BASES DE DATOS RELACIONALES

JESÚS MANUEL JORGE SANTISO

Universidad de La Laguna

PROPÓSITO

- ESTE LIBRO DE APUNTES PRETENDE SER UNA INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES
- HA SIDO ESCRITO POR EL PROFESOR JESÚS MANUEL JORGE SANTISO Y CONTIENE LAS TRANSPARENCIAS UTILIZADAS POR EL AUTOR AL IMPARTIR LA ASIGNATURA DE BASES DE DATOS EN EL GRADO EN INGENIERÍA INFORMÁTICA DE LA UNIVERSIDAD DE LA LAGUNA
- EL MATERIAL CONTENIDO EN EL TEXTO SE CORRESPONDE CON LAS CLASES IMPARTIDAS DURANTE EL CURSO 2020/21

LICENCIA DE USO

- **BASES DE DATOS RELACIONALES**
  - POR JESÚS MANUEL JORGE SANTISO
- ESTE TEXTO ES DE LIBRE DISTRIBUCIÓN (“GRATIS”)
- SE PUEDE DISTRIBUIR A OTRAS PERSONAS LIBREMENTE, SIEMPRE Y CUANDO NO SE MODIFIQUE
- SE DEBE RECONOCER Y DAR CRÉDITO AL AUTOR ORIGINAL (JESÚS MANUEL JORGE SANTISO) SI TODOS O PARTE DE LOS CONTENIDOS DE ESTE TEXTO SE USAN EN LA IMPARTICIÓN DE CURSOS, SEMINARIOS O SIMILARES
- ESTE TEXTO SE DISTRIBUYE "TAL CUAL", SIN GARANTÍA DE NINGÚN TIPO, IMPLÍCITA NI EXPLÍCITA
- SI ENCUENTRA ALGÚN ERROR AGRADECERÍA QUE ME LO COMUNICASE EN LA DIRECCIÓN DE CORREO JJORGE@ULL.EDU.ES

ÍNDICE DE CONTENIDOS

TEMA 1. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS

- PROCESAMIENTO TRADICIONAL DE DATOS
- SISTEMAS DE BASES DE DATOS
- ABSTRACCIÓN E INDEPENDENCIA DE LA INFORMACIÓN
- MODELOS DE DATOS
- LENGUAJES DE BASES DE DATOS
- USUARIOS DE UN SISTEMA DE BASES DE DATOS
- ESTRUCTURA DE UN SISTEMA DE BASES DE DATOS
- ARQUITECTURA DE APLICACIONES

PROCESAMIENTO TRADICIONAL DE DATOS CARACTERÍSTICAS

- SE CONOCE COMO SISTEMAS DE FICHEROS
- PROLIFERACIÓN DE FICHEROS DE DATOS, ESPECÍFICOS PARA UNA DETERMINADA APLICACIÓN
- PROLIFERACIÓN DE PROGRAMAS, ESCRITOS PARA RESOLVER NECESIDADES CONCRETAS
- PROGRAMAS DESARROLLADOS DE FORMA INDEPENDIENTE, POR UNO O VARIOS PROGRAMADORES, POSIBLEMENTE EN DIVERSOS LENGUAJES Y CON HERRAMIENTAS DIVERSAS
- ÉNFASIS PUESTO EN LOS PROGRAMAS – TRATAMIENTO DE LA INFORMACIÓN ORIENTADO A PROCESOS

PROCESAMIENTO TRADICIONAL DE DATOS DESVENTAJAS

- REDUNDANCIA E INCONSISTENCIA DE DATOS
  - AUMENTO DE COSTES DE ALMACENAMIENTO Y ACCESO
  - AUMENTO DE LA PROBABILIDAD DE INCONSISTENCIA DE DATOS
- DIFICULTAD EN EL ACCESO A LA INFORMACIÓN
  - IMPOSIBILIDAD DE ATENDER DE FORMA PRÁCTICA Y EFICIENTE CONSULTAS AL VUELO (NO PREVISTAS)
- AISLAMIENTO DE DATOS
  - DEPENDENCIA DE LOS DATOS DEL FORMATO DE LOS FICHEROS

PROCESAMIENTO TRADICIONAL DE DATOS DESVENTAJAS

- DIFICULTAD EN LA SUPERVISIÓN DEL ACCESO CONCURRENTE
  - POSIBILIDAD DE INCONSISTENCIA DE DATOS DURANTE ACTUALIZACIONES CONCURRENTES
- DIFICULTAD EN LA PROTECCIÓN DE LA CONFIDENCIALIDAD DE LA INFORMACIÓN
  - ES COMPLEJO EVITAR QUE LA INFORMACIÓN ALMACENADA EN LOS FICHEROS DE DATOS RESULTE ACCESIBLE A LOS USUARIOS DE LOS PROGRAMAS
- PROBLEMAS DE INTEGRIDAD
  - LOS DATOS ALMACENADOS DEBEN SATISFACER CIERTOS TIPOS DE CONDICIONES DE INTEGRIDAD O RESTRICCIONES PARA QUE PUEDAN SER CONSIDERADOS FORMALMENTE VÁLIDOS. LOS DESARROLLADORES DE APLICACIONES DEBEN HACER CUMPLIR ESTAS RESTRICCIONES AÑADIENDO EL CÓDIGO APROPIADO EN LOS DIVERSOS PROGRAMAS DE APLICACIÓN
- PROBLEMAS DE ATOMICIDAD EN LAS TRANSACCIONES
  - SI OCURRE UN FALLO EN EL SISTEMA LOS DATOS SE DEBEN DE RESTAURAR AL ESTADO DE CONSISTENCIA ANTERIOR AL FALLO
- DIFÍCIL REESTRUCTURACIÓN DE LA INFORMACIÓN
  - SUPONE GRAVES TRASTORNOS PARA LOS DESARROLLADORES DE APLICACIONES

PROCESAMIENTO TRADICIONAL DE DATOS DESVENTAJAS

- ESTAS DESVENTAJAS DESACREDITAN Y DETERIORAN LA IMAGEN Y CONFIANZA EN EL SISTEMA DE INFORMACIÓN
- LA SOLUCIÓN DE TODAS ESTAS DESVENTAJAS ES INDEPENDIENTE DE LAS APLICACIONES PARTICULARES
- SE PUEDE HACER UNA GESTIÓN DE LA INFORMACIÓN (DATOS) MÁS RACIONAL Y EFICIENTE, CON PRESENTACIONES DE LOS RESULTADOS DE BÚSQUEDA MÁS HOMOGÉNEOS Y CLAROS

SISTEMAS DE BASES DE DATOS

- SISTEMAS ORIENTADOS A LOS DATOS (NO A LOS PROCESOS)
- SURGEN EN UN INTENTO DE RESOLVER LAS DESVENTAJAS DEL PROCESAMIENTO TRADICIONAL DE DATOS
- PERMITEN MANIPULAR DE MANERA ADECUADA GRANDES VOLÚMENES DE DATOS
  - LOS DATOS SON ALMACENADOS Y RECUPERADOS DE FORMA EFICIENTE Y TRANSPARENTE AL USUARIO
- LOS DATOS SE ALMACENAN EN BANCOS (BASES) DE DATOS
  - ESTÁN CONSTITUIDOS POR UNO O VARIOS FICHEROS QUE QUEDAN OCULTOS AL USUARIO FINAL

SISTEMAS GESTORES DE BASES DE DATOS (SGBD)

- SON LOS PROGRAMAS ENCARGADOS DE TODOS LOS ASPECTOS DE CREACIÓN, ACCESO Y ACTUALIZACIÓN DE LA BASE DE DATOS
- SUS FUNCIONES SE AGRUPAN EN TRES CATEGORÍAS:
  - DESCRIPCIÓN O DEFINICIÓN DE DATOS
    - CREACIÓN Y REESTRUCTURACIÓN DE ELEMENTOS DEL SISTEMA (OBJETOS, ATRIBUTOS, RELACIONES, CONDICIONES DE INTEGRIDAD, …)
  - MANIPULACIÓN DE DATOS
    - INSERCIÓN, BORRADO, MODIFICACIÓN Y CONSULTA
  - CONTROL DE DATOS
    - AUTORIZACIONES DE ACCESO, GESTIÓN DE TRANSACCIONES, CONTROL DE CONCURRENCIA, …

ABSTRACCIÓN DE LA INFORMACIÓN E INDEPENDENCIA DE DATOS

- ABSTRACCIÓN DE LA INFORMACIÓN
  - LOS SISTEMAS GESTORES PROPORCIONAN UNA VISIÓN ABSTRACTA DE LA INFORMACIÓN. SE DISTINGUEN 3 NIVELES DE ABASTRACCIÓN:
    - NIVEL FÍSICO O INTERNO – ESQUEMA FÍSICO O INTERNO
    - NIVEL CONCEPTUAL – ESQUEMA CONCEPTUAL
    - NIVEL EXTERNO O DE VISIÓN – ESQUEMA EXTERNO, SUBESQUEMAS O VISTAS
- INDEPENDENCIA DE DATOS
  - CAPACIDAD DE MODIFICAR UN ESQUEMA DE BASE DE DATOS EN UN NIVEL SIN AFECTAR A LA DEFINICIÓN DE LOS ESQUEMAS DE LOS NIVELES SUPERIORES. HAY 2 TIPOS DE INDEPENDENCIA DE DATOS:
    - INDEPENDENCIA LÓGICA DE DATOS
    - INDEPENDENCIA FÍSICA DE DATOS

MODELOS DE DATOS

- EL MODELO DE DATOS ES EL “LENGUAJE” CON QUE EL ANALISTA DESCRIBE EL MODELO CONCEPTUAL QUE HA CREADO EN SU MENTE DESPUÉS DE OBSERVAR LA REALIDAD
- A LA DESCRIPCIÓN OBTENIDA, USANDO EL MODELO DE DATOS, DEL MODELO CONCEPTUAL, SE LE LLAMA ESQUEMA CONCEPTUAL
- CONJUNTO DE HERRAMIENTAS CONCEPTUALES QUE SE UTILIZAN PARA DESCRIBIR LOS DATOS, SUS RELACIONES, SU SEMÁNTICA Y SUS CONDICIONES DE INTEGRIDAD
- EXISTEN NUMEROSOS MODELOS DE DATOS

COMPONENTES DE LOS MODELOS DE DATOS

- ESTRUCTURAS DE DATOS
  - COLECCIONES DE OBJETOS ABSTRACTOS FORMADOS POR DATOS
- OPERADORES SOBRE LAS ESTRUCTURAS
  - CONJUNTO DE OPERADORES, CON REGLAS BIEN DEFINIDAS, QUE PERMITEN MANIPULAR LAS ESTRUCTURAS DE DATOS
- REGLAS DE INTEGRIDAD
  - CONJUNTO DE CONCEPTOS Y PROCEDIMIENTOS QUE EXPRESAN QUE VALORES DE DATOS SON VÁLIDOS EN EL ESQUEMA CONSIDERADO

CATEGORÍAS DE MODELOS DE DATOS

- MODELOS DE DATOS LÓGICOS
  - BASADOS EN OBJETOS
    - MODELO ENTIDAD RELACIÓN (E-R)
  - BASADOS EN REGISTROS
    - MODELO RELACIONAL
- MODELOS DE DATOS FÍSICOS
  - EN DESUSO

MODELO ENTIDAD RELACIÓN (E-R)

- LA REALIDAD SE DESCRIBE A TRAVÉS DE UNA SERIE DE OBJETOS BÁSICOS LLAMADOS ENTIDADES Y DE RELACIONES ENTRE ESOS OBJETOS
- LOS ELEMENTOS GRÁFICOS QUE APARECEN EN UN DIAGRAMA ENTIDAD RELACIÓN SON:
  - RECTÁNGULOS – REPRESENTAN LAS ENTIDADES
    - SIMPLE – ENTIDAD FUERTE
    - DOBLE – ENTIDAD DÉBIL
  - ELIPSES – REPRESENTAN LOS ATRIBUTOS DE UNA ENTIDAD
    - SIMPLE CONTINUO – ATRIBUTOS UNIVALUADOS
    - DOBLE CONTINUO – ATRIBUTOS PLURALES/MULTIVALUADOS/MULTIVALORADOS
    - SIMPLE DISCONTINUO – ATRIBUTOS DERIVADOS
  - ROMBOS – REPRESENTAN LAS RELACIONES ENTRE ENTIDADES
    - SIMPLE – RELACIONES NORMALES
    - DOBLE – RELACIONES DE DEPENDENCIA
  - LIGAS: PERMITEN CONECTAR ATRIBUTOS CON ENTIDADES Y ENTIDADES CON RELACIONES
- SE PUEDE ESPECIFICAR LA CARDINALIDAD DE LAS RELACIONES EN LOS DIAGRAMAS
  - UNO A UNO, UNO A MUCHOS, MUCHOS A UNO Y MUCHOS A MUCHOS

MODELO RELACIONAL

- LOS DATOS Y LAS RELACIONES ENTRE LOS DATOS SE REPRESENTAN MEDIANTE TABLAS O RELACIONES
- LAS TABLAS SON ARREGLOS BIDIMENSIONALES (TIENEN FILAS Y COLUMNAS)
  - LAS COLUMNAS ALMACENAN LOS VALORES DE UN ATRIBUTO
  - LAS FILAS SE LLAMAN T-UPLAS Y REPRESENTAN UNA ASOCIACIÓN DE VALORES CON ALGÚN TIPO DE SIGNIFICADO
  - CADA TABLA DE LA BASE DE DATOS TIENE UN NOMBRE ÚNICO
  - LOS NOMBRES DE LOS ATRIBUTOS DENTRO DE UNA TABLA SON ÚNICOS

LENGUAJES DE BASES DE DATOS

- LENGUAJES DE DEFINICIÓN DE DATOS (DDL – DATA DEFINITION LANGUAGE)
  - TAMBIÉN CONOCIDOS COMO LENGUAJES DE DEFINICIÓN DE ESQUEMAS
- LENGUAJES DE MANIPULACIÓN DE DATOS (DML – DATA MANIPULATION LANGUAGE)
  - TAMBIÉN CONOCIDOS COMO LENGUAJES DE ACCESO/GESTIÓN/MANEJO DE DATOS
- LENGUAJES DE CONTROL DE DATOS (DCL – DATA CONTROL LANGUAGE)

LENGUAJES DE DEFINICIÓN DE DATOS

- SIRVEN PARA ESPECIFICAR EL ESQUEMA DE LA BASE DE DATOS
- REALIZAN TAREAS RELATIVAS A LA ESTRUCTURA LÓGICA DE LA BASE DE DATOS
  - CREAR/BORRAR/MODIFICAR ESTRUCTURAS DE DATOS PARA ALMACENAR INFORMACIÓN
  - ASIGNAR NOMBRES A CAMPOS, DEFINIR SUS TIPOS, LONGITUDES, …
  - DEFINIR RELACIONES ENTRE LOS DATOS
  - DEFINIR CONDICIONES DE INTEGRIDAD
- LA EJECUCIÓN DE LAS SENTENCIAS DEL DDL ACTUALIZAN EL DICCIONARIO O DIRECTORIO DE DATOS

LENGUAJES DE MANIPULACIÓN DE DATOS

- PERMITEN OBTENER ACCESO A LOS DATOS
  - CONSULTA O RECUPERACIÓN DE DATOS
  - INSERCIÓN
  - MODIFICACIÓN
  - BORRADO
- LAS OPERACIONES DE INSERCIÓN/MODIFICACIÓN/BORRADO SE CONOCEN COMO OPERACIONES DE ACTUALIZACIÓN DE DATOS

CLASIFICACIÓN DE LOS DML

- PROCEDIMENTALES
  - EL USUARIO TIENE QUE ESPECIFICAR QUE DATOS QUIERE Y EL PROCEDIMIENTO (SECUENCIA DE OPERACIONES) PARA OBTENERLOS
  - EJ: ÁLGEBRA RELACIONAL
- DECLARATIVOS
  - EL USUARIO SOLO TIENE QUE ESPECIFICAR QUE DATOS QUIERE
  - EJ: CÁLCULO RELACIONAL DE T-UPLAS Y DE DOMINIOS

CLASIFICACIÓN DE LOS DML

- CONVERSACIONALES
  - SE UTILIZAN DE FORMA INTERACTIVA Y SÍNCRONA DESDE UN TERMINAL
  - POR CADA SENTENCIA EJECUTADA EL SISTEMA PRESENTA LOS RESULTADOS
- DIFERIDOS
  - LAS SENTENCIAS DEBEN ESTAR AGRUPADAS FORMANDO UN PROGRAMA O PROCEDIMIENTO
  - AL EJECUTARSE EL PROGRAMA SE PRESENTAN LOS RESULTADOS

CLASIFICACIÓN DE LOS DML

- HUÉSPED
  - LAS SENTENCIAS DEL DML (HUÉSPED O INVITADO) DEBEN ESTAR EMBEBIDAS/INMERSAS/INCRUSTADAS/EMPOTRADAS DENTRO DE UN PROGRAMA ESCRITO EN OTRO LENGUAJE DE PROPÓSITO GENERAL (COMO C, JAVA, …) QUE ACTÚA COMO ANFITRIÓN
  - LAS INSTRUCCIONES DEL DML NECESITAN SER EJECUTADAS DESDE EL LENGUAJE ANFITRIÓN. ESTO SE PUEDE LOGRAR DE DOS FORMAS:
    - EXTENDIENDO LA SINTAXIS DEL LENGUAJE HUÉSPED. PARA ELLO SE NECESITAN PRECOMPILADORES ESPECÍFICOS
    - PROPORCIONANDO UN API ESPECÍFICO PARA INVOCAR LA FUNCIONALIDAD DEL LENGUAJE HUÉSPED
      - ODBC – OPEN DATA BASE CONNECTIVITY
      - JDBC – JAVA DATA BASE CONNECTIVITY
- INDEPENDIENTES
  - EL DML NO NECESITA APOYARSE EN NINGÚN OTRO LENGUAJE PARA CREAR PROCEDIMIENTOS O PROGRAMAS

TIPOS DE PROCESAMIENTO DE DATOS

- PROCESAMIENTO OLTP
  - ONLINE TRANSACTION PROCESSING – PROCESAMIENTO DE TRANSACCIONES EN TIEMPO REAL
  - ES UN TIPO DE PROCESAMIENTO EN EL QUE PREDOMINAN LAS OPERACIONES DE ACTUALIZACIÓN (INSERCIÓN, MODIFICACIÓN Y/O BORRADO) SOBRE LAS DE CONSULTA
  - SE CORRESPONDE CON EL FUNCIONAMIENTO DÍA A DÍA DE LA EMPRESA
  - SE UTILIZAN BASES DE DATOS RELACIONALES FUERTEMENTE NORMALIZADAS
- PROCESAMIENTO OLAP
  - ONLINE ANALYTICAL PROCESSING – PROCESAMIENTO ANALÍTICO EN TIEMPO REAL
  - ES UN TIPO DE PROCESAMIENTO DE DATOS EN EL QUE PREDOMINAN LAS CONSULTAS SOBRE GRANDES VOLÚMENES DE DATOS CON EL FIN DE ANALIZAR LA INFORMACIÓN
  - NO SUELE HABER OPERACIONES DE ACTUALIZACIÓN, SALVO INSERCIONES INCREMENTALES
  - SE UTILIZAN BASES DE DATOS MULTIDIMENSIONALES (CUBOS OLAP)

USUARIOS DE UN SISTEMA DE BASES DE DATOS

- OPERADORES DE LOS PROGRAMAS DE APLICACIONES
  - GRUPO DE USUARIOS MAS NUMEROSO
  - PROCESAMIENTO OLTP
- DESARROLLADORES DE PROGRAMAS DE APLICACIONES
  - UTILIZAN DML, EN MODO HUÉSPED O INDEPENDIENTE
- ADMINISTRADOR DE LA BASE DE DATOS (DBA)
  - UTILIZAN DDL Y DCL FUNDAMENTALMENTE
  - ENCARGADOS DEL DISEÑO Y MANTENIMIENTO RUTINARIO DE LA BASE DE DATOS
- ANALISTAS DE DATOS
  - PROCESAMIENTO OLAP

ESTRUCTURA DE UN SISTEMA DE BASES DE DATOS

- GESTOR DE ALMACENAMIENTO
  - PROPORCIONA LA INTERFAZ ENTRE LOS DATOS DE BAJO NIVEL DE LA BASE DE DATOS Y LOS PROGRAMAS DE APLICACIONES
  - SUS PRINCIPALES COMPONENTES SON:
    - GESTOR DE ARCHIVOS
    - GESTOR DE MEMORIA INTERMEDIA
    - GESTOR DE TRANSACCIONES
    - GESTOR DE AUTORIZACIONES E INTEGRIDAD
- PROCESADOR DE CONSULTAS
  - INTÉRPRETE DDL/DCL – INTERPRETA LAS INSTRUCCIONES DDL/DCL Y REGISTRA LAS DEFINICIONES EN EL DICCIONARIO DE DATOS
  - COMPILADOR DML – TRADUCE CADA INSTRUCCIÓN DML A UN PLAN DE EVALUACIÓN (SECUENCIA DE OPERACIONES EN BAJO NIVEL)
  - MOTOR DE EVALUACIÓN DE CONSULTAS – EJECUTA LOS PLANES DE EVALUACIÓN GENERADOS POR EL COMPILADOR DML

ARQUITECTURA DE APLICACIONES

- ARQUITECTURA CLIENTE/SERVIDOR
  - DOS CAPAS
    - MÁQUINA CLIENTE – APLICACIONES
    - SERVIDOR DEL SISTEMA DE BASES DE DATOS
  - TRES CAPAS
    - MÁQUINA CLIENTE – FRONT END/FORMULARIOS
    - SERVIDOR DE APLICACIONES
    - SERVIDOR DEL SISTEMA DE BASES DE DATOS

TEMA 2. EL MODELO RELACIONAL

- INTRODUCCIÓN
- CONCEPTOS BÁSICOS DEL MODELO RELACIONAL
- VALORES NULOS
- CONDICIONES DE INTEGRIDAD
- ÁLGEBRA RELACIONAL
- CÁLCULO RELACIONAL

INTRODUCCIÓN

- FUE PROPUESTO POR EDGAR FRANK CODD (1923-2003) EN 1970
- SU TRABAJO RECIBIÓ EL PRESTIGIOSO PREMIO TURING DE LA ACM (ASSOCIATION OF COMPUTING MACHINERY)
- A LO LARGO DE LA DÉCADA DE 1970 SE FUERON DESARROLLANDO SISTEMAS GESTORES DE BASES DE DATOS BASADOS EN EL MODELO RELACIONAL
  - EL PRIMER PROTOTIPO FUE EL SISTEMA R DESARROLLADO POR EL IBM SAN JOSE RESEARCH LABORATORY
- DESDE SU ESCALADA EN EL DOMINIO EN LA DÉCADA DE 1980, EL MODELO RELACIONAL HA MANTENIDO SU SUPREMACÍA SOBRE LOS DEMÁS MODELOS DE DATOS
  - ES UN HECHO INSÓLITO EN EL MUNDO DE LA INFORMÁTICA

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- UNA BASE DE DATOS RELACIONAL CONSISTE EN UN CONJUNTO DE **TABLAS** O **RELACIONES,** A CADA UNA DE LAS CUALES SE LE ASIGNA UN NOMBRE ÚNICO
- DE MANERA INFORMAL UNA TABLA PUEDE VERSE COMO UN ARREGLO BIDIMENSIONAL (MATRIZ), DONDE LAS FILAS SE CORRESPONDEN CON **REGISTROS** (**T-UPLAS**) Y LAS COLUMNAS CON LOS **ATRIBUTOS**
- UNA TABLA TIENE UN NÚMERO FINITO DE FILAS Y COLUMNAS
  - EL ORDEN DE LAS FILAS SE CONSIDERA IRRELEVANTE
  - TODAS LAS FILAS TIENEN QUE SER DISTINTAS
  - EL ORDEN DE LAS COLUMNAS ES IMPORTANTE
- CADA T-UPLA REPRESENTA UNA UNA ASOCIACIÓN O RELACIÓN ENTRE UN CONJUNTO DE VALORES
- LOS NOMBRES DE LOS ATRIBUTOS DENTRO DE UNA TABLA SON ÚNICOS
- CADA ATRIBUTO TIENE ASOCIADO UN **DOMINIO,** ES DECIR, UN CONJUNTO DE VALORES PERMITIDO QUE ASUMIREMOS QUE SON **ATÓMICOS** (INDESCOMPONIBLES)

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- FORMALMENTE UNA RELACIÓN SE DEFINE COMO UN SUBCONJUNTO FINITO DEL PRODUCTO CARTESIANO DE LOS DOMINIOS DE SUS ATRIBUTOS ($D_1 \times D_2 \times \ldots \times D_N$)
- UN RELACIÓN SE COMPONE DE SU **ESQUEMA** (**INTENSIÓN**) Y DE SU CONTENIDO (**EXTENSIÓN**)
  - EL ESQUEMA SE SUPONE INVARIANTE EN EL TIEMPO
  - LA EXTENSIÓN VARÍA CONSTANTEMENTE
- EL **ESQUEMA DE UNA RELACIÓN** CONSISTE EN EL NOMBRE DE LA RELACIÓN Y UNA LISTA DE LAS DEFINICIONES DE SUS ATRIBUTOS
  - SE REPRESENTA PONIENDO EL NOMBRE DE LA RELACIÓN Y ENTRE PARÉNTESIS, SEPARADOS POR COMAS, SUS ATRIBUTOS, CON SUS CORRESPONDIENTES DOMINIOS SEPARADOS DEL NOMBRE DEL ATRIBUTO POR AL MENOS UN ESPACIO EN BLANCO.
    - EJ: EMPLEADO(DNI NUMBER, NOMBRE CHAR(10), FECHA_ALTA DATE)
  - LA DEFINICIÓN EXACTA DEL DOMINIO DE CADA ATRIBUTO NO ES RELEVANTE HASTA QUE NO SE IMPLEMENTE CON UN DDL, POR LO CUAL LA OMITIREMOS DE MOMENTO
    - EJ: EMPLEADO(DNI, NOMBRE, FECHA_ALTA)

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- LOS DOMINIOS DE UN ATRIBUTO PUEDEN SER:
  - CUALITATIVOS
    - SIN CARÁCTER NUMÉRICO INTRÍNSECO
    - EJ: NOMBRE, SEXO {HOMBRE, MUJER}, ESTADO CIVIL {SOLTERO, CASADO, DIVORCIADO, VIUDO}, …
  - CUANTITATIVOS
    - ATRIBUTOS DE TIPO NUMÉRICO
    - EJ: ESTATURA, EDAD, SALARIO, …

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- EL **ESQUEMA DE UNA BASE DE DATOS** ESTÁ FORMADO POR LAS ESQUEMAS DE CADA UNA DE SUS TABLAS
  - CLIENTE(DNI, NBC, CDC)
  - CUENTA(CS, NC, DNI, SLD)
  - PRÉSTAMO(CS, NP, DNI, I)
  - SUCURSAL(CS, NS, CDS)
- PONER ATRIBUTOS COMUNES EN LOS ESQUEMAS DE RELACIONES ES UNA FORMA “NATURAL” DE RELACIONAR T-UPLAS DE RELACIONES DISTINTAS

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- EL GRADO DE UNA T-UPLA ES EL NÚMERO DE COMPONENTES O VALORES QUE TIENE
- EL GRADO DE UNA RELACIÓN ES EL NÚMERO DE ATRIBUTOS QUE PERTENECEN A SU ESQUEMA Y COINCIDE CON EL GRADO DE SUS T-UPLAS
- DOS T-UPLAS SON IGUALES CUANDO LOS VALORES DE TODAS SUS COMPONENTES COINCIDEN
- UNA VARIABLE DE TIPO T-UPLA ES UNA VARIABLE QUE REPRESENTA UNA T-UPLA
- LAS VARIABLES DE TIPO T-UPLA TIENEN COMO DOMINIO UNA TABLA
  - SE UTILIZA LA NOTACIÓN t[NS] PARA DENOTAR EL VALOR DE t EN EL ATRIBUTO NS
  - DE MANERA ALTERNATIVA t[1] O t_1 DENOTA EL VALOR DE T EN EL PRIMER ATRIBUTO DE LA RELACIÓN A LA QUE PERTENECE t
- UNA VARIABLE DE TIPO DOMINIO ES UNA VARIABLE QUE REPRESENTA UN VALOR DEL DOMINIO DE UN ATRIBUTO

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL

- TAMBIÉN PODEMOS REPRESENTAR UNA T-UPLA MEDIANTE LA NOTACIÓN: $<t_1, ..., t_N>$
- LAS VARIABLES $t_1, ..., t_N$ SON VARIABLES DE TIPO DOMINIO
- LAS VARIABLES SE PUEDEN CUANTIFICAR EXISTENCIAL (∃) Y UNIVERSALMENTE (∀) Y EXPRESAR PREDICADOS LÓGICOS CON ELLAS
  - DOM(t) = CUENTA; (∃ t) (T[SLD] > 100000)
  - DOM(t) = CUENTA; (∀ t) (T[SLD] > 100)

SUPERCLAVE

- UN CONJUNTO DE ATRIBUTOS K ES UNA SUPERCLAVE DE UNA RELACIÓN R CUANDO PERMITE IDENTIFICAR DE FORMA ÚNICA A LAS T-UPLAS DE R, ES DECIR, TODAS LAS T-UPLAS DE R TIENEN VALORES DISTINTOS AL SER EVALUADAS EN LA SUPERCLAVE
  - TODAS LAS TABLAS TIENEN AL MENOS UNA SUPERCLAVE DADO QUE TODAS LAS FILAS DE LA TABLA TIENEN QUE SER DISTINTAS
  - EN EL PEOR DE LOS CASOS, LA SUPERCLAVE ES EL CONJUNTO DE TODOS LOS ATRIBUTOS DE LA TABLA

CLAVE

- UN CONJUNTO DE ATRIBUTOS K ES UNA CLAVE DE UNA RELACIÓN R SI:
  - K ES SUPERCLAVE DE R
  - K ES MINIMAL EN EL SENTIDO DE LA INCLUSIÓN DE CONJUNTOS
- TODAS LAS TABLAS TIENEN AL MENOS UNA CLAVE (PUESTO QUE TIENEN AL MENOS UNA SUPERCLAVE)
- NO TODAS LAS CLAVES TIENEN EL MISMO CARDINAL (MISMO NÚMERO DE ATRIBUTOS)
- A LOS ATRIBUTOS DE CUALQUIER CLAVE SE LES LLAMA ATRIBUTOS PRINCIPALES

CLAVE PRIMARIA (PRIMARY KEY)

- A LA CLAVE QUE SE USARÁ HABITUALMENTE PARA IDENTIFICAR LAS T-UPLAS DE LA RELACIÓN SE LE LLAMA **CLAVE PRIMARIA** (**PRINCIPAL**)
- A LAS DEMÁS CLAVES DE LA TABLA SE LES CONOCE COMO **CLAVES ALTERNATIVAS** (**SECUNDARIAS**)

CLAVE PRIMARIA (PRIMARY KEY)

- EXISTEN UNA SERIE DE FACTORES A TENER EN CONSIDERACIÓN A LA HORA DE SELECCIONAR LA CLAVE PRIMARIA DE UNA TABLA
  - ESTABILIDAD: ESTA PROPIEDAD HACE REFERENCIA A QUE LOS VALORES DE LA CLAVE NO CAMBIEN CON EL TIEMPO
  - SIMPLICIDAD: SE PREFIEREN CLAVES SENCILLAS, CON POCOS ATRIBUTOS Y DE TIPO NUMÉRICO PREFERIBLEMENTE A CLAVES LARGAS DE TIPO ALFANUMÉRICO
  - FIABILIDAD: EXISTE ALGÚN MECANISMO DE VALIDACIÓN O DE AUTODETECCIÓN O CORRECCIÓN DE ERRORES
  - UNIVERSALIDAD: INDICA QUE SU USO Y CONOCIMIENTO ESTÁ MUY EXTENDIDO (DNI EN EL CASO DE LAS PERSONAS, ISBN EN EL CASO DE LOS LIBROS, …)

CLAVE AJENA (FOREIGN KEY)

- UN CONJUNTO DE ATRIBUTOS SE DICE QUE FORMAN UNA **CLAVE EXTERNA** (**AJENA**) EN UNA TABLA R SI HACEN REFERENCIA (RELACIÓN DE EXISTENCIA) A UNA CLAVE DE UNA TABLA S (NO NECESARIAMENTE DISTINTA DE R)
- LOS DOMINIOS DE LOS ATRIBUTOS DE LA CLAVE DE S Y DE LA CLAVE AJENA EN R DEBEN COINCIDIR
- LA TABLA QUE CONTIENE LA CLAVE AJENA SE DENOMINA TABLA HIJA (REFERENCIANTE) Y LA QUE CONTIENE LA CLAVE DE LA QUE DEPENDE LA CLAVE AJENA SE DENOMINA TABLA PADRE (REFERENCIADA)
    - ESTO DA LUGAR A UNA JERARQUÍA DE DEPENDENCIAS ENTRE LAS TABLAS O JERARQUÍA REFERENCIAL, CON POSIBLEMENTE VARIOS NIVELES
- UNA TABLA PUEDE TENER VARIOS PADRES DISTINTOS, NO NECESARIAMENTE EN EL MISMO NIVEL, PORQUE SE HAN DEFINIDO SENDAS CLAVES AJENAS QUE REFERENCIAN CLAVES DE TABLAS DISTINTAS
- UNA TABLA PUEDE AUTOREFERENCIARSE, ES DECIR, LA CLAVE AJENA APUNTA A UNA CLAVE DENTRO DE LA PROPIA RELACIÓN
    - LA TABLA ES A LA VEZ PADRE E HIJA.

CLAVE AJENA (FOREIGN KEY)

- ESTAS DEPENDENCIAS PUEDEN SER REPRESENTADAS GRÁFICAMENTE MEDIANTE UN GRAFO, DENOMINÁNDOSE AL MISMO DIAGRAMA O GRAFO DE DEPENDENCIA REFERENCIAL DE LA BASE DE DATOS
- LAS TABLAS DEL PRIMER NIVEL DE ESTE DIAGRAMA SE CONSIDERAN INDEPENDIENTES, MIENTRAS QUE LAS DE LOS NIVELES MÁS ANIDADOS O PROFUNDOS SON DEPENDIENTES
- LA DEPENDENCIA REFERENCIAL ESTABLECE UN ORDEN DE CREACIÓN Y ELIMINACIÓN DE OBJETOS EN LA BASE DE DATOS
  - SE CREAN DESDE EL NIVEL MÁS EXTERNO AL MÁS INTERNO Y SE ELIMINAN EN SENTIDO CONTRARIO

VALORES NULOS

- LOS VALORES NULOS SE EMPLEAN PARA REPRESENTAR INFORMACIÓN INCOMPLETA O DESCONOCIDA EN LA BASE DE DATOS
- EN EL MODELO RELACIONAL SE REPRESENTAN MEDIANTE ?
- CUANDO SE PERMITE A UN ATRIBUTO QUE CONTENGA VALORES NULOS, ÉSTE PASA A SER UN ELEMENTO MÁS DEL DOMINIO DEL ATRIBUTO, DISTINTO A TODOS LOS DEMÁS
- SE DEBE ESTABLECER UN CRITERIO BIEN DEFINIDO PARA ORDENAR LOS ELEMENTOS DEL DOMINIO DEL ATRIBUTO (INCLUIDO EL NULO)
  - O BIEN CONSIDERARLOS LOS DE MENOR VALOR O LOS DE MAYOR VALOR

VALORES NULOS

- LOS VALORES NULOS AÑADEN FLEXIBILIDAD A LA BASE DE DATOS, PORQUE A VECES DETERMINADOS VALORES NO SON CONOCIDOS EN EL MOMENTO DE INTRODUCIR LA INFORMACIÓN EN EL SISTEMA O PUEDE QUE TODAVÍA NO HAYAN SIDO GENERADOS
  - PUEDE SER QUE UN CLIENTE NO TENGA NÚMERO DE TELÉFONO
  - O QUE TODAVÍA NO SE TENGA LA NOTA DE UNA ASIGNATURA (POR NO HABERSE CELEBRADO EL EXAMEN)
  - O QUE NO SE TENGA FECHA DE LAS ÚLTIMAS VACACIONES PORQUE EL EMPLEADO ACABA DE INCORPORARSE A LA EMPRESA

VALORES NULOS

- PARA LOS PROPÓSITOS DE ESTA ASIGNATURA, CONSIDERAREMOS QUE TODOS LOS VALORES NULOS TIENEN EL MISMO GRADO DE DESCONOCIMIENTO
- UNA T-UPLA ES DUPLICADA DE OTRA CUANDO TODAS SUS COMPONENTES COINCIDEN, CONSIDERANDO QUE UN VALOR NULO SÓLO ES IGUAL A NULO
- DIREMOS QUE UN CONJUNTO DE ATRIBUTOS TOMA EL VALOR NULO CUANDO ALGUNO DE ESOS ATRIBUTOS TIENE VALOR NULO
- PERMITIR VALORES NULOS EN EL DOMINIO DE UN ATRIBUTO HACE QUE HAYA QUE REVISAR ALGUNOS CONCEPTOS DEL MODELO RELACIONAL Y LOS RESULTADOS DE LAS OPERACIONES ARITMÉTICAS, COMPARACIONALES Y LÓGICAS.

VALORES NULOS

- OPERACIONES ARITMÉTICAS: EN CUALQUIER OPERACIÓN ARITMÉTICA, SI ALGÚN OPERANDO ES NULO, EL RESULTADO ES NULO (5 + ? = ?)
- OPERADORES COMPARACIONALES: EN CUALQUIER COMPARACIÓN, SI ALGÚN OPERANDO ES NULO, EL RESULTADO ES NULO (3 < ? ≡ ?)
  - SE PASA DE UNA LÓGICA BINARIA (V Y F) A UNA LÓGICA TERNARIA (V, F Y ?)
- OPERADORES LÓGICOS: LAS TABLAS DE VERDAD DE LOS OPERADORES LÓGICOS TIENEN QUE SER EXTENDIDAS

TABLAS DE VERDAD OPERADORES LÓGICOS CON VALORES NULOS

∧	V	F	?
V	V	F	?
F	F	F	F
?	?	F	?

∨	V	F	?
V	V	V	V
F	V	F	?
?	V	?	?

¬	
V	F
F	V
?	?

VALORES NULOS

- SE DEBE AÑADIR UN OPERADOR LÓGICO PARA COMPROBAR SI EL VALOR DE UN ATRIBUTO ES NULO O NO
  - EL OPERADOR LO DENOMINAREMOS ISNULL Y DEVUELVE VERDADERO SI EL ATRIBUTO TOMA EL VALOR NULO Y FALSO EN CASO CONTRARIO
  - SU PROTOTIPO SERÍA: BOOLEAN ISNULL(TIPO ATRIBUTO)

CONDICIONES DE INTEGRIDAD

- SON CONDICIONES QUE TIENEN QUE SATISFACER LOS DATOS PARA SER FORMALMENTE VÁLIDOS
- HAY UNA SERIE DE CONDICIONES DE INTEGRIDAD QUE SE CONSIDERAN NATIVAS EN EL MODELO RELACIONAL DE DATOS
  - LAS MÁS SIMPLES CONSISTEN EN RESTRINGIR LOS VALORES DE LOS ATRIBUTOS USANDO DOMINIOS PRECISOS
    - EJEMPLO: NATURALES ENTRE 1 Y 10, FECHAS POSTERIORES AL 01-01-2015, MUNICIPIOS DE LA PROVINCIA DE SANTA CRUZ DE TENERIFE, ETC.
  - SE PUEDEN ESTABLECER RESTRICCIONES DE UNICIDAD DE VALORES SOBRE UN ATRIBUTO O UN CONJUNTO DE ELLOS
    - IMPLICA QUE NO PUEDEN HABER DOS T-UPLAS EN LA TABLA CON LOS MISMOS VALORES EN ESOS ATRIBUTOS.
  - SE PUEDE RESTRINGIR EL DOMINIO DEL ATRIBUTO PARA QUE NO TOME VALORES NULOS
    - POR DEFECTO SE PERMITEN LOS NULOS

INTEGRIDAD DE CLAVES PRIMARIAS

- LA REGLA DE INTEGRIDAD DE CLAVES PRIMARIAS IMPONE QUE NINGUNO DE LOS ATRIBUTOS DE LA CLAVE PRIMARIA PUEDE CONTENER VALORES NULOS
  - DE ESTA FORMA SE EVITAN AMBIGÜEDADES A LA HORA DE IDENTIFICAR LAS T-UPLAS
  - ES EQUIVALENTE A IMPONER UNA RESTRICCIÓN EN EL DOMINIO DE LOS ATRIBUTOS (NOT NULL, ADEMÁS DE LA UNICIDAD POR SER CLAVE)
- LA REGLA SE DEFINE DE FORMA AUTOMÁTICA AL ESTABLECER LA CLAVE PRIMARIA DENTRO DE UNA RELACIÓN
- LA REGLA DE INTEGRIDAD DE CLAVES PRIMARIAS TAMBIÉN SE CONOCE COMO RESTRICCIÓN DE INTEGRIDAD DE ENTIDADES

INTEGRIDAD REFERENCIAL

- LA REGLA DE INTEGRIDAD REFERENCIAL IMPONE QUE TODO VALOR NO NULO DE LA CLAVE AJENA DEBE EXISTIR EN LA CLAVE REFERENCIADA
- EL SISTEMA DEBE VELAR POR EL MANTENIMIENTO DE LA INTEGRIDAD REFERENCIAL CUANDO SE REALIZAN OPERACIONES DE ACTUALIZACIÓN EN LA BASE DE DATOS
- EL ADMINISTRADOR ESPECIFICA QUE ACCIONES COMPENSATORIAS O DE RESTAURACIÓN DE LA CONSISTENCIA ADOPTARÁ EL SISTEMA EN CASO DE ACTUALIZACIONES EN LAS TABLAS PADRE O HIJA
- LAS ACCIONES COMPENSATORIAS SE ESPECIFICAN EN LA TABLA HIJA, CUANDO SE DECLARA LA CLAVE AJENA

INTEGRIDAD REFERENCIAL

- ACCIONES COMPENSATORIAS CUANDO SE ACTUALIZA UNA FILA EN LA TABLA PADRE:
  - INSERCIÓN: NO HAY QUE COMPROBAR NADA
  - BORRADO: COMPROBAR SI HAY FILAS DEPENDIENTES EN LA TABLA HIJA. EN CASO NEGATIVO, ACEPTAR EL BORRADO. EN CASO AFIRMATIVO LAS ACCIONES POSIBLES SON PROPAGACIÓN EN CASCADA, PROPAGACIÓN CON NULOS O RECHAZAR
  - MODIFICACIÓN: COMPROBAR SI HAY FILAS DEPENDIENTES EN LA TABLA HIJA. EN CASO NEGATIVO, ACEPTAR LA MODIFICACIÓN. EN CASO AFIRMATIVO LAS ACCIONES POSIBLES SON PROPAGACIÓN EN CASCADA O RECHAZAR

INTEGRIDAD REFERENCIAL

- ACCIONES COMPENSATORIAS CUANDO SE ACTUALIZA UNA FILA EN LA TABLA HIJA:
  - BORRADO: NO HAY QUE COMPROBAR NADA. SE ACEPTA SIEMPRE
  - INSERCIÓN: HAY QUE COMPROBAR EN LA TABLA PADRE SI EXISTE EL VALOR DE LA CLAVE AJENA
    - SI NO EXISTE RECHAZAR. EN CASO CONTRARIO ACEPTAR
  - MODIFICACIÓN: HAY QUE COMPROBAR EN LA TABLA PADRE SI EXISTE EL NUEVO VALOR DE LA CLAVE AJENA
    - SI NO EXISTE RECHAZAR. EN CASO CONTRARIO ACEPTAR

COMPONENTES DEL MODELO RELACIONAL DE DATOS

- ESTRUCTURAS DE DATOS.
  - RELACIONES, ATRIBUTOS, T-UPLAS, ETC.
- OPERADORES
  - LOS OPERADORES FUNDAMENTALES DEL ÁLGEBRA RELACIONAL: PROYECCIÓN, SELECCIÓN, UNIÓN, DIFERENCIA Y PRODUCTO CARTESIANO
- CONDICIONES DE INTEGRIDAD
  - DOMINIOS, CONCEPTOS DE CLAVES, POSIBILIDAD DE VALORES NULOS, REGLA DE INTEGRIDAD DE CLAVES PRIMARIAS, REGLA DE INTEGRIDAD REFERENCIAL, ETC.

ÁLGEBRA RELACIONAL

- INTRODUCCIÓN
- OPERADORES ESENCIALES
- OPERADORES DERIVADOS
- POTENCIA EXPRESIVA DEL ÁLGEBRA RELACIONAL
- CONSULTAS EQUIVALENTES
- PROPIEDADES DE LOS OPERADORES
- VISTAS
- ÁLGEBRA EXTENDIDA

ÁLGEBRA RELACIONAL

- ES UN LENGUAJE DE CONSULTAS PROCEDIMENTAL "PURO"
- SISTEMA CERRADO DE OPERACIONES DEFINIDAS SOBRE RELACIONES
  - TANTO LOS OPERANDOS COMO LOS RESULTADOS SON RELACIONES
- ESTÁ FORMADO POR UNA SERIE DE OPERADORES QUE PERMITEN MANIPULAR LAS TABLAS
- LOS OPERADORES SE CLASIFICAN EN:
  - OPERADORES ESENCIALES (BÁSICOS, FUNDAMENTALES, PRIMITIVOS, …)
  - OPERADORES DERIVADOS (NO BÁSICOS, NO ESENCIALES, …)

ÁLGEBRA RELACIONAL

- ALGUNOS DE LOS OPERADORES ESTÁN BASADOS EN LA TEORÍA DE CONJUNTOS Y SE LES LLAMA OPERADORES DE CONJUNTOS
  - UNIÓN, INTERSECCIÓN, DIFERENCIA, PRODUCTO CARTESIANO
- OTROS OPERADORES SON ESPECÍFICAMENTE RELACIONALES
  - PROYECCIÓN Y SELECCIÓN
- LOS DEMÁS OPERADORES SON MIXTOS
  - YUNCIÓN, YUNCIÓN NATURAL Y COCIENTE

OPERADORES FUNDAMENTALES

- NO SE PUEDEN PONER EN FUNCIÓN DE LOS DEMÁS OPERADORES
- SI SUPRIMES ALGUNO LE QUITAS POTENCIA EXPRESIVA AL LENGUAJE
- LOS OPERADORES FUNDAMENTALES SON:
  - SELECCIÓN
  - PROYECCIÓN
  - PRODUCTO CARTESIANO
  - UNIÓN
  - DIFERENCIA

SELECCIÓN

- SINTAXIS
  - S(F)(R)
- OPERADOR UNARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE HAN SELECCIONADO LAS FILAS DE R QUE SATISFACEN EL PREDICADO LÓGICO F
- LA TABLA RESULTADO TIENE LOS MISMOS ATRIBUTOS QUE R
- |S(F)(R)| ≤ |R| (REDUCCIÓN EN PROFUNDIDAD)
- F SE CONSTRUYE CON:
  - CONSTANTES Y ATRIBUTOS DE R
  - OPERADORES ARITMÉTICOS: +, -, *, /, **, (,)
  - OPERADORES COMPARACIONALES: <, >, <=, >=, =, <>, !=, ≠
  - OPERADORES LÓGICOS: ∧, ∨, ¬, (,)

EJERCICIOS

- CLIENTE(DNI, NBC, CDC)
- PRÉSTAMO(CS, NP, DNI, I)
- CUENTA(CS, NC, DNI, SLD)
- SUCURSAL(CS, NS, CDS)

- LISTAR TODAS LAS FILAS DE LA TABLA PRÉSTAMO CORRESPONDIENTES A LA SUCURSAL CON CÓDIGO1

- LISTAR TODAS LAS FILAS DE LA TABLA PRÉSTAMO CORRESPONDIENTES A LA SUCURSAL CON CÓDIGO1 Y POR UN IMPORTE SUPERIOR A 100000 EUROS

PROYECCIÓN

- SINTAXIS
  - P(L)(R)
- OPERADOR UNARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE HAN SELECCIONADO LAS COLUMNAS DE R QUE APARECEN EN LA LISTA L
  - SI HUBIESE ALGUNA T-UPLA REPETIDA SE ELIMINA
- LA TABLA RESULTADO TIENE LOS ATRIBUTOS QUE ESTÁN EN L
- REDUCCIÓN EN ANCHURA Y POSIBLEMENTE EN PROFUNDIDAD ($|P(L)(R)| \leq |R|$)

EJERCICIOS

- LISTAR LOS DNI DE LOS CLIENTES DEL BANCO
- LISTAR LOS DNI DE LOS CLIENTES DEL BANCO CON PRÉSTAMOS Y LOS CÓDIGOS DE LAS SUCURSALES DONDE LOS TIENEN
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS

PRODUCTO CARTESIANO

- SINTAXIS
  - $R \times S$
- OPERADOR BINARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS POSIBLES QUE SE OBTIENEN CONCATENANDO UNA DE R CON OTRA DE S
- LA TABLA RESULTADO TIENE LOS ATRIBUTOS DE R Y LOS DE S A CONTINUACIÓN
- SI R Y S TIENEN ATRIBUTOS COMUNES, PARA PODER DISTINGUIRLOS, SE ANTEPONE DELANTE DEL NOMBRE DEL ATRIBUTO EL NOMBRE DE LA TABLA DE LA QUE PROCEDE
- $|R \times S| = |R| * |S|$

EJERCICIOS

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN ALGUNA SUCURSAL DE LA LAGUNA

- CIUDADES EN QUE RESIDEN LOS CLIENTES QUE TIENEN ALGUNA CUENTA CON UN SALDO SUPERIOR A 100000 EUROS

- DNI DE LOS CLIENTES CON CUENTAS EN AL MENOS DOS SUCURSALES DISTINTAS

UNIÓN

- SINTAXIS
  - R U S
- OPERADOR BINARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS DE R Y S
  - SI HAY ALGUNA T-UPLA COMÚN SOLO SE CUENTA UNA VEZ
- PARA PODER APLICAR ESTE OPERADOR TIENE QUE CUMPLIRSE QUE:
  - R Y S TIENEN EL MISMO GRADO
  - LOS DOMINIOS DE LOS ATRIBUTOS I-ÉSIMOS EN AMBAS TABLAS DEBEN COINCIDIR
- $|R \cup S| \leq |R| + |S|$

DIFERENCIA

- SINTAXIS
  - R - S
- OPERADOR BINARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS DE R QUE NO PERTENECEN A S
- PARA PODER APLICAR ESTE OPERADOR TIENE QUE CUMPLIRSE QUE:
  - R Y S TIENEN EL MISMO GRADO
  - LOS DOMINIOS DE LOS ATRIBUTOS I-ÉSIMOS EN AMBAS TABLAS DEBEN COINCIDIR
- $|R - S| \leq |R|$

EJERCICIOS

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO O UNA CUENTA O AMBAS COSAS EN LA SUCURSAL CON CÓDIGO1

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO EN LA SUCURSAL CON CÓDIGO1PERO NO TIENEN UNA CUENTA EN DICHA SUCURSAL

- DNI DE LOS CLIENTES QUE TIENEN UNA CUENTA EN LA SUCURSAL CON CÓDIGO1PERO NO TIENEN UN PRÉSTAMO EN DICHA SUCURSAL

OPERADORES DERIVADOS

- SE PUEDEN PONER EN FUNCIÓN DE LOS OPERADORES ESENCIALES
- FACILITAN LA ESCRITURA DE CONSULTAS
- SI SUPRIMES ALGUNO NO LE QUITAS POTENCIA EXPRESIVA AL LENGUAJE
- LOS OPERADORES DERIVADOS SON:
  - INTERSECCIÓN
  - YUNCIÓN (COMBINACIÓN, REUNIÓN, ASOCIACIÓN, CONCATENACIÓN, JOIN, …)
  - YUNCIÓN NATURAL (COMBINACIÓN NATURAL, REUNIÓN NATURAL, NATURAL JOIN, …)
  - COCIENTE (DIVISIÓN)

INTERSECCIÓN

- SINTAXIS
  - $R \cap S$
- OPERADOR BINARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS QUE ESTÁN EN R Y EN S SIMULTÁNEAMENTE, CONTÁNDOLAS UNA SOLA VEZ
- PARA PODER APLICAR ESTE OPERADOR TIENE QUE CUMPLIRSE QUE:
  - R Y S TIENEN EL MISMO GRADO
  - LOS DOMINIOS DE LOS ATRIBUTOS I-ÉSIMOS EN AMBAS TABLAS DEBEN COINCIDIR
- $|R \cap S| \leq MIN(|R|, |S|)$
- $R \cap S = R - (R - S) = S - (S - R)$

EJERCICIOS

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO Y UNA CUENTA EN LA SUCURSAL CON CÓDIGO1

- CÓDIGOS DE LAS SUCURSALES QUE TIENEN ALGUNA CUENTA CON TITULARES DE SANTA CRUZ Y DE LA LAGUNA

YUNCIÓN

- SINTAXIS
  - R Y(F) S
- OPERADOR BINARIO
- DA LUGAR A UNA A UNA NUEVA TABLA EN LA QUE SE INCLUYEN LAS T-UPLAS DEL PRODUCTO CARTESIANO DE R Y S QUE SATISFACEN EL PREDICADO F
- $R\ Y(F)\ S = S(F)(R \times S)$

YUNCIÓN NATURAL

- SINTAXIS
  - R * S
- OPERADOR BINARIO
- ES UN TIPO ESPECIAL DE YUNCIÓN EN EL QUE EL PREDICADO ESTÁ ENUNCIADO DE FORMA IMPLÍCITA E IMPONE LA IGUALDAD DE VALORES EN LOS ATRIBUTOS HOMÓNIMOS DE AMBAS TABLAS. EN LA TABLA RESULTADO SE SUPRIME UNA COLUMNA DE CADA 2 HOMÓNIMAS
- $R * S = P(L_R \cup L_S)(S(L_{R.(R \cap S)} = L_{S.(R \cap S)}) (R \times S))$
- SI R Y S NO TIENEN ATRIBUTOS COMUNES $R * S = R \times S$
- SI R Y S TIENEN LOS MISMOS ATRIBUTOS $R * S = R \cap S$

EJERCICIOS

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN ALGUNA SUCURSAL DE LA LAGUNA
- CIUDADES EN QUE RESIDEN LOS CLIENTES QUE TIENEN ALGUNA CUENTA CON UN SALDO SUPERIOR A 100000 EUROS
- DNI DE LOS CLIENTES CON AL MENOS UNA CUENTA EN UNA SUCURSAL EN LA CIUDAD EN QUE VIVEN

COCIENTE

- SINTAXIS
  - R / S
- OPERADOR BINARIO
- PARA PODER APLICAR EL OPERADOR TODOS LOS ATRIBUTOS DE S TIENEN QUE ESTAR CONTENIDOS ESTRICTAMENTE EN LOS DE R ($L_S \subset L_R$)
- LA TABLA DE RESULTADO CONTIENE AQUELLAS FILAS DE $P(L_R - L_S)(R)$ TALES QUE AL CONCATENARLES CUALQUIER FILA DE S PERTENECE A R
- $R / S = P(L_R - L_S)(R) - P(L_R - L_S)((P(L_R - L_S)(R) \times S) - R)$
- SE LLAMA COCIENTE POR LA ANALOGÍA: $((R / S) \times S) \subseteq R$ (COCIENTE POR DIVISOR ES MENOR O IGUAL QUE EL DIVIDENDO)

EJERCICIOS

- DNI DE LOS CLIENTES QUE TIENEN AL MENOS UNA CUENTA EN TODAS LAS SUCURSALES DE ‘LA LAGUNA’

OPERACIONES DE ACTUALIZACIÓN

- INSERCIÓN
  - $R \cup \{t\}$, $R \cup E$
- BORRADO
  - $R - \{t\}$, $R - E$
- MODIFICACIÓN
  - $(R - \{t^1\}) \cup \{t^2\}$

OPERADOR COMPLEMENTO

- SI SE DEFINIESE UN OPERADOR COMPLEMENTO HABRIA QUE HACERLO COMO:
  - $\overline{R} = (D_1 \times D_2 \times \ldots \times D_N) - R$
- NO SE PUEDE DEFINIR UN OPERADOR COMPLEMENTO EN ÁLGEBRA RELACIONAL PORQUE EL CONJUNTO DE T-UPLAS COMPLEMENTARIO DE UNA RELACIÓN EN GENERAL NO ES FINITO
  - ALGUNO DE LOS DOMINIOS DE LOS ATRIBUTOS DE R PUEDE SER INFINITO

POTENCIA EXPRESIVA DEL ÁLGEBRA

- LA POTENCIA EXPRESIVA DE UN LENGUAJE DE CONSULTA ES EL CONJUNTO DE TODAS LAS CONSULTAS QUE SE PUEDEN RESOLVER UTILIZANDO ESE LENGUAJE
- EN PARTICULAR, LA POTENCIA EXPRESIVA DEL ÁLGEBRA ESTÁ DETERMINADA POR LOS OPERADORES ESENCIALES CONSIDERADOS
  - PARA INCREMENTAR LA POTENCIA SE NECESITARÍAN MÁS OPERADORES ESENCIALES
- PROBAREMOS QUE EL ÁLGEBRA RELACIONAL Y EL CÁLCULO RELACIONAL TIENEN LA MISMA POTENCIA EXPRESIVA
  - CUALQUIER CONSULTA QUE SE PUEDA HACER EN ÁLGEBRA RELACIONAL TIENE EQUIVALENTE EN CÁLCULO RELACIONAL, Y VICEVERSA.
- CUANDO UN LENGUAJE DE CONSULTA TIENE AL MENOS LA MISMA POTENCIA EXPRESIVA QUE EL ÁLGEBRA SE DICE QUE ES RELACIONALMENTE COMPLETO
- EL ALGEBRA RELACIONAL ES UN LENGUAJE PATRÓN QUE PERMITE COMPARAR LA POTENCIA EXPRESIVA DE OTROS

CONSULTAS EQUIVALENTES

- DOS CONSULTAS SON EQUIVALENTES CUANDO PROPORCIONAN EL MISMO RESULTADO PARA TODOS LOS VALORES POSIBLES DE LAS VARIABLES
- CUANDO DOS CONSULTAS SON EQUIVALENTES INTERESA SELECCIONAR AQUELLA QUE:
  - SEA MÁS SIMPLE (CONSUME MENOS RECURSOS)
  - SEA MÁS LEGIBLE (RESULTA MÁS CLARA)
- PARA PROBAR FORMALMENTE QUE DOS CONSULTAS SON EQUIVALENTES SE UTILIZAN LAS PROPIEDADES DE LOS OPERADORES

PROPIEDADES DE LOS OPERADORES

- LOS OPERADORES DEL ÁLGEBRA SATISFACEN UN CONJUNTO DE PROPIEDADES QUE NOS PERMITEN TRANSFORMAR UNA CONSULTA EN OTRA EQUIVALENTE A LA MISMA
- ESTO NOS PERMITE DETERMINAR SI DOS CONSULTAS SON EQUIVALENTES O BIEN SIMPLIFICAR UNA DADA

PROPIEDADES DE LOS OPERADORES

- $\cup$, $\cap$ SON CONMUTATIVAS Y ASOCIATIVAS
  - $R \cup S = S \cup R$
  - $(R \cup S) \cup T = R \cup (S \cup T)$
- PROPIEDAD DISTRIBUTIVA DE LA UNIÓN RESPECTO DE LA INTERSECCIÓN
  - $(R \cup S) \cap T = (R \cap T) \cup (S \cap T)$
- PROPIEDAD DISTRIBUTIVA DE LA INTERSECCIÓN RESPECTO DE LA UNIÓN
  - $(R \cap S) \cup T = (R \cup T) \cap (S \cup T)$

PROPIEDADES DE LOS OPERADORES

- $\times$, * SON CONMUTATIVAS SI NO IMPORTA EL ORDEN DE LOS ATRIBUTOS DE LA TABLA DE RESULTADO
  - $R \times S = S \times R$
- $\times$, * SON ASOCIATIVAS
  - $(R \times S) \times T = R \times (S \times T)$
- CONMUTATIVA DE LA SELECCIÓN Y EL PRODUCTO CARTESIANO
  - $S(F_1 \wedge F_2)\ (R \times S) = (S(F_1)\ (R)) \times (S(F_2)\ (S))$
  - F_1 SOLO SE REFIERE A LOS ATRIBUTOS DE R Y F_2 SOLO SE REFIERE A LOS ATRIBUTOS DE S
- CONMUTATIVA DE LA SELECCIÓN Y LA YUNCIÓN NATURAL
  - $S(F_1 \wedge F_2)\ (R * S) = (S(F_1)\ (R)) * (S(F_2)\ (S))$
  - F_1 SOLO SE REFIERE A LOS ATRIBUTOS DE R Y F_2 SOLO SE REFIERE A LOS ATRIBUTOS DE S

PROPIEDADES DE LOS OPERADORES

- CASCADA DE SELECCIONES
  - $S(F_1)(S(F_2)(\ldots(S(F_N)(R)))) = S(F_1 \wedge F_2 \wedge \ldots \wedge F_N)(R)$
- CASCADA DE PROYECCIONES
  - $P(L_1)(P(L_2)(\ldots(P(L_N)(R)))) = P(L_1)(R)$
  - SUPONEMOS $L_1 \subseteq L_2 \subseteq \ldots \subseteq L_N$
- CONMUTATIVA ENTRE LA SELECCIÓN Y LA PROYECCIÓN
  - $P(L)(S(F)(R)) = S(F)(P(L)(R))$
  - SUPONEMOS QUE EN F SOLO INTERVIENEN ATRIBUTOS DE LA LISTA L

PROPIEDADES DE LOS OPERADORES

- CONMUTATIVA DE LA PROYECCIÓN CON EL PRODUCTO CARTESIANO
  - $P(L)\ (R \times S) = (P(L \cap L_R)\ (R)) \times (P(L \cap L_S)\ (S))$
  - L_R Y L_S SON LAS LISTAS DE ATRIBUTOS DE R Y S, RESPECTIVAMENTE
- CONMUTATIVA DE LA SELECCIÓN CON LOS OPERADORES DE CONJUNTO {∪, ∩, -}
  - $S(F)(R\ \Theta\ S) = S(F)(R)\ \Theta\ S(F)(S)$, DONDE $\Theta \in \{\cup, \cap, -\}$
- CONMUTATIVA DE LA PROYECCIÓN CON LA ∪
  - $P(L)(R \cup S) = P(L)(R) \cup P(L)(S)$

VISTAS (VIEWS)

- SON EXPRESIONES ALGEBRAICAS A LAS QUE SE LE ASIGNA UN NOMBRE
  - TENEMOS ASÍ EXPRESIONES PREFABRICADAS QUE PUEDEN SER REUTILIZADAS PARA CONSTRUIR EXPRESIONES MÁS COMPLEJAS
- EJ: $V1 = S(F_1 \wedge F_2)\ (R \times S)$
- NO PUEDEN SER RECURSIVAS EN SU DEFINICIÓN
- CUANDO SE CREA UNA VISTA SÓLO SE ALMACENA SU DEFINICIÓN EN EL DICCIONARIO, NO SU CONTENIDO
  - POR ESO SE LES CONOCE TAMBIÉN COMO RELACIONES VIRTUALES
  - HAY QUE RECALCULARLAS CADA VEZ QUE SE UTILIZAN ⇒ SON LENTAS
- LAS VISTAS PERMITEN TAMBIÉN CREAR SINÓNIMOS PARA TABLAS Y CAMBIAR LOS NOMBRES DE LOS ATRIBUTOS
  - V1 = R //SINÓNIMO PARA R. LOS NOMBRES DE LOS ATRIBUTOS DE V1 SON LOS DE R
  - V1(L) = R //SINÓNIMO PARA R. LOS NOMBRES DE LOS ATRIBUTOS DE V1 SON LOS DE L

VISTAS

- LAS VISTAS PRESENTAN ALGUNAS DESVENTAJAS RELATIVAS A SU IMPLEMENTACIÓN EN LOS SISTEMAS GESTORES DE BASES DE DATOS RELACIONALES COMERCIALES:
  - EL CONTENIDO DE LA VISTA NO SE ALMACENA EN LA BASE DE DATOS, SOLO SU DEFINICIÓN EN EL DICCIONARIO DE LA BASE DE DATOS
  - REQUIEREN MAS TIEMPO DE PROCESAMIENTO QUE LAS TABLAS, DEBIDO A QUE HAY QUE RECALCULARLAS CADA VEZ QUE SE UTILIZAN
  - EN GENERAL, LAS VISTAS NO SON ACTUALIZABLES PUES EL SISTEMA TIENE QUE REPERCUTIR LAS ACTUALIZACIONES EN LAS TABLAS EN LAS QUE SE BASA LA VISTA
    - SOLO LAS VISTAS MÁS SENCILLAS SON ACTUALIZABLES

USOS DE LAS VISTAS

- SIMPLIFICAR LA FORMULACIÓN DE EXPRESIONES COMPLEJAS
  - HACEN EL PAPEL DE LAS SUBRUTINAS EN LOS LENGUAJES DE PROGRAMACIÓN, OCULTANDO COMPLEJIDAD DEL CÓDIGO Y PUDIENDO SER INVOCADAS TANTAS VECES COMO SE QUIERAN
- PROTEGER LA CONFIDENCIALIDAD DEL ACCESO A LOS DATOS
  - SE CREAN VISTAS MEDIANTE PROYECCIONES Y SELECCIONES PARA OCULTAR COLUMNAS Y FILAS DE LAS TABLAS Y SE CONCEDE ACCESO A LAS VISTAS PERO NO A LAS TABLAS EN LA QUE SE BASAN
- EVITAR QUE CAMBIOS EN EL DISEÑO AFECTEN A LOS PROGRAMAS YA EXISTENTES
  - CUANDO SE SUPRIME O SUSTITUYE UNA RELACIÓN POR OTRAS DISTINTAS, LOS PROGRAMAS DE APLICACIONES QUE ACCEDÍAN A ESA RELACIÓN FALLAN POR NO EXISTIR LA MISMA
  - ES POSIBLE CREAR UNA VISTA A PARTIR DEL NUEVO DISEÑO QUE RECREE LA TABLA ORIGINAL, DE FORMA QUE LAS OPERACIONES DE CONSULTA DE LOS PROGRAMAS SIGAN FUNCIONANDO

VISTAS MATERIALIZADAS

- ALGUNOS SISTEMAS GESTORES DE BASES DE DATOS IMPLEMENTAN EL CONCEPTO DE VISTA MATERIALIZADA (MATERIALIZED VIEW)
- SON VISTAS ESPECIALES EN LAS QUE, ADEMÁS DE SU DEFINICIÓN, TAMBIÉN SE ALMACENA SU CONTENIDO
  - AUMENTAN LOS REQUERIMIENTOS DE ESPACIO DEL SISTEMA
- ENLENTECEN LAS ACTUALIZACIONES PUES TIENEN QUE ACTUALIZARSE CUANDO SE ACTUALIZAN LAS TABLAS EN LAS QUE SE BASAN LAS VISTAS
- SON MÁS RÁPIDAS PARA CONSULTAS QUE LAS VISTAS TRADICIONALES PUES YA ESTÁN CALCULADAS CUANDO SE USAN

ÁLGEBRA EXTENDIDA

- SI PERMITIMOS EL USO DE VALORES NULOS EN EL MODELO RELACIONAL PODEMOS EXTENDER LOS OPERADORES DEL ÁLGEBRA CON OTROS NUEVOS:
  - UNIÓN EXTERNA
  - YUNCIÓN NATURAL EXTERNA (A IZQUIERDA, DERECHA Y COMPLETA)
  - YUNCIÓN POSIBILISTA

UNIÓN EXTERNA

- SINTAXIS
  - R UE S
- OPERADOR BINARIO QUE GENERALIZA A LA UNIÓN
- PARA PODER APLICAR ESTE OPERADOR TIENE QUE CUMPLIRSE QUE:
  - LOS ATRIBUTOS HOMÓNIMOS DE R Y S TIENEN EL MISMO DOMINIO
- DA LUGAR A UNA A UNA NUEVA TABLA CON ATRIBUTOS L_R U L_S EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS DE R Y S COMPLETADOS CON NULOS
  - SI HAY ALGUNA T-UPLA COMÚN SOLO SE CONSIDERA UNA VEZ

YUNCIÓN NATURAL EXTERNA
NATURAL OUTER JOIN

- SINTAXIS
  - R YE S
- OPERADOR BINARIO QUE GENERALIZA A LA YUNCIÓN NATURAL
- DA LUGAR A UNA A UNA NUEVA TABLA CON ATRIBUTOS L_R U L_S EN LA QUE SE INCLUYEN TODAS LAS T-UPLAS DE LA YUNCIÓN NATURAL, PERO AÑADIENDO ADEMÁS AL RESULTADO LAS T-UPLAS DE LOS OPERANDOS QUE O BIEN TENGAN ? EN EL ATRIBUTO COMÚN O BIEN EL VALOR DE ÉSTE NO EXISTA EN LA OTRA RELACIÓN
  - LAS T-UPLAS AÑADIDAS SE COMPLETAN CON ?
- ESTA VERSIÓN SE DENOMINA YUNCIÓN NATURAL EXTERNA COMPLETA (FULL NATURAL OUTER JOIN)
- CUANDO SE AÑADEN A LA TABLA RESULTADO SÓLO LAS TUPLAS DEL OPERANDO IZQUIERDO (O DERECHO), SE HABLA DE YUNCIÓN NATURAL EXTERNA A IZQUIERDA (O DERECHA)
  - LEFT NATURAL OUTER JOIN
  - RIGHT NATURAL OUTER JOIN

YUNCIÓN POSIBILISTA

- SINTAXIS
  - R YP S
- OPERADOR BINARIO
- ES UN CASO PARTICULAR DE YUNCIÓN DONDE EL PREDICADO ES IMPLÍCITO Y SELECCIONA DEL PRODUCTO CARTESIANO AQUELLAS T-UPLAS CON VALORES IGUALES EN SUS ATRIBUTOS HOMÓNIMOS, CONSIDERANDO QUE ? ES IGUAL A ? O A CUALQUIER OTRO VALOR
- SI NO HAY ATRIBUTOS COMUNES EL OPERADOR YP COINCIDE CON ×

CÁLCULO RELACIONAL

- INTRODUCCIÓN
- NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS
- REGLAS DE INFERENCIA LÓGICA
- CÁLCULO RELACIONAL DE T-UPLAS
- CÁLCULO RELACIONAL DE DOMINIOS

CÁLCULO RELACIONAL

- EL CÁLCULO RELACIONAL ES UN LENGUAJE DE CONSULTA TEÓRICO DECLARATIVO PURO (NO PROCEDIMENTAL)
- SE BASE EN LA LÓGICA DE PREDICADOS
- EXISTEN 2 FORMAS DE CÁLCULO RELACIONAL:
  - CÁLCULO RELACIONAL DE T-UPLAS (CRT)
    - LAS VARIABLES REPRESENTAN T-UPLAS DE ALGUNA TABLA
  - CÁLCULO RELACIONAL DE DOMINIOS (CRD)
    - LAS VARIABLES REPRESENTAN VALORES DEL DOMINIO DE ALGÚN ATRIBUTO

NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS

- UNA PROPOSICIÓN (LÓGICA) ES UNA ORACIÓN DECLARATIVA QUE PUEDE SER VERDADERA O FALSA, PERO NO AMBAS COSAS A LA VEZ
  - EJ: ESTÁ LLOVIENDO
- EL CÁLCULO O LÓGICA PROPOSICIONAL ES EL ÁREA DE LA LÓGICA QUE TRATA SOBRE LAS PROPOSICIONES LÓGICAS
- LAS FÓRMULAS SE CREAN A PARTIR DE LAS PROPOSICIONES EXISTENTES USANDO OPERADORES LÓGICOS

NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS

- EN LOS PREDICADOS LÓGICOS PUEDEN APARECER OPERADORES ARITMÉTICOS, COMPARACIONALES, LÓGICOS, CONSTANTES Y VARIABLES
  - LAS VARIABLES PUEDEN ESTAR CUANTIFICADAS (EXISTENCIAL O UNIVERSALMENTE)
- LA LÓGICA DE PREDICADOS ESTUDIA LAS PROPOSICIONES CON MAYOR GRADO DE DETALLE QUE LA LÓGICA PROPOSICIONAL
  - SE DISTINGUE LO QUE SE AFIRMA (PREDICADO) DE QUIEN SE AFIRMA (OBJETO)

OPERADORES LÓGICOS

Conectiva	Notación	Ejemplo de uso	Análogo natural	Ejemplo de uso en el lenguaje natural	Tabla de verdad
Negación	$\neg, \sim$	$\neg P$	no	**No** está lloviendo.	P | $\neg P$ 1 | 0 0 | 1
Conjunción	$\wedge, \&, \cdot$	$P \wedge Q$	y	Está lloviendo **y** la calle está mojada.	P | Q | $P \wedge Q$ 1 | 1 | 1 1 | 0 | 0 0 | 1 | 0 0 | 0 | 0
Disyunción	$\vee$	$P \vee Q$	o	Está lloviendo **o** la calle está mojada.	P | Q | $P \vee Q$ 1 | 1 | 1 1 | 0 | 1 0 | 1 | 1 0 | 0 | 0
Condicional material	$\rightarrow, \supset$	$P \rightarrow Q$	si... entonces	**Si** está lloviendo, **entonces** la calle está mojada.	P | Q | $P \rightarrow Q$ 1 | 1 | 1 1 | 0 | 0 0 | 1 | 1 0 | 0 | 1
Bicondicional	$\leftrightarrow, \equiv$	$P \leftrightarrow Q$	si y solo si	Está lloviendo **si y solo si** la calle está mojada.	P | Q | $P \leftrightarrow Q$ 1 | 1 | 1 1 | 0 | 0 0 | 1 | 0 0 | 0 | 1
Negación conjunta	$\downarrow$	$P \downarrow Q$	ni... ni	**Ni** está lloviendo **ni** la calle está mojada.	P | Q | $P \downarrow Q$ 1 | 1 | 0 1 | 0 | 0 0 | 1 | 0 0 | 0 | 1
Disyunción excluyente	$\nleftrightarrow, \oplus, \not\equiv, W, \veebar$	$P \nleftrightarrow Q$	o bien... o bien	**O bien** está lloviendo, **o bien** la calle está mojada.	P | Q | $P \nleftrightarrow Q$ 1 | 1 | 0 1 | 0 | 1 0 | 1 | 1 0 | 0 | 0

CÁLCULO RELACIONAL DE T-UPLAS

- DADA LA IMPLICACIÓN $P \rightarrow Q$
  - A P SE LE LLAMA ANTECEDENTE, PREMISA O HIPÓTESIS
  - A Q SE LE LLAMA CONSECUENTE, CONSECUENCIA, CONCLUSIÓN, TESIS
  - $Q \rightarrow P$ ES EL RECÍPROCO
  - $\neg Q \rightarrow \neg P$ ES EL CONTRARRECÍPROCO
  - $\neg P \rightarrow \neg Q$ ES LA INVERSA

NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS

- DOS FÓRMULAS SON EQUIVALENTES CUANDO TIENEN LA MISMA TABLA DE VERDAD
  - REPRESENTAREMOS LA EQUIVALENCIA ENTRE FÓRMULAS CON EL SIGNO $\Leftrightarrow$
- EJEMPLOS:
  - $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$
  - $\neg P \rightarrow \neg Q \Leftrightarrow Q \rightarrow P$
  - $P \leftrightarrow Q \Leftrightarrow P \rightarrow Q \wedge Q \rightarrow P$

PRECEDENCIA OPERADORES LÓGICOS

OPERADOR	PRECEDENCIA
¬	1
∧	2
∨	3
→	4
↔	5

NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS

- TAUTOLOGÍA
  - FÓRMULA QUE SIEMPRE ES VERDADERA, NO IMPORTA LOS VALORES DE VERDAD DE LAS PROPOSICIONES QUE LA COMPONEN
  - EJ: $P \vee \neg P$
- CONTRADICCIÓN
  - FÓRMULA QUE SIEMPRE ES FALSA, NO IMPORTA LOS VALORES DE VERDAD DE LAS PROPOSICIONES QUE LA COMPONEN
  - EJ: $P \wedge \neg P$
- CONTINGENCIA
  - FÓRMULA QUE NO ES NI UNA TAUTOLOGÍA NI UNA CONTRADICCIÓN

EQUIVALENCIAS LÓGICAS

- LEYES DE IDENTIDAD
  - $P \wedge V \Leftrightarrow P$
  - $P \vee F \Leftrightarrow P$
- LEYES DE DOMINACIÓN
  - $P \wedge F \Leftrightarrow F$
  - $P \vee V \Leftrightarrow V$
- LEYES IDEMPOTENTES
  - $P \vee P \Leftrightarrow P$
  - $P \wedge P \Leftrightarrow P$

EQUIVALENCIAS LÓGICAS

- LEY DE LA DOBLE NEGACIÓN
  - $\neg(\neg P) \Leftrightarrow P$
- LEYES CONMUTATIVAS
  - $P \vee Q \Leftrightarrow Q \vee P$
  - $P \wedge Q \Leftrightarrow Q \wedge P$
- LEYES ASOCIATIVAS
  - $(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$
  - $(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$

EQUIVALENCIAS LÓGICAS

- LEYES DISTRIBUTIVAS
  - $(P \vee Q) \wedge R \Leftrightarrow (P \wedge R) \vee (Q \wedge R)$
  - $(P \wedge Q) \vee R \Leftrightarrow (P \vee R) \wedge (Q \vee R)$
- LEYES DE MORGAN
  - $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$
  - $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$

EQUIVALENCIAS LÓGICAS

- LEYES DE ABSORCIÓN
  - $P \vee (P \wedge Q) \Leftrightarrow P$
  - $P \wedge (P \vee Q) \Leftrightarrow P$
- LEYES DE NEGACIÓN
  - $P \vee \neg P \Leftrightarrow V$
  - $P \wedge \neg P \Leftrightarrow F$

EQUIVALENCIAS LÓGICAS RELACIONADAS CON IMPLICACIONES

- $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$
- $P \rightarrow Q \Leftrightarrow \neg P \vee Q$
- $P \vee Q \Leftrightarrow \neg P \rightarrow Q$
- $P \wedge Q \Leftrightarrow \neg(P \rightarrow \neg Q)$
- $\neg(P \rightarrow Q) \Leftrightarrow P \wedge \neg Q$
- $(P \rightarrow Q) \wedge (P \rightarrow R) \Leftrightarrow (P \vee Q) \rightarrow R$
- $(P \rightarrow R) \wedge (Q \rightarrow R) \Leftrightarrow P \rightarrow (Q \wedge R)$
- $(P \rightarrow Q) \vee (P \rightarrow R) \Leftrightarrow P \rightarrow (Q \vee R)$
- $(P \rightarrow R) \vee (Q \rightarrow R) \Leftrightarrow (P \wedge Q) \rightarrow R$

EQUIVALENCIAS LÓGICAS RELACIONADAS CON CUANTIFICADORES

- SUPONGAMOS QUE EL PREDICADO P ES FUNCIÓN DE LA VARIABLE x Y LO DENOTAREMOS POR Px
  - $(\forall x)(Px) \Leftrightarrow \neg(\exists x)\neg(Px)$
  - $(\exists x)(Px) \Leftrightarrow \neg(\forall x)\neg(Px)$
  - $\neg(\forall x)(Px) \Leftrightarrow (\exists x)\neg(Px)$
  - $\neg(\exists x)(Px) \Leftrightarrow (\forall x)\neg(Px)$

NOCIONES BÁSICAS SOBRE LÓGICA DE PREDICADOS

- EL PREDICADO P PUEDE ESTAR EN FUNCIÓN DE VARIAS VARIABLES x, y, … Y LO DENOTAREMOS POR Pxy
- USAREMOS LA SIGUIENTE NOTACIÓN:
  - $(\exists\ x, y)(Pxy) \Leftrightarrow (\exists\ x)\ ((\exists\ y)(Pxy)) \Leftrightarrow (\exists\ y)\ ((\exists\ x)(Pxy))$
  - $(\forall\ x, y)(Pxy) \Leftrightarrow (\forall\ x)\ ((\forall\ y)(Pxy)) \Leftrightarrow (\forall\ y)\ ((\forall\ x)(Pxy))$

REGLAS DE INFERENCIA LÓGICA

- ADICIÓN
  - $P \rightarrow P \vee Q$
- SIMPLIFICACIÓN
  - $P \wedge Q \rightarrow P$
- CONJUNCIÓN
  - $((P) \wedge (Q)) \rightarrow P \wedge Q$
- MODUS PONENS
  - $P \wedge (P \rightarrow Q) \rightarrow Q$

REGLAS DE INFERENCIA LÓGICA

- MODUS TOLLENS
  - $\neg Q \land (P \rightarrow Q) \rightarrow \neg P$
- SILOGISMO HIPOTÉTICO
  - $((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)$
- SILOGISMO DISYUNTIVO
  - $((P \lor Q) \land (\neg P)) \rightarrow Q$
- LEY DE RESOLUCIÓN
  - $((P \lor Q) \land (\neg P \lor R)) \rightarrow (Q \lor R)$

CÁLCULO RELACIONAL DE T-UPLAS

- ES UN TIPO ESPECIAL DE CÁLCULO RELACIONAL DONDE LAS VARIABLES REPRESENTAN T-UPLAS DE UNA TABLA
- UNA CONSULTA EN CÁLCULO RELACIONAL DE T-UPLAS SE EXPRESA COMO:
  - $\{t / P(t)\} \Leftrightarrow$ CONJUNTO DE TODAS LAS T-UPLAS t QUE SATISFACEN EL PREDICADO LÓGICO P(t)
  - t ES LA VARIABLE LIBRE
- EN LOS PREDICADOS LÓGICOS P(t) PUEDEN APARECER OPERADORES ARITMÉTICOS, COMPARACIONALES, LÓGICOS, OPERADOR $\in$, CONSTANTES Y VARIABLES
  - TODAS LAS VARIABLES TIENEN QUE ESTAR CUANTIFICADAS (EXISTENCIAL O UNIVERSALMENTE) Y SU DOMINIO CLARAMENTE ESPECIFICADO, SALVO LA VARIABLE LIBRE
  - A LAS VARIABLES QUE NO SON LIBRES SE LES LLAMA LIGADAS

INTRODUCCIÓN INFORMAL AL CRT

- CLIENTE(DNI, NBC, CDC)
- PRÉSTAMO(CS, NP, DNI, I)

CUENTA(CS, NC, DNI, SLD)

SUCURSAL(CS, NS, CDS)

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN LA SUCURSAL CON CÓDIGO1
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN ALGUNA SUCURSAL DE LA LAGUNA

INTRODUCCIÓN INFORMAL AL CRT

- DNI DE LOS CLIENTES CON CUENTAS EN AL MENOS DOS SUCURSALES DISTINTAS
- DNI DE LOS CLIENTES CON AL MENOS UNA CUENTA EN UNA SUCURSAL EN LA CIUDAD EN QUE VIVEN
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO O UNA CUENTA O AMBAS COSAS EN LA SUCURSAL CON CÓDIGO1
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO Y UNA CUENTA EN LA SUCURSAL CON CÓDIGO1

INTRODUCCIÓN INFORMAL AL CRT

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO EN LA SUCURSAL CON CÓDIGO 1 PERO NO TIENEN UNA CUENTA EN DICHA SUCURSAL
- DNI DE LOS CLIENTES QUE TIENEN UNA CUENTA EN TODAS LAS SUCURSALES DE 'LA LAGUNA'
- DNI DEL CLIENTE CON LA CUENTA CON MENOR SALDO
- DNI DEL CLIENTE CON LA CUENTA DE MENOR SALDO EN LA SUCURSAL CON CÓDIGO1

CÁLCULO RESTRINGIDO DE T-UPLAS

- NO SE PUEDE PERMITIR EN CRT CONSULTAS QUE GENEREN UN NÚMERO INFINITO DE FILAS, COMO LA SIGUIENTE:

  $\{t \mid t \notin PRESTAMO\}$

- UN PREDICADO O EXPRESIÓN ES **SEGURO** (O **SANO**) SI PUEDE EVALUARSE CON UN PROCESO O ALGORITMO FINITO
- TODAS LAS EXPRESIONES DEL ÁLGEBRA RELACIONAL SON SEGUROS
- EL CÁLCULO RESTRINGIDO DE T-UPLAS ES EL CÁLCULO RELACIONAL DE T-UPLAS RESTRINGIDO A PREDICADOS QUE TIENEN ALGUNA EXPRESIÓN ALGEBRAICA EQUIVALENTE

POTENCIA EXPRESIVA DEL CÁLCULO RESTRINGIDO DE T-UPLAS

- EL ÁLGEBRA RELACIONAL ES TAN EXPRESIVO COMO EL CÁLCULO RESTRINGIDO DE T-UPLAS
  - TODOS LOS PREDICADOS TIENEN EQUIVALENTE EN ÁLGEBRA RELACIONAL
- RECÍPROCAMENTE, EL CÁLCULO RESTRINGIDO DE T-UPLAS ES TAN EXPRESIVO COMO EL ÁLGEBRA RELACIONAL
  - TODOS LOS OPERADORES ESENCIALES DEL ÁLGEBRA RELACIONAL (PROYECCIÓN, SELECCIÓN, PRODUCTO CARTESIANO, UNIÓN Y DIFERENCIA) TIENEN EQUIVALENTE EN CRT

DEFINICIÓN FORMAL CRT

- UNA CONSULTA EN CRT TIENE LA FORMA: {t / P(t) }
- LOS PREDICADOS ESTÁN CONSTITUIDOS CON ÁTOMOS Y OPERADORES
- LOS ÁTOMOS PUEDEN SER:
  - $s \in R$, DONDE s ES UNA VARIABLE DE TIPO T-UPLA Y R UNA RELACIÓN
  - $s[A]\ \Theta\ u[B]$, DONDE s Y u SON VARIABLES DE TIPO T-UPLA, A Y B SON ATRIBUTOS EN LOS QUE s Y u ESTÁN DEFINIDOS, RESPECTIVAMENTE, Y Θ ES UN OPERADOR COMPARACIONAL
  - $s[A]\ \Theta\ C$, DONDE s ES UNA VARIABLE DE TIPO T-UPLA, A UN ATRIBUTO EN EL QUE s ESTÁ DEFINIDA, C ES UNA CONSTANTE DEL DOMINIO DEL ATRIBUTO A, Y Θ ES UN OPERADOR COMPARACIONAL

DEFINICIÓN FORMAL CRT

- LOS PREDICADOS SE CONSTRUYEN CON ÁTOMOS EMPLEANDO LAS SIGUIENTES REGLAS:
  - UN ÁTOMO ES UN PREDICADO
  - SI P Y Q SON PREDICADOS ENTONCES TAMBIÉN LO SON:
    - $\neg P$
    - (P)
    - $P \vee Q$
    - $P \wedge Q$
  - SI Px ES UN PREDICADO, ENTONCES TAMBIÉN LO SON:
    - $(\forall x)(Px)$
    - $(\exists x)(Px)$
  - TODAS LAS VARIABLES TIENEN QUE ESTAR CUANTIFICADAS SALVO LA VARIABLE LIBRE
  - NINGUNA OTRA COSA ES UN PREDICADO

CÁLCULO RELACIONAL DE DOMINIOS

- ES UN TIPO ESPECIAL DE CÁLCULO RELACIONAL DONDE LAS VARIABLES REPRESENTAN VALORES DEL DOMINIO DE UN ATRIBUTO DE UNA TABLA
- UNA CONSULTA EN CÁLCULO RELACIONAL DE T-UPLAS SE EXPRESA COMO:
  - $\{<x_1, x_2, \ldots, x_n> / P(x_1, x_2, \ldots, x_n)\} \Leftrightarrow$ CONJUNTO DE TODAS LAS T-UPLAS $<x_1, x_2, \ldots, x_n>$ QUE SATISFACEN EL PREDICADO LÓGICO $P(x_1, x_2, \ldots, x_n)$
  - $x_1, x_2, \ldots, x_n$ SON LAS VARIABLES LIBRES
- EN LOS PREDICADOS LÓGICOS $P(x_1, x_2, \ldots, x_n)$ PUEDEN APARECER OPERADORES ARITMÉTICOS, COMPARACIONALES, LÓGICOS, OPERADOR $\in$, CONSTANTES Y VARIABLES
  - TODAS LAS VARIABLES TIENEN QUE ESTAR CUANTIFICADAS (EXISTENCIAL O UNIVERSALMENTE) Y SU DOMINIO CLARAMENTE ESPECIFICADO, SALVO LAS VARIABLES LIBRES
  - A LAS VARIABLES QUE NO SON LIBRES SE LES LLAMA LIGADAS

DEFINICIÓN FORMAL CRD

- UNA CONSULTA EN CRT TIENE LA FORMA: $\{<x_1, x_2, \ldots, x_n> / P(x_1, x_2, \ldots, x_n)\}$
- LOS PREDICADOS ESTÁN CONSTITUIDOS CON ÁTOMOS Y OPERADORES
- LOS ÁTOMOS PUEDEN SER:
  - $<x_1, x_2, \ldots, x_n> \in R$, DONDE $x_1, x_2, \ldots, x_n$ SON VARIABLES DE TIPO DOMINIO Y R UNA RELACIÓN
  - $x \Theta y$, DONDE x E y SON VARIABLES DE TIPO DOMINIO Y Θ ES UN OPERADOR COMPARACIONAL
  - $x \Theta C$, DONDE x ES UNA VARIABLE DE TIPO DOMINIO, C ES UNA CONSTANTE DEL DOMINIO DE x Y Θ ES UN OPERADOR COMPARACIONAL

DEFINICIÓN FORMAL CRD

- LOS PREDICADOS SE CONSTRUYEN CON ÁTOMOS EMPLEANDO LAS SIGUIENTES REGLAS:
  - UN ÁTOMO ES UN PREDICADO
  - SI P Y Q SON PREDICADOS ENTONCES TAMBIÉN LO SON:
    - $\neg P$
    - (P)
    - $P \vee Q$
    - $P \wedge Q$
  - SI Px ES UN PREDICADO, ENTONCES TAMBIÉN LO SON:
    - $(\forall x)(Px)$
    - $(\exists x)(Px)$
  - TODAS LAS VARIABLES TIENEN QUE ESTAR CUANTIFICADAS SALVO LA VARIABLE LIBRE
  - NINGUNA OTRA COSA ES UN PREDICADO

EJEMPLOS CRD

- CLIENTE(DNI, NBC, CDC)
- PRÉSTAMO(CS, NP, DNI, I)

CUENTA(CS, NC, DNI, SLD)

SUCURSAL(CS, NS, CDS)

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN LA SUCURSAL CON CÓDIGO1
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO SUPERIOR A 100000 EUROS EN ALGUNA SUCURSAL DE LA LAGUNA

EJEMPLOS CRD

- DNI DE LOS CLIENTES CON CUENTAS EN AL MENOS DOS SUCURSALES DISTINTAS
- DNI DE LOS CLIENTES CON AL MENOS UNA CUENTA EN UNA SUCURSAL EN LA CIUDAD EN QUE VIVEN
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO O UNA CUENTA O AMBAS COSAS EN LA SUCURSAL CON CÓDIGO1
- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO Y UNA CUENTA EN LA SUCURSAL CON CÓDIGO1

EJEMPLOS CRD

- DNI DE LOS CLIENTES QUE TIENEN UN PRÉSTAMO EN LA SUCURSAL CON CÓDIGO1PERO NO TIENEN UNA CUENTA EN DICHA SUCURSAL
- DNI DE LOS CLIENTES QUE TIENEN UNA CUENTA EN TODAS LAS SUCURSALES DE 'LA LAGUNA'
- DNI DEL CLIENTE CON LA CUENTA CON MENOR SALDO
- DNI DEL CLIENTE CON LA CUENTA DE MENOR SALDO EN LA SUCURSAL CON CÓDIGO1

TEMA 3. SQL

- INTRODUCCIÓN
- HISTORIA
- CONCEPTOS FUNDAMENTALES
- DML
- DDL
- DCL

INTRODUCCIÓN

- SQL – STRUCTURED QUERY LANGUAGE
- ES EL LENGUAJE ESTÁNDAR INDUSTRIAL DE FACTO PARA BASES DE DATOS RELACIONALES
- HA SUFRIDO NUMEROSAS REVISIONES ANSI
  - SQL (ANSI-86), SQL1 (ANSI-89), SQL2 (ANSI-92), SQL3 (ANSI-99), …

HISTORIA

- EN 1974 SE INTRODUCE, POR PARTE DE DONALD CHAMBERLIN Y OTROS, SEQUEL (STRUCTURED ENGLISH QUERY LANGUAGE) Y SE IMPLEMENTÓ EN UN PROTOTIPO LLAMADO SEQUEL-XRM (1974-1975)
- ENTRE 1976 Y 1977 SE PRODUCE UNA REVISIÓN DEL LENGUAJE, SEQUEL/2
- EN 1976, EL LENGUAJE CAMBIA DE NUEVO DE NOMBRE POR MOTIVOS LEGALES Y PASA A LLAMARSE SQL
- EL PROTOTIPO (SYSTEM R), BASADO EN ESTE LENGUAJE, SE ADOPTÓ Y UTILIZÓ INTERNAMENTE EN IBM
- EN 1979 ORACLE FUE LA PRIMERA COMPAÑÍA QUE HIZO UNA IMPLEMENTACIÓN COMERCIAL DE SQL
- EN 1983 IBM EMPEZÓ A VENDER DB2 (SGBD RELACIONAL COMERCIAL)
- A LO LARGO DE LOS 80 OTRAS COMPAÑÍAS COMO SYBASE, … EMPEZARON A COMERCIALIZAR SISTEMAS GESTORES DE BASES DE DATOS BASADOS EN SQL

CONCEPTOS FUNDAMENTALES

- EL LENGUAJE ESTÁ CONSTITUIDO POR UN NÚMERO RELATIVAMENTE PEQUEÑO DE SENTENCIAS
- TODAS LAS SENTENCIAS SQL ACABA EN ‘;’ (DELIMITADOR FINAL DE SENTENCIA)
- EL LENGUAJE ES INSENSITIVO A LAS MAYÚSCULAS Y MINÚSCULAS
  - SE RECOMIENDA COMO REGLA DE ESTILO QUE LAS KEYWORDS DE SQL SE ESCRIBAN EN MAYÚSCULAS
  - LAS SENTENCIAS SE PUEDEN DIVIDIR EN TANTAS LINEAS COMO SE QUIERAN
- LAS SENTENCIAS ESTÁN CONSTITUIDAS POR CLÁUSULAS, ALGUNAS SON OBLIGATORIAS Y OTRAS SON OPCIONALES
  - SE RECOMIENDA QUE LAS CLÁUSULAS DISTINTAS DE UNA MISMA SENTENCIA SE ESCRIBAN EN LÍNEAS DISTINTAS

DML

- LAS SENTENCIAS SQL DEL DML SON:
  - **SELECT**
    - PERMITE REALIZAR CONSULTAS
  - **INSERT**
    - PERMITE AÑADIR FILAS A UNA TABLA
  - **DELETE**
    - PERMITE BORRAR FILAS DE UNA TABLA
  - **UPDATE**
    - PERMITE MODIFICAR FILAS DE UNA TABLA

CONSULTAS

SELECT {**ALL** | **DISTINCT**} {* | columna1 ["alias1"], columna2 ["alias2"], ...}

FROM nombre_tabla [[**AS**] [alias_tabla]];

- LA CLÁUSULA **SELECT** EQUIVALE AL OPERADOR PROYECCIÓN DEL ÁLGEBRA RELACIONAL
- SE CONSIDERA UN ERROR HISTÓRICO DESAFORTUNADO HABERLA DENOMINADO **SELECT**
  - SE CONFUNDE CON EL OPERADOR SELECCIÓN DEL ÁLGEBRA RELACIONAL
- POR DEFECTO SE PERMITEN DUPLICADOS EN LA TABLA DE RESULTADOS DEL SELECT
  - SI SE QUIEREN ELIMINAR LOS DUPLICADOS HAY QUE ESPECIFICAR LA KEYWORD **DISTINCT** O **UNIQUE**
- * EQUIVALE A PROYECTAR SOBRE TODOS LOS ATRIBUTOS DE LA TABLA

EJEMPLOS CONSULTAS ELEMENTALES

```
SELECT *                    /* EQUIVALE A SELECT DNI,NBC, CDC */
FROM CLIENTE;

SELECT NBC "NOMBRE DEL CLIENTE"
FROM CLIENTE;

SELECT DISTINCT NBC
FROM CLIENTE;               /* P(NBC)(CLIENTE)*/
```

EJEMPLOS

```
SQL> SELECT *
  2  FROM CLIENTE;

       DNI NBC                              CDC
---------- -------------------------------- ---------------
      1111 JUAN                             SANTA CRUZ
      2222 CARLOS                           TACORONTE
      3333 MARA                             SANTA CRUZ
      4444 CARMEN                           LA LAGUNA
      5555 PEDRO                            SANTA CRUZ
      6666 CARLOS                           SANTA CRUZ

6 rows selected.
```

EJEMPLOS

```
SQL> SELECT UNIQUE CDC CIUDAD
  2  FROM CLIENTE;

CIUDAD
---------------
LA LAGUNA
SANTA CRUZ
TACORONTE
```

EJEMPLOS

```
SQL> SELECT *
  2  FROM CUENTA;

        CS          NC         DNI         SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

```
SQL> SELECT *
  2  FROM SUCURSAL;

        CS NS                   CDS
---------- -------------------- ---------------
         1 S1                   SANTA CRUZ
         2 S2                   SANTA CRUZ
         3 S3                   LA LAGUNA
         4 S4                   SANTA CRUZ
         5 S5                   LA LAGUNA
         6 S6                   CANDELARIA
         7 S7                   TACORONTE

7 rows selected.
```

EJEMPLOS

```
SQL> SELECT *
  2  FROM PRESTAMO;

        CS         NP        DNI          I
---------- ---------- ---------- ----------
         1        100       1111       5000
         1        100       2222       5000
         2        100       1111       1000
         2        100       3333       1000
         3        100       4444      50000
         4        100       5555       5000
         5        100       6666       1000
         5        200       3333       2000
         7        100       5555       1000

9 rows selected.
```

CLÁUSULA FROM

- LA SINTAXIS DE LA CLÁUSULA FROM ES:

  FROM tabla1 [[**AS**] [alias_tabla1]], tabla2 [[**AS**] [alias_tabla2]], …

- **FROM** EQUIVALE AL PRODUCTO CARTESIANO DEL ALGEBRA RELACIONAL
- CALCULA EL PRODUCTO CARTESIANO DE LAS TABLAS LISTADAS EN EL **FROM**
- A LAS TABLAS SE LE PUEDEN PONER ALIAS

CLÁUSULA FROM

- LA SINTÁXIS DE LA CLÁSULA **FROM** SE PUEDE ENRIQUECER CUANDO PARTICIPAN AL MENOS DOS TABLAS EN LA MISMA:
  - tabla1 {**CROSS JOIN**|**NATURAL** [**INNER**] **JOIN**} tabla2
- **CROSS JOIN** HACE EL PRODUCTO CARTESIANO DE LAS 2 TABLAS
  - AUMENTA LA LEGIBILIDAD
- **NATURAL JOIN** HACE LA YUNCIÓN NATURAL DE LAS 2 TABLAS
  - IMPONE LA CONDICIÓN DE IGUALDAD DE VALORES EN ATRIBUTOS HOMÓNIMOS EN EL PRODUCTO CARTESIANO Y PROYECTA SOBRE LA UNIÓN DE LOS ATRIBUTOS DE LAS 2 TABLAS

EJEMPLOS

```
SELECT *                        /* CUENTA X CLIENTE */
FROM CUENTA CROSS JOIN CLIENTE;

SELECT DISTINCT DNI, CS, CDC    /* P(DNI, CS, CDC) (CUENTA * CLIENTE) */
FROM CUENTA NATURAL JOIN CLIENTE;
```

EJEMPLOS

```
SQL> SELECT CS, NC, DNI, CDC
  2  FROM CLIENTE NATURAL JOIN CUENTA;

        CS         NC        DNI CDC
---------- ---------- ---------- ---------------
         1        100       1111 SANTA CRUZ
         2        100       1111 SANTA CRUZ
         1        100       2222 TACORONTE
         6        500       2222 TACORONTE
         2        100       3333 SANTA CRUZ
         5        100       3333 SANTA CRUZ
         3        100       4444 LA LAGUNA
         3        100       5555 SANTA CRUZ
         4        300       6666 SANTA CRUZ

9 rows selected.
```

EJEMPLOS

```
SQL> SELECT *
  2  FROM CLIENTE NATURAL JOIN CUENTA CU, SUCURSAL S
  3  WHERE CDS = 'LA LAGUNA' AND CU.CS = S.CS;

       DNI NBC                            CDC                 CS         NC        SLD         CS NS                    CDS
---------- ------------------------------ -------------- ---------- ---------- ---------- ---------- -------------------- ----------
      4444 CARMEN                         LA LAGUNA               3        100       5000          3 S3                   LA LAGUNA
      5555 PEDRO                          SANTA CRUZ              3        100       5000          3 S3                   LA LAGUNA
      3333 MARA                           SANTA CRUZ              5        100      10000          5 S5                   LA LAGUNA
```

CLÁUSULA FROM

- LA SINTÁXIS DE LA CLÁSULA **FROM** SE PUEDE ENRIQUECER CUANDO PARTICIPAN AL MENOS DOS TABLAS EN LA MISMA
  - tabla1 [**INNER**] **JOIN** tabla2 {**ON** condición_join | **USING** (lista_columnas)}}
- **[INNER] JOIN ON** condición_join HACE LA YUNCIÓN DE LAS 2 TABLAS SEGÚN EL PREDICADO INDICADO EN condición_join
- **[INNER] JOIN USING** (lista_columnas) HACE UNA YUNCIÓN CON UN PREDICADO IMPLÍCITO QUE IMPONE LA IGUALDAD DE VALORES EN LAS COLUMNAS INDICADAS EN lista_columnas
  - GENERALIZA A LA YUNCIÓN NATURAL
  - LAS COLUMNAS UTILIZADAS EN LA CLÁUSULA **USING** TIENEN QUE SER ATRIBUTOS COMUNES A LAS 2 TABLAS

EJEMPLO

SELECT DISTINCT CDC

FROM CUENTA **INNER JOIN** CLIENTE **USING** (DNI)

WHERE (CS = 1);

/* P(CDC) S(CS = 1) (CUENTA * CLIENTE) */

EJEMPLOS

SELECT DISTINCT CUENTA.DNI

FROM CUENTA **INNER JOIN** CLIENTE **ON** ((CUENTA.DNI = CLIENTE.DNI) **AND** (CS = 1));

/* P(CUENTA.DNI) (CUENTA Y((CUENTA.DNI = CLIENTE.DNI) ∧ (CS = 1)) CLIENTE) */

CLÁUSULA FROM

- LA SINTÁXIS DE LA CLÁSULA **FROM** SE PUEDE ENRIQUECER CUANDO PARTICIPAN AL MENOS DOS TABLAS EN LA MISMA:
  - tabla1 {**LEFT** [**OUTER**] **JOIN**|**RIGHT** [**OUTER**] **JOIN**|**FULL** [**OUTER**] **JOIN**} tabla2
- HACE LA YUNCIÓN NATURAL EXTERNA A IZQUIERDA, A DERECHA O COMPLETA, RESPECTIVAMENTE, DE LAS 2 TABLAS

CLÁUSULA WHERE

SELECT {**ALL** | **DISTINCT**} {* | columna1 [“alias1”], columna2 [“alias2”], ...}

FROM tabla1 [[**AS**] [alias_tabla1]], tabla2 [[**AS**] [alias_tabla2]], …

[**WHERE** condición];

- ES UNA CLÁUSULA OPCIONAL QUE SI APARECE TIENE QUE IR JUSTO DESPUÉS DE LA CLÁUSULA FROM
- EQUIVALE AL OPERADOR SELECCIÓN DEL ÁLGEBRA RELACIONAL

CONDICIÓN DEL WHERE

- ES UN PREDICADO LÓGICO EN EL QUE PUEDEN APARECER:
  - CUALQUIER ATRIBUTO DE LAS TABLAS DEL **FROM**
  - CONSTANTES DE LOS DOMINIOS DE LOS ATRIBUTOS
  - OPERADORES ARITMÉTICOS
    - (,) +, -, *, /,
  - OPERADORES COMPARACIONALES
    - =, <>, !=, <, <=, >, >=
  - OPERADORES LÓGIGOS
    - (,), **AND**, **OR**, **NOT**, … EQUIVALENTES A (,), ∧, ∨, ¬, … DEL ÁLGEBRA RELACIONAL

EJEMPLOS

```
SELECT DISTINCT DNI                    /* P(DNI) (S(CS = 1)(CUENTA)) */
FROM CUENTA
WHERE CS = 1;
```

```
SELECT DISTINCT C1.DNI     FROM CLIENTE C1, CUENTA C2
WHERE (CDC = 'LA LAGUNA') AND (C1.DNI = C2.DNI);
```

/* P(CLIENTE.DNI) (S((CDC = 'LA LAGUNA') ∧ (CLIENTE.DNI = CUENTA.DNI)) (CLIENTE X CUENTA))*/

EJEMPLOS

SELECT DISTINCT DNI /* P(DNI) (S((CS = 1) v (CS = 2)) (CUENTA)) */

FROM CUENTA

WHERE (CS = 1) **OR** (CS = 2);

OPERADOR LÓGICO BETWEEN

- LA SINTAXIS ES:

  expresión1 [**NOT**] **BETWEEN** expresión2 **AND** expresión3

EQUIVALE A

  - (expresión1 >= expresión2) **AND** (expresión1 <= expresión3)

- LA CLÁUSULA **NOT** NIEGA EL PREDICADO Y EQUIVALE A
  - (expresión1 < expresión2) **OR** (expresión1 > expresión3)

EJEMPLO

```
SELECT DNI
FROM CUENTA
WHERE CS BETWEEN 2 AND 5;
```

```
SELECT DNI
FROM CUENTA
WHERE (CS = 1) AND (NC NOT BETWEEN 200 AND 400);
```

OPERADOR LÓGICO IN

- LA SINTAXIS ES:
  - expresión [**NOT**] **IN** (conjunto_de_valores)
- DEVUELVE VERDADERO SI EL VALOR DE LA EXPRESIÓN PERTENECE AL CONJUNTO
- EQUIVALE AL OPERADOR PERTENECE (O NO PERTENECE SI SE USA LA KEYWORD **NOT**)
- EL CONJUNTO DE VALORES PUEDE VENIR DADO DE FORMA EXPLÍCITA O IMPLÍCITA
- DE FORMA EXPLÍCITA LOS VALORES DEL CONJUNTO SE SEPARAN POR COMAS
- DE FORMA IMPLÍCITA LOS VALORES DEL CONJUNTO SON EL RESULTADO DE UNA SUBCONSULTA (CONSULTA ANIDADA)

EJEMPLOS

```
SELECT DNI
FROM CUENTA
WHERE NC IN (100, 110, 120, 130);
```

```
SELECT DNI
FROM CUENTA
WHERE (NC = 100) OR (NC = 110) OR (NC = 120) OR (NC = 130);
```

EJEMPLO

CIUDADES EN QUE RESIDEN LOS CLIENTES DEL BANCO QUE TIENEN ALGUNA CUENTA EN LA SUCURSAL CON CÓDIGO 1

```
SELECT CDC
FROM CLIENTE
WHERE DNI IN    (SELECT DNI
                 FROM CUENTA
                 WHERE CS = 1);
```

```
SQL> SELECT CDC
  2  FROM CLIENTE
  3  WHERE DNI IN (SELECT DNI
  4                FROM CUENTA
  5                WHERE CS = 1);

CDC
---------------
SANTA CRUZ
TACORONTE

SQL>
```

EJEMPLO

DNI DE LOS CLIENTES DEL BANCO QUE NO TIENEN CUENTA EN LA SUCURSAL CON CÓDIGO 1

```
SELECT DNI
FROM CLIENTE
WHERE DNI NOT IN (SELECT DNI
                  FROM CUENTA
                  WHERE CS = 1);
```

CUALIFICADOR ANY

- LA SINTAXIS ES
  - expresión operador_comparacional **ANY** (conjunto_de_valores)
- DEVUELVE VERDADERO SI EL VALOR DE LA EXPRESIÓN VERIFICA EL OPERADOR COMPARACIONAL PARA ALGÚN ELEMENTO DEL CONJUNTO
- EL CONJUNTO DE VALORES PUEDE VENIR DADO DE FORMA EXPLÍCITA O IMPLÍCITA
- = **ANY** EQUIVALE A **IN**
- <> **ANY** NUNCA DEVUELVE FALSO SI EL CONJUNTO TIENE AL MENOS 2 ELEMENTOS DISTINTOS

CUALIFICADOR ALL

- LA SINTAXIS ES
  - expresión operador_comparacional **ALL** (conjunto_de_valores)
- DEVUELVE VERDADERO SI EL VALOR DE LA EXPRESIÓN VERIFICA EL OPERADOR COMPARACIONAL PARA TODOS LOS ELEMENTOS DEL CONJUNTO
- EL CONJUNTO DE VALORES PUEDE VENIR DADO DE FORMA EXPLÍCITA O IMPLÍCITA
- <> **ALL** EQUIVALE A **NOT IN**
- = **ALL** NUNCA DEVUELVE VERDADERO SI EL CONJUNTO TIENE AL MENOS 2 ELEMENTOS DISTINTOS

EJEMPLOS

CLIENTES CON AL MENOS UNA CUENTA CON UN SALDO SUPERIOR AL DE ALGUNA DE LAS CUENTAS DEL CLIENTE CON DNI 1111

```
SELECT DNI
FROM CUENTA
WHERE SLD > ANY (SELECT SLD
                 FROM CUENTA
                 WHERE DNI = 1111);
```

EJEMPLOS

CLIENTES CON AL MENOS UNA CUENTA CON UN SALDO SUPERIOR AL DE CUALQUIERA DE LAS CUENTAS DEL CLIENTE CON DNI 1111

```
SELECT DNI
FROM CUENTA
WHERE SLD > ALL (SELECT SLD
                 FROM CUENTA
                 WHERE DNI = 1111);
```

LIKE

- LA SINTAXIS ES:

  expresión [**NOT**] **LIKE** 'formato'

- DEVUELVE VERDADERO SI LA COMPARACIÓN DE LA EXPRESIÓN SE AJUSTA AL FORMATO
- expresión DEBE DEVOLVER UNA CADENA DE CARACTERES
- EL FORMATO (O PATRÓN) PUEDE INCLUIR METACARACTERES:
  - "%" DESIGNA UNA SERIE DE 0 A N CARACTERES ARBITRARIA
  - "_" DESIGNA UN ÚNICO CHARACTER ARBITRARIO

EJEMPLOS

SELECT DNI

FROM CLIENTE

WHERE NBC LIKE 'M_R%'; -- MAR, MARÍA, MARCOS, MARTA, …

SELECT DNI

FROM CLIENTE

WHERE NBC LIKE '___%'; -- NOMBRES CON AL MENOS 3 CARACTERES

IS NULL

- LA SINTAXIS ES:

  expresión **IS** [**NOT**] **NULL**

- DEVUELVE VERDADERO SI EL VALOR DE LA EXPRESIÓN ES NULO. FALSO EN CASO CONTRARIO

EJEMPLO

CLIENTES PARA LOS QUE SE CONOCE SU CIUDAD DE RESIDENCIA

SELECT DNI

FROM CLIENTE

WHERE CDC **IS NOT NULL;**

EXISTS

- [**NOT**] **EXISTS** (subconsulta)

- DEVUELVE EL VALOR VERDADERO SI LA TABLA DE RESULTADOS DE LA SUBCONSULTA TIENE AL MENOS UNA FILA (ES NO VACÍA)
- EQUIVALE AL CUANTIFICADOR EXISTENCIAL DEL CÁLCULO RELACIONAL
- PERMITE REALIZAR EN SQL CONSULTAS EN ESTILO CÁLCULO RELACIONAL

EJEMPLOS

DNI DE LOS CLIENTES DEL BANCO QUE NO TIENEN CUENTAS EN LA SUCURSAL CON CÓDIGO 1

```
SELECT DNI
FROM CLIENTE C1
WHERE NOT EXISTS IN (SELECT *
                     FROM CUENTA C2
                     WHERE (CS = 1) AND (C1.DNI = C2.DNI);
```

EJEMPLOS

DNI DE LOS CLIENTES DEL BANCO QUE NO TIENEN CUENTAS EN LA SUCURSAL CON CÓDIGO 1

```
SELECT DNI
FROM CLIENTE
WHERE DNI NOT IN      (SELECT DNI
                       FROM CUENTA
                       WHERE (CS = 1);
```

EJEMPLOS

- CLIENTES CON CUENTAS EN TODAS LAS SUCURSALES DE LA LAGUNA

- EN TÉRMINOS DEL CUANTIFICADOR UNIVERSAL LA CONSULTA ANTERIOR EQUIVALE A DECIR QUE PARA CUALQUIERA QUE SEA LA SUCURSAL DE LA LAGUNA EL CLIENTE TIENE CUENTA EN LA MISMA

- EN TÉRMINOS DEL CUANTIFICADOR EXISTENCIAL LA CONSULTA ANTERIOR EQUIVALE A DECIR QUE NO EXISTE UNA SUCURSAL DE LA LAGUNA EN LA QUE EL CLIENTE NO TENGA CUENTA

EJEMPLOS

```
SELECT DNI
FROM CUENTA C1
WHERE NOT EXISTS (SELECT *
                  FROM SUCURSAL S1
                  WHERE (CDS = 'LA LAGUNA')  AND NOT EXISTS (SELECT *
                                                             FROM CUENTA C2
                                                             WHERE C2.DNI =C1.DNI AND
                                                             (C2.CS = S1.CS)));
```

SUBCONSULTAS

- LAS SUBCONSULTAS TAMBIÉN SE CONOCEN COMO CONSULTAS ANIDADAS
- LA CONSULTA PRINCIPAL PUEDE TENER SUBCONSULTAS Y ÉSTAS A SU VEZ SUBSUBCONSULTAS, CON MÚLTIPLES NIVELES DE ANIDAMIENTO
- SE CLASIFICAN SEGÚN EL CONTENIDO DE LA TABLA DE RESULTADOS DE LA MISMA
  - SUBCONSULTAS QUE RETORNAN UN ÚNICO VALOR (SUBCONSULTAS ESCALARES)
  - SUBCONSULTAS QUE DEVUELVEN UN CONJUNTO DE VALORES (EN UNA COLUMNA)
  - SUBCONSULTAS QUE DEVUELVEN MÚLTIPLES COLUMNAS

SUBCONSULTAS ESCALARES

- DNI DE LOS CLIENTES QUE VIVEN EN LA MISMA CIUDAD QUE EL CLIENTE CON DNI 1111

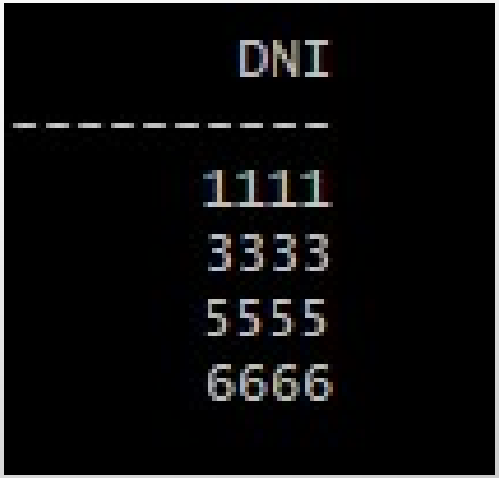
```
SELECT DNI
FROM CLIENTE
WHERE CDC  =  (SELECT CDC
               FROM CLIENTE
               WHERE DNI = 1111);
```

```
      DNI
----------
      1111
      3333
      5555
      6666
```

CONSULTA EQUIVALENTE

- DNI DE LOS CLIENTES QUE VIVEN EN LA MISMA CIUDAD QUE EL CLIENTE CON DNI 1111

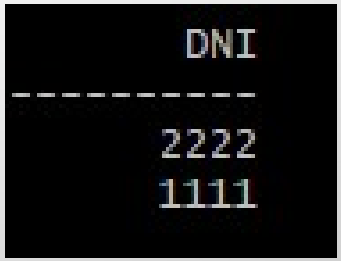
```
SELECT C1.DNI
FROM CLIENTE C1, CLIENTE C2
WHERE (C1.CDC = C2.CDC) AND (C2.DNI = 1111);
```



SUBCONSULTAS QUE DEVUELVEN UNA ÚNICA COLUMNA

- DNI DE LOS TITULARES DE LA CUENTA CON MAYOR SALDO EN LA SUCURSAL CON CÓDIGO 1

```
SELECT DNI
FROM CUENTA
WHERE (CS = 1) AND (SLD >= ALL  (SELECT SLD
                                 FROM CUENTA
                                 WHERE CS = 1));
```



SUBCONSULTAS QUE DEVUELVEN MÚLTIPLES COLUMNAS

- HALLAR LOS DNI DE LOS COTITULARES DE LAS CUENTAS DEL CLIENTE CON DNI 1111

```
SELECT DNI
FROM CUENTA
WHERE (CS, NC) IN (SELECT CS, NC
                   FROM CUENTA
                   WHERE DNI = 1111);
```

```
      DNI
----------
      1111
      2222
      1111
      3333
```

CLÁUSULAS OPERADORES DE CONJUNTO

- LOS OPERADORES DE CONJUNTO SOPORTADOS DIRECTAMENTE EN SQL SON LA UNIÓN, LA INTERSECCIÓN Y LA DIFERENCIA
- TODOS ESTOS OPERADORES TIENEN IGUAL PRECEDENCIA
- SI UNA CONSULTA SQL TIENE VARIAS CLÁUSULAS RELATIVAS A OPERADORES DE CONJUNTO SE EVALÚAN DE IZQUIERDA A DERECHA

UNION

- consulta1 **UNION** [**ALL**] consulta2;
  - HACE LA UNIÓN ENTRE LAS 2 CONSULTAS ELIMINANDO LOS DUPLICADOS
  - SI SE ESPECIFICA **ALL** HACE LA UNIÓN ENTRE LAS 2 CONSULTAS SIN ELIMINAR LOS DUPLICADOS
  - SE APLICAN LAS MISMAS RESTRICCIONES QUE PARA EL OPERADOR UNIÓN DEL ÁLGEBRA RELACIONAL
    - TABLAS CON EL MISMO GRADO
    - DOMINIOS DE LOS ATRIBUTOS IÉSIMOS COMPATIBLES
  - LOS NOMBRES DE LOS ATRIBUTOS DE LA TABLA DE RESULTADO SON LOS DEL OPERANDO DE LA IZQUIERDA

EJEMPLOS

```
SQL> SELECT *
  2  FROM CUENTA
  3  UNION
  4  SELECT *
  5  FROM PRESTAMO;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       1111       5000
         1        100       2222       1500
         1        100       2222       5000
         2        100       1111       1000
         2        100       1111       3000
         2        100       3333       1000
         2        100       3333       3000
         3        100       4444       5000
         3        100       4444      50000
         3        100       5555       5000
         4        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         5        100       6666       1000
         5        200       3333       2000
         6        500       2222       1000
         7        100       5555       1000

18 rows selected.
```

EJEMPLOS

```
SQL> SELECT CS, NC, DNI
  2  FROM CUENTA
  3  UNION
  4  SELECT CS, NP, DNI
  5  FROM PRESTAMO;

        CS         NC        DNI
---------- ---------- ----------
         1        100       1111
         1        100       2222
         2        100       1111
         2        100       3333
         3        100       4444
         3        100       5555
         4        100       5555
         4        300       6666
         5        100       3333
         5        100       6666
         5        200       3333
         6        500       2222
         7        100       5555

13 rows selected.
```

EJEMPLOS

```
SQL> SELECT CS, NC, DNI
  2  FROM CUENTA
  3  UNION ALL
  4  SELECT CS, NP, DNI
  5  FROM PRESTAMO;

        CS         NC        DNI
---------- ---------- ----------
         1        100       1111
         1        100       2222
         2        100       1111
         2        100       3333
         3        100       4444
         3        100       5555
         4        300       6666
         5        100       3333
         6        500       2222
         1        100       1111
         1        100       2222
         2        100       1111
         2        100       3333
         3        100       4444
         4        100       5555
         5        100       6666
         5        200       3333
         7        100       5555

18 rows selected.
```

INTERSECCIÓN

- consulta1 **INTERSECT** consulta2;
  - HACE LA INTERSECCIÓN ENTRE LAS 2 CONSULTAS ELIMINANDO LOS DUPLICADOS
  - SE APLICAN LAS MISMAS RESTRICCIONES QUE PARA EL OPERADOR INTERSECCIÓN DEL ÁLGEBRA RELACIONAL
    - TABLAS CON EL MISMO GRADO
    - DOMINIOS DE LOS ATRIBUTOS IÉSIMOS COMPATIBLES
  - LOS NOMBRES DE LOS ATRIBUTOS DE LA TABLA DE RESULTADO SON LOS DEL OPERANDO DE LA IZQUIERDA

EJEMPLOS

```
SQL> SELECT CS, NC, DNI
  2  FROM CUENTA
  3  INTERSECT
  4  SELECT CS, NP, DNI
  5  FROM PRESTAMO;

        CS         NC        DNI
---------- ---------- ----------
         1        100       1111
         1        100       2222
         2        100       1111
         2        100       3333
         3        100       4444
```

DIFERENCIA

- consulta1 **MINUS** consulta2;
  - HACE LA DIFERENCIA ENTRE LAS 2 CONSULTAS
  - SE APLICAN LAS MISMAS RESTRICCIONES QUE PARA EL OPERADOR DIFERENCIA DEL ÁLGEBRA RELACIONAL
    - TABLAS CON EL MISMO GRADO
    - DOMINIOS DE LOS ATRIBUTOS IÉSIMOS COMPATIBLES
  - LOS NOMBRES DE LOS ATRIBUTOS DE LA TABLA DE RESULTADO SON LOS DEL OPERANDO DE LA IZQUIERDA

EJEMPLOS

```
SQL> SELECT CS, NC, DNI
  2  FROM CUENTA
  3  MINUS
  4  SELECT CS, NP, DNI
  5  FROM PRESTAMO;

        CS          NC         DNI
---------- ---------- ----------
         3        100       5555
         4        300       6666
         5        100       3333
         6        500       2222
```

EJEMPLOS

- CLIENTES QUE TIENEN CUENTAS EN LAS MISMAS SUCURSALES QUE EL CLIENTE CON DNI 2222

```
SQL> SELECT DNI
  2  FROM CUENTA C1
  3  WHERE NOT EXISTS (SELECT CS
  4                    FROM CUENTA
  5                    WHERE DNI = 2222
  6                    MINUS
  7                    SELECT CS
  8                    FROM CUENTA C2
  9                    WHERE C1.DNI = C2.DNI);

       DNI
----------
      2222
      2222
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

- CLIENTES QUE SON COTITULARES DE LAS MISMAS CUENTAS DE LA SUCURSAL CON CÓDIGO 1 QUE EL CLIENTE CON DNI 1111

```
SQL> SELECT DNI
  2  FROM CUENTA C1
  3  WHERE NOT EXISTS (SELECT NC
  4                    FROM CUENTA
  5                    WHERE (CS = 1) AND (DNI = 1111)
  6                    MINUS
  7                    SELECT NC
  8                    FROM CUENTA C2
  9                    WHERE (CS = 1) AND (C2.DNI = C1.DNI));

       DNI
----------
      1111
      2222
      1111
      2222
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

ORDER BY

- LA CLÁUSULA **ORDER BY** ES OPCIONAL Y PERMITE ESPECIFICAR EL ORDEN EN QUE SE VERÁN LAS T-UPLAS EN LA TABLA DE RESULTADO
- SI ESTA CLÁUSULA APARECE EN LA CONSULTA TIENE QUE SER LA ÚLTIMA
- LAS SUBCONSULTAS NO ADMITEN CLÁUSULA **ORDER BY**
- LA SINTAXIS ES:
  - **ORDER BY** expresión1 [**ASC** | **DESC**], expresión2 [**ASC** | **DESC**], …
- POR DEFECTO EL ORDEN ES ASCENDENTE (**ASC**), PERO SE PUEDE ESPECIFICAR EXPLÍCITAMENTE O BIEN INVERTIRLO (**DESC**)

SINTAXIS DEL SELECT

SELECT {**ALL** | **DISTINCT**} {* | columna1 [“alias1”], columna2 [“alias2”], ...}

FROM tabla1 [[**AS**] [alias_tabla1]], tabla2 [[**AS**] [alias_tabla2]], …

[**WHERE** condición]

[cláusulas_operadores de conjunto]

[**ORDER BY** expresión1 [**ASC** | **DESC**], expresión2 [**ASC** | **DESC**], …];

EJEMPLOS

```
SQL> SELECT CDC, DNI
  2  FROM CLIENTE
  3  ORDER BY CDC DESC;

CDC                       DNI
-------------- ----------
TACORONTE               2222
SANTA CRUZ              6666
SANTA CRUZ              1111
SANTA CRUZ              3333
SANTA CRUZ              5555
LA LAGUNA               4444

6 rows selected.
```

```
SQL> SELECT CDC, DNI
  2  FROM CLIENTE
  3  ORDER BY CDC DESC, DNI;

CDC                       DNI
-------------- ----------
TACORONTE               2222
SANTA CRUZ              1111
SANTA CRUZ              3333
SANTA CRUZ              5555
SANTA CRUZ              6666
LA LAGUNA               4444

6 rows selected.
```

FUNCIONES DE GRUPO

- LAS FUNCIONES DE GRUPO DEVUELVEN UN ÚNICO VALOR COMO RESUMEN DE LA INFORMACIÓN RELATIVA AL GRUPO AL QUE SE APLICAN
- SE LES CONOCE TAMBIÉN COMO FUNCIONES INCORPORADAS, FUNCIONES DE AGREGADOS, …
- LAS PRINCIPALES FUNCIONES DE GRUPO INCLUIDAS EN SQL SON:
  - **AVG** - CALCULA LA MEDIA
  - **COUNT** – CUENTA
  - **MAX** – CALCULA EL MÁXIMO
  - **MIN** – CALCULA EL MÍNIMO
  - **SUM** – CALCULA LA SUMA
  - **VARIANCE** – CALCULA LA VARIANZA
  - **STDDEV** – CALCULA LA DESVIACIÓN TÍPICA

FUNCIONES DE GRUPO

- LAS FUNCIONES DE GRUPO SÓLO PUEDEN APARECER EN LA CLÁUSULA **SELECT** O EN LA CLÁUSULA **HAVING**
- SI SE PROYECTA SOBRE UNA FUNCIÓN DE GRUPO, LAS OTRAS PROYECCIONES TIENEN QUE SER SOBRE FUNCIONES DE GRUPO O ATRIBUTOS POR LOS QUE SE HAYA AGRUPADO
- LA SINTAXIS DE TODAS LAS FUNCIONES DE GRUPO, EXCEPTO **COUNT**, ES:
  - función_grupo([**DISTINCT** | **ALL**] expresión)
- LA SINTAXIS DEL COUNT ES:
  - **COUNT** ({ * | [**DISTINCT** | **ALL**] expresión })
- TODAS LAS FUNCIONES DE GRUPO, EXCEPTO **COUNT**(*), IGNORAN LOS NULOS
- CON LAS FUNCIONES **MIN** Y **MAX** LOS MODIFICADORES **DISTINCT** Y **ALL** NO AFECTAN AL RESULTADO

EJEMPLOS

```
SQL> SELECT COUNT(*)
  2  FROM CUENTA;

  COUNT(*)
----------
         9
```

```
SQL> SELECT COUNT(SLD)
  2  FROM CUENTA;

COUNT(SLD)
----------
         9
```

```
SQL> SELECT COUNT(DISTINCT SLD)
  2  FROM CUENTA;

COUNT(DISTINCTSLD)
------------------
                 5
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

- NÚMERO DE CUENTAS DE LA SUCURSAL CON CÓDIGO 1

```
SQL> SELECT COUNT(*)
  2  FROM CUENTA
  3  WHERE CS = 1;

  COUNT(*)
----------
         2
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

- NÚMERO DE CUENTAS DE LA SUCURSAL CON CÓDIGO 1

```
SQL> SELECT COUNT(*)
  2  FROM (SELECT DISTINCT CS, NC FROM CUENTA)
  3  WHERE CS = 1;

  COUNT(*)
----------
         1
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

```
SQL> SELECT AVG(SLD), SUM(SLD), VARIANCE(SLD),STDDEV(SLD)
  2  FROM (SELECT DISTINCT CS, NC, SLD FROM CUENTA);

  AVG(SLD)    SUM(SLD) VARIANCE(SLD) STDDEV(SLD)
---------- ---------- ------------- -----------
5083.33333      30500    16441666.7  4054.83251
```

EJEMPLOS

```
SQL> SELECT MAX(SLD)
  2  FROM CUENTA;

  MAX(SLD)
----------
     10000
```

```
SQL> SELECT MAX(DISTINCT SLD)
  2  FROM CUENTA;

MAX(DISTINCTSLD)
----------------
           10000
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

GROUP BY

- [**GROUP BY** lista_expresiones]
- ES UNA CLÁUSULA OPCIONAL Y SI APARECE EN EL **SELECT** DEBE SEGUIR AL **WHERE** (SI SE USA) O AL **FROM**
- AGRUPA LAS FILAS DE LA TABLA DE RESULTADO HACIENDO QUE LOS VALORES DE LAS EXPRESIONES CONSIDERADAS SEÁN ÚNICAS EN CADA GRUPO
- UNA VEZ SE AGRUPA, SOLO SE PUEDE PROYECTAR SOBRE LAS EXPRESIONES DEL **GROUP BY** Y SOBRE LAS FUNCIONES DE GRUPO
- HABRA TANTOS VALORES EN LA TABLA DE RESULTADO COMO GRUPOS FORMADOS

SINTAXIS DEL SELECT

SELECT {**ALL** | **DISTINCT**} {* | columna1 [“alias1”], columna2 [“alias2”], ...}

FROM tabla1 [[**AS**] [alias_tabla1]], tabla2 [[**AS**] [alias_tabla2]], …

[**WHERE** condición]

[cláusulas_operadores de conjunto]

[**GROUP BY** lista_expresiones]

[**ORDER BY** expresión1 [**ASC** | **DESC**], expresión2 [**ASC** | **DESC**], …];

EJEMPLOS

NÚMERO DE CUENTAS DISTINTAS QUE TIENE CADA SUCURSAL

```
SQL> SELECT CS, COUNT(DISTINCT NC)
  2  FROM CUENTA
  3  GROUP BY CS;

        CS COUNT(DISTINCTNC)
---------- ----------------
         1                1
         6                1
         2                1
         4                1
         5                1
         3                1

6 rows selected.
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

NÚMERO DE TITULARES QUE TIENE CADA CUENTA

```
SQL> SELECT CS, NC, COUNT(*)
  2  FROM CUENTA
  3  GROUP BY CS, NC;

        CS         NC   COUNT(*)
---------- ---------- ----------
         1        100          2
         2        100          2
         3        100          2
         4        300          1
         5        100          1
         6        500          1

6 rows selected.
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

DNI DE LOS CLIENTES CON AL MENOS 2 CUENTAS CON UN SALDO SUPERIOR A 2000 EUROS

```
SQL> SELECT DNI
  2  FROM CUENTA
  3  WHERE SLD > 2000
  4  GROUP BY DNI
  5  HAVING COUNT(*) >= 2;

       DNI
----------
      3333
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

HAVING

- ES UNA CLÁUSULA OPCIONAL
- SI APARECE TIENE QUE SEGUIR OBLIGATORIAMENTE A UNA CLÁUSULA **GROUP BY**
- LA SINTAXIS ES:
  - **HAVING** condición
- SELECCIONA LOS GRUPOS QUE SATISFACEN LA CONDICIÓN
- EN LA CONDICIÓN DEL HAVING SÓLO PUEDEN APARECER ATRIBUTOS POR LOS QUE SE HAYA AGRUPADO, CONSTANTES, SUBCONSULTAS Y FUNCIONES DE GRUPO
- EL **HAVING** ES A LOS GRUPOS LO QUE EL **WHERE** A LAS T-UPLAS

SINTAXIS DEL SELECT

SELECT {**ALL** | **DISTINCT**} {* | columna1 [“alias1”], columna2 [“alias2”], ...}

FROM tabla1 [[**AS**] [alias_tabla1]], tabla2 [[**AS**] [alias_tabla2]], …

[**WHERE** condición]

[cláusulas_operadores de conjunto]

[**GROUP BY** lista_expresiones

[**HAVING** condición]]

[**ORDER BY** expresión1 [**ASC** | **DESC**], expresión2 [**ASC** | **DESC**], …];

EJEMPLOS

DNI DE LOS CLIENTES CON AL MENOS 2 CUENTAS

```
SQL> SELECT DNI
  2  FROM CUENTA
  3  GROUP BY DNI
  4  HAVING COUNT(*) >= 2;

       DNI
----------
      2222
      1111
      3333
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

EJEMPLOS

DNI DE LOS CLIENTES CON AL MENOS 2 CUENTAS EN UNA MISMA SUCURSAL

```
SQL> SELECT DNI
  2  FROM CUENTA
  3  GROUP BY DNI, CS
  4  HAVING COUNT(*) >= 2;

no rows selected
```

```
SQL> SELECT *
  2  FROM CUENTA;

        CS         NC        DNI        SLD
---------- ---------- ---------- ----------
         1        100       1111       1500
         1        100       2222       1500
         2        100       1111       3000
         2        100       3333       3000
         3        100       4444       5000
         3        100       5555       5000
         4        300       6666      10000
         5        100       3333      10000
         6        500       2222       1000

9 rows selected.
```

INSERCIÓN

INSERT INTO nombre_tabla [(columna1, columna2, columna3, ...)]
VALUES (valor1, valor2, valor3, ...);

- INSERTA LA T-UPLA ESPECIFICADA EN LA TABLA
- SI SE ESPECIFICAN LAS COLUMNAS DE LA LISTA, EL NÚMERO DE VALORES TIENE QUE COINCIDIR CON EL DE COLUMNAS
  - LAS COLUMNAS NO LISTADAS SE COMPLETAN CON VALORES POR DEFECTO (**NULL** SI NO SE ESPECIFICÓ UN DEFECTO DE FORMA EXPLÍCITA)

EJEMPLOS

```
SQL> INSERT INTO CLIENTE
  2  VALUES(8888, 'ANTONIO','SANTA CRUZ');

1 row created.

SQL> INSERT INTO CLIENTE(DNI)
  2  VALUES(9999);

1 row created.
```

```
SQL> SELECT *
  2  FROM CLIENTE;

       DNI NBC                            CDC
---------- ------------------------------ ---------------
      1111 JUAN                           SANTA CRUZ
      2222 CARLOS                         TACORONTE
      3333 MARA                           SANTA CRUZ
      4444 CARMEN                         LA LAGUNA
      5555 PEDRO                          SANTA CRUZ
      6666 CARLOS                         SANTA CRUZ
      8888 ANTONIO                        SANTA CRUZ
      9999

8 rows selected.
```

INSERCIÓN DE UNA TABLA DE RESULTADOS

INSERT INTO nombre_tabla[(lista_columnas)]

consulta_sql;

- LA TABLA TIENE QUE EXISTIR
- SI SE ESPECIFICAN LAS COLUMNAS DE LA LISTA, EL NÚMERO DE COLUMNAS SOBRE LAS QUE SE PROYECTA EN LA CONSULTA TIENE QUE COINCIDIR CON EL DE COLUMNAS
  - LAS COLUMNAS NO LISTADAS SE COMPLETAN CON VALORES POR DEFECTO (**NULL** SI NO SE ESPECIFICÓ UN DEFECTO DE FORMA EXPLÍCITA)

EJEMPLOS

```
SQL> INSERT INTO CLIENTES_CON_CUENTA_CS_1
  2  SELECT *
  3  FROM CLIENTE
  4  WHERE DNI IN (SELECT DNI
  5                FROM CUENTA
  6                WHERE CS = 1);

2 rows created.

SQL> SELECT *
  2  FROM CLIENTES_CON_CUENTA_CS_1;

       DNI NBC                            CDC
---------- ------------------------------ ---------------
      1111 JUAN                           SANTA CRUZ
      2222 CARLOS                         TACORONTE
```

BORRADO

DELETE FROM nombre_tabla

[**WHERE** condición];

- LA CLÁUSULA **WHERE** ESTABLECE QUE CONDICIONES TIENEN QUE CUMPLIR LAS FILAS DE LA TABLA PARA SER BORRADAS
- SI NO SE ESPECIFICA LA CLÁUSULA **WHERE** ES UN BORRADO INCONDICIONAL Y SE ELIMINAN TODAS LAS FILAS DE LA TABLA

EJEMPLOS

```
SQL> DELETE FROM CLIENTES_CON_CUENTA_CS_1;

2 rows deleted.
```

```
SQL> DELETE FROM CUENTA
  2  WHERE CS = 1;

2 rows deleted.
```

```
SQL> DELETE FROM CUENTA
  2  WHERE DNI IN (SELECT DNI
  3                FROM CLIENTE
  4                WHERE CDC = 'LA LAGUNA');

1 row deleted.
```

MODIFICACIÓN

UPDATE nombre_tabla

SET col1 = value1, col2 = value2, ...

[**WHERE** condición];

- LA CLÁUSULA **WHERE** ESTABLECE QUE CONDICIONES TIENEN QUE CUMPLIR LAS FILAS DE LA TABLA PARA SER ACTUALIZADAS
- SI NO SE ESPECIFICA LA CLÁUSULA **WHERE** ES UNA MODIFICACIÓN INCONDICIONAL Y SE ACTUALIZAN TODAS LAS FILAS DE LA TABLA

EJEMPLOS

```
SQL> UPDATE CLIENTE
  2  SET NBC = 'NIEVES', CDC = 'LA OROTAVA'
  3  WHERE DNI = 9999;

1 row updated.
```

```
SQL> SELECT *
  2  FROM CLIENTE;

       DNI NBC                            CDC
---------- ------------------------------ --------------
      1111 JUAN                           SANTA CRUZ
      2222 CARLOS                         TACORONTE
      3333 MARA                           SANTA CRUZ
      4444 CARMEN                         LA LAGUNA
      5555 PEDRO                          SANTA CRUZ
      6666 CARLOS                         SANTA CRUZ
      8888 ANTONIO                        SANTA CRUZ
      9999 NIEVES                         LA OROTAVA

8 rows selected.
```

EJEMPLOS

```
SQL> UPDATE CLIENTE
  2  SET CDC = NULL;

8 rows updated.
```

```
SQL> SELECT *
  2  FROM CLIENTE;

       DNI NBC                            CDC
---------- ------------------------------ ---------------
      1111 JUAN
      2222 CARLOS
      3333 MARA
      4444 CARMEN
      5555 PEDRO
      6666 CARLOS
      8888 ANTONIO
      9999 NIEVES

8 rows selected.
```

DDL

- INTRODUCCIÓN
- CREACIÓN DE UNA BASE DE DATOS
- ESQUEMAS
- TABLAS
- VISTAS
- ÍNDICES
- ASERTOS
- DISPARADORES

INTRODUCCIÓN

- LAS SENTENCIAS DE DEFINICIÓN DE DATOS (DDL) PERMITEN CREAR LOS OBJETOS (TABLAS, VISTAS, ÍNDICES, DISPARADORES, SINÓNIMOS, PROCEDIMIENTOS, …) DE LA BASE DE DATOS
- EN GENERAL, LOS OBJETOS DE LA BASE DE DATOS SE PUEDEN CREAR, MODIFICAR SU ESTRUCTURA Y SER ELIMINADOS

CREACIÓN DE UNA BASE DE DATOS

CREATE DATABASE nombre_base_de_datos

[**USER SYS IDENTIFIED BY** password

| **USER SYSTEM IDENTIFIED BY** password

| **CHARACTER SET** conjunto_caracteres

| TABLESPACE_CLAUSES

| SET_TIME_ZONE_CLAUSE

| …];

CREACIÓN DE UNA BASE DE DATOS

- LAS CLÁUSULAS **USER SYS ...** Y **USER SYSTEM ...** PERMITEN ESTABLECER CONTRASEÑAS PARA LOS USUARIOS **SYS** Y **SYSTEM**
  - ESTAS CLÁUSULAS NO SON OBLIGATORIAS
  - SIN EMBARGO, SI SE ESPECIFICA CUALQUIERA DE ELLAS, SE DEBEN ESPECIFICAR AMBAS
  - SI NO SE ESPECIFICAN ESTAS CLÁUSULAS, ORACLE CREA CONTRASEÑAS PREDETERMINADAS change_on_install PARA EL USUARIO SYS Y manager PARA EL USUARIO SYSTEM
- LA CLÁUSULA **CHARACTER SET** ESPECIFICA EL JUEGO DE CARACTERES QUE UTILIZA LA BASE DE DATOS PARA ALMACENAR LOS DATOS
  - LOS JUEGOS DE CARACTERES ADMITIDOS Y EL VALOR PREDETERMINADO DE ESTE PARÁMETRO DEPENDEN DEL SISTEMA OPERATIVO

EJEMPLOS

CREATE DATABASE mi_base_de_datos1;

CREATE DATABASE mi_base_de_datos2

USER SYS IDENTIFIED BY sys_password

USER SYSTEM IDENTIFIED BY system_password

DEFAULT TEMPORARY TABLESPACE temp

DEFAULT TABLESPACE users;

ESQUEMAS

- UN ESQUEMA ES UN CONJUNTO DE TABLAS, VISTAS Y PRIVILEGIOS AGRUPADOS BAJO UN MISMO NOMBRE (EL DEL USUARIO)
- EL USO EXPLÍCITO DE UN ESQUEMA MEDIANTE LA INSTRUCCIÓN **CREATE SCHEMA AUTHORIZATION** APLICA UNA ANALOGÍA DEL CONCEPTO DE TRANSACCIÓN A LAS INSTRUCCIONES DDL PARA LA CREACIÓN DE TABLAS Y VISTAS Y LA ASIGNACIÓN DE PRIVILEGIOS MEDIANTE **GRANT**
- SI UNA DE LAS INSTRUCCIONES DDL ESPECIFICADAS EN LA INSTRUCCIÓN **CREATE SCHEMA AUTHORIZATION** FALLA, EL CONJUNTO ENTERO DE INSTRUCCIONES SE ANULA

CREATE SCHEMA AUTHORIZATION

CREATE SCHEMA AUTHORIZATION nombre_esquema

{ sentencia_creación_tabla | sentencia_creación_vista| sentencia_grant}

… ;

- EL nombre_esquema USADO EN LA SENTENCIA ES EL NOMBRE DEL USUARIO QUE HA ABIERTO LA SESIÓN ACTUAL

CREACIÓN DE TABLAS

CREATE TABLE [nombre_esquema.]nombre_tabla

(lista_definición_columnas

[, lista_restricciones_tabla]);

- EL NOMBRE DE LA TABLA TIENE QUE SER ÚNICO DENTRO DEL ESQUEMA DE LA BASE DE DATOS Y NO PUEDE SER UNA PALABRA RESERVADA DE SQL

DEFINICIÓN DE COLUMNAS

- LA SINTAXIS PARA DEFINIR UNA COLUMNA ES:

  nombre_columna tipo_dato [**DEFAULT** expresión] [restricciones_columna]

- EXISTEN NUMEROSOS TIPOS DE DATOS PARA ALMACENAR LOS VALORES DE LOS CAMPOS DE LAS TABLAS (**NUMBER**, **VARCHAR**, **DATE**, …)
- LAS RESTRICCIONES DE TIPO COLUMNA PUEDEN RESTRINGIR EL DOMINIO DEL ATRIBUTO PARA QUE NO TOME VALORES NULOS, O NO TOME VALORES DUPLICADOS, O PERTENEZCA A UN DETERMINADO CONJUNTO, O SATISFAGA UNA DETERMINADA CONDICIÓN, O SEA UNA CLAVE PRIMARIA O SEA UNA CLAVE AJENA

EJEMPLOS

```
CREATE TABLE CLIENTE
(DNI NUMBER(4) PRIMARY KEY,
NBC VARCHAR2(30),
CDC VARCHAR2(15));

CREATE TABLE SUCURSAL
(CS NUMBER(3) PRIMARY KEY,
NS VARCHAR2(20),
CDS VARCHAR2(15));
```

EJEMPLOS

CREATE TABLE CUENTA

(CS **NUMBER**(3),

NC **NUMBER**(6),

DNI **NUMBER**(4),

SLD **FLOAT DEFAULT** 100.0 **NOT NULL CHECK** (SLD >= 100.0),

PRIMARY KEY (CS, NC, DNI),

CONSTRAINT FK_CUENTA_CLIENTE **FOREIGN KEY** (DNI) **REFERENCES** CLIENTE **ON DELETE CASCADE,**

CONSTRAINT FK_CUENTA_SUCURSAL **FOREIGN KEY** (CS) **REFERENCES** SUCURSAL **ON DELETE CASCADE);**

TIPOS DE DATOS

- EXISTEN NUMEROSOS TIPOS DE DATOS PARA ALMACENAR LOS VALORES DE LOS CAMPOS DE LAS TABLAS
- A CONTINUACIÓN VAMOS A ESTUDIAR LOS PRINCIPALES TIPOS DE DATOS SOPORTADOS EN ORACLE

CHAR(N)

- CADENA DE CARACTERES DE LONGITUD FIJA, TIENE UN TAMAÑO *N* BYTES
- SI NO SE ESPECIFICA *N*, ORACLE LE ASIGNA UN TAMAÑO DE 1 BYTE
- EL TAMAÑO MÁXIMO EN BD ES 2000 BYTES Y EL MÍNIMO 1 BYTE
- EL TAMAÑO MÁXIMO EN PL/SQL ES 32767 BYTES Y EL MÍNIMO 1 BYTE
- **CHARACTER** ES SINÓNIMO DE **CHAR**
- **CHARACTER** Y **CHAR** ESTÁN SOPORTADOS DIRECTAMENTE EN ANSI SQL

VARCHAR2(*N*)

- CADENA DE CARACTERES DE LONGITUD VARIABLE, TIENE UN TAMAÑO MÁXIMO DE *N* BYTES
  - ES OBLIGATORIO ESPECIFICAR EL TAMAÑO
- EL TAMAÑO MÁXIMO EN BD ES 4000 BYTES Y EL MÍNIMO 1 BYTE
- EL TAMAÑO MÁXIMO EN PL/SQL ES 32767 BYTES Y EL MÍNIMO 1 BYTE
- **STRING** Y **VARCHAR** SON SINÓNIMOS DE **VARCHAR2**
- USANDO **VARCHAR2** EN LUGAR DE **CHAR** AHORRAMOS ESPACIO DE ALMACENAMIENTO
- SU DESVENTAJA ES QUE SI SE ESCRIBEN MUCHAS VECES HAY QUE HACER UN MAYOR ESFUERZO DE MANTENIMIENTO DEL SISTEMA PARA MANTENER LA EFICIENCIA (COMPACTAR)
- EL EQUIVALENTE ANSI SQL DE **VARCHAR2**(*N*) ES **CHAR VARYING**(*N*)

NUMBER(P, S)

- NÚMERO DE *P* DIGITOS (PRECISIÓN), DE LOS CUALES *S* SON DECIMALES (ESCALA – DÍGITOS A LA DERECHA DEL PUNTO DECIMAL)
- EL TAMAÑO DE *P* VA DE 1 A 38 Y EL *S* DESDE -84 A 127
- NO ES OBLIGATORIO ESPECIFICAR NI LA PRECISIÓN NI LA ESCALA
- EN ANSI SQL LOS DECIMALES DE PUNTO FIJO SE REPRESENTAN MEDIANTE **NUMERIC**(*P*, *S*) Y **DECIMAL**(*P*, *S*), EQUIVALENTES A **NUMBER**(*P*, *S*)
- EN ANSI SQL LOS ENTEROS SE REPRESENTAN MEDIANTE **INTEGER**, **INT** Y **SMALLINT**, Y SON EQUIVALENTES A **NUMBER**(38)
- EN ANSI SQL LOS DECIMALES DE PUNTO FLOTANTE SE REPRESENTAN COMO **DOUBLE PRECISION**, **FLOAT** Y **REAL** Y SON EQUIVALENTES A **NUMBER**

DATE

- PERMITE ESPECIFICAR UNA FECHA
- EL RANGO DE VALORES PERMITIDOS ES DESDE EL 1 DE ENERO DEL 4712 AC HASTA EL 31 DE DICIEMBRE DEL 9999 DC
- ESTE TIPO DE DATOS ESTÁ SOPORTADO DIRECTAMENTE EN ANSI SQL
- LA FUNCIÓN ORACLE PARA PROPORCIONAR LA FECHA Y HORA DEL SISTEMA ES **SYSDATE**
  - **CURRENT_DATE** ES LA VERSIÓN ANSI DE **SYSDATE** Y TAMBIÉN ESTÁ SOPORTADA EN ORACLE

LONG

- CADENA DE CARACTERES DE LONGITUD VARIABLE
  - ES UNA VERSIÓN MÁS GRANDE DE VARCHAR2
- EL TAMAÑO MÁXIMO ES 2 GIGABYTES

CLOB

- CADENA DE CARACTERES DE LONGITUD VARIABLE
  - ES UNA VERSIÓN MÁS GRANDE DE VARCHAR2
- EL TAMAÑO MÁXIMO ES 4 GIGABYTES

TIMESTAMP(F)

- EL **TIMESTAMP** ES UN FECHA QUE CONTIENE UNA GRANULARIDAD SUPERIOR AL TIPO **DATE**
  PERMITE ALMACENAR FRACCIONES DE SEGUNDO
- SE UTILIZA PRINCIPALMENTE PARA ESTABLECER MARCAS TEMPORALES
- CON F DEFINIMOS EL NUMERO DE DÍGITOS QUE QUEREMOS EN LA FRACCIÓN DE SEGUNDO
  - F PUEDE VARIAR DESDE 0 HASTA 9, EL VALOR POR DEFECTO ES 6.
- ESTE TIPO DE DATOS ESTÁ SOPORTADO DIRECTAMENTE EN ANSI SQL
- LA FUNCIÓN ORACLE PARA PROPORCIONAR UNA MARCA TEMPORAL ES **SYSTIMESTAMP**
  - **CURRENT_TIMESTAMP** ES LA VERSIÓN ANSI DE **SYSTIMESTAMP** Y TAMBIÉN ESTÁ SOPORTADA EN ORACLE

SINTAXIS RESTRICCIONES DE COLUMNA

[**CONSTRAINT** nombre_restricción]

{ [**NOT**] **NULL**

| {**UNIQUE**|**PRIMARY KEY**}

| **REFERENCES** nombre_tabla[(col)] [**ON DELETE** {**CASCADE**|**SET NULL**}]

| **CHECK** (predicado_lógico)

} [**ENABLE** | **DISABLE**]

EJEMPLO

CREATE TABLE CLIENTE

(DNI **NUMBER**(4) **PRIMARY KEY,**

NBC **VARCHAR2**(30) **CONSTRAINT** CK_NBC_CLIENTE **CHECK** (NBC = **UPPER**(NBC)),

CDC **VARCHAR2**(15) **CONSTRAINT** CK_CDC_CLIENTE **CHECK** (CDC **IN** (´LA LAGUNA', 'TACORONTE'));

SINTAXIS RESTRICCIONES DE TABLA

[**CONSTRAINT** nombre_restricción]

{ {**UNIQUE**|**PRIMARY KEY**} (lista_columnas)

| **FOREIGN KEY** (lista_columnas)

  REFERENCES nombre_tabla[(lista_columnas)] [**ON DELETE** {**CASCADE**|**SET NULL**}]

| **CHECK** (predicado_lógico)

} [**ENABLE** | **DISABLE**]

EJEMPLO

CREATE TABLE PRESTAMO

(CS **NUMBER**(3),

NP **NUMBER**(6),

DNI **NUMBER**(4),

I **FLOAT DEFAULT** 1000.0 **NOT NULL CHECK** (I >= 0.0),

PRIMARY KEY (CS, NP, DNI),

CONSTRAINT FK_PRESTAMO_CLIENTE **FOREIGN KEY** (DNI) **REFERENCES** CLIENTE **ON DELETE CASCADE,**

CONSTRAINT FK_PRESTAMO_SUCURSAL **FOREIGN KEY** (CS) **REFERENCES** SUCURSAL **ON DELETE CASCADE**);

EJEMPLOS

```
CREATE TABLE DEPARTAMENTO
(CD NUMBER(2)  PRIMARY KEY,
D VARCHAR2(60));
```

```
CREATE TABLE AREA
(CAR NUMBER(3)  PRIMARY KEY,
AR VARCHAR2(60),
CD NUMBER(2)  REFERENCES DEPARTAMENTO ON DELETE CASCADE);
```

EJEMPLOS

```
CREATE TABLE PROFESOR
(DNI NUMBER(8) PRIMARY KEY,
P VARCHAR2(60),
CAR NUMBER(3),
CAT VARCHAR2(5),
FOREIGN KEY (CAR) REFERENCES AREA ON DELETE SET NULL);
```

EJEMPLOS

```
CREATE TABLE ASIGNATURA
(CAS NUMBER(3) PRIMARY KEY,
A VARCHAR2(50) NOT NULL,
T CHAR(4) NOT NULL,
CUR NUMBER(1) CHECK (CUR BETWEEN 1 AND 5),
CAR NUMBER(3) REFERENCES AREA ON DELETE SET NULL,
CT NUMBER(3,1) DEFAULT 0.0,
CP NUMBER(3,1) DEFAULT 0.0,
CL NUMBER(3,1) DEFAULT 0.0);
```

EJEMPLOS

```
CREATE TABLE PLAN_DOCENTE
(DNI NUMBER(8),
CAS NUMBER(3),
CTA NUMBER(3,1)DEFAULT 0.0,
CPA NUMBER(3,1)DEFAULT 0.0,
CLA NUMBER(3,1)DEFAULT 0.0,
FI DATE DEFAULT SYSDATE,
FF DATE,
PRIMARY KEY (DNI, CAS, FI),
FOREIGN KEY (CAS) REFERENCES ASIGNATURA ON DELETE CASCADE,
FOREIGN KEY (DNI) REFERENCES PROFESOR ON DELETE CASCADE,
CHECK (FF >= FI));
```

CREACIÓN DE TABLAS COMO RESULTADO DE CONSULTAS

CREATE [**OR REPLACE**] **TABLE** [nombre_esquema.]nombre_tabla[(lista_columnas)]

AS consulta_sql;

CREATE TABLE CUENTA_CS_1

AS **SELECT** *

FROM CUENTA

WHERE CS = 1;

ELIMINACIÓN DE TABLAS

- **DROP TABLE** [nombre_esquema.] nombre_tabla [**CASCADE CONSTRAINTS**] [**PURGE**];

- ELIMINA EL CONTENIDO Y LA DEFINICIÓN DE LA TABLA DEL DICCIONARIO DE LA BASE DE DATOS
- LA CLÁUSULA **CASCADE CONSTRAINTS** ELIMINA LAS REGLAS DE INTEGRIDAD REFERENCIAL DEFINIDAS SOBRE LA TABLA
- POR DEFECTO, **DROP TABLE** COLOCA LA TABLA Y LOS OBJETOS DEPENDIENTES DE LA MISMA EN LA PAPELERA DE RECICLAJE
- LA CLÁSUSULA **PURGE** ELIMINA LA TABLA Y LIBERA EL ESPACIO ASOCIADO A LA MISMA EN UN ÚNICO PASO
  - EL BORRADO DE UNA TABLA NO SE PUEDE REVERTIR CON ESTA CLÁUSULA

ELIMINACIÓN DE TABLAS

TRUNCATE TABLE [nombre_esquema.] nombre_tabla [**CASCADE**];

- ELIMINA EL CONTENIDO DE LA TABLA PERO NO LA DEFINICIÓN DE LA MISMA
- ES EQUIVALENTE A **DELETE FROM** [nombre_esquema.] nombre_tabla;
- LA CLÁUSULA **CASCADE** TRUNCA DE FORMA RECURSIVA TODAS LAS TABLAS DEPENDIENTES (A TRAVÉS DE REGLAS DE INTEGRIDAD REFERENCIAL)

MODIFICACIÓN DE LA ESTRUCTURA DE UNA TABLA

ALTER TABLE [nombre_esquema.]nombre_tabla

{**ADD** (definición_col[, …])

|**MODIFY** (modificación_col[, …])

|**DROP** { **COLUMN** nombre_col |(nombre_col[, …]) [**CASCADE CONSTRAINTS**]

|**RENAME COLUMN** nombre_viejo **TO** nombre_nuevo

|**ADD** restricción_nivel_tabla

|**DROP** { **CONSTRAINT** nombre_restricción | **PRIMARY KEY** |**UNIQUE**(lista_columnas) } [**CASCADE**]

|**RENAME CONSTRAINT** nombre_viejo **TO** nombre_Nuevo

| {**ENABLE** | **DISABLE**} **CONSTRAINT** nombre_restricción };

EJEMPLOS

ALTER TABLE CLIENTE

ADD (TELEFONO **NUMBER**(6));

ALTER TABLE CLIENTE

MODIFY (TELEFONO **NUMBER**(9));

ALTER TABLE CLIENTE
DROP COLUMN TELÉFONO;

EJEMPLOS

ALTER TABLE CLIENTE

MODIFY (CDC **CHAR**(30) NOT NULL);

ALTER TABLE CLIENTE

RENAME COLUMN CDC **TO** CIUDAD_CLIENTE;

VISTAS

- UNA VISTA ES UNA TABLA VIRTUAL (O LÓGICA) BASADA EN UNA O MÁS TABLAS O VISTAS
- LA VISTA NO PUEDE SER RECURSIVA
- LA VISTA NO CONTIENE DATOS EN SÍ MISMA
  - SON LENTAS PUES HAY QUE RECALCULARLAS CADA VEZ QUE SE UTILIZAN
- LAS TABLAS EN LAS QUE SE BASA UNA VISTA SE DENOMINAN TABLAS BASE

VISTAS

CREATE [**OR REPLACE**] {**NO FORCE** | **FORCE**} **VIEW** [nombre_esquema.]nombre_vista[(lista_columnas)]

AS consulta_sql;

- EN LA LISTA DE COLUMNAS SE PUEDEN ESPECIFICAR TAMBIEN RESTRICCIONES A NIVEL DE COLUMNA Y A NIVEL DE TABLA
- LA CLÁUSULA **FORCE** PERMITE CREAR LA VISTA INDEPENDIENTEMENTE DE SI EXISTEN LAS TABLAS BASE DE LA VISTA O SI EL PROPIETARIO DEL ESQUEMA QUE CONTIENE LA VISTA TIENE PRIVILEGIOS SOBRE ELLOS
  - ESTAS CONDICIONES DEBEN SER VERDADERAS ANTES DE REALIZAR UN SELECT, INSERT, UPDATE, O DELETE SOBRE LA VISTA
- HAY QUE ESPECIFICAR **NO FORCE** SI DESEA CREAR LA VISTA SOLO SI LAS TABLAS BASE EXISTEN Y EL PROPIETARIO DEL ESQUEMA QUE CONTIENE LA VISTA TIENE PRIVILEGIOS SOBRE ELLAS
  - ES LA OPCIÓN PREDETERMINADA

EJEMPLO

```
CREATE VIEW CUENTA_CS_10
AS      SELECT *
        FROM CUENTA
        WHERE CS = 10;
```

MODIFICACIÓN DE LA ESTRUCTURA DE UNA VISTA

ALTER VIEW [nombre_esquema.]nombre_vista

{**ADD** restricción_nivel_tabla

|**DROP** { **CONSTRAINT** nombre_restricción | **PRIMARY KEY** |**UNIQUE**(LISTA_COLUMNAS) } };

BORRADO DE UNA VISTA

- **DROP VIEW** [nombre_esquema.]nombre_vista;

- LOS OBJETOS QUE SE BASEN EN UNA VISTA ELIMINADA DEJAN DE FUNCIONAR

ÍNDICES

- UN ÍNDICE ES UNA ESTRUCTURA DE DATOS AUXILIAR QUE SIRVE PARA ACELERAR LAS BÚSQUEDAS DENTRO DE UNA TABLA
- LOS ÍNDICES PUEDEN SER CREADOS, ALTERADOS Y BORRADOS DE FORMA EXPLÍCITA
- SI UNA TABLA TIENE CREADO UN ÍNDICE SOBRE UN CONJUNTO DE SUS COLUMNAS Y SE REQUIERE HACER UNA BÚSQUEDA QUE LAS INVOLUCRE, EL SISTEMA USARA EL ÍNDICE DE FORMA TRANSPARENTE AL USUARIO
- CUANDO SE ACTUALIZAN LOS DATOS DE LA TABLA SOBRE LA QUE SE CONSTRUYE EL ÍNDICE, ÉSTE TAMBIÉN TIENE QUE SER ACTUALIZADO
    - LOS ÍNDICES ENLENTECEN LAS ACTUALIZACIONES PUES DICHAS ACTUALIZACIONES TIENEN QUE REPERCUTIRSE EN EL ÍNDICE

ÍNDICES

- ES CONVENIENTE INDEXAR LAS COLUMNAS MÁS FRECUENTEMENTE USADAS EN LA CLÁUSULA **WHERE** DEL **SELECT**
- LOS VALORES NULOS NO SON INDEXADOS
- EN ORACLE LOS **ÍNDICES NORMALES** SON **ÁRBOLES-B**
- LAS CLAVES PRIMARIAS Y LAS RESTRICCIONES DE UNICIDAD SE IMPLEMENTAN A TRAVÉS DE ÍNDICES (IMPLÍCITOS)
- AL CREAR UNA BASE DE DATOS NUEVA ES MÁS RÁPIDO CREAR LAS TABLAS, INSERTAR LAS FILAS Y POSTERIORMENTE CREAR LOS ÍNDICES QUE CREAR LOS ÍNDICES CON LAS TABLAS VACÍAS
  - EVITA QUE TENGAN QUE SER ACTUALIZADOS LOS ÍNDICES CONSTANTEMENTE

CREACIÓN DE ÍNDICES

CREATE [**UNIQUE** | **BITMAP**] **INDEX** [nombre_esquema.]nombre_índice

ON nombre_tabla(col1 [**ASC**|**DESC**] [, …])

[**USABLE** | **UNUSABLE**];

- LA CLÁUSULA **UNIQUE** GARANTIZA QUE LAS COLUMNAS DEL ÍNDICE NO TENGAN VALORES DUPLICADOS DE FORMA CONJUNTA
  - SE TRATA DE UN ÍNDICE NORMAL
- LA CLÁUSULA **BITMAP** HACE QUE EL ÍNDICE SEA DE TIPO BITMAP (MAPAS DE BITS)
  - CADA BIT DEL MAPA DE BITS CORRESPONDE A UN POSIBLE IDENTIFICADOR DE FILA. SI EL BIT ESTÁ ESTABLECIDO, SIGNIFICA QUE LA FILA CON EL ID DE FILA CORRESPONDIENTE CONTIENE EL VALOR DE LA CLAVE

CREACIÓN DE ÍNDICES

- LA CLÁUSULA **USABLE** INDICA QUE EL ÍNDICE ES UTILIZABLE. UN ÍNDICE **UNUSABLE** (INUTILIZABLE) DEBE RECONSTRUIRSE, O ELIMINARSE Y VOLVERSE A CREAR, ANTES DE QUE PUEDA USARSE.
- **ASC** O **DESC** SE UTILIZA PARA INDICAR SI EL ÍNDICE DEBE CREARSE EN ORDEN ASCENDENTE O DESCENDENTE SEGÚN EL JUEGO DE CARACTERES DE LA BASE DE DATOS

BORRADO DE UN ÍNDICE

DROP INDEX [nombre_esquema.]nombre_índice

[**ON** nombre_tabla];

- SI NO SE ESPECIFICA LA TABLA SOBRE LA QUE SE BORRA EL ÍNDICE, EL NOMBRE DEL ÍNDICE DEBE SER ÚNICO

MODIFICACIÓN DE UN ÍNDICE

ALTER INDEX [nombre_esquema.]nombre_índice

{ { **ENABLE** | **DISABLE** }

| **UNUSABLE**

| **RENAME TO** nombre_nuevo

| **REBUILD** {**COMPRESS** | **NOCOMPRESS**} };

MODIFICACIÓN DE UN ÍNDICE

- LA CLÁUSULA **REBUILD** RECONSTRUYE UN ÍNDICE EXISTENTE
  - CUANDO SE BORRAN DATOS EN LAS TABLAS SE CREAN 'HUECOS' EN LOS ÍNDICES. PARA ELIMINAR ESTOS HUECOS HAY QUE RECONSTRUIR EL ÍNDICE
  - SI EL ÍNDICE ESTÁ MARCADO **UNUSABLE,** UNA RECONSTRUCCIÓN EXITOSA LO MARCARÁ **USABLE**
  - LA OPCIÓN COMPRESS ACTIVA LA COMPRESIÓN DE CLAVES, LA CUAL ELIMINA LAS OCURRENCIAS REPETIDAS DE LOS VALORES CLAVES
    - PARA ÍNDICES ÚNICOS LA CLAVE DEBE TENER AL MENOS 2 ATRIBUTOS
    - PARA ÍNDICES NO ÚNICOS LA CLAVE PUEDE TENER UN SOLO ATRIBUTO

CREACIÓN DE SINÓNIMOS

CREATE [**OR REPLACE**] [**PUBLIC**] **SYNONYM** [nombre_esquema.] nombre_sinonimo

FOR [nombre_esquema.] nombre_objecto [@dblink] ;

- **PUBLIC** PERMITE CREAR UN SINÓNIMO PÚBLICO
- LOS SINÓNIMOS PÚBLICOS SON ACCESIBLES PARA TODOS LOS USUARIOS
  - SIN EMBARGO, CADA USUARIO DEBE TENER LOS PRIVILEGIOS ADECUADOS SOBRE EL OBJETO SUBYACENTE PARA PODER UTILIZAR EL SINÓNIMO

ELIMINACIÓN DE SINÓNIMOS

DROP [**PUBLIC**] **SYNONYM** [nombre_esquema.] nombre_sinónimo [**FORCE**];

- LA CLÁUSULA **FORCE** PERMITE ELIMINAR EL SINÓNIMO INCLUSO SI TIENE TABLAS DEPENDIENTES O TIPOS DEFINIDOS POR EL USUARIO

RENOMBRAR OBJETOS

RENAME nombre_viejo **TO** nombre_nuevo;

- SE PUEDEN RENOMBRAR TABLAS, VISTAS Y SINÓNIMOS PRIVADOS
- NO SE PUEDE CAMBIAR EL NOMBRE DE UN SINÓNIMO PÚBLICO
  - EN SU LUGAR, HAY QUE ELIMINAR EL SINÓNIMO PÚBLICO Y LUEGO VOLVER A CREAR EL SINÓNIMO PÚBLICO CON EL NUEVO NOMBRE
- NO SE PUEDE CAMBIAR EL NOMBRE DE UN SINÓNIMO QUE TENGA TABLAS DEPENDIENTES O TIPOS DE OBJETOS DEPENDIENTES VÁLIDOS DEFINIDOS POR EL USUARIO

COMENTARIOS SOBRE OBJETOS

COMMENT ON

{ **COLUMN** [nombre_esquema.] {nombre_tabla. | nombre_vista. | nombre_vista_materializada. } nombre_columna

| **TABLE** [nombre_esquema.] {nombre_tabla | nombre_vista} }

IS string_comentario;

COMENTARIOS SOBRE OBJETOS

- LAS TABLAS DEL DICCIONARIO DE LA BASE DE DATOS QUE GUARDAN LOS COMENTARIOS SOBRE TABLAS Y COLUMNAS SON **USER_TAB_COMMENTS** Y **USER_COL_COMMENTS,** RESPECTIVAMENTE

```
SQL> COMMENT ON TABLE CUENTA IS 'ESTA TABLA CONTIENE INFORMACION SOBRE LAS CUENTAS';

Comment created.

SQL> SELECT COMMENTS
  2  FROM USER_TAB_COMMENTS
  3  WHERE TABLE_NAME = 'CUENTA';

COMMENTS
--------------------------------------------------------------------------------
ESTA TABLA CONTIENE INFORMACION SOBRE LAS CUENTAS
```

```
SQL> DESCRIBE USER_TAB_COMMENTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TABLE_NAME                                NOT NULL VARCHAR2(128)
 TABLE_TYPE                                         VARCHAR2(11)
 COMMENTS                                           VARCHAR2(4000)
 ORIGIN_CON_ID                                      NUMBER
```

```
SQL> DESCRIBE USER_COL_COMMENTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TABLE_NAME                                NOT NULL VARCHAR2(128)
 COLUMN_NAME                               NOT NULL VARCHAR2(128)
 COMMENTS                                           VARCHAR2(4000)
 ORIGIN_CON_ID                                      NUMBER
```

ASERTOS

- SON CONDICIONES DE INTEGRIDAD GENERALES
- EXPRESAN UNA CONDICIÓN QUE HABITUALMENTE AFECTA A VARIAS TABLAS
- LAS CONDICIONES DE INTEGRIDAD A NIVEL DE COLUMNA Y A NIVEL DE TABLA PUEDEN VERSE COMO CASOS PARTICULARES DE ASERTOS
- LOS ASERTOS ESTÁN INCLUIDOS EN EL ANSI SQL, PERO ORACLE NO LOS SOPORTA

ASERTOS

CREATE ASSERTION nombre_aserto **CHECK**(predicado);

ALTER ASSERTION nombre_aserto {**ENABLE** | **DISABL**E};

DROP ASSERTION nombre_aserto;

EJEMPLOS

- CADA CLIENTE QUE TENGA UN PRÉSTAMO DEBE TENER ABIERTA AL MENOS UNA CUENTA CON SALDO SUPERIOR A 1000 EUROS

- LA SUMA DE LOS IMPORTES DE LOS PRÉSTAMOS DE CADA SUCURSAL DEBE SER MENOR QUE LA SUMA DE TODOS LOS SALDOS DE LAS CUENTAS DE ESA SUCURSAL

DISPARADOR (TRIGGER)

- ES UN CONJUNTO DE ÓRDENES E INSTRUCCIONES QUE EL SISTEMA GESTOR EJECUTA DE MANERA AUTOMÁTICA COMO EFECTO SECUNDARIO O COLATERAL DE LA MODIFICACIÓN DE LA BASE DE DATOS Y LA SATISFACCIÓN DE UN DETERMINADO PREDICADO (OPCIONAL)
- TAMBIÉN SE LES CONOCE CON EL NOMBRE DE INICIADOR, DESENCADENADOR, DETONANTE
- LAS **BASES DE DATOS ACTIVAS** SON BASES DE DATOS QUE SOPORTAN DISPARADORES Y OTROS MECANISMOS QUE PERMITEN A LA BASE DE DATOS REALIZAR ACCIONES CUANDO SUCEDAN EVENTOS
- HAY QUE EVITAR QUE LOS DISPARADORES SEAN RECURSIVOS
- LOS DISPARADORES SE BASAN EN EL MODELO EVENTO-CONDICIÓN-ACCIÓN

MODELO EVENTO-CONDICIÓN-ACCIÓN

- CUANDO OCURRE UN DETERMINADO EVENTO O SUCESO EN LA BASE DE DATOS Y SE CUMPLE LA CONDICIÓN ESPECIFICADA SE EJECUTAN LAS ACCIONES DEL DISPARADOR
- EL EVENTO PUEDE SER CUALQUIER OPERACIÓN DE ACTUALIZACIÓN DE LA BASE DE DATOS
- LA COMPROBACIÓN DE LA OCURRENCIA DEL EVENTO PUEDE REALIZARSE A PRIORI O A POSTERIORI DEL MISMO
- LA ESPECIFICACIÓN DE LA CONCIDIÓN ES OPCIONAL

SINTAXIS DISPARADOR

```
CREATE [OR REPLACE] TRIGGER <NOMBRE_TRIGGER>
{BEFORE|AFTER} {DELETE|INSERT|UPDATE [OF COL1, COL2, ..., COLN]}
[OR {DELETE|INSERT|UPDATE [OF COL1, COL2, ..., COLN]...]}
ON <NOMBRE_TABLA>
[FOR EACH ROW [WHEN (<CONDICION>)]]
[DECLARE]
  -- VARIABLES LOCALES
BEGIN
  -- SENTENCIAS
[EXCEPTION]
  -- SENTENCIAS CONTROL DE EXCEPCION
END <NOMBRE_TRIGGER>;
```

SINTAXIS DISPARADOR

- EL USO DE **OR REPLACE** PERMITE SOBREESCRIBIR UN TRIGGER EXISTENTE
  - SI SE OMITE, Y EL TRIGGER EXISTE, SE PRODUCIRÁ, UN ERROR
- LOS TRIGGERS PUEDEN DEFINIRSE PARA LAS OPERACIONES INSERT, UPDATE O DELETE
- SE DEBE ESPECIFICAR SI SE EJECUTAN ANTES (**BEFORE**) O DESPUÉS (**AFTER**) DE LA OPERACIÓN DE ACTUALIZACIÓN
- SI INCLUIMOS EL MODIFICADOR **OF** EL TRIGGER SOLO SE EJECUTARÁ CUANDO LA SENTENCIA SQL AFECTE A LOS CAMPOS INCLUIDOS EN LA LISTA

SINTAXIS DISPARADOR

- EL MODIFICADOR **FOR EACH ROW** INDICA QUE EL BLOQUE PL/SQL DEL TRIGGER SE EJECUTARÁ POR CADA FILA AFECTADA POR LA OPERACIÓN DE ACTUALIZACIÓN
  - EL DISPARADOR SE DENOMINA DISPARADOR POR FILA
  - SI SE ACOMPAÑA DEL MODIFICADOR WHEN, SE ESTABLECE UNA RESTRICCIÓN Y EL TRIGGER SOLO ACTUARÁ, SOBRE LAS FILAS QUE SATISFAGAN LA RESTRICCIÓN
- SI SE OMITE **FOR EACH ROW** EL BLOQUE PL/SQL DEL TRIGGER SE EJECUTARÁ UNA SOLA VEZ POR CADA INSTRUCCIÓN DE ACTUALIZACIÓN
  - EL DISPARADOR SE DENOMINA DISPARADOR POR SENTENCIA

RESUMEN SINTAXIS DISPARADOR

Valor	Descripción
INSERT, DELETE, UPDATE	Define qué tipo de orden DML provoca la activación del disparador.
BEFORE , AFTER	Define si el disparador se activa antes o después de que se ejecute la orden.
FOR EACH ROW	Los disparadores con nivel de fila se activan una vez por cada fila afectada por la orden que provocó el disparo. Los disparadores con nivel de orden se activan sólo una vez, antes o después de la orden. Los disparadores con nivel de fila se identifican por la cláusula FOR EACH ROW en la definición del disparador.

VARIABLES OLD Y NEW

- DENTRO DEL ÁMBITO DE UN TRIGGER DISPONEMOS DE LAS VARIABLES **OLD** Y **NEW**
- ESTAS VARIABLES SE UTILIZAN DEL MISMO MODO QUE CUALQUIER OTRA VARIABLE PL/SQL, CON LA SALVEDAD DE QUE NO ES NECESARIO DECLARARLAS
- CONTIENEN UNA COPIA DEL REGISTRO ANTES (OLD) Y DESPUÉS (NEW) DE LA ACCIÓN SQL (INSERT, UPDATE, DELTE) QUE HA EJECUTADO EL TRIGGER
- UTILIZANDO ESTAS VARIABLES PODEMOS ACCEDER A LOS DATOS QUE SE ESTÁN INSERTANDO, ACTUALIZANDO O BORRANDO
- SE REFERENCIAN COMO VARIABLES HOST MEDIANTE EL PREFIJO ":"
- SON DE TIPO **%ROWTYPE**
  - PARA ACCEDER A LOS CAMPOS SE USA EL OPERADOR "." (:OLD.nombre_campo, :NEW.nombre_campo)
- SE PUEDEN UTILIZAR EN LA CLÁUSULA WHEN O EN EL BLOQUE DE INSTRUCCIONES DEL DISPARADOR

CLÁUSULA REFERENCING

- LOS NOMBRES OLD Y NEW ESTÁN DEFINIDOS DE MANERA PREDETERMINADA, PERO ES POSIBLE UTILIZAR OTROS NOMBRES USANDO LA CLÁUSULA **REFERENCING** CUYA SINTAXIS ES:
  - REFERENCING OLD [AS] NUEVO_NOMBRE_OLD NEW [AS] NUEVO_NOMBRE_NEW
- ESTA CLÁUSULA SE INCLUYE JUSTO ANTES DE LA CLÁUSULA FOR EACH ROW (SI EXISTE)

VALORES DE OLD Y NEW

ACCION SQL	OLD	NEW
INSERT	No definido; todos los campos toman valor NULL.	Valores que serán insertados cuando se complete la orden.
UPDATE	Valores originales de la fila, antes de la actualización.	Nuevos valores que serán escritos cuando se complete la orden.
DELETE	Valores, antes del borrado de la fila.	No definidos; todos los campos toman el valor NULL.

EJEMPLO DE DISPARADOR

- DISPARADOR QUE INSERTA UN REGISTRO EN LA TABLA PRECIOS_PRODUCTOS CADA VEZ QUE INSERTAMOS UN NUEVO REGISTRO EN LA TABLA PRODUTOS
- EL PRECIO POR DEFECTO SERÁ 100
- LA FECHA POR DEFECTO ES LA DEL SISTEMA

EJEMPLO DE DISPARADOR

```
CREATE OR REPLACE TRIGGER TR_PRODUCTOS_01 AFTER INSERT ON PRODUCTOS
FOR EACH ROW
BEGIN
  INSERT INTO PRECIOS_PRODUCTOS(CO_PRODUCTO,PRECIO,FX_ACTUALIZACION)
  VALUES (:NEW.CO_PRODUCTO,100,SYSDATE);
END;
```

SECUENCIA DE EJECUCIÓN DE DISPARADORES

- UNA MISMA TABLA PUEDE TENER VARIOS TRIGGERS
- LOS DISPARADORES SE ACTIVAN AL EJECUTARSE LA SENTENCIA SQL
  - SI EXISTE, SE EJECUTA EL DISPARADOR DE TIPO BEFORE (DISPARADOR PREVIO) CON NIVEL DE SENTENCIA
  - PARA CADA FILA A LA QUE AFECTE LA SENTENCIA :
    - SE EJECUTA SI EXISTE, EL DISPARADOR DE TIPO BEFORE CON NIVEL DE FILA
    - SE EJECUTA LA PROPIA SENTENCIA
    - SE EJECUTA SI EXISTE, EL DISPARADOR DE TIPO AFTER (DISPARADOR POSTERIOR) CON NIVEL DE FILA.
  - SE EJECUTA, SI EXISTE, EL DISPARADOR DE TIPO AFTER CON NIVEL DE SENTENCIA.

DISPARADORES Y CONTROL DE TRANSACCIONES

- EL DISPARADOR SE ACTIVA COMO PARTE DE LA EJECUCIÓN DE LA ORDEN QUE PROVOCÓ EL DISPARO, Y FORMA PARTE DE LA MISMA TRANSACCIÓN QUE DICHA ORDEN
- CUANDO LA ORDEN QUE PROVOCA EL DISPARO ES CONFIRMADA O CANCELADA, SE CONFIRMA O CANCELA TAMBIÉN EL TRABAJO REALIZADO POR EL DISPARADOR
- UN DISPARADOR NO PUEDE EMITIR NINGUNA ORDEN DE CONTROL DE TRANSACCIONES: **COMMIT, ROLLBACK** O **SAVEPOINT**

DETERMINACIÓN DE LA OPERACIÓN QUE EJECUTÓ EL DISPARADOR

- DENTRO DE UN DISPARADOR EN EL QUE SE DISPARAN DISTINTOS TIPOS DE ÓRDENES DML (INSERT, UPDATE Y DELETE), HAY TRES FUNCIONES BOOLEANAS QUE PUEDEN EMPLEARSE PARA DETERMINAR DE QUÉ OPERACIÓN SE TRATA
- ESTOS PREDICADOS SON INSERTING, UPDATING Y DELETING
- SU COMPORTAMIENTO ES EL SIGUIENTE:

Predicado	Comportamiento
INSERTING	TRUE si la orden de disparo es INSERT; FALSE en otro caso.
UPDATING	TRUE si la orden de disparo es UPDATE; FALSE en otro caso.
DELETING	TRUE si la orden de disparo es DELETE; FALSE en otro caso.

BORRADO DE UN DISPARADOR

- **DROP TRIGGER** NOMBRE-DISPARADOR;

SQL DCL GESTIÓN DE TRANSACCIONES

- UNA TRANSACCIÓN ES UN CONJUNTO DE OPERACIONES QUE EL SISTEMA DE BASES DE DATOS CONTEMPLA COMO UNA UNIDAD (COMO UN ÁTOMO – INDIVISIBLE)
- POSIBLEMENTE LA TRANSACCIÓN ESTÁ FORMADA POR UNA O VARIAS OPERACIONES DE LECTURA/ESCRITURA
- UNA ÚNICA SENTENCIA SQL SE CONSIDERA QUE ES ATÓMICA, TANTO SI COMPLETA SU EJECUCIÓN SIN ERRORES COMO SI FALLA Y DEJA LA BASE DE DATOS SIN MODIFICAR
- EN SQL NO HAY UNA SENTENCIA EXPLÍCITA DE COMIENZO DE TRANSACCIÓN
- EL INICIO DE UNA TRANSACCIÓN COMIENZA IMPLÍCITAMENTE CON LA PRIMERA SENTENCIA EJECUTABLE SQL DESPUÉS DE UNA CONEXIÓN O DEL FINAL DE OTRA TRANSACCIÓN

COMMIT WORK

- **COMMIT** [**WORK**];

- CONFIRMA Y FINALIZA LA TRANSACCIÓN ACTUAL, DANDO COMIENZO A UNA NUEVA
- EN EL MOMENTO EN QUE SE CONFIRMA UNA TRANSACCIÓN LOS CAMBIOS PRODUCIDOS POR LA MISMA EN LA BASE DE DATOS SE HACEN VISIBLES A LOS DEMÁS USUARIOS

SAVEPOINT

SAVEPOINT punto_referencia;

- PERMITE ESTABLECER PUNTOS DE REFERENCIA (O DE CONTROL, SALVAGUARDA, GRABACIÓN, …) A LO LARGO DE UNA TRANSACCIÓN
- LOS PUNTOS DE CONTROL PERMITEN MEMORIZAR EL ESTADO DE LOS DATOS EN UN INSTANTE DETERMINADO DURANTE LA EJECUCIÓN DE LA TRANSACCIÓN, PERMITIENDO DESHACER PARCIALMENTE LAS TRANSACCIONES HASTA DICHOS PUNTOS

ROLLBACK WORK

ROLLBACK [**WORK**] [**TO** [**SAVEPOINT**] punto_referencia];

- DESHACE (RETROCEDE) LA TRANSACCIÓN ACTUAL HACIENDO QUE ABORTE LA MISMA
- SI SE ESPECIFICA LA CLÁUSULA **TO SAVEPOINT** DESHACE PARCIALMENTE LA TRANSACCIÓN HASTA EL PUNTO DE REFERENCIA ESPECIFICADO

EJEMPLOS

DELETE FROM SUCURSAL **WHERE** CDS = 'LA LAGUNA';

SAVEPOINT S1;

INSERT INTO CLIENTE **VALUES**(8888, 'CARLOS', 'ARAFO');

ROLLBACK WORK TO SAVEPOINT S1;

COMMIT WORK;

FINALIZACIÓN IMPLÍCITA DE UNA TRANSACCIÓN

- SI EL PROGRAMA ACABA SIN REALIZAR UN **COMMIT** O UN **ROLLBACK** DE LA TRANSACCIÓN ACTUAL, LOS CAMBIOS O BIEN SE COMPROMETEN O BIEN SE RETROCEDEN
  - EN LA NORMA NO SE ESPECIFICA CUÁL DE LAS DOS ACCIONES TIENE LUGAR, Y DEPENDE DE LA IMPLEMENTACIÓN DEL SISTEMA DE BASES DE DATOS
  - EN ORACLE, SE COMPROMETE LA TRANSACCIÓN
- ORACLE COMPROMETE IMPLÍCITAMENTE (SIN NECESIDAD DE **COMMIT**) LA TRANSACCIÓN ACTUAL ANTES Y DESPUÉS DE EJECUTAR CUALQUIER SENTENCIA DDL

INTRODUCCIÓN SEGURIDAD

- LOS DATOS GUARDADOS EN LA BASE DE DATOS DEBEN ESTAR PROTEGIDOS CONTRA LOS ACCESOS NO AUTORIZADOS Y DE LA DESTRUCCIÓN O ALTERACIÓN DE LOS DATOS MALINTENCIONADOS
- LOS SGBD PROPORCIONAN MECANISMOS PARA PROTEGERSE DE DICHAS INCIDENCIAS
- NO ES POSIBLE LA PROTECCIÓN ABSOLUTA DE LA BASE DE DATOS CONTRA EL USO MALINTENCIONADO, PERO SE PUEDE ELEVAR LO SUFICIENTE EL COSTE PARA QUIEN LO COMETE COMO PARA DISUADIR LA MAYOR PARTE, SI NO LA TOTALIDAD, DE LOS INTENTOS DE TENER ACCESO A LA BASE DE DATOS SIN LA AUTORIZACIÓN ADECUADA

TIPOS DE ACCESO MALINTENCIONADO

- LECTURA O COPIA NO AUTORIZADA DE LOS DATOS
  - SE LE CONOCE COMO ROBO DE INFORMACIÓN
  - CONTRABANDO DE INFORMACIÓN, EXTORSIÓN, CHANTAJE, PUBLICACIÓN DE INFORMACIÓN SENSIBLE, …
- MODIFICACIÓN NO AUTORIZADA DE LOS DATOS
  - PARA CREAR CONFUSIÓN Y AFECTAR A LA TOMA DE DECISIONES, PARA OBTENER BENEFICIOS PERSONALES, PARA PERJUDICAR A TERCEROS, …
- DESTRUCCIÓN NO AUTORIZADA DE LOS DATOS
  - IMPOSIBILIDAD DE UTILIZAR LOS DATOS COMO EVIDENCIA DE HECHOS, VENGANZAS, …

MEDIDAS DE SEGURIDAD EN LA BASE DE DATOS SEGÚN LOS NIVELES

- **SISTEMA DE BASES DE DATOS**
  - ES RESPONSABILIDAD DEL SISTEMA DE BASES DE DATOS ASEGURARSE DE QUE NO SE VIOLEN LAS RESTRICCIONES DE ACCESO Y AUTORIZACIÓN SOBRE LOS DATOS
  - PUEDE QUE ALGUNOS USUARIOS DEL SISTEMA DE BASES DE DATOS SÓLO ESTÉN AUTORIZADOS A TENER ACCESO A UNA PARTE LIMITADA DE LA BASE DE DATOS (RESTRICCIONES SOBRE EL CONTENIDO)
  - PUEDE QUE OTROS USUARIOS ESTÉN AUTORIZADOS A FORMULAR CONSULTAS PERO TENGAN PROHIBIDO MODIFICAR LOS DATOS (RESTRICCIONES SOBRE LAS OPERACIONES)

MEDIDAS DE SEGURIDAD EN LA BASE DE DATOS SEGÚN LOS NIVELES

- **SISTEMA OPERATIVO**
  - INDEPENDIENTEMENTE DE LO SEGURO QUE PUEDA SER EL SISTEMA DE BASES DE DATOS, LA DEBILIDAD DE LA SEGURIDAD DEL SISTEMA OPERATIVO PUEDE SERVIR COMO MEDIO PARA EL ACCESO NO AUTORIZADO A LA BASE DE DATOS
- **RED**
  - DADO QUE CASI TODOS LOS SISTEMAS DE BASES DE DATOS PERMITEN EL ACCESO REMOTO MEDIANTE TERMINALES O REDES, LA SEGURIDAD EN EL SOFTWARE DE RED ES TAN IMPORTANTE COMO LA SEGURIDAD DEL SISTEMA OPERATIVO, TANTO EN INTERNET COMO EN LAS REDES PRIVADAS DE LAS EMPRESAS

MEDIDAS DE SEGURIDAD EN LA BASE DE DATOS SEGÚN LOS NIVELES

- **FÍSICO**
  - LOS SITIOS QUE CONTIENEN LOS SISTEMAS INFORMÁTICOS DEBEN ESTAR PROTEGIDOS FÍSICAMENTE CONTRA LA ENTRADA NO AUTORIZADA DE INTRUSOS
- **HUMANO**
  - LOS USUARIOS DEBEN SER AUTORIZADOS CUIDADOSAMENTE PARA REDUCIR LA POSIBILIDAD DE QUE ALGUNO DE ELLOS DÉ ACCESO A INTRUSOS A CAMBIO DE SOBORNOS U OTROS FAVORES

CONTROL DE ACCESO A LA BASE DE DATOS

- EL PRIMER PASO PARA QUE UNA PERSONA TENGA ACCESO A UNA BASE DE DATOS CONSISTE EN DISPONER DE UNA CUENTA DE USUARIO
- ALGUNOS SISTEMAS DE BASES DE DATOS NO DISPONEN DE CUENTAS DE USUARIO Y DELEGAN ESTAS FUNCIONES EN EL SISTEMA OPERATIVO
  - ACCESS
- ESTA CUENTA ES HABITUALMENTE CREADA Y GESTIONADA POR EL ADMINISTRADOR DE LA BASE DE DATOS MEDIANTE SENTENCIAS SQL
  - CREATE USER, ALTER USER USER, DROP USER, …

CREATE USER

CREATE USER username

IDENTIFIED BY password

[**DEFAULT TABLESPACE** tablespace]

[**QUOTA** {**SIZE** | **UNLIMITED**} **ON** tablespace]

[**PASSWORD EXPIRE**]

[**ACCOUNT** {**LOCK** | **UNLOCK**}];

CREATE USER

- **CREATE USER** username – NOMBRE DEL USUARIO QUE VA A SER CREADO
- **IDENTIFIED BY** password – ESPECIFICA EL PASSWORD PARA QUE UN USUARIO LOCAL PUEDA LOGEARSE EN LA BASE DE DATOS
- **DEFAULT TABLESPACE** – ESPECIFICA EL ESPACIO EN QUE SE CREARÁN LOS OBJETOS COMO LAS TABLAS O LAS VISTAS DEL USUARIO
- **QUOTA** – ESPECIFICA EL MÁXIMO ESPACIO QUE PUEDE USAR EL USUARIO EN EL ESPACIO DE TABLAS
- **PASSWORD EXPIRE** – FUERZA A QUE EL USUARIO TENGA QUE CAMBIAR EL PASSWORD LA PRIMERA VEZ QUE ENTRA EN LA BASE DE DATOS
- **ACCOUNT {LOCK | UNLOCK}** – BLOQUEA|DESBLOQUEA A UN USUARIO Y DESACTIVA|ACTIVA SU ACCESO

CREATE USER - EJEMPLOS

CREATE USER john

IDENTIFIED BY abcd1234;

CREATE USER jane

IDENTIFIED BY abcd1234

PASSWORD EXPIRE;

CREATE USER bar

IDENTIFIED BY abcd1234

QUOTA 5m **ON users;**

ALTER USER

ALTER USER username

{ **IDENTIFIED** { **BY** password [**REPLACE** old_password] }

| **DEFAULT TABLESPACE** tablespace

| **DEFAULT ROLE** { role [, role]... | **ALL** [**EXCEPT** role [, role]...] | **NONE** }

| **PASSWORD EXPIRE**

| **ACCOUNT** { **LOCK** | **UNLOCK** } };

DROP USER

DROP USER username [**CASCADE**];

- ELIMINA EL USUARIO ESPECIFICADO
- SI EL USUARIO POSEE OBJETOS EN LA BASE DE DATOS HABRÁ QUE ELIMINARLOS PRIMERO MANUALMENTE, LO CUAL ES BASTANTE TEDIOSO
- LA OPCIÓN **CASCADE** PERMITE QUE ORACLE BORRE TODOS LOS OBJETOS DEL USUARIO DE FORMA AUTOMÁTICA ANTES DE ELIMINAR AL USUARIO

TIPOS DE AUTORIZACIÓN

- LAS AUTORIZACIONES DE ACCESO A LOS DATOS PUEDEN CLASIFICARSE COMO:
  - PRIVILEGIOS SOBRE OBJETOS
    - SE CONCEDEN SOBRE OBJETOS (TABLAS O VISTAS) EXISTENTES DE LA BASE DE DATOS
    - HAY UN NÚMERO REDUCIDO DE PRIVILEGIOS SOBRE OBJETOS
  - PRIVILEGIOS SOBRE EL SISTEMA
    - PERMITEN A LOS USUARIOS CREAR SUS PROPIOS OBJETOS O INFLUIR CON DISTINTO GRADO SOBRE OTROS ASPECTOS Y OBJETOS DE LA BASE DE DATOS
    - HAY UNA AMPLIA VARIEDAD DE PRIVILEGIOS DEL SISTEMA

PRIVILEGIOS SOBRE OBJETOS

- **SELECT** – AUTORIZACIÓN DE LECTURA
- **INSERT**[(col, …)] – AUTORIZACIÓN DE INSERCIÓN
- **UPDATE** [(col, …)] – AUTORIZACIÓN DE MODIFICACIÓN
- **DELETE** – AUTORIZACIÓN DE BORRADO
- **INDEX** – AUTORIZACIÓN PARA CREACIÓN DE ÍNDICES
- **ALTER** – AUTORIZACIÓN PARA MODIFICAR EL ESQUEMA DE UN OBJETO
- **REFERENCES**[(col, …)] – AUTORIZACIÓN PARA CREAR CLAVES AJENAS
- **EXECUTE** – AUTORIZACIÓN PARA LA EJECUCIÓN DE CÓDIGO PL/SQL

PRIVILEGIOS SOBRE OBJETOS

- LOS USUARIOS PUEDEN RECIBIR TODOS LOS TIPOS DE AUTORIZACIÓN (DE MANERA ABREVIADA EN SQL, **ALL PRIVILEGES**), NINGUNO DE ELLOS O UNA COMBINACIÓN DETERMINADA DE LOS MISMOS
- SI EL OBJETO SOBRE EL QUE SE CONCEDEN PRIVILEGIOS ES UNA VISTA SÓLO SON APLICABLES LOS PRIVILEGIOS DE LECTURA, INSERCIÓN, MODIFICACIÓN Y BORRADO

PRIVILEGIOS SOBRE EL SISTEMA

- **CREATE SESSION –** CREAR SESIONES DE TRABAJO
- **CREATE TABLE –** CREAR TABLAS,
- **CREATE VIEW –** CREAR VISTAS
- **CREATE PROCEDURE –** CREAR PROCEDIMIENTOS
- **CREATE SYNONYM –** CREAR SINÓNIMOS
- **CREATE USER –** CREAR USUARIOS
- **LOCK ANY TABLE –** ESTABLECER BLOQUEOS
- …

TRANSMISIÓN DE PRIVILEGIOS

- UN USUARIO AL QUE SE LE CONCEDE UN PRIVILEGIO NO ESTÁ AUTORIZADO DE MANERA PREDETERMINADA A CONCEDÉRSELO A OTROS USUARIOS
- SI SE DESEA CONCEDER UN PRIVILEGIO A UN USUARIO Y PERMITIRLE QUE LO TRANSMITA A OTROS USUARIOS HAY QUE INDICARLO DE MANERA ESPECÍFICA
- LA TRANSMISIÓN DE UNA AUTORIZACIÓN DE UN USUARIO A OTRO PUEDE REPRESENTARSE MEDIANTE UN GRAFO
- EL CREADOR DE UN OBJETO (TABLA, VISTA, ...) ES EL PROPIETARIO DEL OBJETO Y OBTIENE TODOS LOS PRIVILEGIOS SOBRE EL MISMO, INCLUYENDO EL PRIVILEGIO DE CONCEDER PRIVILEGIOS A OTROS

GRAFO DE TRANSMISIÓN DE PRIVILEGIOS

- LOS NODOS DE ESTE GRAFO SON LOS USUARIOS
- SE INCLUYE UN ARCO ENTRE DOS NODOS U_I Y U_J SI U_I TRANSMITE EL PRIVILEGIO A U_J
- LA RAÍZ DEL GRAFO ES EL ADMINISTRADOR DE LA BASE DE DATOS
- UN USUARIO TIENE AUTORIZACIÓN SI Y SÓLO SI HAY UN CAMINO DESDE LA RAÍZ DEL GRAFO HASTA EL NODO QUE REPRESENTA A ESE USUARIO

CONCESIÓN DE PRIVILEGIOS

- EN SQL LA SENTENCIA PARA CONCEDER PRIVILEGIOS ES **GRANT** (QUE, ACERTADAMENTE, SIGNIFICA CONCEDER)
- DEPENDIENDO DE QUE SE TRATE DE PRIVILEGIOS SOBRE OBJETOS O SOBRE EL SISTEMA, DISTINGUIMOS DOS VARIANTES DE LA INSTRUCCIÓN

CONCESIÓN DE PRIVILEGIOS SOBRE OBJETOS

GRANT {privilegio[(lista_columnas)][, ...] | **ALL**[**PRIVILEGES**]}

ON objeto

TO {lista_usuarios | **PUBLIC**}

[**WITH GRANT OPTION**];

EJEMPLOS CONCESIÓN DE PRIVILEGIOS SOBRE OBJETOS

GRANT SELECT, INSERT, UPDATE(cdc)**, DELETE**

ON cliente

TO john;

GRANT ALL PRIVILEGES

ON sucursal

TO public

WITH GRANT OPTION;

CONCESIÓN DE PRIVILEGIOS DEL SISTEMA

GRANT lista_privilegios_sistema

TO {lista_usuarios | **PUBLIC**}

[**WITH ADMIN OPTION**];

EJEMPLOS CONCESIÓN DE PRIVILEGIOS DEL SISTEMA

GRANT CREATE SESSION

TO john;

GRANT CREATE TABLE

TO john

WITH ADMIN OPTION;

ELIMINACIÓN DE PRIVILEGIOS

- LA SENTENCIA PARA RETIRAR UNA AUTORIZACIÓN ES **REVOKE**
- DEPENDIENDO DE QUE SE TRATE DE PRIVILEGIOS SOBRE OBJETOS O SOBRE EL SISTEMA, DISTINGUIMOS DOS VARIANTES DE LA INSTRUCCIÓN
- LA RETIRADA DE UN PRIVILEGIO A UN USUARIO PUEDE HACER QUE OTROS USUARIOS TAMBIÉN LO PIERDAN
- ESTE COMPORTAMIENTO (PREDETERMINADO EN LA MAYORÍA DE LOS SISTEMAS DE BASES DE DATOS) SE DENOMINA **RETIRADA EN CASCADA** O **CADENA**

ELIMINACIÓN DE PRIVILEGIOS SOBRE OBJETOS

REVOKE {privilegio[, ...] | **ALL**[**PRIVILEGES**]}

ON objeto

FROM {lista_usuarios | **PUBLIC**}

[**CASCADE CONSTRAINT**];

- LA OPCIÓN **CASCADE CONSTRAINT** ELIMINA CUALQUIER CONDICIÓN DE INTEGRIDAD REFERENCIAL DEFINIDA USANDO EL PRIVILEGIO **REFERENCES** QUE SE INVOQUE EN LA INSTRUCCIÓN

EJEMPLOS ELIMINACIÓN DE PRIVILEGIOS SOBRE OBJETOS

REVOKE DELETE

ON cliente

FROM john;

REVOKE REFERENCES

ON sucursal

FROM john

CASCADE CONSTRAINT;

ELIMINACIÓN DE PRIVILEGIOS DEL SISTEMA

REVOKE lista_privilegios_sistema

FROM {lista_usuarios | **PUBLIC**};

VISTAS Y SEGURIDAD

- LAS VISTAS CONSTITUYEN UN MEDIO PARA PROPORCIONAR A UN USUARIO UN MODELO PERSONALIZADO DE LA BASE DE DATOS
  - LAS VISTAS OCULTAN A LOS USUARIOS LOS DATOS QUE NO NECESITA VER
- SIMPLIFICAN EL USO DEL SISTEMA
  - PUES EL USUARIO RESTRINGE SU ATENCIÓN SÓLO A LOS DATOS DE SU INTERÉS
- MEJORAN LA SEGURIDAD
  - AUNQUE A UN USUARIO SE LE NIEGUE EL ACCESO DIRECTO A UNA RELACIÓN, PUEDE QUE SE LE PERMITA EL ACCESO A PARTE DE ESA RELACIÓN MEDIANTE UNA VISTA
  - DISMINUYEN EL GRANO DE LOS OBJETOS SOBRE LOS QUE SE CONCEDE AUTORIZACIÓN

AUTORIZACIONES Y VISTAS

- PARA QUE UN USUARIO PUEDA CREAR UN VISTA NECESITA LA AUTORIZACIÓN DEL SISTEMA **CREATE VIEW** Y LA AUTORIZACIÓN DE LECTURA (**SELECT**) SOBRE TODOS LOS OBJETOS SOBRE LOS QUE SE BASA LA VISTA
- CON RESPECTO A LAS OPERACIONES DE ACTUALIZACIÓN SOBRE VISTAS CREADAS POR UN USUARIO, EL USUARIO SÓLO POSEERÁ SOBRE LA VISTA AQUELLOS PRIVILEGIOS QUE TENGA SOBRE CADA UNO DE LOS OBJETOS SOBRE LOS QUE SE BASA LA VISTA
- PARA QUE UN USUARIO PUEDA CONCEDER PRIVILEGIOS SOBRE UNA VISTA PROPIETARIA DEBE TENER ESE PRIVILEGIO CON LA OPCIÓN DE PODER TRANSMITIRLO SOBRE TODAS LAS TABLAS EN LAS QUE SE BASA LA VISTA

EL CONCEPTO DE ROL

- PARA SIMPLIFICAR LA GESTIÓN DE USUARIOS, ÉSTOS PUEDEN AGRUPARSE PARA FORMAR UN CONJUNTO O GRUPO DE USUARIOS (ROL) CON LAS MISMAS NECESIDADES RESPECTO DEL SISTEMA
  - CADA GRUPO DE USUARIOS (ROL) INTERACTÚA CON EL SISTEMA DE UNA FORMA ESPECÍFICA
- SE PUEDEN CREAR ROLES Y CONCEDER/ELIMINAR AUTORIZACIONES A ESTOS ROLES DE IGUAL MODO QUE SE CONCEDEN/ELIMINAN A USUARIOS INDIVIDUALES
- LOS ROLES SE CONCEDEN/ELIMINAN POSTERIORMENTE A LOS DISTINTOS USUARIOS COMO SI FUERAN UN PRIVILEGIO DEL SISTEMA
- SI SE TRANSMITE UN ROL CON LA OPCIÓN **WITH ADMIN OPTION,** EL USUARIO PUEDE TRANSMITIR Y ALTERAR EL ROL.

CREACIÓN DE ROLES

CREATE ROLE nombre_rol

{[NOT IDENTIFIED], [IDENTIFIED BY password]**};**

- SI SE ESPECIFICA LA CLÁUSULA **IDENTIFIED BY** LOS USUARIOS DEBEN ESPECIFICAR EL PASSWORD PARA ACTIVAR EL ROL
- CUANDO SE CREA EL ROL, ÉSTE NO TIENE NINGÚN PRIVILEGIO

BORRADO DE ROLES

DROP ROLE nombre_rol;

- PARA PODER BORRAR UN ROL EL USUARIO DEBE HABER CREADO EL ROL O HABÉRSELE CONCEDIDO EL ROL CON EL PRIVILEGIO **WITH ADMIN OPTION** O TENER EL PRIVILEGIO DEL SISTEMA **DROP ANY ROLE**

ACTIVAR/DESACTIVAR ROLE

SET ROLE {nombre_rol1 [**IDENTIFIED BY** password] [, ...] | **ALL** [**EXCEPT** nombre_rol2 [, ...]] | **NONE**};

- SI EL ROL TIENE UN PASSWORD SE DEBE ESPECIFICAR PARA ACTIVAR EL ROL
- **ALL** ACTIVA TODOS LOS ROLES QUE SE HAN CONCEDIDO AL USUARIO PARA LA SESIÓN ACTUAL EXCEPTO LOS QUE APAREZCAN DESPUÉS DE LA CLÁUSULA **EXCEPT**
  - NO SE PUEDE USAR ESTA OPCIÓN CON ROLES QUE TENGAN PASSWORD
- LA CLÁUSULA **NONE** DESACTIVA TODOS LOS ROLES EN LA SESIÓN ACTUAL

MODIFICAR UN ROL

ALTER ROLE nombre_rol

{**NOT IDENTIFIED** | **IDENTIFIED BY** password};

- PERMITE CAMBIAR LA FORMA DE ACTIVACIÓN DE UN ROL (MEDIANTE PASSWORD O SIN ÉL)
- PARA PODER ALTERAR UN ROL EL USUARIO DEBE TENER EL ROL CON EL PRIVILEGIO **WITH ADMIN OPTION** O TENER EL PRIVILEGIO DEL SISTEMA **ALTER ANY ROLE**

EJEMPLOS

```
CREATE ROLE mi_rol1
IDENTIFIED BY xyz123;

CREATE ROLE mi_rol2;

GRANT SELECT, INSERT, UPDATE, DELETE
ON cliente
TO mi_rol1;
```

EJEMPLOS

GRANT SELECT

ON sucursal

TO mi_rol2;

- **GRANT** mi_rol1, mi_rol2 **TO** John;

- **SET ROLE** mi_rol1 **IDENTIFIED BY** xyz123, mi_rol2;

CADENAS DE ROLES

- PUEDEN CREARSE CADENAS DE ROLES
  - ROLES CONCEDIDOS A OTROS ROLES
- POR EJEMPLO, EL ROL *EMPLEADO* SE PUEDE CONCEDER A TODOS LOS *CAJEROS*. A SU VEZ, EL ROL *CAJERO* SE PUEDE CONCEDER A TODOS LOS *GESTORES*
  - EL ROL *GESTOR* HEREDA TODOS LOS PRIVILEGIOS CONCEDIDOS A LOS ROLES *EMPLEADO* Y *CAJERO* Y TIENE, ADEMÁS, LOS PRIVILEGIOS CONCEDIDOS DIRECTAMENTE A *GESTOR*

TEMA 4. GESTIÓN DE TRANSACCIONES

- CONCEPTO DE TRANSACCIÓN
- PROPIEDADES ACID
- ESTADOS DE UNA TRANSACCIÓN
- ACCESO CONCURRENTE Y TRANSACCIONES
- PLANIFICACIÓN DE UNA TRANSACCIÓN
- PLANIFICACIÓN SECUENCIAL
- SECUENCIALIDAD O SERIABILIDAD DE UNA PLANIFICACIÓN EN CUANTO A CONFLICTOS
- RECUPERABILIDAD

CONCEPTO DE TRANSACCIÓN

- SE LLAMA **TRANSACCIÓN** A UNA COLECCIÓN DE OPERACIONES (DE ACCESO Y ACTUALIZACIÓN) QUE FORMAN UNA **UNIDAD LÓGICA DE TRABAJO** PARA EL SISTEMA
  - EJ: TRANSFERENCIA DE FONDOS ENTRE DOS CUENTAS CORRIENTES
- EL CONCEPTO DE TRANSACCIÓN ES FUNDAMENTAL EN LOS SISTEMAS DE BASES DE DATOS Y, PARTICULARMENTE, EN AQUELLOS QUE HACEN PROCESAMIENTO OLTP
- LAS OPERACIONES CRÍTICAS DE UNA TRANSACCIÓN SON LAS DE LECTURA Y ESCRITURA

TRANSFERENCIA DE FONDOS ENTRE DOS CUENTAS CORRIENTES

- T_I: LEER(A);

  A = A – 50;

  ESCRIBIR(A);

  LEER(B);

  B = B + 50;

  ESCRIBIR(B);

PROPIEDADES ACID

- **ATOMICITY (ATOMICIDAD):** O TODAS LAS OPERACIONES DE LA TRANSACCIÓN SE REALIZAN ADECUADAMENTE EN LA BASE DE DATOS O NINGUNA DE ELLAS
- **CONSISTENCY (CONSISTENCIA):** LA EJECUCIÓN AISLADA DE LA TRANSACCIÓN (ES DECIR, SIN OTRA TRANSACCIÓN QUE SE EJECUTE CONCURRENTEMENTE) CONSERVA LA CONSISTENCIA DE LA BASE DE DATOS
- **ISOLATION (AISLAMIENTO):** AUNQUE SE EJECUTEN VARIAS TRANSACCIONES CONCURRENTEMENTE, EL SISTEMA GARANTIZA QUE PARA CADA PAR DE TRANSACCIONES T_I Y T_J, SE CUMPLE QUE PARA LOS EFECTOS DE T_I, O BIEN T_J HA TERMINADO SU EJECUCIÓN ANTES DE QUE COMIENCE T_I, O BIEN QUE T_J HA COMENZADO SU EJECUCIÓN DESPUÉS DE QUE TERMINE T_I
- **DURABILITY (PERSISTENCIA):** TRAS LA FINALIZACIÓN CON ÉXITO DE UNA TRANSACCIÓN, LOS CAMBIOS REALIZADOS EN LA BASE DE DATOS PERMANECEN, INCLUSO SI HAY FALLOS EN EL SISTEMA

ESTADOS DE UNA TRANSACCIÓN

- **ACTIVA:** ES EL ESTADO INICIAL. LA TRANSACCIÓN PERMANECE EN ESTE ESTADO DURANTE SU EJECUCIÓN
- **ABORTADA:** DESPUÉS DE HABER DESHECHO (O RETROCEDIDO) LOS CAMBIOS EFECTUADOS POR LA TRANSACCIÓN Y RESTABLECIDO LA BASE DE DATOS A SU ESTADO ANTERIOR AL COMIENZO DE LA TRANSACCIÓN
- **FALLIDA:** TRAS DESCUBRIR QUE NO PUEDE CONTINUAR LA EJECUCIÓN NORMAL DE LA TRANSACCIÓN
- **CONFIRMADA O COMPROMETIDA:** TRAS COMPLETARSE CON ÉXITO. UNA TRANSACCIÓN COMPROMETIDA QUE HAYA HECHO MODIFICACIONES TRANSFORMA LA BASE DE DATOS LLEVÁNDOLA A UN NUEVO ESTADO CONSISTENTE, QUE PERMANECE INCLUSO SI HAY UN FALLO EN EL SISTEMA
  - CUANDO UNA TRANSACCIÓN SE HA COMPROMETIDO NO SE PUEDEN DESHACER SUS EFECTOS ABORTÁNDOLA
  - LA ÚNICA FORMA DE DESHACER LOS CAMBIOS DE UNA TRANSACCIÓN COMPROMETIDA ES EJECUTANDO UNA **TRANSACCIÓN COMPENSADORA**. LA RESPONSABILIDAD DE CREAR Y EJECUTAR TRANSACCIONES COMPENSADORAS RECAE SOBRE EL USUARIO
- **PARCIALMENTE CONFIRMADA O COMPROMETIDA:** DESPUÉS DE EJECUTARSE LA ÚLTIMA INSTRUCCIÓN

FLUJO DE ESTADOS EN UNA TRANSACCIÓN

- UNA TRANSACCIÓN COMIENZA EN EL ESTADO ACTIVA
- CUANDO ACABA SU ÚLTIMA INSTRUCCIÓN PASA AL ESTADO DE PARCIALMENTE COMPROMETIDA
  - EN ESTE PUNTO LA TRANSACCIÓN HA TERMINADO SU EJECUCIÓN, PERO ES POSIBLE QUE AÚN TENGA QUE SER ABORTADA, PUESTO QUE LOS DATOS ACTUALES PUEDEN ESTAR TODAVÍA EN LA MEMORIA PRINCIPAL Y PUEDE PRODUCIRSE UN FALLO EN EL HARDWARE ANTES DE QUE SE COMPLETE CON ÉXITO
- EL SISTEMA DE BASE DE DATOS ESCRIBE EN DISCO LA INFORMACIÓN SUFICIENTE PARA QUE, INCLUSO AL PRODUCIRSE UN FALLO (EN MEMORIA PRINCIPAL), PUEDAN REPRODUCIRSE LOS CAMBIOS HECHOS POR LA TRANSACCIÓN AL REINICIAR EL SISTEMA TRAS EL FALLO. CUANDO SE TERMINA DE ESCRIBIR ESTA INFORMACIÓN, LA TRANSACCIÓN PASA AL ESTADO COMPROMETIDA
- UNA TRANSACCIÓN LLEGA AL ESTADO FALLIDA DESPUÉS DE QUE EL SISTEMA DETERMINE QUE DICHA TRANSACCIÓN NO PUEDE CONTINUAR SU EJECUCIÓN NORMAL (POR EJEMPLO, A CAUSA DE ERRORES DE HARDWARE O LÓGICOS). UNA TRANSACCIÓN DE ESTE TIPO SE DEBE RETROCEDER
- DESPUÉS DEL ESTADO FALLIDA SE PASA AL ESTADO ABORTADA

OPCIONES DEL SISTEMA CON TRANSACCIONES ABORTADAS

- SI LA TRANSACCIÓN SE ENCUENTRA EN EL ESTADO ABORTADA, EL SISTEMA TIENE DOS OPCIONES:
  - **REINICIAR** LA TRANSACCIÓN, PERO SÓLO SI LA TRANSACCIÓN SE HA ABORTADO A CAUSA DE ALGÚN ERROR HARDWARE O SOFTWARE QUE NO LO HAYA PROVOCADO LA LÓGICA INTERNA DE LA TRANSACCIÓN
  - **CANCELAR** LA TRANSACCIÓN. NORMALMENTE SE HACE ESTO SI HAY ALGÚN ERROR INTERNO LÓGICO QUE SÓLO SE PUEDE CORREGIR ESCRIBIENDO DE NUEVO EL PROGRAMA DE APLICACIÓN, O DEBIDO A UNA ENTRADA INCORRECTA O DEBIDO A QUE NO SE HAN ENCONTRADO LOS DATOS DESEADOS EN LA BASE DE DATOS

ACCESO CONCURRENTE Y TRANSACCIONES

- EL ACCESO CONCURRENTE GENERA PROBLEMAS AL EJECUTAR SIMULTÁNEAMENTE VARIAS TRANSACCIONES, PUES SI NO SE CONTROLA CONVENIENTEMENTE PUEDE ORIGINAR INCONSISTENCIA EN LOS DATOS, A PESAR DE QUE CADA TRANSACCIÓN CONSIDERADA DE FORMA INDIVIDUAL SEA CONSISTENTE
- LAS RAZONES PARA PERMITIR LA CONCURRENCIA SON QUE SE MEJORAN EL RENDIMIENTO Y LA UTILIZACIÓN DEL SISTEMA Y SE REDUCE EL TIEMPO DE ESPERA DE LAS TRANSACCIONES
- ES RESPONSABILIDAD DEL SISTEMA DE BASE DE DATOS CONTROLAR LA INTERACCIÓN ENTRE LAS TRANSACCIONES PARA EVITAR QUE SE DESTRUYA LA CONSISTENCIA DE DATOS (PROPIEDAD DE AISLAMIENTO)

PLANIFICACIÓN DE TRANSACCIONES

- UNA **PLANIFICACIÓN** (O **PLAN** O **HISTORIA**) P DE N TRANSACCIONES T_1, T_2, ..., T_N ES UNA ORDENACIÓN DE TODAS LAS INSTRUCCIONES DE DICHAS TRANSACCIONES QUE DEBE CONSERVAR EL ORDEN EN QUE APARECEN LAS INSTRUCCIONES EN CADA TRANSACCIÓN INDIVIDUAL
- UNA PLANIFICACIÓN ES **VÁLIDA** SI CONSERVA LA CONSISTENCIA DE LA BASE DE DATOS

PLANIFICACIÓN DE TRANSACCIONES

- T_1:

```
LEER(A);
A = A – 50;
ESCRIBIR(A);
LEER(B);
B = B + 50;
ESCRIBIR(B);
```

- T_2:

```
LEER(A);
TEMP = A * 0.1;
A = A – TEMP;
ESCRIBIR(A);
LEER(B);
B = B + TEMP;
ESCRIBIR(B);
```

PLANIFICACIÓN VÁLIDA

T_1	T_2
Leer(A); A = A – 50; Escribir(A);	
	Leer(A); temp = A * 0.1; A = A – temp; Escribir(A);
Leer(B); B = B + 50; Escribir(B);	
	Leer(B); B = B + temp; Escribir(B);

PLANIFICACIÓN NO VÁLIDA

T_1	T_2
Leer(A); A = A – 50;	
	Leer(A); temp = A * 0.1; A = A – temp; Escribir(A);
	Leer(B);
Escribir(A);	
Leer(B);	
B = B + 50;	
Escribir(B);	
	B = B + temp;
	Escribir(B);

PLANIFICACIONES SECUENCIALES

- UNA PLANIFICACIÓN SE DICE QUE ES **SECUENCIAL** O **EN SERIE** SI LAS INSTRUCCIONES PERTENECIENTES A UNA ÚNICA TRANSACCIÓN ESTÁN JUNTAS EN DICHA PLANIFICACIÓN, ES DECIR, SE EJECUTAN DE MANERA CONSECUTIVA Y SIN OPERACIONES INTERCALADAS DE LA OTRA TRANSACCIÓN
- PARA UN CONJUNTO DE N TRANSACCIONES EXISTEN N! PLANIFICACIONES SECUENCIALES VÁLIDAS DISTINTAS
- LAS PLANIFICACIONES SECUENCIALES SIEMPRE SON VÁLIDAS (SI LO SON CADA UNA DE LAS TRANSACCIONES QUE LA COMPONEN)

PLANIFICACIÓN SECUENCIAL <T_1, T_2 >

T_1	T_2
Leer(A); A = A – 50; Escribir(A); Leer(B); B = B + 50; Escribir(B);	
	Leer(A); temp = A * 0.1; A = A – temp; Escribir(A); Leer(B); B = B + temp; Escribir(B);

PLANIFICACIÓN SECUENCIAL <T_2, T_1>

T_1	T_2
	Leer(A); temp = A * 0.1; A = A – temp; Escribir(A); Leer(B); B = B + temp; Escribir(B);
Leer(A); A = A – 50; Escribir(A); Leer(B); B = B + 50; Escribir(B);	

PLANIFICACIÓN NO SECUENCIAL "EQUIVALENTE" A LA PLANIFICACIÓN SECUENCIAL $<T_1, T_2>$

T_1	T_2
Leer(A); A = A – 50; Escribir(A);	
	Leer(A); temp = A * 0.1; A = A – temp; Escribir(A);
Leer(B); B = B + 50; Escribir(B);	
	Leer(B); B = B + temp; Escribir(B);

INSTRUCCIONES DE UNA PLANIFICACIÓN EN CONFLICTO

- LAS ÚNICAS OPERACIONES SIGNIFICATIVAS DE UNA TRANSACCIÓN SON, DESDE EL PUNTO DE VISTA DE LA PLANIFICACIÓN, LAS INSTRUCCIONES DE ACCESO A LOS DATOS, ES DECIR, LEER Y ESCRIBIR
- SE DICE QUE DOS INSTRUCCIONES CONSECUTIVAS DE UN PLAN ESTÁN EN **CONFLICTO** SI SATISFACEN LAS TRES CONDICIONES SIGUIENTES:
  - (I) PERTENECEN A DIFERENTES TRANSACCIONES
  - (II) TIENEN ACCESO AL MISMO ELEMENTO X
  - (III) AL MENOS UNA DE LAS OPERACIONES ES ESCRIBIR(X)

INSTRUCCIONES DE UNA PLANIFICACIÓN EN CONFLICTO

T_1	T_2
Leer(A);	
Escribir(A);	
	Leer(A);
	Escribir(A);
Leer(B);	
Escribir(B);	
	Leer(B);
	Escribir(B);

PLANIFICACIONES EQUIVALENTES EN CUANTO A CONFLICTOS

- CONSIDEREMOS UNA PLANIFICACIÓN P EN LA CUAL HAY DOS INSTRUCCIONES CONSECUTIVAS I_I E I_J PERTENECIENTES A LAS TRANSACCIONES T_I Y T_J, RESPECTIVAMENTE ($I \neq J$). SI I_I E I_J NO ESTÁN EN CONFLICTO ENTONCES PODEMOS INTERCAMBIAR EL ORDEN DE I_I E I_J (PUES SU ORDEN DE EJECUCIÓN NO AFECTA AL RESULTADO DE LA TRANSACCIÓN) PARA OBTENER UNA NUEVA PLANIFICACIÓN P' EQUIVALENTE A P
- QUIZÁS PODAMOS PROSEGUIR INTERCAMBIANDO INSTRUCCIONES NO CONFLICTIVAS
- DOS PLANIFICACIONES P Y P' SON **EQUIVALENTES EN CUANTO A CONFLICTOS** SI P SE PUEDE TRANSFORMAR EN P' POR MEDIO DE UNA SERIE DE INTERCAMBIOS DE INSTRUCCIONES CONSECUTIVAS NO CONFLICTIVAS

SECUENCIALIDAD O SERIABILIDAD DE UNA PLANIFICACIÓN EN CUANTO A CONFLICTOS

- UNA PLANIFICACIÓN P ES **SECUENCIABLE O SERIALIZABLE EN CUANTO A CONFLICTOS** SI ES EQUIVALENTE EN CUANTO A CONFLICTOS A ALGUNA PLANIFICACIÓN SECUENCIAL
- CUALQUIER PLANIFICACIÓN SECUENCIABLE (EN CUANTO A CONFLICTOS) ES VÁLIDA
- NO TODAS LAS PLANIFICACIONES SON SECUENCIABLES EN CUANTO A CONFLICTOS
- EXISTEN ALGORITMOS SENCILLOS PARA COMPROBAR SI UNA PLANIFICACIÓN ES SECUENCIABLE EN CUANTO A CONFLICTOS

PLANIFICACIONES SECUENCIABLE EN CUANTO A CONFLICTOS

- PLANIFICACIÓN NO SECUENCIAL
- SECUENCIABLE EN CUANTO A CONFLICTOS
- EQUIVALENTE A $<T_1, T_2>$

T_1	T_2
Leer(A);	
Escribir(A);	
	Leer(A);
	Escribir(A);
Leer(B);	
Escribir(B);	
	Leer(B);
	Escribir(B);

PLANIFICACIONES NO SECUENCIABLE EN CUANTO A CONFLICTOS

T_3	T_4
Leer(A); Escribir(A);	Escribir(A);

ESQUEMAS DE CONTROL DE CONCURRENCIA

- LA MAYOR PARTE DE LOS ESQUEMAS DE CONTROL DE CONCURRENCIA IMPLEMENTADOS EN LOS SISTEMAS DE BASES DE DATOS SE BASAN EN LA APLICACIÓN DE PROTOCOLOS (CONJUNTO DE REGLAS) QUE GARANTIZAN LA SECUENCIALIDAD DE TODAS LAS PLANIFICACIONES EN QUE PARTICIPEN LAS TRANSACCIONES INDIVIDUALES QUE LOS SIGUEN
  - OBSÉRVESE QUE LAS TRANSACCIONES SE ESTÁN INTRODUCIENDO CONTINUAMENTE EN EL SISTEMA Y POR ELLO ES DIFÍCIL DETERMINAR CUANDO COMIENZA Y CUANDO TERMINA UNA PLANIFICACIÓN
- EXISTEN VARIOS PROTOCOLOS DE CONTROL DE CONCURRENCIA QUE GARANTIZAN LA SECUENCIALIDAD:
  - LA TÉCNICA MÁS COMÚN, DENOMINADA **BLOQUEO DE DOS FASES,** ESTÁ BASADA EN EL BLOQUEO DE ELEMENTOS DE DATOS
  - OTROS PROTOCOLOS SON LOS DE **MARCAS TEMPORALES,** LOS **ESQUEMAS MULTIVERSIÓN** (USADO POR ORACLE), ...

RECUPERABILIDAD

- UNA TRANSACCIÓN T_J SE DICE QUE DEPENDE DE T_I SI T_J LEE DATOS QUE HA ESCRITO T_I
- SI T_J DEPENDE DE T_I Y T_I FALLA, POR LA RAZÓN QUE SEA, ES NECESARIO:
  - DESHACER T_I PARA GARANTIZAR LA ATOMICIDAD DE T_I
  - ABORTAR T_J PARA MANTENER LA CONSISTENCIA.
- SI VARIAS TRANSACCIONES DEPENDEN DE T_I Y T_I FALLA, PUEDE QUE SEA NECESARIO RETROCEDER VARIAS (MÁS DE UNA) TRANSACCIONES PARA RECUPERAR CORRECTAMENTE EL SISTEMA AL ESTADO PREVIO AL FALLO (RETROCESO EN CASCADA)
  - EFECTO NO DESEABLE

RECUPERABILIDAD

T_5	T_6
Leer(A); Escribir(A);	
	Leer(A); Escribir(A);
Leer(B); Escribir(B);	

- SI EL SISTEMA PERMITE QUE T_6 SE COMPROMETA INMEDIATAMENTE DESPUÉS DE EJECUTAR LA INSTRUCCIÓN ESCRIBIR(A) Y T_5 FALLA ANTES DE COMPLETARSE, SE DEBE:
  - DESHACER T_5 (PARA MANTENER LA ATOMICIDAD)
  - DESHACER T_6 (PARA MANTENER LA CONSISTENCIA)
- SIN EMBARGO, T_6 YA SE HA COMPROMETIDO Y NO PUEDE ABORTARSE. DE ESTE MODO SE LLEGA A UNA SITUACIÓN EN LA CUAL ES IMPOSIBLE RECUPERARSE CORRECTAMENTE DEL FALLO DE T_5.

PLANIFICACIÓN RECUPERABLE

- UNA **PLANIFICACIÓN** SE DICE QUE ES **RECUPERABLE** SI PARA TODO PAR DE TRANSACCIONES T_I Y T_J DE LA MISMA, TALES QUE T_J DEPENDA DE T_I, LA OPERACIÓN COMPROMETER DE T_I APARECE ANTES QUE LA DE T_J
- CUALQUIER PLANIFICACIÓN SECUENCIAL ES RECUPERABLE
- EN UNA PLANIFICACIÓN RECUPERABLE NINGUNA TRANSACCIÓN CONFIRMADA TIENE QUE DESHACERSE JAMÁS

PLANIFICACIÓN SIN CASCADA

- UNA **PLANIFICACIÓN SIN CASCADA** ES AQUÉLLA PARA LA QUE TODO PAR DE TRANSACCIONES T_I Y T_J DE LA MISMA, TALES QUE T_J DEPENDA DE T_I, LA OPERACIÓN COMPROMETER DE T_I APARECE ANTES QUE LA CORRESPONDIENTE OPERACIÓN DE LECTURA DE T_J
- TODA PLANIFICACIÓN SIN CASCADA ES TAMBIÉN RECUPERABLE