lenguajes de consulta
- Introducción ha resultado mas lenta de lo esperado

OO sobre modelo Relacional

- 10 años atrás …
  - modelo basado en objetos desplazaría al relacional en la misma forma que este último desplazó a los anteriores
- razonamiento se ha demostrado inválido y modelo relacional sigue liderando porque
  - a diferencia de los anteriores es un modelo verdaderamente general
  - tiene bases matemáticas muy sólidos
  - SQL es un standard universal (aún no existe un standard equivalente en OODBMS)
  - Desempeño ha mejorado significativamente
  - Aparición de los sistemas objeto-relacionales (soporte de objetos sobre RDBMS)

Perspectivas del Mercado



- Estudio realizado por IDC Market Research
- Ventas en 2001 de sistemas relacionales y objeto-relacionales de US$ 15.600 millones
- Ventas en 2001 de sistemas orientados a objetos de US$ 265 millones
- Proyección de crecimiento
  - sistemas relacionales y objeto-relacionales 18.2 %
  - sistemas orientados a objetos 12.5 %

Enfoque de ODMG (ODL/OQL)

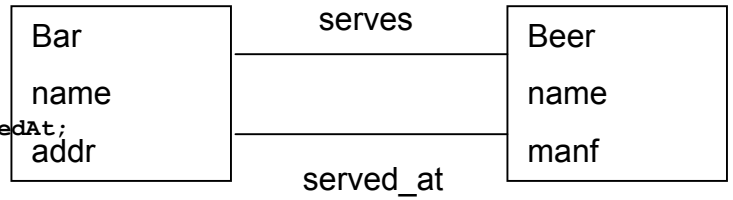
- Intento de estandarizar el modelo de objetos para BD
- ODL - Object Description Language
- OQL - Object Query Language (similar a SQL)
- Declaración de una Clase (Interface) en ODL
  - nombre de la clase
  - declaración de la extensión (extent) - el conjunto que contiene los objetos de la clase
  - declaración de los elementos de la clase
    - atributos
    - métodos
    - relaciones (relationships)
- Sintaxis
  - interface <name>{ <list of elements separated by semicolons > }

Atributos y Relaciones

- los atributos son elementos de tipos que no involucran clases
  - attribute <type> <name>;
- las relaciones conectan un objeto de una clase con uno o mas objetos de otra clase
  - relationship <type> <name> inverse <relationship>;
- Si C está relacionado con D a través de R entonces D debe estar relacionado con C a través de S

Ejemplo

```
interface Bar {
   attribute string name;
   attribute string addr;
   relationship Set<Beer> serves inverse Beer::servedAt;
}
interface Beer {
   attribute string name;
   attribute string manf;
   relationship Set<Bar> servedAt inverse Bar::serves;
}
```

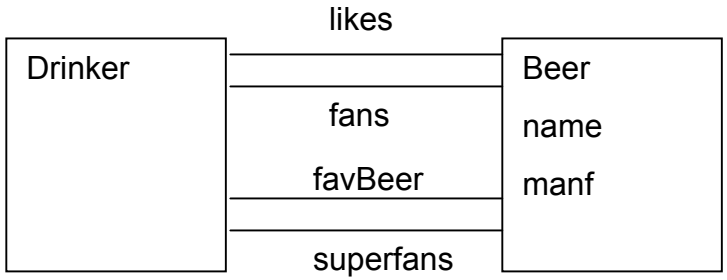


Tipos de Relaciones

- Bar
  - en este caso un objeto puede estar conectado con un solo objeto de esa clase
- Set<Bar>
  - un objeto está conectado a un conjunto de objetos Bar
- Bag<Bar>
  - un objeto está conectado a un Bag de objetos Bar
- List<Bar>
  - un objeto está conectado a una Lista de objetos Bar
- Array<Bar>
  - un objeto está conectado a un Array de objetos Bar

Tipo de Relaciones

- Todas las relaciones son binarias
- Multiplicidad
  - 1 - 1 tiene clases en ambos lados
  - 1 - * tiene en un lado Set<...>
  - * - * tiene Set<...> en ambos lados



Ejemplo

```
interface Drinker { …
  relationship Set<Beer> likes inverse Beer::fans;
  relationship Beer favBeer inverse Beer::superfans;
}
interface Beer { …
  relationship Set<Drinker> fans inverse Drinker::likes;
  relationship Set<Drinker> superfans inverse Drinker::favBeer;
}
```

Tipos de Dato

- tipos básicos
  - int, real/float, string, tipos enumerados, y classes.
- constructores
  - Struct para estructuras
  - Colecciones
    - Set
    - Bag
    - List
    - Array
    - Dictionary
  - Tipos Relationship solo pueden seer una clase o una colección aplicada a una clase

Subclases

- lo usual en lenguajes object-oriented
- Se indica superclase con : y el nombre
- La subclase solo lista la propiedades que son únicas (también hereda propiedades de la superclase)

Ejemplo

```
interface Ale:Beer {

     attribute string color;

}
```

Extensión (Extents)

- Cada clase tiene su *extent*
- representa el conjunto de objetos de esa clase
- se indica después del nombre de la clase junto con las claves (extent <extent name >)
- por convención se acostumbra usar singular para la clase y plural para el extents

Ejemplo

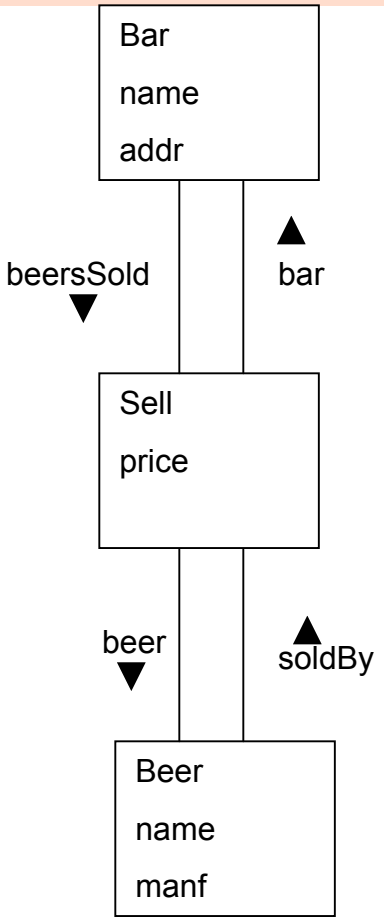
```
interface Beer

   (extent Beers key name) { …

}
```

OQL

- Usa ODL como el lenguaje de definición del esquema
- tipos de dato son los mismos de ODL
- Set(Struct) y Bag(Struct) juegan el rol de tablas
- Se usan path expresions. Supongamos que x es un objeto de la clase C
  - si a es un atributo de C entonces x.a es el valor del atributo
  - si r es una relación de C entonces x.r es el valor al cual x está conectado (podría ser un Set)
  - si m es un método de C entonces x.m es el resultado de aplicar m a x

Ejemplo

```
interface Sell (extent Sells) {
 attribute real price;
 relationship Bar bar inverse Bar::beersSold;
 relationship Beer beer inverse Beers::soldBy;
}
interface Bar (extent Bars) {
 attribute string name;
 attribute string addr;
 relationship Set<Sell> beersSold inverse Sell::bar;
}
interface Beer (extent Beers) {
 attribute string name;
 attribute string manf;
 relationship Set<Sell> soldBy inverse Sell::beer;
}
```



Select

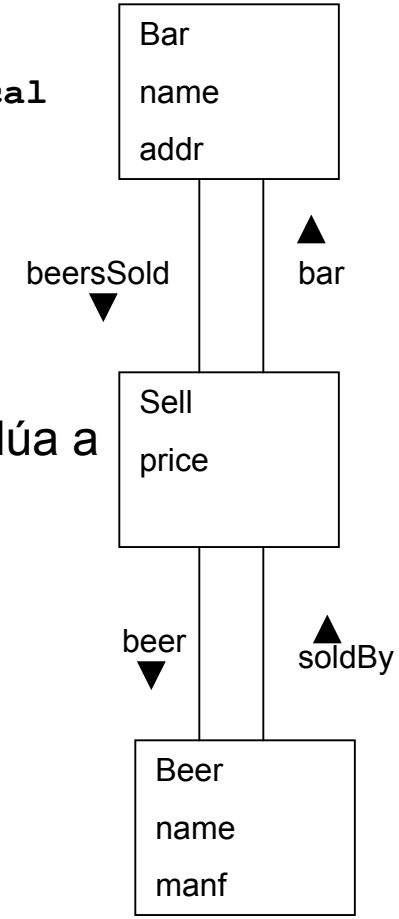
- Similar al Select de SQL

```
SELECT <list of values>
FROM <list of collections and names for typical members>
WHERE <condition>
```

- cada término del FROM tiene la forma

```
<collection><member name>
```

- una colección puede ser un extent o una expresión que evalúa a una colección (b.beersSold)



Ejemplos

```
SELECT s.beer.name, s.price
FROM Sells s
WHERE s.bar.name = "Joe's Bar"

SELECT s.beer.name, s.price
FROM Bars b, b.beersSold s
WHERE b.name = "Joe's Bar"
```

Futuro de ODL/OQL

- ODMG se disolvió después de producir la especificación ODMG 3.0 que incluye un modelo de objetos ODL, OQL y bindings para C++, Smalltalk y Java
- el binding con Java fue entregado a JCP (Java Community Process) y dió origen a JDO (Java Data Objects)
- Probablemente JDO se transforme en la manera de acceder en forma transparente a BD desde Java (proporciona lo que se conoce como transparent persistence)