El nivel de datos de la arquitectura Java EE

Autores: SimonPickin

Pablo Basanta-Val

Dirección: Departamento de Ingeniería Telemática

Universidad Carlos III de Madrid

Versión: 1.0

Agradecimientos: Jesús Villamor Lugo, Natividad Martínez Madrid, IT/UCIIIM, Marty Hall

Contenido

- Terminología y conceptos
- El Modelo de Datos Relacional (MDR)
  - Bases de Datos Relacionales (BDR)
  - Álgebra relacional
- SQL básico
  - DDL (*Data Definition Language*)
  - DML (*Data Manipulation Language*)
- Conexión de aplicaciones Java a bases de datos: JDBC

Terminología de Bases de Datos

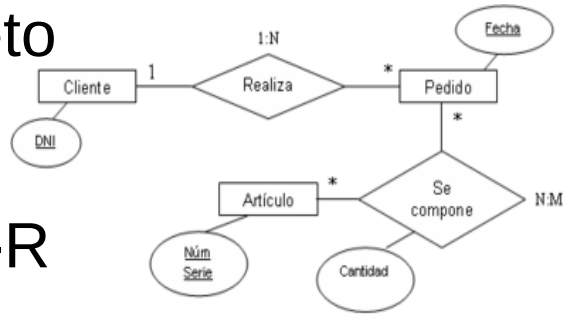
- Base de datos (BD)
  - una colección de datos relacionados
  - colección coherente de datos con significado inherentes
- Datos (D)
  - hechos conocidos que pueden registrarse y que tienen un significado implícito (susceptible de ser modelados)
- Universo de Discurso (UdD)
  - parte del mundo sujeto a ser modelado en una BD
- Sistema de Gestión de Bases de Datos (SGBD)
  - software que facilita la definición, construcción y manipulación de las BDDD

Modelos de BBDD: Conceptos

- Modelo de Datos (MD)
  - colección de conceptos que sirven para describir la estructura de una base de datos
  - M = {G, O}
    - G: conjunto de reglas de generación de datos
      - DDL: Lenguaje de definición de datos
      - restricciones sobre los datos
    - O: conjunto de operaciones permitidas sobre los datos
      - DML: Lenguaje de manipulación de datos
- Esquema de la BD (nivel intencional)
  - descripción de la estructura de la base de datos
- Estado de la BD (nivel extensional)
  - los datos que la BD contiene en un determinado momento

Modelo Entidad-Relación (E-R)

- Creado por P.P. Chen en 1976
- Modelo de datos que utiliza la teoría de conjuntos para modelar el diseño computacional del mundo real
- No es un modelo de datos completo
  - tiene G pero no tiene O



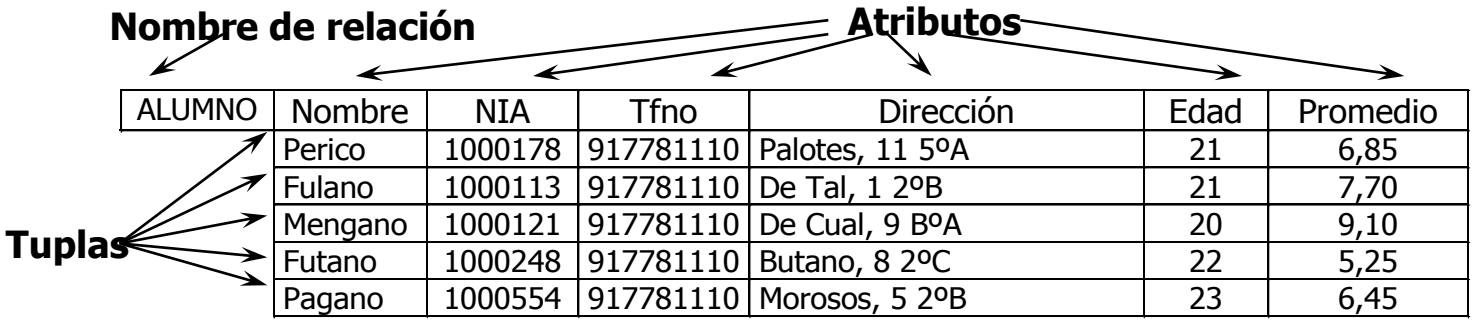
- Pasos para definir un diagrama E-R
  - seleccionar los conjuntos de entidades
  - describir los propiedades de cada conjunto de entidades
  - seleccionar el conjunto de relaciones
  - describir los propiedades posibles del conjunto de relaciones

Contenido

- Terminología y conceptos
- El Modelo de Datos Relacional (MDR)
  - Bases de Datos Relacionales (BDR)
  - Álgebra relacional
- SQL básico
  - DDL (*Data Definition Language*)
  - DML (*Data Manipulation Language*)
- Conexión de aplicaciones Java a bases de datos: JDBC

El Modelo de Datos Relacional (MDR): Visión General

- Introducido por Tedd Codd (IBM Research) en 1970
  - Utiliza el concepto de relación matemática
  - tiene sus bases teóricas en la teoría de conjuntos y la lógica de predicados de primer orden
- Analogía Relación ≈ Tabla
  - Relación ≈ Tabla
  - Tupla ≈ Fila
  - Atributo ≈ Cabecera de una columna
  - Dominio ≈ Los tipos de valores de una columna



ALUMNO	Nombre	NIA	Tfno	Dirección	Edad	Promedio
	Perico	1000178	917781110	Palotes, 11 5ºA	21	6,85
	Fulano	1000113	917781110	De Tal, 1 2ºB	21	7,70
	Mengano	1000121	917781110	De Cual, 9 BºA	20	9,10
	Futano	1000248	917781110	Butano, 8 2ºC	22	5,25
	Pagano	1000554	917781110	Morosos, 5 2ºB	23	6,45

MDR: Conceptos (I)

- Dominio D_i
  - conjunto de valores atómicos
- Esquema de Relación $R(A_1, A_2, ..., A_n)$
  - R, el nombre de la relación, A_1, A_2, ..., A_n, el nombre de sus atributos (a veces, el rango de valores admitidos es parte de la definición)
- Atributo A_i
  - un nombre que participa en una relación y al que está asociado un dominio D_i que define el conjunto de valores que puede tomar
- Una tupla t
  - una lista ordenada de n valores $t = <v_1, v_2, ..., v_n>$, donde cada valor v_i es un elemento de dom(A_i) o bien un valor nulo especial (que denota un valor desconocido, inaplicable o todavía no asignado)

MDR: Conceptos (II)

- Una instancia (o un estado) de una relación *r(R)*
  - Un subconjunto del producto cartesiano de los dominios de R

    $$r(R) \subseteq (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$$

    Las tuplas de una relación no tienen por qué estar ordenadas
- El grado de una relación
  - El número de atributos *n* de su esquema de relación
- La cardinalidad de una relación
  - El número de tuplas de la relación

Dominios

- Dominios usuales
  - datos numéricos estándar,
  - números reales,
  - caracteres,
  - cadenas de caracteres,
  - fechas,
  - horas,
  - dinero,
  - tipos enumerados
  - …

MDR: Restricciones relacionales (I)

- **Restricción de Dominio**
  - el valor de cada atributo *A* debe ser un valor atómico del dominio *dom(A)*; es decir, no se permiten valores que son conjuntos

- ¿Cómo manejar atributos de valores múltiples?
  - utilizar múltiples tuplas, una por valor
  - utilizar múltiples atributos, uno por valor
  - utilizar múltiples relaciones

MDR: Restricciones relacionales (II)

- **Restricción de claves** (**restricción de tupla única**)
  - todas las tuplas de una relación deben ser distintas

Clave primaria

- **Clave**
  - un subconjunto del conjunto de atributos de una relación
- **Superclave** *SC*
  - aquel conjunto de atributos cuyos valores caracterizan las tuplas de una relación: $\forall$ tuples $t_i = (a^1_i, \ldots a^m_i)$, $t_j = (a^1_j \ldots a^m_j)$,

    $a^k_i = a^k_j$, $\forall a^k \in SK \rightarrow t_i = t_j$ (*restricción de unicidad*)
  - gracias a la restricción de claves, toda relación tiene por lo menos una superclave (el conjunto de todos sus atributos)
- **Superclave mínima**
  - superclave tal que ninguno de sus subconjuntos también lo es
- **Clave primaria** *CP.* Convención: (atributos subrayados)
  - *clave candidata*: cualquiera de las superclaves mínimas
  - *clave primaria*: una clave escogida de entre las claves candidatas cuyos valores van a identificar las tuplas de una relación

MDR: Restricciones relacionales (III)

- **Restricción de integridad de entidades**
  - ningún valor de la clave primaria puede ser nulo

Clave externa

- Dos claves son compatibles si:
  - tienen el mismo número de atributos
  - atributos correspondientes comparten el mismo dominio
- Clave externa (*foreign key*)

  FK es una clave externa de la relación R_1 si:
  - FK es compatible con una clave candidata (normalmente *CP*) de la relación R_2
  - los valores de *FK* concuerdan con los del *CK* correspondiente, es decir:

    $\forall$ tuples t_1 of the current state $r_1(R_1)$,
    $\exists$ tuple t_2 in the current state $r_2(R_2)$ s.t.
    $t_1[FK] = t_2[CK]$ or $t_1[FK]$ = null

  Puede ser el caso que $R_1 = R_2$

MDR: Restricciones relacionales (IV)

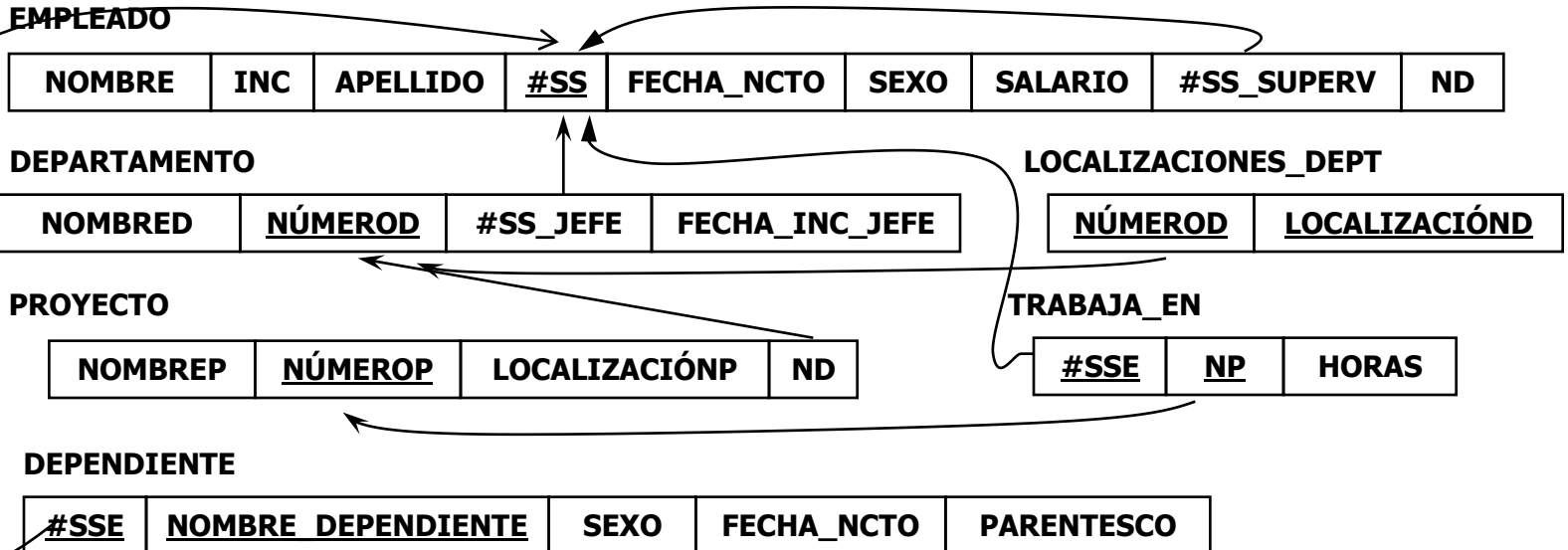
- **Restricción de integridad referencial**
  - Ninguna de las relaciones de un esquema relacional tiene un valor de una clave foránea (foreign key) que no aparezca en alguna tupla de la relación referenciada por la clave.
- Borrar una tupla de R_2 que contiene un valor al que se refiere una clave externa de R_1 rompería la integridad referencial

MDR: Restricciones relacionales (V)

- **Restricciones de integridad semántica**
  - aquellas que limitan el rango de valores de determinados atributos
    - ej_1. El salario de un empleado no debe ser mayor que el de su supervisor
    - ej_2. El número máximo de horas semanales de un trabajador es 56
- Para aplicar tales restricciones a las BDR se utilizan mecanismos de control semántico de los datos
  - disparadores (*triggers*) y aserciones

Bases de Datos Relacionales como Modelos Relacionales

- Esquema de Base de Datos Relacional *S*
  - conjunto de esquemas de relaciones $S = \{R_1, R_2, ..., R_m\}$ + conjunto de restricciones de integridad *RI*
- Un estado de Base de Datos Relacional *BD de S*
  - conjunto de estados de relaciones $BD = \{r_1, r_2, ..., r_m\}$ tal que todos sus r_i (estados de sus R_1) satisfacen las restricciones *RI*



Álgebra relacional: operaciones sobre conjuntos: 1. Unión, 2. Diferencia y 3. Intersección

- $R \cup S$
  - incluye todas las tuplas de las dos relaciones (eliminando duplicados)
- $R - S$
  - incluye todas las tuplas que están en *R* pero no en *S*
- $R \cap S$
  - incluye todas las tuplas que están en *R* y también en *S*

Compatibilidad para las operaciones $\cup$, $-$ *y* $\cap$

- las dos relaciones tienen el mismo grado
- atributos correspondientes comparten el mismo dominio

habitualmente llamado “compatibilidad de unión”

Álgebra relacional: operaciones sobre conjuntos 4. Producto Cartesiano

- $R \times S$
  - el resultado de $R(A_1, A_2, ..., A_n) \times S(B_1, B_2, ..., B_m)$ es una relación Q con n+m atributos $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$ en ese orden, con una tupla por cada combinación de tuplas: una de R y una de S
  - Debido a que en el modelo relacional las tuplas están desordenadas, el producto es conmutativo

Álgebra relacional: operaciones relacionales
5. Selección

- σ *<condición de la selección>* (R)
  - Sirve para seleccionar un subconjunto de tuplas dentro de una relación
    - *<condición de la selección>* ::= [*cláusula operadorClausula*] *cláusula*
    - *operadorCláusula ::=* $\wedge$ | $\vee$
    - *cláusula ::=* [$\neg$] *subcláusula*
    - *subcláusula ::=* *nombré atributo operadorComparación variConstant*
    - *variConstant ::=* *valorConstante* | *nombreAtributo*
    - *operadorComparación ::=* = | $\neq$ | > | $\geq$ | < | $\leq$
      estos operadores se aplican a dominios que son valores ordenados (como numéricos o fechas)

- Ejemplo: Seleccione todas las tuplas de empleados que
  - o bien trabajan en el dpto. 4 y ganan más de 25.000 dólares al año
  - o bien trabajan en el dpto. 5 y ganan más de 30.000 dólares al año

$$\sigma_{(ND=4 \wedge SALARIO>25000) \vee (ND=5 \wedge SALARIO > 30000)} (EMPLEADO)$$

Álgebra relacional: operaciones relacionales. 6. Proyección

- $\pi_{<lista\ de\ atributos>}(R)$
  - selecciona ciertos atributos de la relación y desecha los demás
  - el resultado de proyectar elimina tuplas repetidas
    - el resultado es otra relación

- Ejemplo: Deme el sexo y el salario de los empleados

$$\pi_{SEXO,\ SALARIO}(EMPLEADO)$$

Álgebra relacional: otras operaciones
7. Join (θ-join)

- $R \underset{i\ \theta\ j}{|\times|} S$ (ó $\triangleright\!\triangleleft$ como una mariposa)

  - El resultado es el conjunto de tuplas formadas al concatenar una tupla de R con otra tupla de S de manera que se cumpla la condición θ, donde θ relaciona $R.a_i$ y $S.a_j$ para $i,j = 1,\ldots n$.

    - es equivalente a una selección sobre el producto Cartesiano:

$$T = \underset{i\ \theta\ j}{|\times|} S = \sigma_{i\ \theta j}\ (R \times S)$$

Álgebra relacional: otras operaciones
8. Equi-join

- $R \underset{i=j}{|\times|} S$

  – es un θ - JOIN en el que la condición θ es igualdad entre valores de atributos de *R* y valores de atributos de *S*

    - es equivalente a una selección sobre el producto Cartesiano:

$$T = \underset{i=j}{|\times|} S = \sigma_{i=j} (R \times S)$$

Álgebra relacional: otras operaciones
9. Join natural

- $R \mid\times\mid S$
  - El resultado es el conjunto de tuplas formadas al concatenar una tupla de *R* con una tupla de *S* que tiene los mismos valores para los atributos comunes, de manera que se eliminen las columnas duplicadas
    - es equivalente a un equi-join sobre atributos con el mismo nombre seguido por una proyección para eliminar columnas duplicadas

      $$R \mid\times\mid S = \pi_{i1, i2, ..., im} [\sigma_{R.ai=S.aj, \text{ donde } nombre(ai)=nombre(aj)} (R \times S)]$$

    - ***requiere atributos homónimos***
      - *atributos con el mismo nombre definidos sobre el mismo dominio*
  - Es uno de los operadores más utilizados en las consultas
    - es la forma de "navegar" entre relaciones con atributos homónimos

Álgebra relacional: otras operaciones
10. Semijoin

- $R \ltimes S$ *(símbolo como mariposa sin cierre del ala dcha.)*
  - El resultado es el conjunto de tuplas situadas a la izquierda del operador, tal que son exactamente las tuplas que participarían en el join natural entre R y S
    - es equivalente a la proyección del join natural de *R* y *S* sobre los atributos de *R*:

$$R \ltimes S = \pi_{R.A1,\ldots,R.An} (R \mid\times\mid S)$$

    - es equivalente a la selección sobre R donde la condición de selección es la igualdad de valores de atributos homónimos de *R* y *S*.

$$R \ltimes S = \sigma_{R.ai=S.aj,\ donde\ nombre(ai)=nombre(aj)}(R)$$

    - ***Requiere atributos homónimos***

Álgebra relacional: más operaciones

- División $R \div S$
  - El resultado de $R(A_1, A_2, ..., A_{m+n}) \div S(B_1, B_2, ..., B_n)$ es una relación T con m atributos cuyas extensiones cumplen lo siguiente:
    - La concatenación de todas y cada una de las tuplas de T con todas y cada una de las tuplas de S, resultan tuplas contenidas en R

      $T \times S \subset R$

      $(R \div S) \times S \subset R$

- Outer join, left outer join, right outer join, anti-join,…

Bases de Datos Relacionales
Un posible estado de la BDR de ejemplo (I)

EMPLEADO	NOMBRE	APELLIDO	#SS	FECHA_NCTO	DIRECCIÓN	SEXO	SALARIO	NSS_SUPERV	ND
	Juan	Garrido	123456789	1965-01-09	Atocha, 11 5ºA	H	30000	333445555	5
	Francisco	Rodríguez	333445555	1965-12-08	Castellana, 1 2ºB	H	40000	888665555	5
	Alicia	González	999887777	1968-07-19	Juan Bravo, 9 5ºA	F	25000	987654321	4
	Jennifer	Ramírez	987654321	1941-06-20	Delicias, 10 4ºA	F	43000	888665555	4
	Ramón	Luque	666884444	1962-09-15	Oporto, 2 3ºC	H	38000	333445555	5
	Josefina	Manrique	453453453	1972-07-31	Sepúlveda, 7 5ºA	F	25000	333445555	5
	Adán	López	987987987	1969-03-29	Ocaña, 5 21ºF	H	25000	987654321	4
	Jaime	Pérez	888665555	1937-11-10	Tembleque, 5 4ºA	H	55000	null	1

DEPARTAMENTO	NOMBRED	NÚMEROD	NSS_JEFE	FECHA_INC_JEFE
	Investigación	5	333445555	1988-05-22
	Administración	4	987654321	1995-01-01
	Dirección	1	888665555	1981-06-19

DEPENDIENTE	#SSE	NOMBRE_DEPENDIENTE	SEXO	FECHA_NCTO	PARENTESCO
	333445555	Alicia	M	1986-04-05	HIJA
	333445555	Teodoro	H	1983-10-25	HIJO
	333445555	Julia	M	1958-95.93	ESPOSA
	987654321	Abner	H	1942-02-28	ESPOSA
	123456789	Miguel	H	1988-01-04	HIJO
	123456789	Alicia	M	1988-12-30	HIJA
	123456789	Elísabet	M	1967-05-05	ESPOSA

Bases de Datos Relacionales
Un posible estado de la BDR de ejemplo (II)

LOCALIZACIONES_DEPT	NÚMEROD	LOCALIZACIÓND
	1	Huelva
	4	Salamanca
	5	Barcelona
	5	Sevilla
	5	Barcelona

TRABAJA_EN	#SSE	NP	HORAS
	123456789	1	32,5
	123456789	2	7,5
	666884444	3	40,0
	453453453	1	20,0
	453453453	2	20,0
	333445555	2	10,0
	333445555	3	10,0
	333445555	10	10,0
	333445555	20	10,0
	999887777	30	30,0
	987987987	10	10,0
	987987987	10	35,0
	987987987	30	5,0
	987987987	30	20,0
	987987987	20	15,0
	888665555	20	null

PROYECTO	NOMBREP	NÚMEROP	LOCALIZACIÓNP	ND
	ProjectX	1	Barcelona	5
	ProjectY	2	Sevilla	5
	ProjectZ	3	Huelva	5

Ejercicios del Álgebra Relacional
Consultas sobre el ejemplo dado

- Consulta 1
  - *Obtenga el nombre y la dirección de todos los empleados que trabajan en el departamento 'Investigación'*
- Consulta 2
  - *Para cada proyecto localizado en 'Barcelona', obtenga una lista con el número de proyecto, el número de departamento que lo controla, y el apellido, la dirección y la fecha de nacimiento del jefe de dicho departamento*
- Consulta 3
  - *Busque los nombres de los empleados que trabajan en todos los proyectos controlados por el departamento número 5*
- Consulta 4
  - *Prepare una lista con los números de los proyectos en que interviene un empleado cuyo apellido es 'Smith', ya sea como trabajador o como jefe del departamento que controla el proyecto*

Ejercicios del Álgebra Relacional
Soluciones a los ejercicios 1 y 2

1. $\pi_{\text{NOMBRE, APELLIDO, DIRECCIÓN}}(\sigma_{\text{NOMBRED='Investigación'}}($
   $\text{DEPARTMENTO} \mid\times\mid_{\text{NÚMEROD=ND}} \text{EMPLEADO}))$

2. $\pi_{\text{NÚMEROP, NÚMEROD, APPELLIDO, DIRECCIÓN, FECHA\_NCTO}}($
   $\sigma_{\text{LOCALIZACIÓNP='BARCELONA'}}($
   $(\text{PROYECTO} \mid\times\mid_{\text{NÚMEROD=ND}} \text{DEPARTMENTO})$
   $\mid\times\mid_{\text{NSS\_JEFE=\#SS}} \text{EMPLEADO}))$

Contenido

- Terminología y conceptos
- El Modelo de Datos Relacional (MDR)
  - Bases de Datos Relacionales (BDR)
  - Álgebra relacional
- SQL básico
  - DDL (*Data Definition Language*)
  - DML (*Data Manipulation Language*)
- Conexión de aplicaciones Java a bases de datos: JDBC

SQL: conceptos

- SQL = *Structured Query Lenguage*
  - Lenguaje pensado para la consulta de Bases de Datos Relacionales
  - Basado en el cálculo relacional y orientado a tuplas
  - Es un lenguaje declarativo de alto nivel: sólo es necesario especificar cuál es el resultado (∗)
  - La secuencia se deja al optimizador de consultas

(∗) Se debe tomar esta afirmación “con una pizca de sal” puesto que a menudo se da la situación en que una sentencia se ejecutará más eficientemente (y a veces mucho más eficientemente) que otra que es equivalente semánticamente

SQL: conceptos

- Lenguaje de Definición de Datos (DDL)
  - Con sentencias para crear, alterar o borrar elementos de la Base de Datos como tablas atributos, claves, etc.. y definir acceso de control a través de Vistas: CREATE SCHEMA..., CREATE TABLE..., ALTER TABLE..., DROP TABLE..., CREATE VIEW...
- Lenguaje de Manipulación de Datos (DML)
  - Con sentencias para consultar, insertar, actualizar y borrar instancias (o extensiones) a la Base de Datos: SELECT... FROM... WHERE... , INSERT INTO..., UPDATE... SET... WHERE, DELETE FROM... WHERE...

SQL-DDL
CREATE SCHEMA

- Creación del nombre del esquema y autorización a usuarios

Schema ::= **CREATE SCHEMA** name
AUTHORIZATION user
[*schema-element-list*]

schema-element-list ::=
base-table-definition
view-definition
grant-operation

- Ejemplo
  - Crear el esquema relacional EMPRESA con autorización al usuario JSMITH

```
CREATE SCHEMA EMPRESA AUTHORIZATION JSMITH;
```

(**CREATE DATABASE** en MySQL)

SQL-DDL
CREATE TABLE - Declaración de tabla

- Creación del nombre de cada una de las relaciones (tablas) del esquema

base-table-definition ::= **CREATE TABLE** base-table
({*base-table-element-commalist*}$^{+}$ [*key-spec-commalist*]
[*constraint-spec-commalist*] **)**

base-table-element ::= *column-definition*
| *unique-constraint-definition*

column-definition ::= *column-data-type* [**NOT NULL** [**DISTINCT**]]

unique-constraint-definition ::= {*column-commalist*}$^{+}$

column-data-type ::= { **CHAR**[**ARACTER**] | **VARCHAR** } [(length)]
| **NUMERIC** [{precision [,*scale*]}]
| **DEC**[**IMAL**][{precision [,scale]}]
| **INTEGER** | **INT** | **SMALLINT** | **FLOAT** [{precision}] | **REAL**
| **DOUBLE PRECISION** | **INTERVAL** | **DATE**
| **TIMESTAMP** | **MONEY** | **TIME** [**WITH TIME ZONE**]

- Ejemplo
  - Crear la relación PROYECTO con NOMBRE (no nulo), NUMEROP (no nulo), LOCALIZACIÓNP y NUMD

```
CREATE TABLE PROYECTO (NOMBRE VARCHAR(15) NOT NULL, NUMEROP INT NOT NULL,
LOCALIZACIONP VARCHAR(15), NUMD INT);
```

SQL-DDL
CREATE TABLE - Declaración de Claves

- Al crear la tabla, se pueden declarar las claves primarias, columnas con valores únicos y las claves externas

> *key-spec* ::= **PRIMARY KEY (** {*column-commalist*}$^{+}$**)**
> | **UNIQUE (**column**)**
> | **FOREIGN KEY (**column**) REFERENCES** table**(**column**)**

- Ejemplo
  - Crear la relación PROYECTO con NOMBRE (no nulo), NUMEROP (no nulo), LOCALIZACIÓNP y NUMD, definiendo como clave primaria NUMEROP, que el atributo NOMBRE no se repita y que el atributo NUMD sea una clave externa de DEPARTAMENTO

```
CREATE TABLE PROYECTO (
    NOMBRE VARCHAR(15) NOT NULL, NUMEROP INT NOT NULL,
    LOCALIZACIONP VARCHAR(15), NUMD INT,
PRIMARY KEY (NUMEROP),
UNIQUE (NOMBRE),
FOREIGN KEY (NUMD) REFERENCES DEPARTAMENTO(NUMEROD));
```

SQL-DDL
DROP TABLE

- Se pueden borrar las tablas que previamente se hayan definido tanto a nivel intencional (la estructura de la base de datos) como extensional (su contenido)

> *drop-table-definition* ::= **DROP TABLE** column [**CASCADE** | **RESTRICT**]

  - CASCADE: Con esta opción, todas las restricciones y vistas que hagan referencia a la tabla, se desecharán automáticamente
  - RESTRICT: la tabla sólo se desecha si no se hace referencia a ella en ninguna restricción

- Ejemplo
  - Borrar la tabla EMPRESA

```
DROP TABLE EMPRESA CASCADE;
```

SQL-DDL
ALTER TABLE

- Si queremos modificar la estructura interna de la tabla a nivel intencional, es decir añadir o quitar atributos

alter-table-definition ::=	**ALTER TABLE** table
	{ *add-dec*
	| *drop-dec*
	| … (hay más posibilidades)
add-dec ::=	**ADD** column *column-data-type*
drop-dec ::=	**DROP** column [**CASCADE**|**RESTRICT**]

- Ejemplo
  - Añadir a la tabla *empleado* el atributo *puesto*

```
ALTER TABLE EMPLEADO ADD PUESTO VARCHAR(12);
```

SQL-DML

- Lenguaje de Manipulación de Datos (DML): Confiere las capacidades relacionales

- Son pocas las sentencias pero muy potentes
  - **ELECT... FROM... WHERE...**
  - **INSERT ... INTO...**
  - **UPDATE... SET... WHERE...**
  - **DELETE ... FROM... WHERE...**

Introducción a SQL –DML (1/2)

- Resulta fácil deducir de las sentencias SQL las operaciones del álgebra relacional
  - **Proyección** (π): la cláusula **SELECT**
  - **Producto Cartesiano** (**X**): la cláusula **FROM**
  - **Condición de selección** (σ): la cláusula **WHERE**
  - **Equi-join** (**|X|**): la cláusula **FROM** (el producto Cartesiano) junto con la cláusula **WHERE** que implica condiciones de igualdad sobre atributos compatibles o incluso homónimos. (en SQL, la condición de unión debe darse explícitamente, incluso cuando hay atributos homónimos iguales)

- Una tabla se corresponde con una relación
  - **SELECT-FROM-WHERE**
    - genera una nueva tabla vía producto Cartesiano, selección y proyección
  - **INSERT**, **UPDATE** y **DELETE**
    - modifica la tabla original insertando, actualizando o borrando tuplas

Introducción a SQL –DML (2/2)

- SQL-DML no basado del todo en álgebra relacional
  - las tablas no son exactamente relaciones
  - algunos teoremas del álgebra relacional no se aplican
- Diferencias entre SQL y el álgebra relacional
  - las tablas SQL pueden tener filas duplicadas
    - son multiconjuntos mientras que relaciones son conjuntos
    - los duplicados se eliminan con el operador SQL **`DISTINCT`**
  - SQL **`null`** no es un valor
    - la lógica de SQL es de tres valores (true, false, desconocido)
  - las tablas SQL tienen que tener al menos una columna
    - el álgebra relacional usa la noción de relaciones de grado cero
  - el orden de las columnas en las tablas SQL es significativo
    - producto cartesiano y unión no son conmutativas en SQL
  - …

SQL-DML: SELECT
SELECT FROM WHERE

- Selecciona tuplas de una o varias relaciones (enlazadas por producto cartesiano) que cumplan una condición (la expresada en la cláusula **WHERE**)

```
select-statement ::=
            SELECT <atributos y lista de funciones>
            FROM <lista de tablas>
            [WHERE <condición>]
            [GROUP BY <atributo(s) de agrupación>]
            [HAVING <condición de agrupación>]
            [ORDER BY <lista de atributos>]
```

SQL-DML: SELECT
SELECT FROM WHERE (I)

- Consulta 1 (selección sobre una relación)
  - Recupere la fecha de nacimiento y dirección del empleado(s) cuyo nombre es 'John Smith'

```
SELECT FECHA_NAC, DIRECCIÓN
FROM EMPLEADO
WHERE NOMBRE='John' AND APELLIDO='Smith';
```

- Consulta 2 (selección con producto cartesiano)
  - De cada proyecto ubicado en 'Barcelona', haga una lista con el número de proyecto, el número del departamento controlador y el apellido, dirección y fecha de nacimiento del jefe de departamento

```
SELECT NUMEROP, NUMD, APELLIDO, DIRECCION, FECHA_NAC
FROM PROYECTO, DEPARTAMENTO, EMPLEADO
WHERE NUMD=NUMEROD AND
      NSS_JEFE=NSS AND
      LOCALIZACIONP='Barcelona';
```

SQL-DML: SELECT
SELECT sin WHERE (II)

- Consulta 3
  - Recupere todos los números de seguridad social y los nombres de los departamentos en los que trabaje cualquier empleado de la empresa

```
SELECT NSS, NOMBRED
FROM EMPLEADO, DEPARTAMENTO
WHERE DNUM = DN;
```

SQL-DML: SELECT
SELECT sin WHERE

- En SQL se puede omitir la cláusula WHERE
  - Indica una selección incondicional de tuplas
    - no se aplica el operador σ
    - si intervienen más de una relación, se aplica el operador producto cartesiano (**X**)

- Consulta 4
  - Liste los nombres completos de los empleados junto con sus números de seguridad social

```
SELECT NOMBRE, APELLIDO, #SS
FROM EMPLEADO;
```

SQL-DML: SELECT
Uso del asterisco (*)

- La cláusula SELECT puede ir seguida de asterisco
  - Significa que escogemos todos los atributos de las relaciones de la selección

- Consulta 5 (uso del asterisco)
  - Obtenga los valores de todos los atributos de los empleados que trabajan en el departamento de 'Investigación'

```
SELECT *
FROM EMPLEADO, DEPARTAMENTO
WHERE NOMBRED='Investigación' AND ND=NUMEROD;
```

SQL-DML: SELECT
Otras cláusulas

- [**GROUP BY** <atributo(s) de agrupación>]
  - especifica los atributos que se utilizarán para agrupar (tuplas con el mismo valor de estos atributos estarán en la misma fila agrupada)
- [**HAVING** <condición de agrupación>]
  - filtra las filas agrupados (como WHERE filtra filas); una fila agrupada solo se incluirá en el resultado si satisface la condición
- [**ORDER BY** <lista de atributos>]
  - especifica que los resultados de una petición se ordenarán de acuerdo al valor de los atributos especificados; ascendente (**ASC**) por defecto o descendiente (**DESC**).

SQL-DML : INSERT
Añadir una tupla o varias

- En su forma más simple **INSERT** sirve para añadir una tupla (o más) a una relación; esas tuplas se especifican explícitamente.

```
insert-statement ::= INSERT INTO table [(column-commalist)]
                         {VALUES(insert-atom-commalist)}
```

  - Los valores deben enumerarse en el mismo orden (*) en que se especificaron los atributos
    - el *column-commalist* puede usarse para evitar la especificación explícita de valores NULL; el hecho de omitir un atributo de la lista significa que hay que insertar el valor NULL en la columna correspondiente

- Inserción 1 (inserción simple)
  - Inserte el empleado Richard Marini (con los valores obligatorios)

```
INSERT INTO EMPLEADO(NOMBRE, APELLIDO, ND, NSS)
VALUES ('Richard', 'Marini', 4, '653298653');
```

* El orden es el en el que se declararon las columnas a la hora de crearlas con CREATE TABLE o CREATE VIEW

SQL-DML : UPDATE
Actualización de tuplas

- Con **UPDATE** se pueden cambiar los valores de los atributos de las tuplas que satisfacen la cláusula **WHERE** de la manera especificada en la cláusula **SET**

```
update-statement ::= UPDATE table
                     SET assignment-commalist
                       [where-clause]
```

  - Se seleccionan las tuplas de una sola relación
    - Pero existe el concepto de ***disparo referencial*** :
      la actualización de un valor de clave primaria puede propagarse a los valores de clave externa de otras relaciones

- Actualización 1 (actualización de una sola tupla)
  - Cambie el lugar y el número del departamento controlador del proyecto número 10 a 'Sevilla' y 5 respectivamente

```
UPDATE PROYECTO
SET LOCALIZACIONP='Barcelona', DNUM=5
WHERE NUMEROP=10;
```

SQL-DML : DELETE
Borrado de tuplas

- DELETE elimina tuplas de la relación

```
delete-statement ::= DELETE FROM table [where-clause]
```

  - si el resultado de la cláusula **WHERE** es **TRUE**, se pasa la tupla a la cláusula **DELETE**
  - la omisión de la cláusula **WHERE** indica que se deben eliminar todas las tuplas de la relación
  - el borrado puede propagarse por ***disparo referencial***
    - así se evita invalidar las ***restricciones de integridad referencial*** del DDL

SQL-DML : DELETE
Borrado de tuplas

- Borrado 1 (borrado simple)
  - Borre el empleado de NSS='123456789'

```
DELETE FROM EMPLEADO
WHERE NSS='123456789';
```

- Borrado 2 (borrado múltiple)
  - Borre los empleados del departamento de investigación

```
DELETE FROM EMPLEADO
WHERE ND IN (SELECT NUMEROD FROM DEPARTAMENTO
WHERE NOMBRED='Investigación');
```

- Borrado 3 (vaciado de la tabla)
  - Borre todos los empleados

```
DELETE FROM EMPLEADO;
```

SQL-DML : funciones de conjuntos

- Utilizado con peticiones con agrupación (**`GROUP BY`**)
  - también peticiones con agregación
- Syntaxis: ***`función`*** **`( [DISTINCT|ALL]`** ***`columna`*** **`)`**
  donde ***`función`*** es uno de:
  - **`COUNT`**: contar filas (resultado es un entero)
    - **`COUNT(*)`**: cuenta todas las filas
  - **`SUM`**: suma aritmética de valores numéricos de ***`columna`***
  - **`AVG`**: media aritmética de valores numéricos de ***`columna`***
    - igual a **`SUM/COUNT`**.
  - **`MIN`**: valor mínimo encontrado en ***`columna`***
  - **`MAX`**: valor máximo encontrado en ***`columna`***

  el tipo del resultado es el de la columna salvo para **`COUNT`**
- Se saltan los valores null
- Los especificadores **`DISTINCT`** y **`ALL`** son opcionales.
  - **`ALL`** (defecto): utiliza todos los valores no nulos
  - **`DISTINCT`**: utiliza solo los valores distintos (salta los duplicados)

SQL-DML: funciones de conjuntos, ejemplos

- Función de conjuntos 1, **SUM**
  - Liste los nombres completos de los empleados junto con el número total de horas que cada uno de ellos trabaja en proyectos

```
SELECT NOMBRE, APELLIDO, SUM(HORAS)
FROM EMPLEADO, TRABAJA_EN
WHERE #SS_EMP = #SS
GROUP BY #SS;
```

- Función de conjuntos 2, **COUNT**
  - Liste los nombres de los departamentos junto con el número de empleados que trabaja en cada uno de ellos

```
SELECT NOMBRED, COUNT(*)
FROM EMPLEADO, DEPARTAMENTO
WHERE NUMEROD = ND
GROUP BY NOMBRED;
```

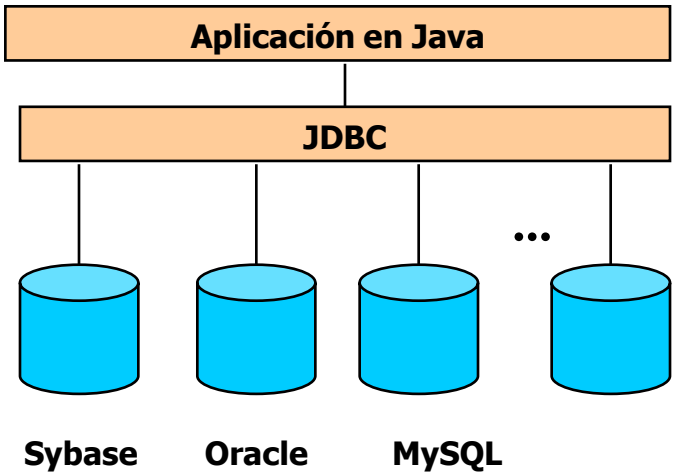
Contenido

- Terminología y conceptos
- El Modelo de Datos Relacional (MDR)
  - Bases de Datos Relacionales (BDR)
  - Álgebra relacional
- SQL básico
  - DDL (*Data Definition Language*)
  - DML (*Data Manipulation Language*)
- Conexión de aplicaciones Java a bases de datos: JDBC

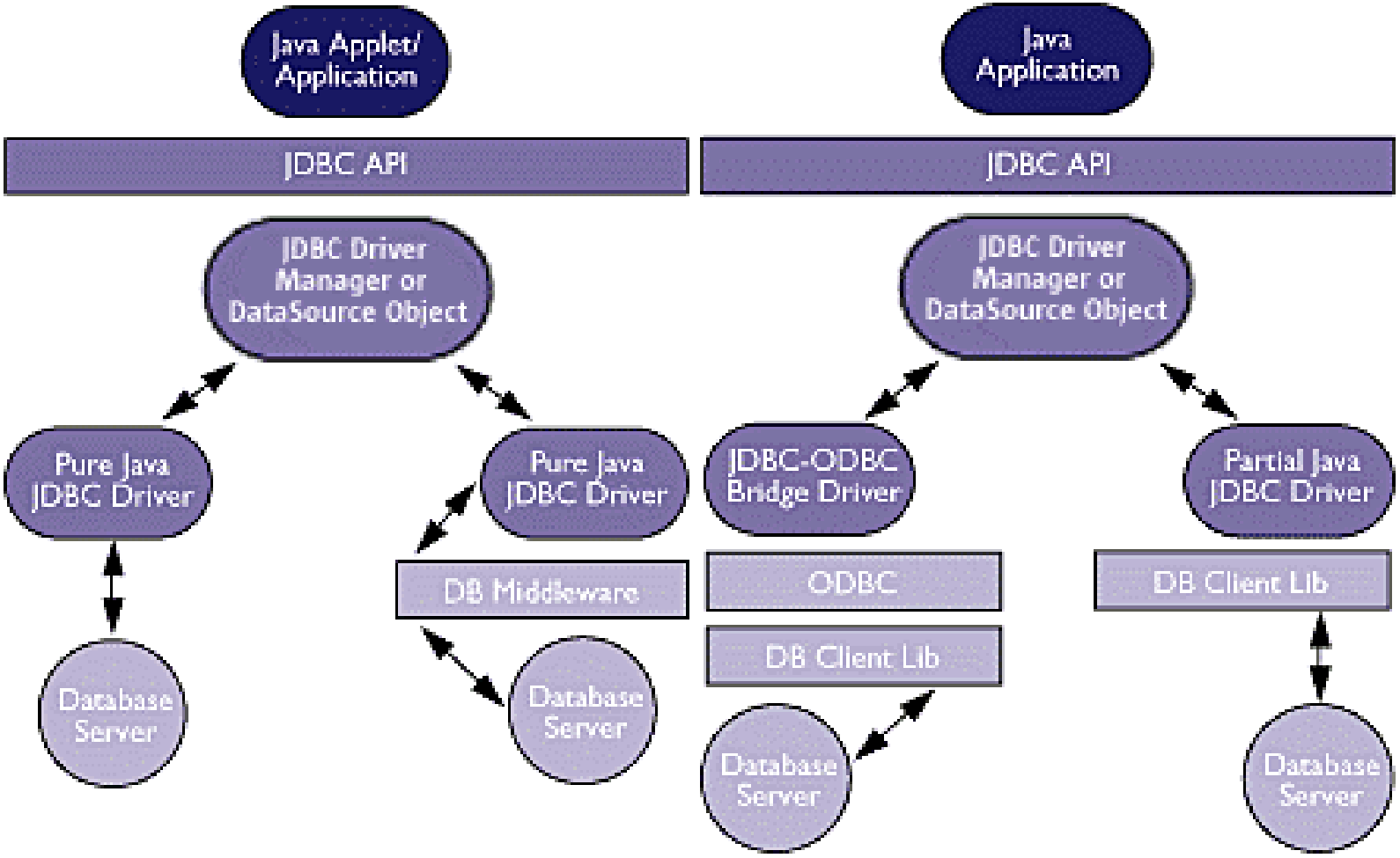
Visión general

JDBC (Java Database Connectivity)

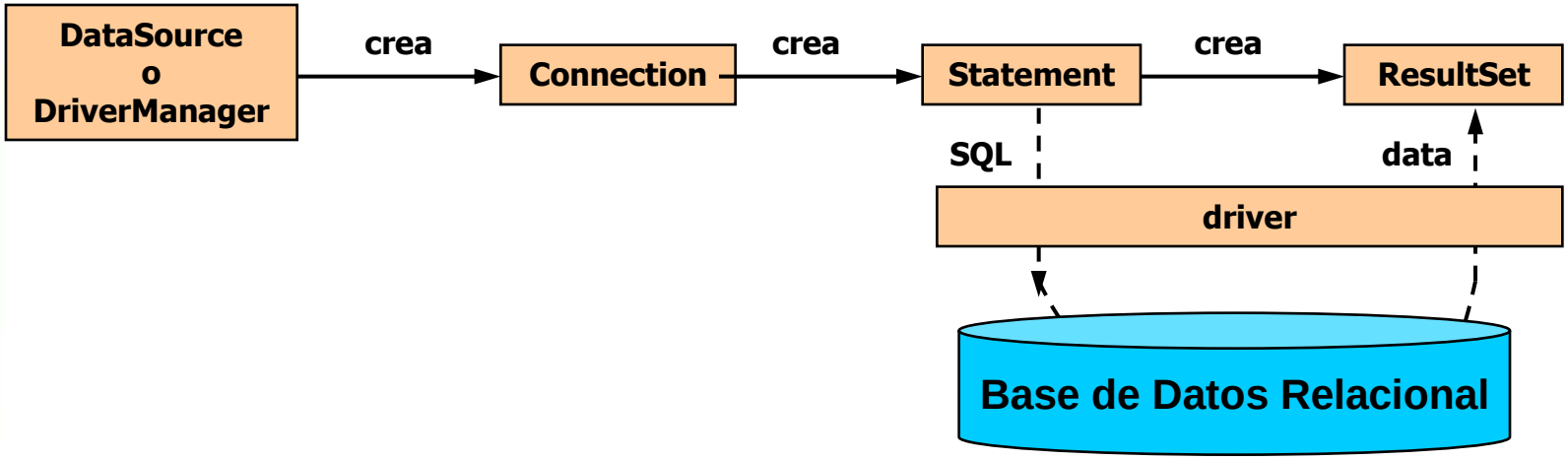
- Es una interfaz para trabajar con BDR que comprende
  - un controlador (*driver*) para la BDR
  - una interfaz de API para interactuar con la BDR
- Es un estándar (de facto) que permite consultar de la misma forma SGBD de varios fabricantes (parecido a ODBC)
- El API JDBC 3.0 (parte de J2SE desde 2002) permite SQL-99
- Reside en **java.sql**, también **javax.sql** (elementos opcionales)



Tipos de *driver* JDBC



Uso de JDBC: pasos básicos



Uso de un DataSource en JDBC

```
// 1. Importar las clases necesarias
import java.sql.*;

// 2. Obtener un objeto DataSource del servidor de nombres
Context context = new InitialContext();
DataSource datasource =
      (DataSource)context.lookup("java:comp/env/jdbc/dbname")

// 3. Crear un objeto Connection
//    (username/password en configuración del DataSource)
Connection con = datasource.getConnection();

// 4. Crear un objeto Statement
Statement stmt = con.createStatement();

// 5. Ejecutar petición,… (e.g. "select * from employee")
ResultSet res = stmt.executeQuery("select * from employee");

// 6. Extraer los datos del objeto ResultSet
while (res.next()) { … }

// 7. Liberar el objeto Connection (sin esperar al recolector
// de basura: una conexión BD es un recurso valioso)
con.close();
```

Uso de un `DataManager` JDBC (anticuado)

- Reemplazar los pasos 2 & 3 por

```
// 2. Cargar el driver JDBC
Class.forName("jdbc.DriverXYZ");
// 3. Identificar el origin de los datos
String url="jdbc:sgbd:db";
// 4. Crear un objeto Connection
Connection con = DriverManager.getConnection(url,usr,pass);
```

- En el paso 2:
  - el método **`forName`** de la clase **`Class`** devuelve el objeto de la clase **`Class`** asociada al nombre proporcionado como parámetro (carga dinámica de clases)
  - la carga de la clase conlleva la creación de una instancia del *driver* que, a continuación se registra con el **`DriverManager`**

Métodos de la clase `Statement`

La clase **`Statement`** contiene los métodos

- **`ResultSet executeQuery(String sql) throws SQLException`**
  - ejecuta un comando SELECT y devuelve los resultados
  - el ResultSet es una tabla con un cursor
- **`int executeUpdate(String sql) throws SQLException`**
  - ejecuta un comando INSERT, UPDATE o DELETE
  - devuelve el número de tuplas insertadas, actualizadas o borradas
  - también utilizado para ejecutar comandos DDL de SQL

Más métodos de la clase `Statement`

La clase **`Statement`** contiene los métodos

- **`int[] executeBatch() throws SQLException`**
  - entrega un lote de comandos a la base de datos
  - devuelve el resultado de ejecutar cada comando (si ejecutan todos con éxito)
  - para añadir un comando al lote actual:

    `void addBatch(String sql) throws SQLException`
- **`ResultSet getResultSet()`**
  - devuelve el resultado de la ejecución de la última sentencia

Véase API para más métodos de la clase **`Statement`**.

Recuperar valores de ResultSet (1/4)

- Tras la consulta hay que obtener los resultados

```
String query="SELECT * FROM EMPLOYEE";
ResultSet res= stmt.executeQuery(query);
while(res.next()){
   String nombre= res.getString("NAME");
   int nd= res.getInt("DN");
   ...
}
```

- El método **`next`** de la clase **`ResultSet`**
  - desplaza el cursor a la siguiente fila
- Véase el API para más métodos de **`ResultSet`**

Recuperar valores de ResultSet (2/4)

- Existen múltiples métodos para recoger los distintos tipos ofrecidos por SQL
  - en las dos diapositivas siguientes aparecen dos tablas
    - filas: los métodos de **`ResultSet`**
    - columnas: los dominios SQL para los que se usan (o se pueden usar)
  - nótese se puede usar **`getString`** para cualquier dominio
    - pero requiere conversión de tipos posterior
  - consúltese el API de Java para más detalles

Recuperar valores de ResultSet (3/4)

	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	DECIMAL	NUMERIC
getByte	X	x	x	x	x	x	x	x	x
getShort	x	X	x	x	x	x	x	x	x
getInt	x	x	X	x	x	x	x	x	x
getLong	x	x	x	X	x	x	x	x	x
getFloat	x	x	x	x	X	x	x	x	x
getDouble	x	x	x	x	x	X	X	x	x
getBigDecimal	x	x	x	x	x	x	x	X	X
getBoolean	x	x	x	x	x	x	x	x	x
getString	x	x	x	x	x	x	x	x	x
getBytes									
getDate									
getTime									
getTimestamp									
getAsciiStream									
getUnicodeStream									
getBinaryStream									
getObject	x	x	x	x	x	x	x	x	x

Una "x" indica que el método getXXX se puede utilizar legalmente para recuperar el tipo JDBC dado
Una "X" indica que el método getXXX está recomendado para recuperar el tipo JDBC dado.

Recuperar valores de ResultSet (4/4)

	CHAR / VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP
getByte	x	x						
getShort	x	x						
getInt	x	x						
getLong	x	x						
getFloat	x	x						
getDouble	x	x						
getBigDecimal	x	x						
getBoolean	x	x						
getString	X	x	x	x	x	x	x	x
getBytes			X	X	x			
getDate	x	x				X		x
getTime	x	x					X	x
getTimestamp	x	x				x	x	X
getAsciiStream	x	X	x	x	x			
getUnicodeStream	x	X	x	x	x			
getBinaryStream			x	x	X			
getObject	x	x	x	x	x	x	x	x

Una "x" indica que el método getXXX se puede utilizar legalmente para recuperar el tipo JDBC dado
Una "X" indica que el método getXXX está recomendado para recuperar el tipo JDBC dado.

Acceso concurrente a la base de datos

- Integridad de datos en presencia de acceso concurrente
  - evitar "*dirty reads*", "*non-repeatable reads*", "*phantom rows*"
  - niveles de aislamiento SQL de ISO-ANSI + "auto-commit" (solo JDBC)
  - puede que el SGBD no soporte todos los niveles
  - niveles más altos: menos problemas pero ejecución más lenta
- Niveles de aislamiento SQL de ISO-ANSI:
  - read uncommitted, *read committed*, *repeatable read*, *serializable*
- En JDBC, se puede cambiar el nivel de isolación con
  - **`Connection.setTransactionIsolation(int level)`**
- Si se cambia el nivel de aislamiento durante una transacción
  - el efecto depende de la implementación del SGBD
- Ver, por ejemplo:
  - **`http://en.wikipedia.org/wiki/Transaction_isolation_level`**

JDBC: más información (1/2)

- Uso de *scroll* con **`ResultSet`**
  - cursor puede moverse en ambas direcciones
  - cursor puede moverse directamente a una fila concreta
- Manipular el resultado con **`RowSet`** en vez de con **`ResultSet`**
  - pertenece a **`javax.sql.rowset`**
  - uso básico parecido al de **`ResultSet`**
  - Permite añadir eventos cuando una tupla se modifica
- Uso de comandos preparados; clase **`PreparedStatement`**
  - componer comandos parametrizados (con "**`?`**")
  - los "**`?`**" se llenan con valores (permitiendo el re-uso de comandos)
- Uso de procedimientos almacenados (*stored procedures*) de la BD; clase **`CallableStatement`**

JDBC: más información (2/2)

- Uso de la interfaz **`DatabaseMetaData`**
  - para consultar información intencional de la BD
  - ver también
    - **`ResultSetMetaData`**
      información sobre tipos y propiedades de las columnas en un objeto **`ResultSet`**
    - **`ParameterMetaData`**
      información sobre tipos y propiedades de las columnas en un objeto **`PreparedStatement`**

- Uso de transacciones vía métodos de la clase **`Connection`**
  - quitar *auto-commit* y hacer *commit* de operaciones explícitamente
  - llamar a *rollback* si se produce una excepción
  - en una transacción distribuida, es el gestor llama a *commit*, etc.

Algunas notas sobre MySQL (1/2)

- Software de dominio público para Unix y Windows
- Recomendación:
  - seguir el tutorial
  - consultar las páginas del manual
  - usar la herramienta para resolver dudas

Algunas notas sobre MySQL (2/2)

- Dispone adicionalmente del tipo **`Datetime`**
  - formato "**`YYYY-MM-DD hh:mm:ss`**"
- Hay muchas operaciones sobre este tipo (también valen algunas de las que operan sobre tipos **`Date`**), p.e.
  - **`to_days(datetime)`** -> **`int`**
    (número de días desde el año 0 del calendario gregoriano)
- Uso de **`AUTO_INCREMENT`** en la declaración de atributos
  - la inserción de una tupla con valor **`NULL`** para un atributo con esa característica denota que el valor se asignará automáticamente
- Versiones > 3 aceptan **`SELECT`** anidados
- > 4 mantienen la integridad referencial con **`DELETE`** & **`UPDATE`**
  - con tal de que se creen correctamente las claves externas
  - con tal de que se utilice el *database engine* "InnoDB"

Bibliografía en línea

- FirstSQL's SQL tutorial
  http://www.firstsql.com/tutor.htm
- MySQL documentation
  http://dev.mysql.com/doc/
- PostgreSQL documentation
  http://www.postgresql.org/docs/

- Sun's Java SE database technologies page
  http://java.sun.com/javase/technologies/database/index.jsp
- Sun's "getting started with the JDBC API"
  http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html
- Sun's JDBC FAQ
  http://java.sun.com/products/jdbc/faq.html
- JDBC chapter from Marty Hall’s “Core Servlets” book
  http://pdf.coreservlets.com/#ch17